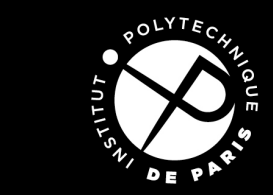
Network-wide intrusion detection through statistical analysis of event logs : an interaction-centric approach

Corentin Larroche



INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2021IPPAT041

Thèse de doctorat

Network-Wide Intrusion Detection through Statistical Analysis of Event Logs: an Interaction-Centric Approach

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

École doctorale n°574 Ecole Doctorale de Mathématiques Hadamard (EDMH)
Spécialité de doctorat : Mathématiques aux interfaces

Thèse présentée et soutenue à Palaiseau, le 25 octobre 2021, par

CORENTIN LARROCHE

Composition du Jury :

Éric Totel Professeur, Télécom SudParis	Président
Nick Heard Professeur, Imperial College London	Rapporteur
Fabrice Rossi Professeur, Université Paris-Dauphine	Rapporteur
Isabelle Chrisment Professeure, Télécom Nancy	Examinatrice
Stephan Clémençon Professeur, Télécom Paris	Directeur de thèse
Johan Mazel Docteur, ANSSI	Encadrant de thèse
Anaël Beaugnon Docteure, Roche	Invitée

Remerciements

First of all, I would like to thank Nick Heard and Fabrice Rossi for the time they have devoted to reviewing this thesis. I was truly honored to see the effort they put in exploring the depths of my research work, and the discussions we had during the defense were both pleasing and inspiring. Je remercie également Éric Totel et Isabelle Chrisment d'avoir accepté de participer à mon jury de thèse, joignant ainsi leur expertise sur les questions de sécurité numérique aux considérations mathématiques et statistiques soulevées par les autres membres.

Mes remerciements suivants vont naturellement à Stephan Clémençon et Johan Mazel pour leur encadrement tout au long de ce chemin sinueux mais tellement enrichissant. Cette thèse fut pour moi l'occasion de découvrir un grand nombre de sujets, et si j'en ai retiré une somme inestimable de connaissances nouvelles, c'est en grande partie grâce à eux. Je les remercie sincèrement pour leur aide (tant scientifique que logistique) et pour leur confiance. Je n'oublie pas non plus ce que la réussite de ce projet doit à Anaël Beaugnon, de par son investissement dans le suivi de la thèse et son regard toujours éclairé sur mes travaux, mais aussi parce qu'elle est à l'origine de tout : le stage qu'elle m'a permis d'effectuer à l'ANSSI fut le début d'une longue et belle aventure, et je lui en suis infiniment reconnaissant.

J'ai aussi une pensée pour les membres actuels et anciens de la SDO qui m'ont fait bénéficier de leur immense savoir et de leurs intuitions sur le vaste sujet de la détection d'intrusion. Géraud, Thibaut, Matthieu, Yohann, Aurélien, Marion, vous avez chacun à votre façon contribué à l'orientation de mes recherches, et même si ce n'est pas toujours évident, il y a bel et bien des traces de nos discussions dans les développements plus ou moins éthérés qui composent ce manuscrit. Merci également à ma hiérarchie à l'ANSSI, et notamment Pierre Chifflier, de m'avoir fait pleinement confiance tout au long de cette thèse, me laissant explorer à ma guise toutes sortes d'idées parfois un peu ésotériques.

Sur une note plus personnelle, je remercie chaleureusement ma famille et mes amis pour leur soutien. Dans cette entreprise de longue haleine et non dénuée d'incertitudes et autres angoisses, j'ai eu la chance d'être bien entouré et de pouvoir régulièrement évacuer mes tracas dans la joie et la bonne humeur. Je vous prie respectueusement de continuer à exister, et j'espère vivre encore bien des moments hors du temps avec vous toutes et tous.

Enfin, j'adresse mes ultimes remerciements à Marie – pour tout, et notamment d'avoir été à mes côtés dans les moments les plus durs. Achever cette thèse n'aura assurément pas été facile, mais cela l'aurait été encore beaucoup moins sans toi, et si je me réjouis d'avoir survécu à la tempête et de revenir enfin à une vie plus tranquille, c'est en grande partie parce que tu es là pour la partager.

Contents

List of Figures

List of Tables

List of Publications

- Corentin Larroche, Johan Mazel, and Stephan Clémençon. Percolation-Based Detection of Anomalous Subgraphs in Complex Networks. In *Proceedings of the Symposium on Intelligent Data Analysis (IDA)*, 2020.
- Corentin Larroche, Johan Mazel, and Stephan Clémençon. Recent Trends in Statistical Analysis of Event Logs for Network-Wide Intrusion Detection. In *Proceedings of the Conference on Artificial Intelligence for Defense (CAID)*, 2020.
- Corentin Larroche, Johan Mazel, and Stephan Clémençon. Anomalous Cluster Detection in Large Networks with Diffusion-Percolation Testing. In *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2021.
- Corentin Larroche, Johan Mazel, and Stephan Clémençon. Dynamically Modelling Heterogeneous Higher-Order Interactions for Malicious Behavior Detection in Event Logs. Preprint, 2021.

List of Abbreviations

APT	**A**dvanced **P**ersistent **T**hreat
C&C	**C**ommand **&** **C**ontrol
DNS	**D**omain **N**ame **S**ystem
IDS	**I**ntrusion **D**etection **S**ystem
IP	**I**nternet **P**rotocol
SIEM	**S**ecurity **I**nformation and **E**vent **M**anagement
TTPs	**T**actics, **T**echniques and **P**rocedures
VPN	**V**irtual **P**rivate **N**etwork
AUC	**A**rea **U**nder the **C**urve
ROC	**R**eceiver **O**perating **C**haracteristic
DR	**D**etection **R**ate
FPR	**F**alse **P**ositive **R**ate
TPR	**T**rue **P**ositive **R**ate
SNR	**S**ignal-to-**N**oise **R**atio
CKF	**C**ollaborative **K**alman **F**iltering
GSP	**G**raph **S**ignal **P**rocessing
HMM	**H**idden **M**arkov **M**odel
HPF	**H**ierarchical **P**oisson **F**actorization
LOC	**L**argest **O**pen **C**luster
MAP	**M**aximum **A** **P**osteriori
MTL	**M**ulti-**T**ask **L**earning
MRF	**M**arkov **R**andom **F**ield
NCE	**N**oise **C**ontrastive **E**stimation
NNG	**N**earest **N**eighbor **G**raph
PMF	**P**robabilistic **M**atrix **F**actorization
SGD	**S**tochastic **G**radient **D**escent
SVD	**S**ingular **V**alue **D**ecomposition

List of Symbols

Global notations

$[n]$	Set of integers from 1 to n
$x_{1:n}$	Set of elements $\{x_1, \dots, x_n\}$
$\odot$	Element-wise product of two vectors
$\otimes$	Outer product of two vectors
$\mathbf{A}^\top$	Transpose of matrix $\mathbf{A}$
$\mathcal{P}(\cdot)$	Power set of a set
$\lvert \cdot \rvert$	Cardinal of a set
Y^X	Set of functions between sets X and Y
$\mathbb{1}_{\{\cdot\}}$	Indicator function of an event
$\mathbb{P}[\cdot]$	Probability of an event
$\mathbb{E}[\cdot]$	Expected value of a random variable
$\mathbb{V}[\cdot]$	Variance of a scalar random variable
$p(\cdot)$	Probability density or mass function
$\mathcal{N}(\mu, \sigma^2)$	Univariate normal distribution with mean μ and variance σ^2
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$

Notations of Part I

$\mathcal{U}$	Set of entities
L	Number of entity types
$\mathcal{W}$	Set of all event types
$\mathcal{H}$	Set of all possible events
N_e	Number of entities involved in a type e event
Ω_e	Tuple of entity types involved in a type e event
Ξ	Class of event sets
ϕ	Mapping from a class of event sets into a set of mathematical objects
θ	Parameters of a model (usually treated as a vector)
Θ	Parameter space of a model
ψ_θ	Anomaly scoring function for mathematical objects
f_θ	Anomaly scoring function for event sets
$\mathcal{F}(\mathcal{S})$	Entity graph induced by the event set $\mathcal{S}$

Notations of Part II

Λ	Class of entity subsets
$\mathcal{Y}$	Output space
T	Interaction function
J	Loss function
$\mathbf{Y}$	User-item rating matrix
$\mathbf{U}$	User latent factor matrix
$\mathbf{V}$	Item latent factor matrix
D	Number of latent attributes
$\mathcal{Y}$	Interaction tensor
$\mathcal{G}$	Graph or hypergraph

g	Embedding map
h	Link prediction function
$\omega = v_{1:\ell}$	Interaction between ℓ entities
κ	Inner compatibility function
κ_θ^ℓ	Compatibility function for the ℓ-th involved entity
$\kappa_\theta^{e,\ell}$	Compatibility function for the ℓ-th involved entity in a type e event
Q	Noise distribution
Q_ℓ	Noise distribution for the ℓ-th involved entity
Q_ℓ^e	Noise distribution for the ℓ-th involved entity in a type e event
K	Number of negative samples per training sample
C	Number of context entities
C_e	Number of context entities for type e events
$\mathbf{x}_v$	Embedding of entity v
$\mathbf{x}_v^T$	Embedding of entity v at time step T
$\boldsymbol{\beta}_e$	Latent factor of event type e
τ_ℓ^e	Type of the ℓ-th entity involved in a type e event
$\sigma_{e,\ell}$	Uncertainty associated with predicting the ℓ-th entity involved in a type e event
$p_{v_\ell \mid e, v_{1:\ell-1};\theta}$	Mid-p-value associated with the ℓ-th entity involved in a type e event, computed with parameter set θ

Notations of Part III

$\mathcal{G}$	Event graph
$\mathcal{V}$	Set of events (vertices of the event graph)
$\mathcal{E}$	Edge set of the event graph
n	Number of vertices of $\mathcal{G}$
$\mathbf{X}$	Signal observed over the event graph (anomaly scores of the events)
$\mathbf{A}$	Weighted adjacency matrix of $\mathcal{G}$
$\mathbf{M}$	Row-normalized weighted adjacency matrix
$\mathbf{L}$	Laplacian of $\mathcal{G}$
$\mathcal{H}$	Bipartite entity-event graph
$\mathcal{U}$	Entity set
$d(v)$	Degree of vertex v
$N(v)$	Neighborhood of vertex v
S	Pairwise event similarity function
τ	Time constant of the similarity function
K	Number of neighbors included in the nearest neighbor event graph
$\mathbf{H}(\lambda)$	Heat kernel with diffusion time λ
$\mathbf{X}^{(r)}$	Signal after r rounds of message passing
R	Number of message passing iterations
γ	Proportion of red team events in the neighborhood of a red team event
Λ	Set of potential clusters
δ	Proportion of vertices in the anomalous cluster
μ	Signal strength
F_0	Null probability distribution
F_1	Alternative probability distribution
$C(p)$	Size of the largest connected component of $\mathcal{G}$ at occupation probability p
$\mathcal{G}(p)$	Subgraph induced by the occupied vertices at threshold p in the canonical coupling
$\mathcal{C}(p)$	Largest connected component of $\mathcal{G}(p)$

$\mathcal{G}_k$	k-th element of the imbedded Markov chain of the process $\{\mathcal{G}(p)\}_{p\in[0,1]}$
$\mathcal{C}_k$	Largest connected component of $\mathcal{G}_k$
$\mathcal{G}(\tau)$	Subgraph induced by the vertices $v_k \in \mathcal{V}$ such that $X_k \geq \tau$
$\mathcal{Q}(\tau)$	Largest connected component of $\mathcal{G}(\tau)$
$\mathcal{H}_k$	Subgraph induced by the vertices corresponding to the k smallest values of the signal
$\mathcal{Q}_k$	Largest connected component of $\mathcal{H}_k$
K_c	Location of the phase transition; smallest index k such that the expected value of $\mathcal{C}_k$ is above $\sqrt{n}$
$T_{\mathcal{G}}(\mathbf{X})$	Test statistic for cluster detection in the signal $\mathbf{X}$ observed over $\mathcal{G}$

Chapter 1

Introduction

1.1 Computer Network Monitoring and the Big Data Paradigm

What do Microsoft Windows Security Auditing, NetFlow and syslog have in common? One possible answer could be the following: while none of them was originally designed for network-wide intrusion detection through statistical methods, they all ended up being used for this purpose.

The syslog protocol was developed in the 1980s to allow all sorts of devices in a given computer network to report events to a central logging server [Gerhards et al., 2009]. Its main purpose was to distinguish the program generating events from those reporting and storing them, making it easier for developers to make their programs send out relevant information about their execution. It was thus not specifically meant to enable intrusion detection, nor was it designed to make events suited for statistical modelling. However, modular and centralized logging is a key component of network-wide security monitoring. Similarly, the NetFlow protocol was introduced in the 1990s to enable traffic monitoring at the network layer[1] for purposes such as billing and troubleshooting [Claise et al., 2004]. Even though its creators were probably not pursuing this specific goal, large-scale statistical analysis of NetFlow data quickly became a prominent research direction in the network security community [Lakhina et al., 2004]. As for Windows Security Auditing, it first appeared in 1996 when the notion of event source was introduced in Windows NT 4.0, allowing for a distinction between security-related events (such as logons and changes to system files) and other purposes of logging, such as troubleshooting. Just like NetFlow data, Windows event logs were not originally designed for large-scale statistical modelling, but the wide range of fine-grained information they contain eventually made them an enthralling prey for an ever-growing data mining community.

In the words of Efron and Hastie, *"[s]omething important changed in the world of statistics in the new millenium"* [Efron and Hastie, 2016]. A typical data analysis problem used to be the clinical trial: given a specific question (to wit, does that drug work as it is supposed to?), the statistician would formalize it in mathematical terms, come up with a well-thought-out procedure to collect the exact amount of necessary data as predicted by the theory, and finally use these observations to answer the question at the desired confidence level. These were the days of scarce data and limited computing power – Efron and Hastie again:

> Before the computer age there was the calculator age, and before "big data" there were small data sets, often a few hundred numbers or fewer, laboriously collected by individual scientists working under restrictive experimental constraints.

[1] In the 7-layer Open Systems Interconnection model, the network layer is the third one. It handles addressing and routing between different local networks. The Internet Protocol (IP) is a well-known network layer protocol.

Needless to say, anomalous behavior detection through large-scale statistical modelling of NetFlow records does not exactly fit in this landscape. Beyond the obvious difference in data volumes, this approach can indeed be seen as the methodological opposite of the clinical trial: data collection came first, and only afterwards did statisticians start to wonder what they could do with this new information source. Hence a natural question: how did the realm of statistics get turned upside down? Efron and Hastie's answer could come down to one word: computers. Also, letting them elaborate a bit more:

> [C]omputer-based technology allows scientists to collect enormous data sets, orders of magnitude larger than those that classic statistical theory was designed to deal with; huge data demands new methodology, and the demand is being met by a burst of innovative computer-based statistical algorithms.

How computers enable statistical methodologies that could barely be fantasized in the calculator age is rather straightforward. The mechanisms through which computers provide statisticians with virtually infinite amounts of data, however, deserve further attention. Indeed, rather than "computers", the key word here might well be "digitalization".

Digitalization[2] is here defined as the process of redesigning all sorts of activities through the use of computers (or similar devices, such as smartphones). For instance, using streaming services to listen to music instead of acquiring physical records is an occurrence of digitalization, and buying CDs online rather than in a physical store is another one. What matters here is this: as more and more activities and aspects of everyday life start being mediated by computers, more and more data becomes available about pretty much everything. Indeed, due to computers' innate ability to record everything happening to them, gathering data about any kind of activity becomes trivial once it has been digitalized: no more costly experimental setups, no more thorough pondering about what should be included in the data before collecting them, all we may want to know is readily accessible. Of course, starting to produce all sorts of records before figuring out how they could be used raises new issues [Jagadish et al., 2014]: not just volume, but also heterogeneity, impractical data formats, errors and missing values.

Computer event logs, such as Windows Security Auditing or NetFlow records mentioned above, are yet a different brand of digitalization byproducts: even better than computer-mediated activities, they relate the lives of computers themselves. Just like any computer-age data source, they contain a fair amount of valuable information concealed by their jaw-dropping volume and baroque format. It is thus somewhat tempting to feed them to some of the latest statistical algorithms and see what comes out – but to what end exactly?

It turns out that the answer to that question also comes from global and sometimes not-so-well-thought-out digitalization. Indeed, as more and more aspects of our lives migrated to the Internet, they incidentally became vulnerable to the ever-growing threat of cyberattacks. The worldwide spread of computer worms such as ILOVEYOU [Knight, 2000] or Slammer [Moore et al., 2003] hinted towards the disruptive potential of nefarious use of computers, but it was only the beginning. Soon enough, various threat actors started targeting equally diverse victims for all sorts of reasons. Advanced intruders with suspected ties to nation states were caught stealing intellectual property and other sensitive data from firms and governments [Mandiant,

[2] Not to be confused with digitization, which is the process of converting information into a digital format – for instance, scanning a handwritten document.

2013, FireEye, 2015], while others used their sophisticated skills for sabotage [Kerr et al., 2010, Park and Walstrom, 2017] or financial gain [Trautman and Ormerod, 2018]. Criminals quickly found opportunities in cyberspace too, with data theft and extortion operations ending up in gigantic leaks of private information [NCSC, 2017], considerable financial losses [IBM, 2020] and, incidentally, deaths [Wetsman, 2020].

In the face of this swarming threat landscape, defenders have to step up their game. Hardening their networks and systems so as to make them more difficult to breach is a necessary, but not sufficient protection: with enough skills and determination, an intruder can always find a way in. They must therefore be able to detect intrusions as quickly as possible, ideally as soon as they happen. This is where event logs come in handy: since they record everything going on within a protected network, they should in particular contain traces of any ongoing malicious activity. Unfortunately, they also contain overwhelming amounts of irrelevant information – so much of it that more often than not, nobody actually bothers looking at them. As a consequence, intruders can manage to stay undetected for a long time despite event logs providing blatant evidence of their presence: 207 days on average according to IBM's 2020 Cost of a Data Breach Report [IBM, 2020].

This situation creates a need for more sophisticated event log analysis tools, and statistical methods developed in the last few decades as a result of the growing availability of large and diverse datasets appear as obvious candidates. In a way, the cycle is complete: after years of sparkling statistical innovation driven by ubiquitous, digitalization-powered data collection, all these brand new tools are now needed to alleviate some of the less desirable side effects of that very same digitalization. This thesis is our modest contribution to this ongoing effort. Drawing inspiration from various research fields such as recommendation systems, bioinformatics, computer vision or dynamical systems, it aims to build effective tools enabling malicious behavior detection in event logs. This is no easy task, and it actually entails several subproblems which are discussed in the following sections. First of all, feeding event logs to a statistical model is not as straightforward as it sounds: these data are in fact peculiarly complex, as explained in Section 1.2. In addition, defining malicious behavior from a data-centric point of view is also nontrivial, let alone detecting it. Statistical intrusion detection methods typically circumvent this issue by resorting to anomaly detection, as discussed in Section 1.3. Our contributions, summarized in Section 1.4, cover both of these aspects.

1.2 Representing and Modelling Complex and Multi-Faceted Data

Two main challenges appear when trying to build a statistical model for event logs. First, the complexity of the data makes it hard to come up with an appropriate mathematical representation. Indeed, none of the usual mathematical objects studied in statistics and data mining fully captures the multiple facets of event logs, which we briefly present in Section 1.2.1. Secondly, once a suitable representation has been designed, building an appropriate statistical model also takes some craft: as discussed in Section 1.2.2, factoring in all the characteristics of the data requires an ad hoc model, which also raises the need for adequate inference procedures.

1.2.1 Building Abstract Representations for Complex Data

The notion of event log considered in this thesis encompasses various real-world data sources, including the Windows Security Auditing and NetFlow records mentioned above. While differences exist between these practical instances, we focus on what they have in common so as to make our contributions broadly applicable. In particular, the main specificity of all the data sources we consider is that they do not naturally fit in any usual mathematical framework.

Event logs are multi-faceted. Event logs are primarily combinatorial data, in that each event is defined by a finite number of fields mostly containing discrete values: user and host names, IP addresses, and so on. However, they are also intrinsically heterogeneous: the contents of the fields, as well as events themselves, can be of different types. In other words, a user is not equivalent to an IP address, and a process creation event does not have the same meaning as a NetFlow record of a communication between two hosts. Finally, each event is associated with a specific timestamp, and the probability of observing a given event is not necessarily constant in time: it may undergo seasonal variations as well as long-term drift. Besides, statistical dependencies can be expected to exist between successive events: for instance, the probability of a process creation event involving user U and host H should intuitively be higher if U recently logged on to H. These two properties endow event logs with a temporal dimension.

Although none of these three characteristics is new to the seasoned statistician, observing all three of them simultaneously is less frequent. As a consequence, none of the usual mathematical objects perfectly represents these multiple facets: matrices and tensors can for instance appropriately encode combinatorial data, but considering events of multiple types containing a variable number of fields makes them inadequate. Similarly, point processes and time series may well capture the temporal aspect of event sequences, but cramming a combinatorial dimension into either of these frameworks is nontrivial.

Two ways to bridge the gap. Since event logs are not adequately represented by usual mathematical objects, existing statistical modelling tools cannot be used directly: the significant semantic gap between the data and the model must be bridged. Most existing work circumvents this problem by adapting the data to the model, simplifying them enough to make them fit into a standard theoretical framework [Yen et al., 2013, Legg et al., 2015, Hu et al., 2017]. This leads to significant information loss, in turn bounding the performance of the whole detection procedure: once relevant aspects of the data have been erased, even the smartest algorithm will be unable to recover them. Therefore, recent contributions have developed a more comprehensive approach, coming up with better suited models instead of oversimplifying the data [Tuor et al., 2018, Amin et al., 2019, Leichtnam et al., 2020b]. In particular, significant progress has been made towards adequately modelling the combinatorial dimension of events. Building upon these recent advances, we propose an abstract representation of event logs revolving around the notion of interaction.

An interaction-centric approach. The fundamental intuition underpinning this thesis is that events should be seen as interactions between entities. In addition to emphasizing their combinatorial aspect, this description also suggests that events result from intertwined individual behaviors. In other words, it relies on the entity-event duality: past events help us infer the role and usual behavior of each entity (user,

host, process, etc.), and future events can then be predicted using this knowledge. What mainly motivates this approach is that malicious events are often rare interactions. For instance, an intruder using stolen credentials to explore a breached network should generate several unusual interactions between the corresponding user account, a source host and various destination hosts. Focusing on interactions thus seems relevant with respect to the task at hand, namely intrusion detection. However, it raises some issues in terms of statistical modelling.

1.2.2 Statistical Modelling for Nonnumerical Data

Modelling interactions between entities at a large scale is a relatively recent topic in statistics – one that typically gained traction with digitalization. Beyond the questions raised by the combinatorial nature of such data, we also need to include the two additional characteristics we identified in the previous section, namely the heterogeneous and temporal aspects of event logs. This makes designing a suitable model particularly tricky.

Modelling combinatorial data. Broadly speaking, a combinatorial random variable is a discrete random variable defined over a combinatorial set, such as the power set of a finite set [Bekkerman et al., 2006]. The main challenge arising when modelling such variables stems from the high dimensionality of the sample space. Taking an event log-related example, let $\mathcal{U}$ be a set of users, $\mathcal{S}$ be a set of source hosts and $\mathcal{D}$ be a set of destination hosts. An authentication event can then be seen as a triple $T = (U, S, D) \in \mathcal{U} \times \mathcal{S} \times \mathcal{D}$ indicating that user U logged on to D from S. Suppose we are trying to estimate the distribution of T from a set of n past events $\{(u_i, s_i, d_i)\}_{i=1}^n$. In a real-world computer network, each of the sets $\mathcal{U}$, $\mathcal{S}$ and $\mathcal{D}$ can be expected to contain thousands (if not tens of thousands) of elements, meaning that the number of possible values of T ranges between 10^9 and 10^{12}. Without any assumption on the underlying probability distribution, the number of samples n must be huge for any kind of inference to be reasonably reliable. Even worse, the size of the sample space grows exponentially with the number of entities involved in the considered events, which has no reason to be as small as three.

In order to build a reliable model for such data, some sort of dimensionality reduction must be performed. To that end, we resort to latent space modelling [Turnbull, 2020]. The key intuition behind this approach is that the propensity of a given entity to interact with others depends on a small number d of latent attributes of that entity. Coming back to the authentication example, let $g : \mathcal{U} \cup \mathcal{S} \cup \mathcal{D} \to \mathbb{R}^d$ be the unknown function mapping each entity onto its attribute vector. A sensible model for the random triple T can be defined as

$$\forall (u, s, d) \in \mathcal{U} \times \mathcal{S} \times \mathcal{D},\ \mathbb{P}\big[T = (u, s, d)\big] = f\big(g(u), g(s), g(d)\big), \tag{1.1}$$

with $f : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$ some affinity function. Inferring the full distribution of T then mostly consists in estimating the d latent attributes of each entity, leading to $d(|\mathcal{U}| + |\mathcal{S}| + |\mathcal{D}|)$ parameters to learn – a steep decrease from the $|\mathcal{U}| \cdot |\mathcal{S}| \cdot |\mathcal{D}|$ parameters associated with naive estimation of the probability mass function.

Latent space models have been proposed for various kinds of interaction data, including some related to event logs and intrusion detection [Turcotte et al., 2016a, Amin et al., 2019, Lee et al., 2021, Sanna Passino et al., 2020]. However, they mostly focus on dyadic interactions, which is problematic when dealing with event logs: as evidenced by the authentication example above, events cannot reasonably be assumed

to involve only two entities. Therefore, part of our work (presented in Chapter 4) consists in designing a latent space model for polyadic interactions.

Dealing with heterogeneity. Another specificity of event logs in comparison with other kinds of interactions is their heterogeneity, whose main practical manifestation is the existence of several event types. This specificity should not be ignored when designing a statistical model: as a typical example, just because user U is likely to log on to server S does not mean that U is expected to modify system files on S. Therefore, our latent space model must factor in the differences between event types. Reusing the notations from Equation 1.1, it seems reasonable to do so by making the affinity function f specific to one event type while reusing the same latent attribute map g across all types. This intuition comes from the field of multi-task learning [Caruana, 1997]: simultaneously learning to perform several related prediction tasks has been observed to yield better performance for each single task than learning to perform it independently. A common interpretation of this phenomenon is that related tasks act as an inductive bias. In other words, by looking for a model which performs well at all tasks, multitask learning algorithms restrain the class of possible models for each task, leading to more efficient learning. Coming back to event logs and latent space models, looking for latent attributes that explain the behavior of an entity across all event types can be expected to yield a more realistic model.

The main challenge in multi-task learning is to find the right balance between different tasks. Indeed, jointly minimizing the error made by the model across several tasks can be formulated as multi-objective optimization problem, which is significantly more complex than the usual setting of single-task learning. For instance, a given update to user U's latent attributes may lead to better predictions for U's authentications, but worse ones for process creations involving U. What should we do in such a situation? In order to answer that question and design a suitable learning procedure for our model, we draw inspiration from previous work on multi-task learning [Kendall et al., 2018].

Adapting to nonstationary data streams. Finally, the temporal dimension of event logs should be considered when building a statistical model. In particular, patterns of interaction inside a monitored network can hardly be assumed to remain constant over time: users take on new organizational roles or start working on new projects, servers start hosting new applications, and so on. In addition, new entities frequently appear – user accounts for new employees or new computers, typically. Therefore, a static model could quickly become obsolete in a real-world security monitoring setting. Fortunately, latent space modelling naturally extends to nonstationary interaction streams.

Indeed, since the probability of a given interaction only depends on the latent attributes of the involved entities, keeping the model up to date simply requires tracking the evolution of these latent attributes. Reusing once again the notations of Equation 1.1, this amounts to replacing the map g with a sequence $\{g_t\}_{t\geq 0}$, where each $t \geq 0$ represents a time step. This makes sense from an event log-oriented perspective: changes in the observed interaction patterns result from entity-related modifications, such as the apparition of a new host in the network or a user changing their habits.

In terms of inference, a parallel can be made between using observed events to track the evolution of the latent map g and the more generic framework of hidden Markov models. Indeed, events happening between time steps $t-1$ and t can be seen as observations drawn from an emission distribution Q_t, which depends on the current latent map g_t. Similarly, the sequence $\{g_t\}_{t\geq 0}$ can be seen as a latent Markov process

characterized by some transition kernel P. Inferring the current latent map g_t given events observed up to time step t then boils down to a filtering problem. We leverage this parallel in Chapter 5 to design an updating procedure for our latent space model.

1.3 Reliably Detecting Relevant Anomalies

Building a statistical model for event logs is a first step towards making something useful out of them. However, it does not directly enable intrusion detection: some additional work is required to dig up malicious behavior using our model. The key idea is that intrusion-related events should be anomalous and thus be given a low predicted probability by the model, as explained in Section 1.3.1. However, unusual events happen constantly in real-world computer networks. As a result, simply looking for unlikely events yields more false positives than true detections, and Section 1.3.2 presents the improvements we propose to alleviate this issue.

1.3.1 Malicious Events Are Anomalous...

What makes intrusion detection remarkably challenging is that malicious behavior cannot be generically and reliably characterized: the only common trait of all its instances is that the legitimate administrator of the targeted network does not want them to happen, and this is never explicitly written in the logs. This essentially leaves defenders with two options: gathering explicit descriptions of as many specific malicious actions as possible, or trying to come up with an indirect but more general characterization. This second approach is the one taken in this thesis.

An indirect characterization. The set of events that could theoretically be triggered by malicious behavior is both extremely large and complex. Consider for instance an intruder trying to breach the targeted network by collecting credentials through a phishing campaign. After completing this first step, the intruder would typically log on to a Virtual Private Network (VPN) endpoint using the stolen credentials, then use these same credentials to move laterally into the internal network and collect information. In terms of event logs, such behavior can be expected to generate all kinds of remote authentication events: the stolen credentials can belong to any user, and the source and destination hosts can also be any of the legitimate nodes of the internal network. Besides, nothing specific sets malicious events apart from benign ones: taken individually, each field of each intrusion-related event looks perfectly legitimate. As mentioned above (and further discussed in Chapter 2), the only characteristic we can reasonably expect malicious events to exhibit is that they involve entities which do not usually interact. Therefore, instead of trying to describe the set of possibly malicious remote authentication events, a more practical approach consists in collecting the interaction patterns corresponding to usual legitimate behavior. Future events can then be considered suspicious if they do not match these patterns. This approach is commonly referred to as anomaly detection.

Detecting combinatorial anomalies. While anomaly detection is a widely studied topic [Chandola et al., 2009], its application to combinatorial data raises specific challenges. Indeed, an anomaly can intuitively be defined as a low-probability sample, and detecting anomalies then boils down to building a statistical model of past normal data and using it to compute predicted probabilities of future samples. However, the joint probability distribution over a combinatorial sample space actually

provides limited information. For instance, suppose we observe a remote authentication $(u, s, d) \in \mathcal{U} \times \mathcal{S} \times \mathcal{D}$ with low predicted probability. There might actually be several explanations for this low probability: u may be expected to log on to d, but not from s. Another explanation could be that u is supposed to interact with neither s nor d. Actually, any non-empty subset of entities involved in a given interaction could provide sufficient evidence to consider the whole interaction suspicious from a security-oriented perspective. Combinatorial anomaly detection algorithms should thus rely on a finer-grained description of the probability distribution. This issue has been addressed in the data mining and machine learning literature [Das and Schneider, 2007, Akoglu et al., 2012, Amin et al., 2019], and we build upon these past contributions to design our event-wise anomaly detection algorithm in Chapter 4.

1.3.2 ...but Not All Anomalous Events Are Malicious

Assuming that intrusion-related events are anomalous, being able to detect anomalous interactions between entities is a necessary condition for intrusion detection. It is, however, not sufficient: many legitimate events are also anomalous from a statistical point of view, and distinguishing them from actual malicious events requires further work – as well as further assumptions.

Anomalies everywhere. Each time a regular employee misclicks and opens the wrong network share in their graphical user interface, an anomalous event is generated. However, this is arguably not something the network administrator wants to be notified about. More generally, many legitimate activities do not occur on a regular basis – consider for instance software deployment or other administrative tasks. Due to their irregular nature, these activities are likely to be considered anomalous by statistical models. It is important to understand that this is not just an imperfection of the model, which could be alleviated with more training data or better algorithms: there is simply a significant part of human behavior that is unpredictable. Therefore, detecting anomalous events is not enough: further processing steps are required to avoid unacceptable false positive rates. In this thesis, these additional steps rely on a fundamental assumption: intrusion-related events are not only anomalous, but also connected to each other regarding the entities they involve.

Leveraging connections between events. Coming back to the phishing example mentioned above, remote authentication events resulting from the exploration of the network by the intruder should involve some shared entities. Indeed, each new lateral movement should start from an already compromised host, and stolen user credentials may also be used more than once. More generally, each action taken by an intruder can be expected to involve some previously compromised entities, creating intersections between the underlying events.

These relationships can typically be abstracted into a graph whose vertices represent events. An edge between two events then encodes some notion of similarity, which should be appropriately defined so that intrusion-related events are mapped onto a densely connected subgraph [Pei et al., 2016, Leichtnam et al., 2020a]. Having built such an event graph, anomaly scores associated with events can be seen as a noisy graph-structured signal. Rare but legitimate events should then correspond to small peaks in this signal, while clusters of malicious events should appear as consistently high plateaus. We derive two postprocessing methods from this intuition. First of all, as discussed in Chapter 6, graph signal processing tools can be used to denoise anomaly scores, smoothing out false positives while preserving the high scores

of malicious events. Secondly, connected clusters of high-scoring events can be detected before starting any manual investigation, helping analysts focus on the most suspicious sections of the logs. Methods aiming to detect such clusters are presented in Chapter 7.

1.4 Contributions of this Thesis

To sum up, this thesis addresses three challenges associated with statistical analysis of event logs and intrusion detection: representing and modelling the data, effectively detecting anomalous events and postprocessing anomaly scores to classify anomalies as benign or malicious.

Main contributions. Our first contribution is a rigorous definition of intrusion detection in event logs as well as a thorough classification of existing methodologies. This formalization of the problem helps us identify the most important characteristics of the data, and we build upon these insights to design an event-wise anomaly detection algorithm relying on latent space modelling and multi-task learning. We compare this algorithm with state-of-the-art baselines and obtain superior detection performance on a real event log dataset. Our implementation of the proposed model is openly available. Finally, we propose an event graph-based approach to anomaly score postprocessing. Our methodology entails two aspects: we first experiment with two graph signal processing tools in order to denoise event-wise anomaly scores using the structure of the event graph. We then study the problem of cluster detection in a graph, proposing two novel detection procedures and comparing them with existing methods on a synthetic benchmark dataset. These detection procedures are also evaluated on real-world event graphs.

Evaluating detection methodologies. Assessing the relevance and usefulness of our contributions is not a simple task. Indeed, the models and algorithms we propose are tailored to a practical use case, making them poorly suited to theoretical analysis. Besides, empirical evaluation is also complex: due to the vast diversity of real-world computer networks and adversarial behaviors, results obtained on one single dataset should be taken with a grain of salt. However, for lack of a better evaluation metric, we assess the applicability and effectiveness of our algorithms on a real event log dataset released by the Los Alamos National Laboratory, namely the "Comprehensive, Multi-Source Cyber-Security Events" dataset [Kent, 2015a, Kent, 2015b]. What makes this dataset especially interesting is that it contains labelled traces of a red team exercise (i.e. an intrusion carried out by security experts in order to assess the network's security). Therefore, the ability of our algorithms to spot malicious activity can be estimated using these labels. We describe the dataset in further detail in Chapter 2.

Thesis outline. The rest of this thesis is divided in three parts, each one covering one of our three main contributions: formalization and classification, event-wise anomaly detection, and graph-based anomaly score postprocessing. Each one of these topics can be seen as one step of a global event log processing pipeline, as illustrated in Figure 1.1.

Part I first sets the stage by formally stating the problem we consider and discussing previous contributions. We define some basic notions in Chapter 2, such as events, event logs and intrusions. Using these definitions, we propose a generic formulation of intrusion detection in event logs and highlight some of the challenging aspects

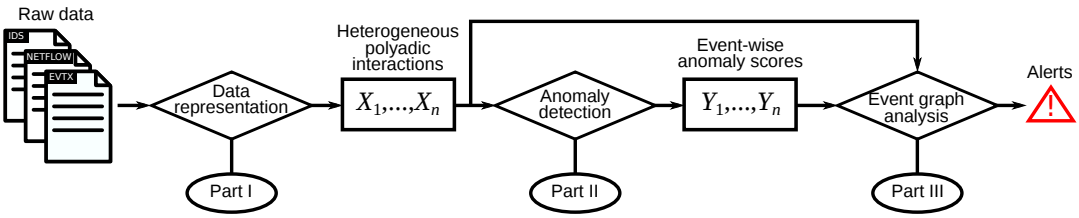


FIGURE 1.1: Event log processing pipeline and corresponding parts of the thesis.

of this problem. We then build upon our formalism to discuss and compare detection algorithms found in the literature, introducing a taxonomy of existing approaches in Chapter 3. This taxonomy emphasizes the representation choices and modelling assumptions of each reviewed work, connecting them to the different facets of the data. Some of the work presented in this part has been published at the *Conference on Artificial Intelligence for Defense* (CAID 2020) [Larroche et al., 2020b].

Part II then presents our event-wise anomaly detection algorithm, as well as the underlying statistical model and inference procedure. The aim of our algorithm is to deal with the three main aspects of event logs, namely their combinatorial, heterogeneous and temporal facets. The first two aspects are addressed in Chapter 4, which describes our latent space model for heterogeneous polyadic interactions. We also discuss the corresponding learning algorithm, emphasizing the additional challenges raised by the existence of several event types. Chapter 5 then factors in the temporal aspect, leveraging the connections between latent space modelling and Bayesian filtering to come up with a parameter updating procedure for our model. Parts of these chapters have been published as a non-peer reviewed preprint [Larroche et al., 2021b].

Finally, Part III deals with event-wise anomaly score postprocessing using event graphs. Chapter 6 describes our event graph construction procedure and presents graph signal processing tools that can be used to denoise the scores. We then focus on cluster detection in Chapter 7, generically defining this task as a combinatorial hypothesis testing problem. To address the limited scalability of existing detection methods, we propose two computationally efficient tests relying on percolation theory. These tests are evaluated against previously published methods on a synthetic benchmark dataset, and we provide a first assessment of their effectiveness on real-world event graphs. Some of the contents of Chapter 7 have been published at the *Symposium on Intelligent Data Analysis* (IDA 2020) [Larroche et al., 2020a] and at the *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* (ESANN 2021) [Larroche et al., 2021a].

Part I

Definitions and State of the Art

Chapter 2

Events, Logs and Intrusions

Before proceeding with any technical development, a thorough and accurate statement of the considered problem is in order. Some key concepts also need to be properly defined, starting with the data we are investigating: what exactly is an event, or an event log? This chapter deals with all these preliminary questions, translating the initial problem into a more formal one and connecting it with some useful mathematical concepts.

2.1 Introduction

It is fairly straightforward to provide an intuitive definition of intrusion detection in event logs: given activity records of a given network in a given time frame, we want to know whether malicious activity has occurred. However, designing statistical methods addressing this problem requires a more formal and thorough definition, which happens to be less obvious.

Starting with event logs themselves, two main challenges stand in the way of their formal specification. First, the generic notion of event logs encompasses many real-world data sources and formats, and unifying all of them into a unique formalism is nontrivial. Secondly, most of these data sources were not primarily designed for statistical modelling, but rather for manual inspection or basic analysis through simple descriptive statistics. Therefore, they do not fit well into usual data representation frameworks, such as Euclidean spaces or time series.

As for the problem of intrusion detection, although there seems to be nothing complex about the concept of an intrusion, properly defining it in terms of traces left in the data is not straightforward. This difficulty partly stems from the vast diversity of offensive tactics, techniques and procedures (TTPs) and end goals of intruders, which make it challenging to define generic characteristics that malicious activity should exhibit. A more profound cause is that an intrusion, or more generally malicious behavior, is mostly defined in terms of intention: from a data-centric perspective, the difference between a legitimate administrator installing software on a computer and an intruder using the administrator's credentials to install malware on the same computer is not obvious. Therefore, it does take some careful thinking to formally characterize what we are trying to detect.

As a consequence of these challenges, some choices and assumptions need to be made: the mathematical framework our models and algorithms rely upon cannot perfectly depict such complexity and diversity. In this thesis, however, we aim to keep our focus as broad as possible, which leads us to privilege rather generic definitions and weak assumptions. These are introduced and justified in the following sections.

We first present our definition of event logs, as well as the challenges it raises in terms of statistical modelling, in Section 2.2. We then define the intrusion detection problem and underline its connections with the field of anomaly detection in Section 2.3. Finally, Section 2.4 provides a concrete example of event log data through an extensive description of the LANL dataset, which is consistently used for our experiments and evaluations throughout this thesis.

2.2 A Generic Definition of Event Logs

Being the main focus of this thesis, event logs are quite logically the first concept we aim to define in this chapter. We start with the most practical aspects by discussing where event logs come from and what they look like in Section 2.2.1, then move on to more formal considerations and provide some basic definitions in Section 2.2.2. Finally, Section 2.2.3 states the definition of event logs underpinning our work and exposes the main resulting challenges.

2.2.1 Multiplicity and Diversity of Existing Data Sources

Event logs can be obtained from various sources, each of which can be characterized by two main elements: the type of activity being recorded, and the level of abstraction at which this activity is described. As for the type of activity, a common distinction can be made between system logs and network logs: while the former record internal actions on a given device, the latter focus on communications between devices. Therefore, these two categories give different perspectives on the underlying activity: system logs can describe anything happening on a given device with an arbitrary level of detail, from authentications and process creations to file-related activity and operating system (OS) administration. Network logs, on the other hand, provide more of an external perspective on these activities through communications which can either be internal to the network or happen between an internal host and an external one. Note that while some information about what happens on a given host can be inferred from its network activity, entirely local actions are only captured by system logs.

Data sources from either category also differ in their level of abstraction. This second distinction relates to the program creating the logs, which can have a more or less high-level point of view on the actions being logged. In the case of system logs, the two extremes are kernel-level auditing and application logs. OS event logs, such as Linux system logs or Windows Event Logs, sit in between in terms of abstraction. Similarly, network logs can be produced at different abstraction layers of the Open Systems Interconnection (OSI) model: while traffic metadata providers such as Cisco's NetFlow [Claise et al., 2004] focus on the network layer, intrusion detection systems (IDS) such as Zeek (formerly known as Bro [Paxson, 1999]) can log the contents of the communications up to the application layer.

These distinctions lead to significant variations in the content and structure of event logs coming from different sources, as evidenced by Figure 2.1. Note that they are highly relevant from a security perspective: different data sources enable observation of different kinds of activity, at different granularities. Therefore, some sources are better suited to detecting a given kind of attack than others: for instance, while network traffic metadata can be sufficient when looking for a data exfiltration, ransomware deployment through encrypted communications may require fine-grained logging on endpoints to be detected. In addition, all data sources are not equally reliable when dealing with a malicious adversary. In particular, an intruder can fairly easily tamper with the local logs on a given host after fully compromising it. Network

```
An account was successfully logged on.

Subject:
        Security ID:            SYSTEM
        Account Name:           WIN-GG82ULGC9GO$
        Account Domain:         WORKGROUP
        Logon ID:               0x3E7

Logon Information:
        Logon Type:             2
        Restricted Admin Mode:  -
        Virtual Account:        No
        Elevated Token:         Yes

Impersonation Level:            Impersonation

New Logon:
        Security ID:            CONTOSO\Administrator
        Account Name:           Administrator
        Account Domain:         WIN-GG82ULGC9GO
        Logon ID:               0x8DCDC
        Linked Logon ID:        0x0
        Network Account Name:   -
        Network Account Domain: -
        Logon GUID:             {00000000-0000-0000-0000-000000000000}

Process Information:
        Process ID:             0x44c
        Process Name:           C:\Windows\System32\svchost.exe

Network Information:
        Workstation Name:       WIN-GG82ULGC9GO
        Source Network Address: 127.0.0.1
        Source Port:            0

Detailed Authentication Information:
        Logon Process:          User32
        Authentication Package: Negotiate
        Transited Services:     -
        Package Name (NTLM only):       -
        Key Length:             0
```

(A) Windows Event Logs (source: Microsoft).

```
{
  "ts": "2021-02-18T09:34:22.546324Z",
  "uid": "Conn_Identifier",
  "id.orig_h": "192.168.0.1",
  "id.orig_p": 49230,
  "id.resp_h": "192.168.0.0",
  "id.resp_p": 53,
  "proto": "udp",
  "trans_id": 15381,
  "query": "www.example.com",
  "qclass": 1,
  "qclass_name": "C_INTERNET",
  "qtype": 1,
  "qtype_name": "A",
  "AA": false,
  "TC": false,
  "RD": true,
  "RA": true,
  "Z": 0,
  "rejected": false
}
```

(B) Zeek DNS logs.

```
Date                    Duration  Proto  Src IP Addr:Port   Dst IP Addr:Port   Packets  Bytes  Flows
2021-02-18 09:34:22.546  0.000     UDP    192.168.0.1:49230  192.168.0.0:53     1        83     1
2021-02-18 09:34:22.553  0.000     UDP    192.168.0.0:53     192.168.0.1:49230  1        147    1
```

(C) NetFlow records.

FIGURE 2.1: Examples of events coming from different sources: a Windows authentication event (2.1a), a DNS request recorded by Zeek (2.1b) and NetFlow records of a DNS exchange (2.1a).

logs, on the other hand, can record communications coming in and out of other devices than the one generating them, which makes them somewhat more robust to the intruder's actions. Security practitioners are thus inclined to emphasize the distinction between event logs coming from different sources.

From a data science perspective, however, it is more interesting to find out what all these data sources have in common. Indeed, we aim to make our methods applicable in as many practical settings as possible, independently of the logging mechanisms which may be set up. In addition, it is often interesting to gather knowledge from several complementary data sources, which requires merging them in a global formalism. Therefore, the first step of our work should be to build an abstract framework encompassing as many concrete data sources as possible. To the best of our knowledge, existing work on event log analysis for intrusion detection relies on ad-hoc definitions of the data, designing mathematical representations that are suited to the data sources and statistical tools under consideration. In contrast, we seek to build a more generic

definition, from which we can then derive case-specific instances. This unifying approach allows us to motivate our statistical modelling choices, as well as to compare existing approaches in a common framework.

2.2.2 Generically Defining Events as Polyadic Interactions

As mentioned above, designing a generic definition of an event requires finding something common to many different kinds of event logs. The underlying intuition of our work is that this common characteristic lies in the relational nature of the data: in other words, events are primarily interactions between entities. These entities are typically user accounts, computers or files, and we consider events that involve several of them: authentication of a given user on a given computer, network communication between two hosts, and so on. Many data sources can be described this way, which suggests that this relational aspect is indeed essential. As a side note, this is consistent with the file formats used for event logs, which often have a relational or hierarchical nature: Comma-Separated Values (CSV), JavaScript Object Notation (JSON) or Extensible Markup Language (XML), for instance, fall into these categories.

More formally, we define a computer network as a heterogeneous set of entities. Letting L denote the number of existing entity types, we write $\mathcal{U}_\ell$ for the set of type ℓ entities (for $\ell \in \{1, \dots, L\} := [L]$), and $\mathcal{U} = \bigcup_{\ell=1}^{L} \mathcal{U}_\ell$ for the whole entity set. Note that the entity set might evolve over time (e.g. when new user accounts are created), but this aspect is left aside for now. Before stating the definition of an event, we first define the underlying notion of event type.

Definition 2.2.1 (Event type)**.** An event type is a triple $(e, N_e, \Omega_e) \in \mathbb{N} \times \mathbb{N}^* \times [L]^{N_e}$, where e is a unique index, N_e is the number of entities involved in a type e event and Ω_e is the tuple formed by the types of the entities involved in a type e event.

Two remarks can be made about this definition. First, it implicitly assumes that all events of a given type involve the same number of entities, and that these entities always have the same type. Note that this implies no loss of generality: since an arbitrary number of event types can be defined, a type that could involve a variable number of entities can simply be divided into as many subtypes as necessary. The same goes for the involved entities' types. Secondly, putting the involved entity types in a specific order is a way to distinguish entities of the same type that play different roles in an event. An event representing a network communication typically illustrates the need for such a distinction: while the source and destination of the connection are both hosts, they are not interchangeable with respect to the meaning of the event.

From now on, let $\mathcal{W}$ denote the set of event types. Each event type $(e, N_e, \Omega_e) \in \mathcal{W}$ is simply referred to through its index $e \in \mathbb{N}$ unless otherwise specified. Harnessing these concepts and notations, we now proceed with our definition of an event.

Definition 2.2.2 (Event)**.** An event is a tuple (t, e, ω, Γ), where $t \in \mathbb{R}_+$ is a timestamp, $e \in \mathbb{N}$ is an event type, $\omega \in \prod_{\ell \in \Omega_e} \mathcal{U}_\ell$ is a tuple of involved entities, and Γ represents some optional additional information.

This generic definition encompasses a wide variety of data sources, as we now demonstrate through practical examples, illustrated in Figure 2.2.

Example 2.2.1 (Process creation)**.** Assume that user `usr@dom` executes the program `cmd.exe` from the graphical user interface on a Windows computer called `wstn`. Such an action is recorded by Windows in the Security Auditing event logs, and it can also be traced by other logging systems, such as Sysmon. These various data sources may

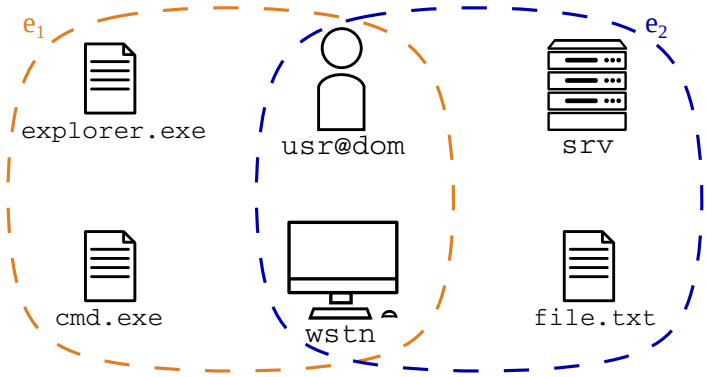


FIGURE 2.2: Two events represented as polyadic interactions: a process creation (e_1, in orange) and a remote access to a file (e_2, in blue).

define different fields to characterize the action, but the latter can be represented in a generic way as a process creation event involving user `usr@dom`, computer `wstn`, parent process `explorer.exe` and child process `cmd.exe` (as illustrated by event e_1 in Figure 2.2), with additional information such as the full paths of the executables. Note, however, that this representation is not unique: it depends not only on the amount of information available through the data sources at hand, but also on deliberate design choices. In particular, defining which fields of a given log line should be treated as entities and which others merely contain additional information is nontrivial: such choices should be guided by domain-specific knowledge on intrusion detection.

Example 2.2.2 (File access on a remote share)**.** Let us now assume that user `usr@dom`, logged in on computer `wstn`, remotely accesses a file called `file.txt` and located at the root of the file share `share` hosted on server `srv`. This can be seen as an "open file" event involving entities `usr@dom`, `wstn`, `srv` and `\\srv\share\file.txt` (as illustrated by event e_2 in Figure 2.2). Note, however, that different data sources can provide different points of view: considering for instance NetFlow data, this action would result in a TCP connection from `wstn` to `srv` on port 445. Windows authentication logs, on the other hand, would depict it as a network authentication event involving `usr@dom`, `wstn` and `srv`, with additional information such as the authentication package used or the process which requested the authentication. While these examples demonstrate the flexibility of our proposed framework, they also emphasize its abstract nature: in practice, real-world detection methodologies rely on simpler, case-specific data representations.

From now on, we write $\mathcal{H}$ for the set of all possible events. Chapter 3 surveys existing representation and modelling frameworks for event logs, emphasizing their connections to our generic (and hopefully unifying) definition. For now, we use this definition as a starting point to discuss the main challenges associated with statistical modelling of event log data.

2.2.3 The Complex Nature of Event Logs and Resulting Challenges

Having formally defined what events are, the immediate next step consists in defining the notion of event log. This definition is rather straightforward: an event log is essentially a sequence of events.

Definition 2.2.3 (Event log)**.** An event log is a sequence $\left\{\left(t^i, e^i, \omega^i, \Gamma^i\right)\right\}_{i=1}^{n}$, where $n \in \mathbb{N}^*$ is the number of logged events and, for each $i \in [n]$, $\left(t^i, e^i, \omega^i, \Gamma^i\right)$ is one event, ordered such that $t^1 \leq \ldots \leq t^n$.

While there is nothing complex about the notion of event log as defined above, coming up with a generative model for such an object is clearly an arduous task. One of the main reasons for this is that event logs are multi-faceted data, mainly exhibiting three important aspects:

Combinatorial. As stated above, events are here primarily defined as interactions between entities, which endows them with a combinatorial nature. In other words, the space of all possible events is essentially a Cartesian product of finite sets. The main challenge resulting from this characteristic is commonly referred to as the curse of dimensionality: since the size of the event space grows exponentially with the number of involved entities, the number of actually observed events in most available datasets quickly becomes much smaller than the number of possible events. Therefore, modelling a probability distribution over the event space requires structural assumptions in order to reduce the effective dimension of the problem.

Heterogeneous. The existence of several entity and event types brings another layer of complexity. Indeed, besides the aforementioned challenge of estimating a probability distribution over the event space, the heterogeneity of events implies that such estimation must be performed for each event type – intuitively, the probability of a given user and a given computer being involved in the same event is not necessarily identical for, say, an authentication and a process creation. However, correlations may exist between these probabilities, and these should be taken into account. In addition, various entity types playing different roles in each event type further complexify the structure of the distribution.

Temporal. Finally, considering event logs instead of isolated events adds a temporal dimension. This aspect can be divided into an absolute component and a relative component: the former implies that the probability of a given event happening varies in time, exhibiting both seasonal patterns and long-term trends. As for the latter, it stems from the dependencies which may exist between certain events. Consider for instance a high-level action yielding several events, such as an administrator deploying software on a set of computers. From a probabilistic point of view, it seems reasonable to assume that the i-th low-level event is not independent from the previous ones.

While building statistical models for data exhibiting one of these characteristics is a research topic in and of itself, considering all three of them at once is yet another kind of problem. As a consequence, accurately modelling all of the information contained in the logs can be considered especially challenging. However, remember that our goal is not to model event logs just for the sake of it: statistical tools are only a means to an end, namely detecting intrusions. Therefore, our search for a suitable model should not be guided by its effectiveness at predicting benign events, but rather by its ability to distinguish benign behaviors from malicious ones. In particular, identifying which of the three aforementioned aspects are crucial and which others may be overlooked is an important step in the process of building better statistical intrusion detection methods. With this in mind, we devote the next section to formally defining intrusions, with a specific focus on their specification in terms of event logs.

2.3 Intrusions from the Point of View of Event Logs

Having formally defined the concept of event logs, we now discuss the notion of intrusion detection. Similarly to the previous section, we start with the most concrete

aspects and incrementally raise the level of abstraction: Section 2.3.1 defines and formalizes the notion of intrusion from a cybersecurity perspective, then Section 2.3.2 introduces a more systematic definition of intrusion detection, along with some useful assumptions. Finally, Section 2.3.3 presents the problem considered in this thesis, emphasizing its similarity with the general concept of anomaly detection.

2.3.1 What Is an Intrusion?

In its most basic and intuitive form, the definition of an intrusion could be the following: someone who is not allowed inside a given perimeter somehow gets into it. Focusing on the information security-related meaning of this notion, however, a few more things can be said.

In this thesis, we mainly focus on sophisticated attacks, carried out by so-called Advanced Persistent Threats (APTs [Tankard, 2011, Sood and Enbody, 2012]). Such attacks are stealthy enough to avoid detection by traditional IDSs, which typically rely on signatures (such as IP addresses of known malicious infrastructure or hashes of known malware samples) or simple heuristics. Starting from outside the network, APTs make their way inside of it in order to achieve some end goals, which often consist in stealing some valuable information, but also extend to sabotage or extortion. Between the initial breach and the final impact, a number of actions can be carried out by the threat actor, each of which being an opportunity to detect the intrusion.

Several attempts have been made to formalize the steps of an advanced intrusion, including Lockheed Martin's well known Cyber Kill Chain [Hutchins et al., 2011]. In this framework, seven steps are defined:

- Reconnaissance: while still outside of the targeted network, the attacker gathers intelligence on it.
- Weaponization: knowing what kind of technologies are used inside the network, the attacker crafts an easily deliverable piece of malware.
- Delivery: the attacker delivers malware to the targeted network (for instance as an attachment to a phishing email).
- Exploitation: malicious code executes on an initial target system.
- Installation: a persistent backdoor is set up on the compromised computer.
- Command and control: a communication channel is created between the backdoor and the attacker's infrastructure.
- Action: the attacker achieves their end goal.

As another example, the MITRE corporation's ATT&CK framework [Strom et al., 2018] defines 14 tactics[1], each of which represents a potential step of an intrusion. It is both more detailed and flexible than the Cyber Kill Chain, but the main elements remain the same. Our work, however, relies on a somewhat simpler description: first of all, the data sources we consider hardly allow us to observe preparatory phases such as reconnaissance or weaponization. In addition, while some specific reconnaissance techniques, such as network scans, may leave traces in event logs, they are not always followed by an actual intrusion. Detecting them is thus less interesting than focusing

[1]Reconnaissance, resource development, initial access, execution, persistence, privilege escalation, defense evasion, credential access, discovery, lateral movement, collection, command and control, exfiltration, impact.

on subsequent steps. Therefore, we focus on what is commonly referred to as post-compromise detection: we assume that the attacker has successfully breached the network's defenses and tries to reach some final objective. The actions they need to take to that end are then simply decomposed as follows (see Figure 2.3 for an illustration):

Initial compromise. The attacker establishes a foothold in the targeted network, using techniques such as phishing, waterholing, vulnerability exploitation or brute force attacks against exposed servers, etc. After completing this stage, they can access at least one host inside the network. Note that from the system's point of view, the attacker appears as a regular user – the one whose credentials were stolen, or the account that was used to run a vulnerable program which the attacker exploited.

Persistence, command and control. After gaining initial access to a first host, the attacker typically tries to make this access permanent. This implies creating a persistent backdoor (i.e. one that is automatically restarted when the host reboots), for instance by registering a new service or scheduled task, or by modifying an already existing one. A command and control (C&C) channel is also established, which often relies on a beacon repeatedly initiating outbound network communications from the compromised host to the attacker's infrastructure, using any of a variety of techniques to make these communications blend in with regular traffic.

Privilege escalation and lateral movement. The attacker's point of entry into the network is usually not their final target. Therefore, they need to propagate to other hosts, which may for instance be hosting sensitive data or applications. They might also need to take hold of other user accounts in order to elevate their rights and privileges, since the initially compromised account often has restricted access to critical resources. From the perspective of event logs, this can be similar to repeated occurrences of the two previous stages in various regions of the network.

Impact. Finally, the attacker accomplishes their end goal: exfiltration of sensitive data, sabotage, ransomware deployment, etc. Note that this stage may happen weeks, months or even years after the initial compromise, and it may itself last for a while (e.g. in the case of espionage). Ideally, we would like to detect the intrusion before this phase begins.

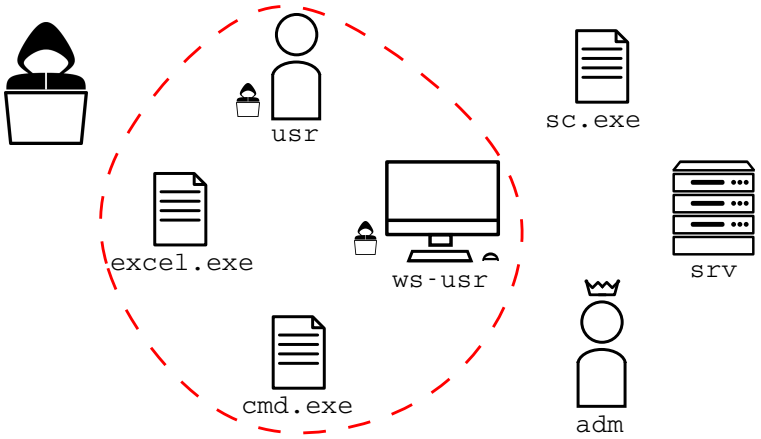
Using this first characterization of the activity to detect, we can now adopt a more formal approach and define the problem in terms of data analysis.

2.3.2 Formal Definition and Assumptions

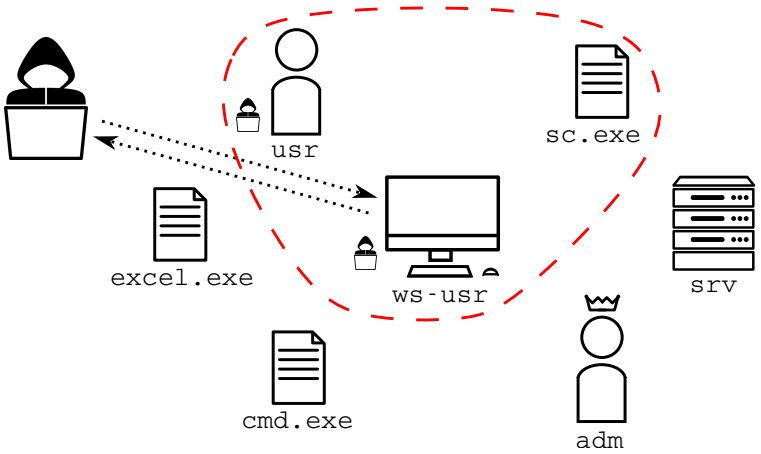
Starting from the data, intrusion detection can intuitively be defined as looking for the events generated by the attacker in a larger set of logged events. This definition can be formally phrased as follows.

Definition 2.3.1 (Intrusion detection: naive definition)**.** Given an event log $\mathcal{L}$, intrusion detection consists in finding a subset $\mathcal{M} \subset \mathcal{L}$ reflecting malicious activity.

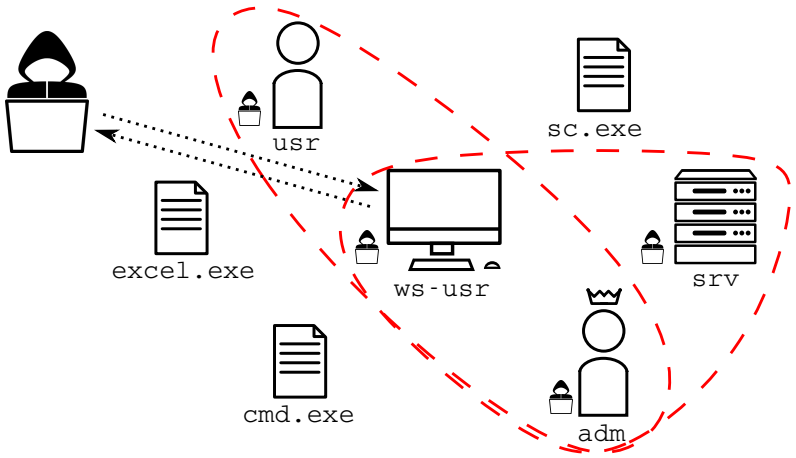
This definition, however, is not very useful as such. There are three main reasons for this: first of all, the diversity of techniques that can be used by an attacker makes



(A) Initial compromise: a normal user opens a weaponized file (in this case, an Excel spreadsheet), which launches a command prompt and starts executing code.



(B) Persistence, command and control: a new service is registered to create a persistent backdoor, and a communication channel is established with the attacker's infrastructure.



(C) Privilege escalation and lateral movement: the attacker uses stolen credentials to authenticate as an administrator, then opens remote sessions on other computers.

FIGURE 2.3: A simple example of an intrusion. Red dashed lines symbolize events, and the presence of the attacker's picture next to a computer or a user account means that this entity is under the attacker's control.

it hard to come up with a both generic and accurate description of malicious activity. Secondly, the same remark applies to the attacker's end goal: for instance, exfiltrating sensitive data does not generate the same kind of events as making critical systems unavailable. Finally and more generally, an intrusion is primarily defined in terms of intention and context: as far as event logs are concerned, there is not much of a difference between malicious actions carried out by some intruder using compromised credentials and a normal user doing the same things for legitimate reasons.

In order to circumvent these difficulties, the problem must be redefined through a slightly different perspective: although characterizing malicious activity hardly seems to be a realistic objective, separating malicious events from benign ones may still be achievable through another approach. However, this implies better specifying the problem, which we accomplish through three elementary assumptions. The first one is rather trivial.

Assumption 1. Malicious event sets are distinguishable from benign ones.

In other words, we simply assume that detection is actually possible – if the set $\mathcal{M}$ looks exactly like any other subset of $\mathcal{L}$, then looking for it is obviously pointless. As mentioned above, the actual challenge lies in exhibiting this difference: since explicitly and generically describing malicious activity is excessively complex, detection methods must rely on other approaches to separate malicious events from benign ones. To that end, a slightly more significant assumption is made regarding the size of $\mathcal{M}$.

Assumption 2. Malicious events are scarce compared to benign ones.

This is a classic assumption in intrusion detection [Portnoy, 2000]. It can be seen as a consequence of the threat model under consideration: since the intruder is supposed to be as discreet as possible, traces of their actions should be much scarcer than the numerous events generated by background activity. In conjunction with Assumption 1, this suggests a close connection with the concept of anomaly detection: what we are actually looking for is a small subset of a given set which "does not fit in". Our last assumption provides a more accurate description of this anomalous subset.

Assumption 3. Malicious events are connected with each other with respect to the entities they involve.

More formally, denoting $\mathcal{M} = \left\{\left(t^i, e^i, \omega^i, \Gamma^i\right)\right\}_{i=1}^{m}$, let $\mathcal{F}(\mathcal{M}) = (\mathcal{V}_\mathcal{M}, \mathcal{E}_\mathcal{M})$ be the graph whose vertex set is defined as

$$\mathcal{V}_\mathcal{M} = \bigcup_{i=1}^{m} \omega^i,$$

and whose edge set $\mathcal{E}_\mathcal{M}$ is defined as follows: for $(u, v) \in \mathcal{V}_\mathcal{M}^2$,

$$(u, v) \in \mathcal{E}_\mathcal{M} \iff \exists i \in [m],\ \{u, v\} \subseteq \omega^i.$$

Then Assumption 3 implies that $\mathcal{F}(\mathcal{M})$, which we call the entity graph induced by $\mathcal{M}$, is connected. From a cybersecurity perspective, this assumption means that the attacker propagates to new entities from already compromised ones. Thus malicious events should make up a connected graph of entities. Using this last assumption, we can come up with a more specific formulation of the anomaly detection problem considered in this thesis.

2.3.3 Intrusion Detection as an Anomaly Detection Problem

Building upon the notions and assumptions introduced in the previous section, we can now design a more formal definition of the problem of intrusion detection in event logs, centered around the concept of anomaly detection.

Definition 2.3.2 (Intrusion detection: formal definition)**.** Given an event log $\mathcal{L}$ and a maximum size ratio $\delta > 0$, intrusion detection consists in finding a subset

$$\mathcal{M} \in \underset{\mathcal{S} \subset \mathcal{L}}{\arg\min}\, p(\mathcal{S}) \quad \text{subject to} \begin{cases} \frac{|\mathcal{S}|}{|\mathcal{L}|} \leq \delta \\ \mathcal{F}(\mathcal{S}) \text{ is connected} \end{cases},$$

where p denotes the probability mass function of benign event sets.

Note that the presence of a maximum size ratio δ results not only from Assumption 2, but also from practical constraints: when actually monitoring the logs from a real computer network, the number of events that can be investigated in a given amount of time is limited. Therefore, solutions to the optimization problem whose size exceeds this limit are of little interest.

One major issue remains: the distribution p is unknown, thus it must be replaced with an adequate function which should be estimated based on the available data. In anomaly detection, this typically boils down to building an anomaly scoring function $f(\cdot)$, which indicates how anomalous a given instance is with respect to a training dataset (as a convention, we consider that anomaly scoring functions give higher scores to anomalous instances). We therefore break the problem into two: estimation of an anomaly scoring function based on a training set of supposedly normal activity, and detection of an anomalous subset in a test set using the scoring function. Note, however, that defining the class of possible anomaly scoring functions requires further thought. In particular, there is no obvious and widely used definition of the set over which f should be defined: should it output one anomaly score for each possible event, or, for instance, take a certain class of event sets as input? Taking this into consideration, the two main problems treated in this thesis can be defined as follows.

Problem 1 – Learning an anomaly scoring function for event sets. Let $\mathcal{T}$ be an event log representing supposedly benign historical activity. Problem 1 consists in defining a class of event sets $\Xi \subset \mathcal{P}(\mathcal{H})$ (where $\mathcal{P}(\cdot)$ denotes the power set of a set), a class of scoring functions $\{f_\theta : \Xi \to \mathbb{R};\, \theta \in \Theta\}$ (with $\Theta \subseteq \mathbb{R}^d$), a training criterion $\ell_\mathcal{T} : \Theta \to \mathbb{R}$ such that f_{θ^*}, with $\theta^* \in \arg\min_{\theta \in \Theta} \ell_\mathcal{T}(\theta)$, is a good anomaly scoring function, and an optimization procedure for $\ell_\mathcal{T}$.

Problem 2 – Detecting a connected subset of anomalous events. Let $\mathcal{L}$ be an event log observed subsequently to $\mathcal{T}$, and $f_{\hat{\theta}}$ be an anomaly scoring function obtained by solving Problem 1. Given a maximum size ratio $\delta > 0$, Problem 2 consists in finding a subset

$$\mathcal{M} \in \underset{\mathcal{S} \in \mathcal{P}(\mathcal{L}) \cap \Xi}{\arg\max}\, f_{\hat{\theta}}(\mathcal{S}) \quad \text{subject to} \begin{cases} \frac{|\mathcal{S}|}{|\mathcal{L}|} \leq \delta \\ \mathcal{F}(\mathcal{S}) \text{ is connected} \end{cases}.$$

Various examples of possible classes of event sets and scoring functions are given in Chapter 3. Note that, similarly to our definition of event logs, this formulation of the problem is meant to be generic: one of our goals is to unify existing contributions, which often rely upon definitions of their own, into a cohesive formalism. To conclude

Table 2.1: Entity types defined in the LANL dataset, along with their arities in the whole dataset and in red team events only.

Name	Arity (all)	Arity (malicious)
User	19 679	104
Computer	15 846	299
Auth. type	27	1
Process	24 742	0

this chapter, we study a real event log dataset and demonstrate the relevance of our definitions and assumptions through this practical example.

2.4 A Practical Example: the LANL Dataset

This section provides an exploratory analysis of the "Comprehensive, Multi-Source Cyber-Security Events" dataset [Kent, 2015a], which was released in 2015 by the Los Alamos National Laboratory [Kent, 2015b]. After briefly describing the contents of the dataset in Section 2.4.1, we discuss in further detail the structure of normal and malicious activity in Sections 2.4.2 and 2.4.3, respectively. Through this study, we aim to motivate the definitions and assumptions stated in previous sections.

2.4.1 Description

The LANL dataset includes several data sources: authentication and process-related events extracted from Windows event logs, NetFlow records for internal network communications and DNS logs. We focus on the first two categories, namely authentication and process management. More specifically, only logons and process creations are considered, ignoring logoffs and process termination events, which seem less relevant from an intrusion detection perspective. Using the formalism introduced in Section 2.2.2, we define four entity types: user (actually a user account, defined by a user name and a domain name), host, process (which actually refers to a process name rather than one actual process running on a specific computer), and authentication type (which is defined as the conjunction of a logon type and an authentication package). Using these entity types, three event types are then defined:

Local authentication. A user is authenticated locally on a given host. Three entities are involved: a user, a host and an authentication type.

Remote authentication. A user authenticates from a source host to a destination host (distinct from the source). Four entities are involved: a user, a source host, a destination host and an authentication type.

Process creation. A user starts a new process on a given host. Three entities are involved: a user, a host and a process.

In addition to selecting these event types only, we filter out all events involving computer accounts or built-in accounts (such as "LOCAL SYSTEM"). Note that this is not always appropriate, since it is technically possible for an attacker to compromise such an account. However, no such activity is reported in the LANL dataset, and this preprocessing step allows us to significantly reduce its size, making it more amenable to various computations.

TABLE 2.2: Events in the LANL dataset: defined event types, ordered types and meaning of the involved entities, and counts (total event count and number of unique entity tuples). The entity types are user (U), host (H), authentication type (T) and process (P).

Name	Entity types	Entity meanings	Count (malicious)	
			Total	Unique
Local auth.	(U, H, T)	Credentials used, host where logon happens, auth. type	26 017 837 (0)	88 259 (0)
Remote auth.	(U, H, H, T)	Credentials used, source host, dest. host, auth. type	103 501 632 (702)	529 354 (433)
Proc. start	(U, H, P)	User creating process, host where process is created, process name	34 986 891 (0)	877 444 (0)

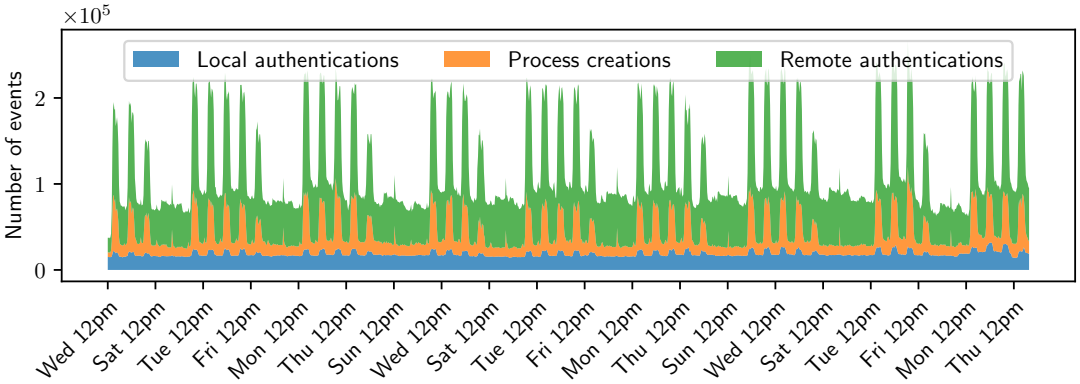


FIGURE 2.4: Number of events of each type observed hourly in the LANL dataset.

The logs cover 58 consecutive days. A red team exercise took place during this period, meaning that a realistic intrusion was performed by security experts in order to assess the level of security of the whole network. Authentications performed by the red team are labelled, providing an example of intrusion to detect. Note that the red team exercise probably also resulted in some process creation events, but no labels are available for these. Despite this limitation, the LANL dataset remains an interesting example of both benign and malicious activity recorded in a real computer network. Therefore, studying its contents is a good way to get a first glimpse of the kind of data we deal with in this thesis.

Starting with basic statistics about the entity and event types defined above, Table 2.1 shows the number of distinct entities for each entity type, as well as the number of entities involved in red team events. Note that the counts do not exactly match those provided by the LANL: we obtain a greater number of users, which is probably due to the fact that we treat (user, domain) pairs with the same user but a different domain as separate users. As for computers and processes, considering only a subset of all the events in the dataset results in smaller entity counts. Regarding events, Table 2.2 shows event counts by type in the whole dataset, along with the number of red team events. A more fine-grained analysis of the observed entities and events is provided in the next two sections.

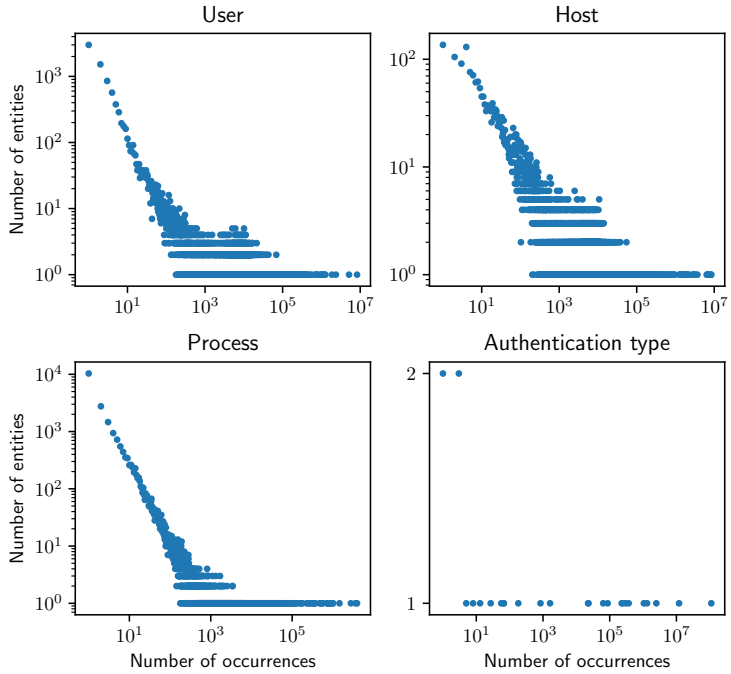


FIGURE 2.5: Distribution of the number of events involving a given entity for each entity type.

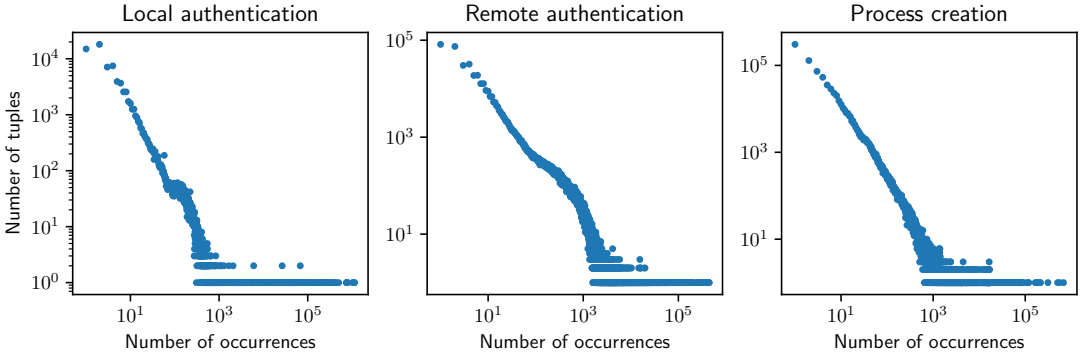
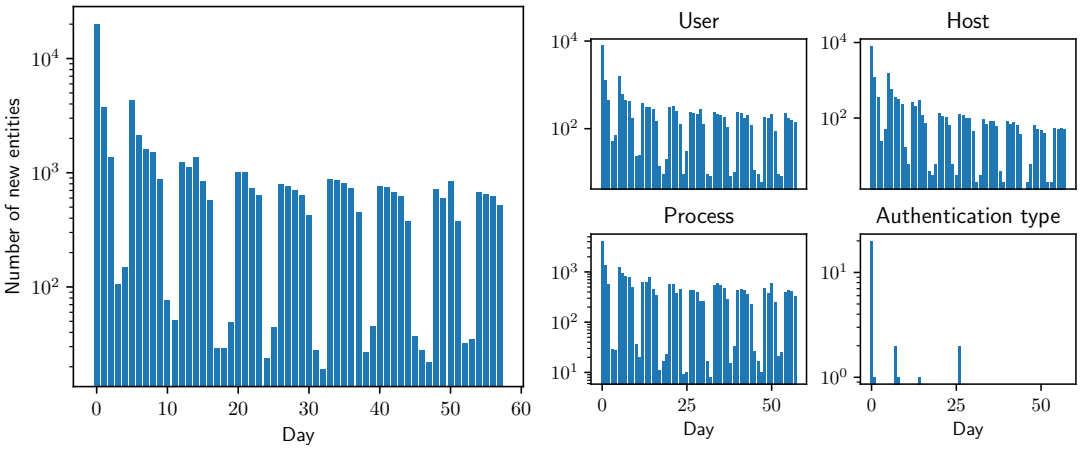


FIGURE 2.6: Distribution of the number of events involving a given entity tuple for each event type.

2.4.2 Characteristics of Normal Activity

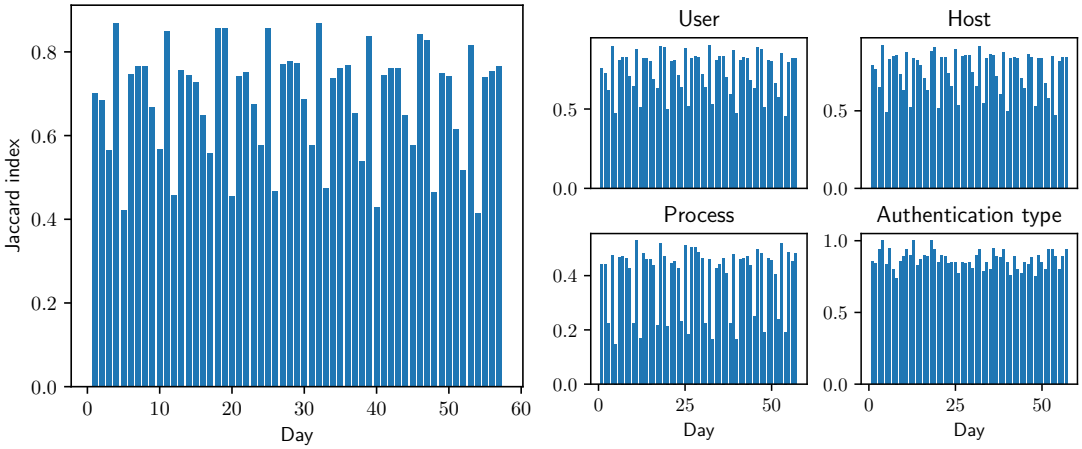
We first focus on normal activity – in other words, everything but the small set of labelled red team events. An obvious observation comes first: normal activity generates a considerable amount of events. Coming back to Table 2.2, the dataset contains approximately 164M benign events, which gives a daily average of 2.83M events in the absence of malicious activity. This volume is expected to vary in time, which can indeed be observed in Figure 2.4: hourly event counts exhibit clear seasonal variations corresponding to the days of the week as well as office hours within each day. Interestingly, these seasonal variations are steeper for remote authentications and process creations than for local authentications, which might suggest that a greater proportion of the latter results from automated activities.

Going a bit deeper, we now examine the content of the events – more specifically, the entities they involve. A simple question can be asked first: are some entities observed more frequently than others? The answer is yes, as evidenced by Figure 2.5: the number of entities involved in n events decreases following a power law pattern



(A) All types (B) By type

FIGURE 2.7: Number of new entities observed on each day in the LANL dataset, aggregated for all entity types (2.7a) and by entity type (2.7b).

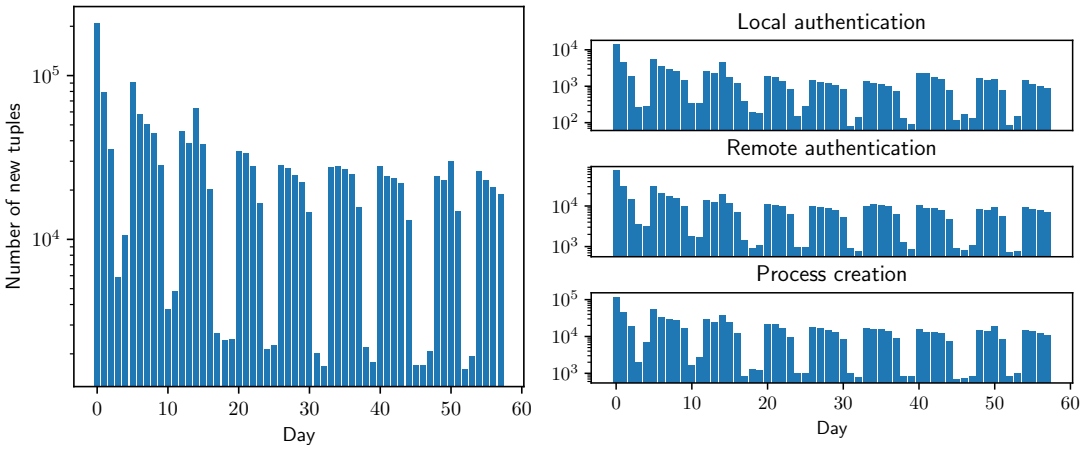


(A) All types (B) By type

FIGURE 2.8: Jaccard index of the entity sets observed on two consecutive days, aggregated for all entity types (2.8a) and by entity type (2.8b).

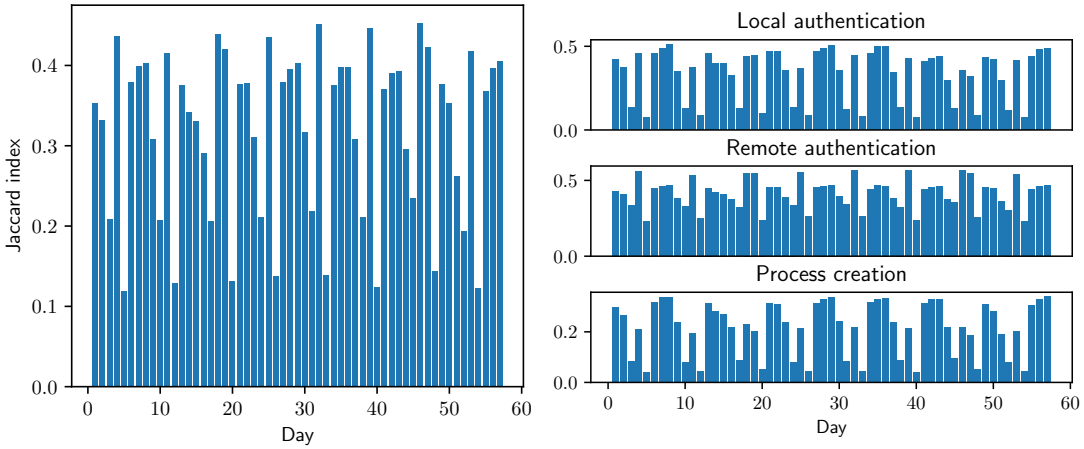
as n increases. This phenomenon happens for each entity type, although it is less evident for authentication types due to a very small entity set. It can be intuitively understood considering the nature of the data: for instance, some computers, such as domain controllers[2], are expected to appear in more events than others. The same goes for users: service accounts associated with very active services will generate more events than accounts of employees whose tasks mostly involve no computer. Going beyond unigram distributions, Figure 2.6 shows the same statistics for entity tuples involved in events: the shape of the distribution also resembles a power law. This is also understandable through domain-specific knowledge: some events, for instance those related to automated network polling, are expected to occur a lot, making a few entity tuples appear often. On the other hand, many occasional actions (typically carried out by human users) will result in different entity tuples appearing in a small

[2]A domain controller is a server which handles user authentication in a computer network. It typically stores information about user accounts and security policy. In particular, for most user accounts, any request for authentication on a host inside the network must be validated by a domain controller.



(A) All types (B) By type

FIGURE 2.9: Number of new entity tuples observed on each day in the LANL dataset, aggregated for all event types (2.9a) and by event type (2.9b).



(A) All types (B) By type

FIGURE 2.10: Jaccard index of the entity tuple sets observed on two consecutive days, aggregated for all event types (2.10a) and by event type (2.10b).

number of events.

Another important aspect is the temporal dimension. In particular, the temporal variability in the observed events must be taken into account. It can intuitively be expected to be significant: activity in a computer includes an important human component, which is intrinsically irregular. Moreover, computer networks evolve in time: new user accounts are created, new computers are added to the network, etc. As a consequence, the set of observed entities cannot be expected to remain constant, as demonstrated by Figure 2.7: the number of yet unseen entities observed each day is always greater than zero, and even after several weeks, it is still in the hundreds. There is, however, a global downward trend, suggesting that most new entities were actually there from the beginning, but had simply not been involved in any event in the first days or weeks. As for short-term variability in the observed data, Figure 2.8 shows that the set of observed entities undergoes significant variations from one day to the next: the Jaccard index between entity sets from consecutive days, defined as the size of their intersection divided by that of their union, is rarely more than 80% (with important differences between event types). Seasonal patterns in this

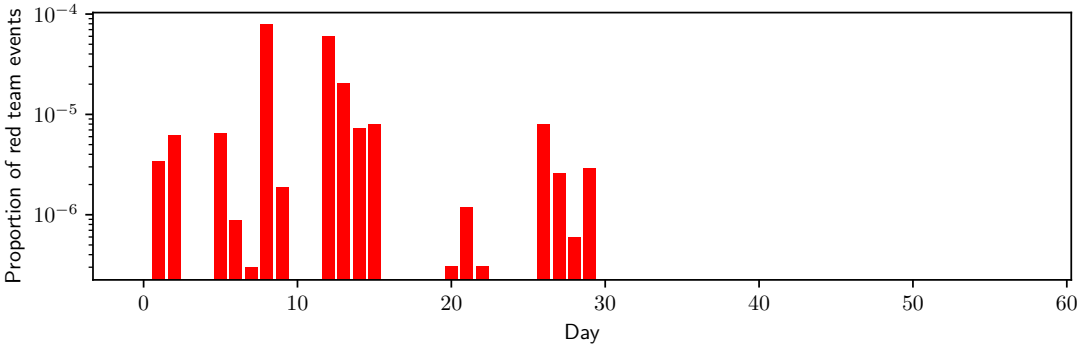


FIGURE 2.11: Proportion of red team events for each day in the LANL dataset.

metric suggest that most of the entities active on weekdays are inactive on weekends, which intuitively makes sense. As evidenced by Figure 2.9 and Figure 2.10, these observations also apply to the entity tuples observed on each given day, sustaining our intuitive claims about the irregularity of normal activity.

In summary, exploratory analysis of benign events in the LANL dataset reveals three characteristics of event logs: important volume, heavy-tailed distributions for the occurrences of entities and entity tuples, and significant temporal variability. From the perspective of statistical modelling and anomaly detection, none of this is good news: large amounts of data imply that scalability may be an issue, while heavy-tailed distributions are hardly suited to anomaly detection. Finally, the nonstationary nature of the data generating process quickly makes any static model obsolete. We can therefore expect anomaly detection in event logs to be a challenging problem.

2.4.3 Characteristics of Malicious Activity

The presence of labelled red team activity is the most interesting aspect of the LANL dataset: even though, as explained in Section 2.3.2, the great diversity of intrusions makes it difficult to define general characteristics from a single example, the presence of supposedly realistic traces of malicious activity is a good opportunity to discuss the relevance of the assumptions we made. The description of malicious activity presented in this section thus revolves around these three assumptions.

Similarly to the previous section, we start with the most obvious observation: Table 2.2 shows that malicious events are vastly outnumbered by benign ones, suggesting that Assumption 2 (scarcity of malicious events) is justified. A more detailed picture is obtained by considering the proportion of malicious events for each day, which is displayed in Figure 2.11. This proportion never exceeds 0.01%, which is arguably quite small. This claim should, however, be slightly attenuated: as mentioned earlier, labelled red team events almost certainly account for only a subset of all events triggered by the intrusion. Moreover, Figure 2.11 shows that red team events only occur on weekdays, which is consistent with the fact that the intruders were actually security experts working inside office hours. Such a property does not necessarily hold in general, and an actual intrusion could obviously happen during a period of low activity. Overall, though, it seems reasonable to assume that the event set we are looking for is small.

Distinguishability (Assumption 1) is up next. One possible way to assess the relevance of this assumption is to compute the proportion of malicious events that also happen without being labelled malicious. In other words, for each malicious event, we check for benign events of the same type, involving the same entities. This search returns matches for approximately 11% of the labelled red team events (76

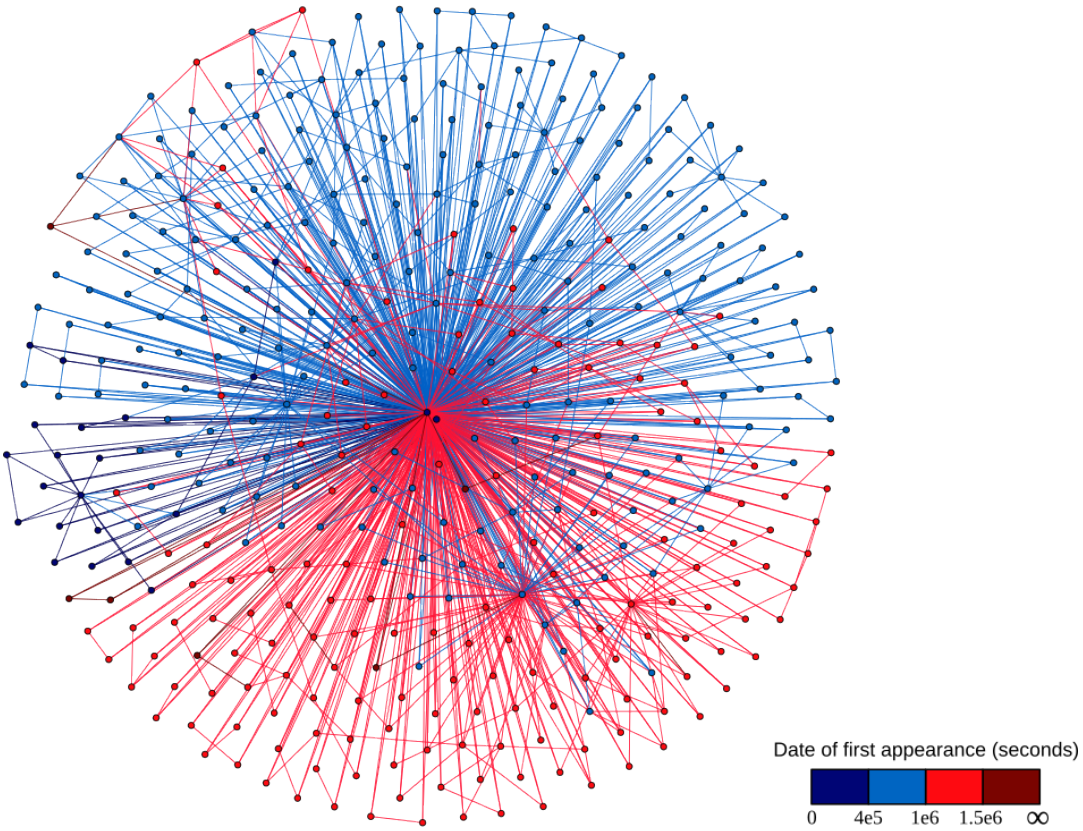


FIGURE 2.12: Entity graph induced by the red team events from the LANL dataset. Each node is an entity, and edges indicate co-occurrence of two entities in a red team event. Nodes and edges are colored according to the date of their first appearance in a red team event.

out of 702), meaning that almost nine malicious events out of ten never happen in the absence of an intrusion. This tends to suggest that malicious events are indeed distinguishable from benign ones based on the entities they involve. Therefore, trying to separate them from the rest through anomaly detection might actually work. Keep in mind, though, that a lot of rare events are also present in the benign event set (see Figure 2.6), making it nontrivial to build this separation in practice.

Finally, in order to evaluate Assumption 3 (connectivity of the set of malicious events), we build the induced entity graph of the red team events, which is displayed in Figure 2.12. This graph is indeed connected, suggesting that Assumption 3 is reasonable. The induced entity graph can also provide some insight on how the red team propagated into the network. To that end, nodes and edges in Figure 2.12 are colored according to the date of their first appearance in a red team event. Two entities seem to stick out, namely those associated with the two high-degree nodes at the center of the figure. The first one is an authentication type ("Network/NTLM"), which was actually used in all of the red team events. More interestingly, the second one is a computer ("C17693"), which was available early on to the red team and was therefore used as a stepping stone to explore the rest of the network. In particular, these two entities are the main bridge between the four time windows which were defined to color the graph (note that these windows were delimited manually based on the apparent clusters of red team activity, which can for instance be observed in Figure 2.11).

2.5 Conclusion

We introduce a theoretical framework to represent event logs, as well as a formal definition of intrusion detection based on event logs. In doing so, we primarily aim to define abstract and generic notions, preserving the intrinsic complexity of the data and making as weak assumptions as possible. Through this approach, we try to build a unifying perspective, allowing us to formalize and compare previously published methodologies. This literature review is the subject of Chapter 3.

The proposed framework relies on the central notion of event. We adopt an interaction-centric approach, considering that events can be primarily defined as polyadic interactions between entities. As a consequence, we treat events as combinatorial data, and we take two other aspects into consideration: first, since events can be of different types, they are also heterogeneous in nature. Secondly, since event logs are sequences of timestamped events, they have a temporal dimension. This multi-faceted nature makes event logs peculiarly complex, hence a need for carefully tailored statistical models.

These statistical models should be built with one specific goal in mind, namely intrusion detection. The latter, though, is not easy to define in a formal and systematic way: in particular, precisely characterizing what intrusion-related events look like hardly seems possible. Therefore, our definition of the problem relies on the concept of anomaly detection: instead of trying to describe the events we look for, we simply assume that they stand out from the norm in some way. Intrusion detection then consists in building an anomaly scoring function, which quantifies how anomalous a given event set is, and using this function to look for a small and cohesive set of anomalous events.

In order to both illustrate and discuss our theoretical framework, we finally study a real event log dataset containing traces of a red team exercise. Besides validating our assumptions about intrusion-related event sets, this exploratory analysis emphasizes some of the challenges associated with our problem. The next chapters present our approach to tackling these challenges.

Chapter 3

A Taxonomy of Anomaly Detection Methods for Event Logs

Chapter 2 introduced some key definitions, formalizing the notions of event, event log and intrusion detection. Using this minimal theoretical framework, we can now review existing work on statistical intrusion detection based on event logs. This chapter outlines the main steps of existing approaches, describing the processing pipeline leading from raw event logs to prioritized pieces of higher-level information. These elementary steps are used to build a taxonomy of previously published detection methodologies. While most existing surveys focus on the models and algorithms used to detect anomalies, we put more emphasis on data representation and implicit assumptions about the underlying data generating process, which we consider more significant when designing a classification.

3.1 Introduction

Given the potential benefits in terms of security, one could expect anomaly detection in event logs to be a rather active research topic. This intuition is indeed correct: since its inception in the late 1980s, with seminal papers such as [Denning, 1987], the field of statistical intrusion detection, including its event log-oriented subdomain, has received a vast amount of contributions. The goal of this chapter is to review and synthesize this literature, using the theoretical framework introduced in Chapter 2 to develop a unified perspective.

The perimeter of our literature review is defined through four characteristics, as schematically depicted in Figure 3.1: first of all, we only consider intrusion detection based on event logs. In other words, only activity-centric methods are included: we aim to directly analyze actions carried out inside the network, and therefore exclude approaches relying on static artifacts related to (or resulting from) these actions. Such artifacts include contents and metadata of documents and executables, memory dumps of computers, etc. Secondly, we are primarily interested in analyzing event logs on a global scale: contributions aiming to model the behavior of a single computer using low-level event logs are excluded from our scope. As for detection methods, we focus on statistical anomaly detection, which can itself be seen as the intersection of two concepts: anomaly detection, as opposed to misuse detection, seeks to characterize the benign rather than the malicious, defining the latter as "not known good" instead of "known bad". In addition, we aim to build this characterization through statistical modelling, excluding systems relying on expert-defined rules.

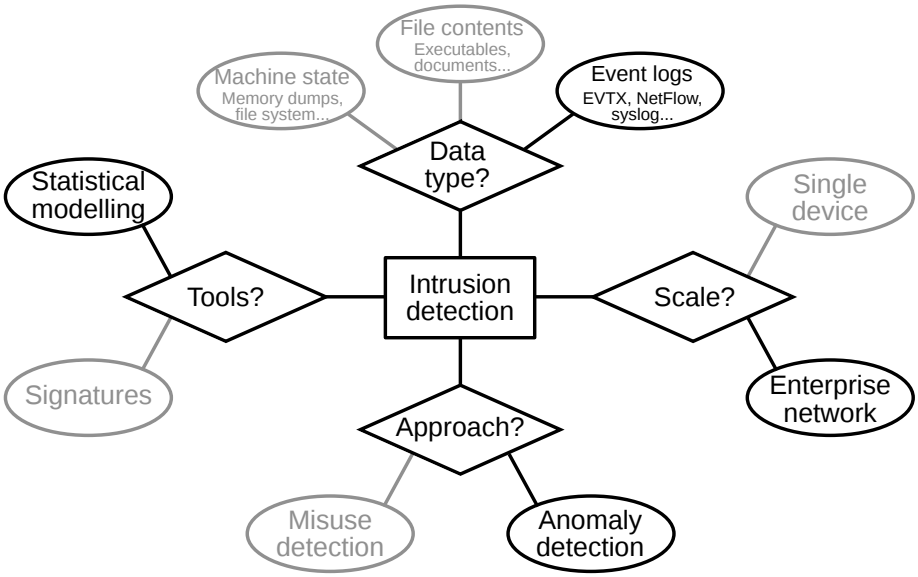


FIGURE 3.1: Perimeter of our literature review.

In Chapter 2, we formally defined anomaly detection in event logs through some key elements, namely the anomaly scoring function f_θ and the class of event sets Ξ on which f_θ is defined. In practice, these objects are parts of a global processing pipeline, represented in Figure 3.2. The first step of this pipeline, which we call the segmentation step, consists in defining the class Ξ: given an event log $\mathcal{L}$, the goal of the segmentation step is to extract meaningful subsets of events from $\mathcal{L}$ – that is, subsets which can be deemed benign or malicious in a way that makes sense from a security perspective. Distinguishing benign event sets from malicious ones is the role of the anomaly scoring function f_θ, whose construction itself entails two main steps. First, the elements of Ξ must be transformed into simpler mathematical objects: indeed, the complex and multi-faceted nature of event logs hinders direct application of standard anomaly detection algorithms to event sets. A representation step is thus needed to map elements of Ξ into another space $\mathcal{X}$, which should be more suited to statistical modelling and anomaly detection. The outcome of the representation step can be formally defined as a map $\phi : \Xi \to \mathcal{X}$. Finally, an anomaly scoring function $\psi_\theta : \mathcal{X} \to \mathbb{R}$ can be defined in what we call the modelling step. The desired function f_θ is then simply obtained as a composite function $f_\theta = \psi_\theta \circ \phi$.

As a logical consequence of the considerable amount of existing contributions, many surveys have also been published. However, they typically put more emphasis on the models and algorithms used for anomaly detection – in other words, they focus on the modelling step (see for instance [Patcha and Park, 2007, Garcia-Teodoro et al., 2009, Tsai et al., 2009, Buczak and Guven, 2015]). In contrast, we consider

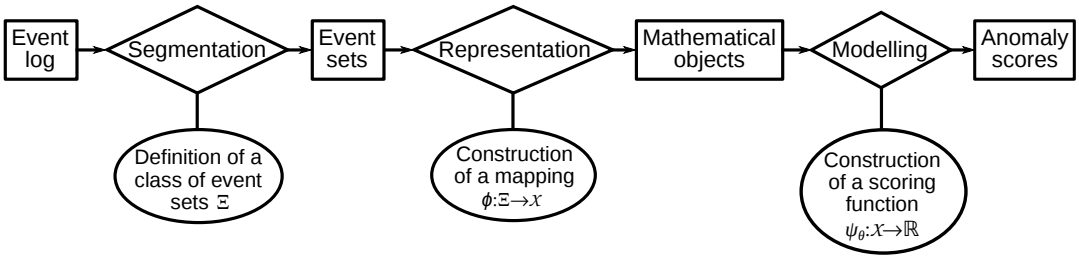


FIGURE 3.2: Processing pipeline associated with statistical intrusion detection in event logs.

the segmentation and representation steps equally if not more important: indeed, the effectiveness of any anomaly scoring function ψ_θ at detecting malicious behavior heavily depends on the quality of the class Ξ and the map ϕ. If the segmentation and representation steps do not actually map malicious event sets to low-density regions of $\mathcal{X}$, no anomaly scoring function will be able to retrieve them. In addition, the modelling step does not simply consist in picking an anomaly detection algorithm: how this algorithm is applied is also crucial, and it reflects implicit assumptions about the underlying generative process from which the data are drawn.

Motivated by the often overlooked importance of segmentation, representation and modelling assumptions, this chapter aims to build a taxonomy of existing methods which revolves around these notions. To that end, we discuss the successive steps of the processing pipeline one after the other: Section 3.2 deals with the segmentation and representation steps. The modelling step and underlying assumptions regarding the process generating the data are then discussed in Section 3.3, focusing on the three main aspects of event logs: combinatorial, temporal and heterogeneous.

3.2 Segmentation and Representation of the Data

In their original, complex and multi-faceted form, event logs are hardly suited to statistical modelling and anomaly detection. Therefore, the first half of the processing pipeline aims to transform them into a collection of simpler mathematical objects. This entails two main steps: in the segmentation step, described in Section 3.2.1, an event log is divided into smaller event sets according to some domain knowledge-based aggregation rules. These event sets are then mapped onto simple mathematical objects in the representation step, which is discussed in Section 3.2.2.

3.2.1 Spatio-Temporal Segmentation

As mentioned in Section 2.3.3, the first step of the anomaly detection pipeline consists in building a class of event sets Ξ over which the anomaly scoring function f_θ can be defined. In practice, these event sets are generally characterized through spatio-temporal constraints: a typical event set of interest can for instance be defined by gathering all events involving a given user inside a given time window. In other words, defining a class Ξ implies two steps: delimiting relevant subsets of entities and segmenting the time axis into a set of windows. This two-dimensional segmentation was formalized by Memory et al. [Memory et al., 2013] through the notion of *extent*. An extent is the conjunction of an entity or group of entities (*entity extent*) and a time period (*temporal extent*), and it is the elementary information unit which can be classified as normal or anomalous.

Regarding the entity extent, a simple and frequent choice is to consider singletons. This idea can be traced back to early work on user profiling, which fundamentally relies on the idea that the data generated by a user summarizes this user's behavior. Early work on statistical analysis of event logs for intrusion detection (such as, among others, [Debar et al., 1992, Forrest et al., 1996, Lee et al., 1997]) focuses on this individual behavior profiling approach. As an illustration, in a highly cited 1998 paper [Lee and Stolfo, 1998], Lee and Stolfo state that

> [t]he elements central to intrusion detection are: resources to be protected in a target system, i.e., user accounts, file systems, system kernels, etc; models that characterize the "normal" or "legitimate" behavior of these resources; techniques that compare the actual system activities with the established models, and identify those that are "abnormal" or "intrusive".

Numerous contributions have kept following this approach since then, mostly focusing on user behavior profiling [Eldardiry et al., 2013, Gavai et al., 2015, Legg et al., 2015, Kent et al., 2015, Rashid et al., 2016, Turcotte et al., 2016b, Wu et al., 2016a, Hu et al., 2017, Tuor et al., 2017, Liu et al., 2019, Powell, 2020] as well as its host-centric counterpart [Yen et al., 2013, Gonçalves et al., 2015, Sexton et al., 2015, Bohara et al., 2016, Veeramachaneni et al., 2016, Bohara et al., 2017, Hogan and Adams, 2018, Siddiqui et al., 2019].

However, there is no reason to limit the analysis to individual behaviors: considering sets of events involving a tuple of entities can yield more fine-grained insights. As a typical example, studying the behavior of a user *on a specific computer* – in other words, aggregating events by user-computer pair – can be considered more appropriate than analyzing the user's overall behavior: for instance, users are not necessarily supposed to carry out the same actions on domain controllers as on their own workstations. More generally, using entity tuples as aggregation keys better reflects the combinatorial nature of events, and it has been an increasingly frequent approach in recent years [Neil et al., 2013a, Turcotte et al., 2014, Schon et al., 2017, Siadati and Memon, 2017, Tang et al., 2017, Adilova et al., 2019, Garchery and Granitzer, 2019, Bowman et al., 2020, Sanna Passino et al., 2020]. Interestingly, it was already suggested by Denning in her seminal 1987 paper [Denning, 1987]:

> An activity profile characterizes the behavior of a given subject (or set of subjects) with respect to a given object (or set thereof), thereby serving as a signature or description of normal activity for its respective subject(s) and object(s).

As for the temporal extent, the most common practice is to divide the time axis into disjoint fixed-length windows. The main question then lies in the choice of a suitable window length. Intuitively, setting either an excessively small or great length can make malicious behavior detection harder: on the one hand, too short time windows might not contain enough events to characterize ongoing activity. However, going too far in the opposite direction might lead to malicious behavior occurring in only a small part of the considered period of time, and getting drowned in normal behavior as a consequence. As no a priori ideal trade-off exists, many possibilities have been explored in the literature, from minutes to hours to days. The two particular cases of null and infinite length, corresponding respectively to a single event[1] and all available events, can be found too. Table 3.1 contains references to some selected contributions using several families of entity subsets and time window lengths.

Note that some contributions also use extents of variable temporal length. In particular, when considering user-computer pairs, aggregating events by session is common practice [Böse et al., 2017, Adilova et al., 2019, Garchery and Granitzer, 2019, Yuan et al., 2019]. A more generic approach consists in defining temporal windows in terms of number of events rather than time elapsed [Heymann and Le Grand, 2013].

3.2.2 Representation through Mathematical Objects

Defining a relevant class of event sets is a way to transform raw event logs into a set of meaningful and comparable objects: instead of looking for "something unusual" in the logs, asking (for instance) whether a given user's behavior in a specific time window is consistent with other windows of supposedly similar activity seems more

[1] This holds under the implicit assumption that two events involving a shared entity cannot happen at the exact same time. Such an assumption is arguably invalid, and we do not use it when building statistical models: its only purpose here is to simplify the discussion by making zero-length windows equivalent to single events.

TABLE 3.1: Selected contributions on statistical intrusion detection in event logs, grouped by type of entity extent and time window length used for segmentation. Note that strictly speaking, using zero-length time windows amounts to considering singletons regardless of the entity extent. The difference between cells in this column lies more on the modelling side, see Section 3.3.

Entities \ Length	**0**	**(0, 1h]**	**[4h, 1d]**	**∞**
User	[Turcotte et al., 2016b] [Garchery and Granitzer, 2020]	[Whitehouse et al., 2016]	[Eldardiry et al., 2013] [Legg et al., 2015] [Hu et al., 2017]	[Kent et al., 2015] [Wu et al., 2016a] [Liu et al., 2019]
Host	-	[Bohara et al., 2016] [Hogan and Adams, 2018] [Siddiqui et al., 2019]	[Yen et al., 2013] [Gonçalves et al., 2015] [Sexton et al., 2015]	[Bohara et al., 2017]
User-resource pair	-	[Sapegin et al., 2015] [Gutflaish et al., 2019]	[Shashanka et al., 2016] [Sanna Passino et al., 2020]	[Turcotte et al., 2016a]
Host-host pair	[Liu et al., 2018b] [Metelli and Heard, 2019] [Price-Williams and Heard, 2020]	[Neil et al., 2013a] [Turcotte et al., 2014] [Schon et al., 2017]	[Lee et al., 2021]	-
Higher-order tuple	-	-	-	[Siadati and Memon, 2017] [Amin et al., 2019]

workable. However, meaningful event sets are still too complex to be fed to standard anomaly detection algorithms: they must be abstracted into simpler mathematical objects first.

More formally, actual anomaly scoring functions $\{\psi_\theta : \mathcal{X} \to \mathbb{R};\ \theta \in \Theta\}$ are defined on specific spaces $\mathcal{X}$, the most frequent case being $\mathcal{X} \subseteq \mathbb{R}^d$ (with $d \geq 1$). Therefore, building an event log-oriented anomaly scoring function $f_\theta : \Xi \to \mathbb{R}$ requires first defining a map $\phi : \Xi \to \mathcal{X}$, and then picking an anomaly scoring function $\psi_\theta : \mathcal{X} \to \mathbb{R}$ such that $f_\theta = \psi_\theta \circ \phi$. This representation step is both crucial and quite challenging. Indeed, it implies finding the right tradeoff between making the data simple enough so that it can be handled by anomaly detection algorithms, and preserving enough information to ensure that detected anomalies remain meaningful from a security perspective. There is no a priori ideal solution to this contradiction: finding a suitable balance can only be done through an empirical approach.

Classical mathematical objects used to represent event logs can be decomposed into three main categories: scalars and vectors, discrete sequences, and combinatorial structures (graphs and hypergraphs). Note that in practice, the definition of the map

ϕ applied to the logs depends on the data sources under consideration. In addition, all data sources are not equally well suited to being represented through a given mathematical object: while, for instance, network traffic metadata can naturally be seen as a weighted and directed graph, commands executed on a given computer more closely resemble a discrete sequence. More details and practical examples of representation maps are given in the next paragraphs.

Scalars and vectors. The most practical mathematical objects in terms of anomaly detection are scalars and vectors – more formally, elements of $\mathbb{R}^d$ with $d \geq 1$, or subsets thereof. Indeed, anomaly detection for such data has been widely studied in the literature, both from a theoretical [Scott and Nowak, 2006] and practical [Chandola et al., 2009] perspective. A widely used approach to translate event sets into such objects is to build vectors of event counts [Gavai et al., 2015, Hu et al., 2017, Tuor et al., 2017, Liu et al., 2018a, Siddiqui et al., 2019]: letting d denote the number of event types, an event set $\mathcal{S} = \left\{\left(t^i, e^i, \omega^i, \Gamma^i\right)\right\}_{i=1}^n$ is converted to a vector $\mathbf{x} \in \mathbb{R}^d$ whose k-th coordinate is $x_k = \sum_{i=1}^n \mathbb{1}_{\{e^i=k\}}$. An even simpler representation can be obtained by dropping event types and taking the total number of events n as a summary of $\mathcal{S}$ [Sapegin et al., 2015].

On the other hand, more complex methods can also be designed by leveraging the entities involved in the events or the specificities of a given data source (in other words, the additional information Γ^i of each event). For instance, the activity of a user can be further aggregated by computer, yielding one event count vector per computer. These vectors are then concatenated into one [Legg et al., 2015]. Alternatively, the number of computers visited by the user can be appended to the count vector [Bhattacharjee et al., 2017]. Regarding the additional information Γ^i, a typical example is the number of packets sent and received when considering network traffic metadata. As these are already numerical data, they can easily be included in a vectorial representation [Gonçalves et al., 2015, Shashanka et al., 2016].

Discrete sequences. Emphasis can alternatively be put on the order in which events happen, representing the set as a temporally ordered discrete sequence. Although slightly less common than numerical data, such objects have also been widely studied in the statistics and data mining literature, including contributions on anomaly detection (see [Chandola et al., 2010] for a survey). Sequential representations of event logs can encompass more or less information. The simplest approach only considers event types, transforming the aforementioned event set $\mathcal{S}$ into a sequence $(e^1, \ldots, e^n)$ [Rashid et al., 2016, Wu et al., 2016a, Adilova et al., 2019]. However, additional information can also be included, such as the time elapsed between consecutive events [Yuan et al., 2019] or details about their content [Turcotte et al., 2016b, Du et al., 2017, Tuor et al., 2018, Garchery and Granitzer, 2020].

Note that defining the notion of anomaly is not as straightforward for discrete sequences as for numerical data. Indeed, while contributions on scalar or vectorial representations of event sets simply define regions of $\mathbb{R}^d$ as anomalous, sequential representations allow for different granularities: a single event can be deemed anomalous with respect to previous ones, but decisions can also be made regarding a subsequence of any length, or even the whole sequence. From an intrusion detection perspective, none of these options stands out as obviously superior (similarly to the choice of a window length discussed in Section 3.2.1): while considering longer subsequences can lead to more reliable decisions by taking more information into account, dividing the data into smaller bits more effectively isolates malicious events, which can help detecting them. Therefore, both options appear in the literature.

Graphs and hypergraphs. Finally, the combinatorial nature of event logs can be leveraged by representing event sets as graphs or hypergraphs. These mathematical objects are particularly flexible and, as a consequence, several representation approaches relying on them have been proposed. The earliest ones focus on data sources whose relational nature is most evident, namely remote authentications and network connections. Kent et al. [Kent and Liebrock, 2013, Kent et al., 2015] introduced the concept of authentication graph of a user, which is a directed graph whose nodes are computers onto which a given user has authenticated in a given time window. An edge from u to v then indicates that the user performed at least one remote authentication from host u to host v. A closely related approach consists in building bipartite access graphs, whose nodes are users and resources (including hosts), with each edge standing for a user accessing a resource [Chen and Malin, 2011, Heymann and Le Grand, 2013, Turcotte et al., 2016a, Moriano et al., 2017, Tang et al., 2017, Gutflaish et al., 2019, Bowman et al., 2020, Sanna Passino et al., 2020]. Network traffic metadata can also be intuitively thought of as a directed graph whose nodes are hosts, with edges representing network communications [Neil et al., 2013a, Neil et al., 2013b, Turcotte et al., 2014, Bohara et al., 2017, Lee et al., 2021].

More recently, however, several contributions have proposed more generic graph-based representations of event logs [Liu et al., 2019, Leichtnam et al., 2020b]. They rely on the combinatorial nature of events to build event graphs – in other words, graphs whose vertices are events. Edges then represent similarity in terms of involved entities or temporal proximity. The intuition underpinning these methods is that malicious events should result in anomalous subgraphs in the event graph, which can be detected through graph mining methods. Unlike authentication graphs or network communication graphs, which can only encode dyadic relationships between entities, event graphs make full use of the combinatorial dimension of events. Another way to overcome this limitation is to represent event logs as hypergraphs or categorical data [Siadati and Memon, 2017, Amin et al., 2019, Eren et al., 2020].

Similarly to discrete sequences, several kinds of anomalies can be defined when considering graphs. At the coarsest scale, a whole graph can be characterized as normal or anomalous, either based on some global structural properties [Kent and Liebrock, 2013, Heymann and Le Grand, 2013, Kent et al., 2015, Moriano et al., 2017] or on the probability given by a random graph model [Gutflaish et al., 2019]. However, finer-grained decisions can sometimes be considered more relevant: for instance, looking for anomalous vertices can uncover compromised user credentials or malicious insiders in user-resource access graphs [Chen and Malin, 2011, Turcotte et al., 2016a] or anomalous events in event graphs [Liu et al., 2019]. The same applies to anomalous edge detection, which has been used to look for malicious use of resources [Tang et al., 2017, Bowman et al., 2020, Sanna Passino et al., 2020] or, more generally, for anomalous associations in event graphs [Leichtnam et al., 2020b]. Finally, an intermediary approach can also be found in the literature, with several contributions aiming to detect anomalous subgraphs in network communication graphs [Neil et al., 2013a, Neil et al., 2013b, Turcotte et al., 2014, Lee et al., 2021].

At the end of the representation step, an initial event log $\mathcal{L}$ has been transformed into a collection of simple mathematical objects, which can be defined with a slight abuse of notation as $\phi(\mathcal{L}) = \{\phi(\mathcal{S}); \mathcal{S} \in \mathcal{P}(\mathcal{L}) \cap \Xi\}$. This transformation relies on two main elements: a class of event sets Ξ, which enables extraction of relevant subsets of $\mathcal{L}$, and a map $\phi : \Xi \to \mathcal{X}$, which transforms these subsets into elements of a more practical space $\mathcal{X}$. The next step then consists in building a good anomaly scoring function $\psi_\theta : \mathcal{X} \to \mathbb{R}$, where "good" here means that the composite function

$f_\theta = \psi_\theta \circ \phi$ returns higher values for malicious event sets. Existing approaches to this modelling step are discussed in the next section.

3.3 Anomaly Detection and Underlying Generative Models

Once raw event logs have been abstracted into simple and usual mathematical objects, any of the numerous existing anomaly detection algorithms can be used to build the anomaly scoring function ψ_θ. However, the chosen algorithm and, perhaps more importantly, the way it is actually applied to the data are highly significant: indeed, they carry implicit assumptions about the underlying generative model. These assumptions pertain to each of the three main characteristics of event logs, and they can essentially be understood in terms of statistical dependencies.

This section covers modelling choices associated with each of these three main characteristics: first of all, Section 3.3.1 deals with the combinatorial aspect, highlighting the difference between aggregation-based models, which essentially treat entity extents as mutually independent, and interaction-based models, which integrate high-order relationships between them. The temporal aspect is then discussed in Section 3.3.2, distinguishing its two components: nonstationarity of the data generating process, and statistical dependencies between successive observations. Finally, Section 3.3.3 reviews the two main paradigms for handling heterogeneous events, namely jointly modelling events of all types and combining marginal models.

3.3.1 Combinatorial Aspect

Because of their combinatorial nature, event logs give rise to intricate association patterns between entities. Beyond first-order associations (i.e. entities appearing in the same event), higher-order relationships can be meaningful as well – for instance, two users often accessing the same resources can be thought of as somehow connected, even though they are not jointly involved in any event. However, taking these high-order relationships into account leads to more complex models. In particular, such relationship-oriented models are necessarily global: the behavior of a given entity can no longer be considered self-contained and entirely characterized by the events involving it. Complex models raise some reliability concerns, thus restricting the focus to local, entity-centric models can arguably be considered a more realistic option.

These two options, which we respectively call interaction-based and aggregation-based modelling, both appear in the literature. Simply put, aggregation-based models see each entity as the sum of its actions, while interaction-based models define it as the sum of its relations. This opposition is illustrated in Figure 3.3, and both paradigms are discussed in more detail in the next paragraphs.

Divide and conquer – Aggregation-based models. Aggregation-based models mostly rely on comparison between supposedly independent and identically distributed instances. Coming back to the generic framework introduced by Memory et al. [Memory et al., 2013], this approach can be generically thought of as comparing an extent with a *baseline*. This baseline can consist of past activity of the same entity extent (*longitudinal baseline*) or, conversely, of the activity of other (and presumably similar) entity extents in the same temporal extent (*cross-sectional baseline*). A combination of these two ideas can also be used, comparing an entity extent's latest activity with its own past activity and that of similar entity extents (*simultaneous*

(A) Raw events (B) Aggregation (C) Interaction

FIGURE 3.3: Illustration of the two main paradigms for handling combinatorial events: given a set of events seen as interactions between entities (Figure 3.3a), aggregation-based models (Figure 3.3b) consider each entity extent as an independent actor whose behavior is described by the events involving it, while interaction-based models (Figure 3.3c) factor in high-order relationships between entities.

baseline). Using our own terminology, the baseline is the set of events used to build the anomaly scoring function ψ_θ, which is then applied to the extent under consideration.

The aggregation-based approach has been widely used, in combination with all kinds of extents and mathematical objects. Its most frequent (and perhaps simplest) application is user behavior monitoring through anomaly detection in Euclidean spaces. This methodology represents the behavior of each user in each temporal extent through a fixed-size vector, then looks for anomalous behavior using classic anomaly detection algorithms such as Isolation Forest [Liu et al., 2008], Local Outlier Factor [Breunig et al., 2000] or One Class Support Vector Machine [Schölkopf et al., 2001], among others. The choice of the baseline is then one of the most important aspects, and all possible kinds have been used: longitudinal [Legg et al., 2015, Hu et al., 2017], cross-sectional [Eldardiry et al., 2013, Gavai et al., 2015, Aldairi et al., 2019] and simultaneous [Bhattacharjee et al., 2017, Tuor et al., 2017, Chattopadhyay et al., 2018, Haidar and Gaber, 2018, Liu et al., 2018a].

This methodology can easily be extended to different mathematical objects, such as graphs [Kent and Liebrock, 2013, Kent et al., 2015, Liu et al., 2019, Powell, 2020] or discrete sequences [Rashid et al., 2016, Wu et al., 2016a, Lu and Wong, 2019]. The main difference is the anomaly detection algorithm used to classify extents, which obviously depends on the type of mathematical object under consideration. The same approach can also straightforwardly be applied to hosts instead of users [Yen et al., 2013, Bohara et al., 2016, Bohara et al., 2017, Siddiqui et al., 2019] and, with no major difference, to user-host [Shashanka et al., 2016, Böse et al., 2017, Adilova et al., 2019] or host-host [Neil et al., 2013a, Neil et al., 2013b, Turcotte et al., 2014] pairs. Note, however, that all types of entity extents are not equally well suited to detecting a given kind of malicious behavior. In particular, using entity tuples as entity extents makes rare associations between entities stand out more easily, which can for instance be useful when looking for lateral movements.

All intertwined – Interaction-based models. Interaction-based modelling differs in a sometimes subtle way from aggregation-based modelling with entity tuples as entity extents. To illustrate this difference, consider the following example: suppose user U remotely authenticates for the very first time onto computer C. How can this event be classified as benign or malicious? Aggregation-based methods could handle

this case by computing how often new user-computer pairs start interacting, how often user U visits new computers or how often computer C is visited by new users. In contrast, interaction-based methods rely on global models of how entities interact, and would therefore ask whether users usually connecting to the same hosts as U also visited C or, more generally, whether users similar to U are expected to authenticate onto hosts similar to C.

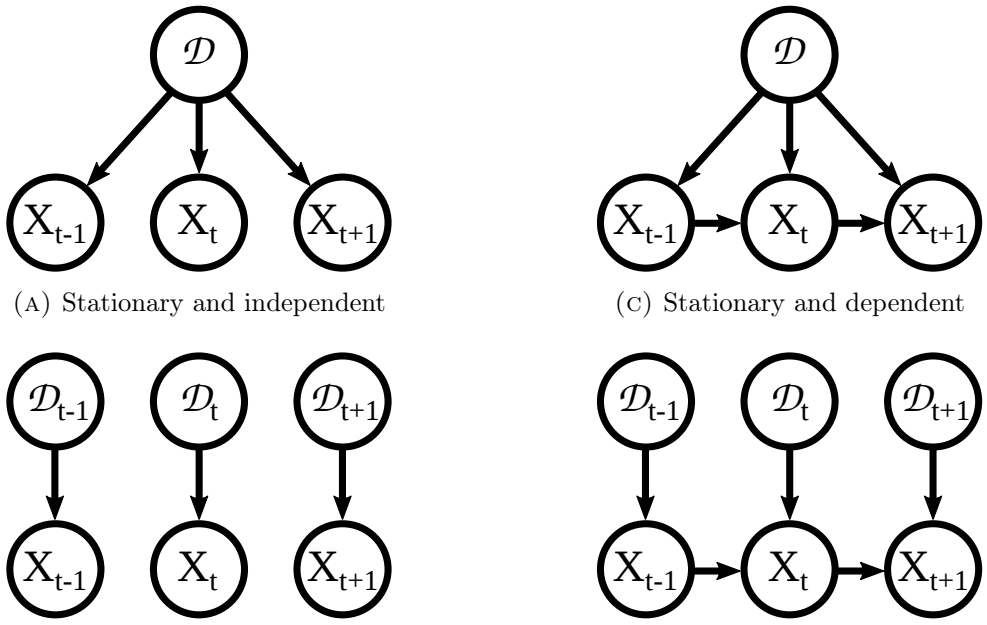
To build such global models, interaction-based methods leverage concepts from combinatorial statistics. These are reviewed more extensively in Chapter 4. A frequently used technique is link prediction in graphs: by learning a classifier predicting the existence of an edge between two given nodes, anomalous interactions between entities can be detected. The classifier can rely either on explicit features of the nodes [Leichtnam et al., 2020b] or latent features inferred from a training dataset of existing links, using techniques such as matrix factorization [Turcotte et al., 2016a, Tang et al., 2017], graph embedding [Wei et al., 2019, Bowman et al., 2020] or other latent space models [Lee et al., 2021, Metelli and Heard, 2019, Sanna Passino et al., 2020]. Similar techniques exist for polyadic interactions as well, which can also be modelled using either explicit [Siadati and Memon, 2017] or inferred [Tuor et al., 2018, Amin et al., 2019, Eren et al., 2020, Garchery and Granitzer, 2020] features describing involved entities. Finally, more global metrics can be used to assess the normality of an entire interaction graph. These can for instance be defined using global structural characteristics [Heymann and Le Grand, 2013, Moriano et al., 2017], graph distances with respect to some baseline [Aksoy et al., 2019] or the likelihood of a random graph model [Gutflaish et al., 2019].

3.3.2 Temporal Aspect

As mentioned in Section 2.2.3, the temporal dimension of event logs can be divided into an absolute component and a relative component. Regarding the absolute component, modelling it amounts to taking into account the nonstationarity of the data generating process. As for the relative component, it pertains to the existence of statistical dependencies between successive observations.

More formally, suppose that an observation X_t is obtained at each time step t. Then, assuming stationarity means that there is a single distribution $\mathcal{D}$ such that $X_t \sim \mathcal{D}$ for all t. Conversely, under the nonstationarity assumption, each observation X_t follows a specific distribution $\mathcal{D}_t$ – note, however, that there may exist time steps $t \neq t'$ such that $\mathcal{D}_t = \mathcal{D}_{t'}$. As for dependence, we focus on first-order Markovian dependencies, leading to the two following possibilities: independent observations, i.e. $X_t \perp\!\!\!\perp X_{t'}$ for all $t \neq t'$, and dependent observations, i.e. $X_t \perp\!\!\!\perp X_{t'} \mid X_{t-1}$ for $t' \notin \{t-1, t\}$. These two subdivisions lead to four possible pairs of assumptions, illustrated in Figure 3.4. Note that these assumptions are usually implicit, but they can be deduced from the detection procedure. In other words, how the anomaly scoring function ψ_θ is built or whether it evolves over time, among other considerations, can be interpreted in terms of temporal characteristics of the underlying data generating process. The next paragraphs successively discuss the absolute and relative aspects, and Table 3.2 lists some selected contributions, grouped according to their implicit assumptions about the temporal dimension.

Absolute component – (Non)stationarity of the data generating process. Coming back once again to [Memory et al., 2013], longitudinal baselines typically carry the implicit assumption that the current behavior of an entity should not differ too much from its past behavior, which can be understood as asserting stationarity of



(A) Stationary and independent

(C) Stationary and dependent

(B) Nonstationary and independent

(D) Nonstationary and dependent

FIGURE 3.4: Illustration of the four possible pairs of assumptions about the temporal dimension of the data. Observations, denoted $X_\bullet$, follow different distributions $\mathcal{D}_\bullet$, and arrows between observations indicate statistical dependencies.

the underlying data generating process. Note that this assumption can be attenuated by only including the most recent observations in the baseline [Legg et al., 2015, Shashanka et al., 2016, Hu et al., 2017], thus implicitly considering the underlying process as slowly evolving. In contrast, using a cross-sectional baseline entails no need for stationarity.

More generally, the anomaly scoring function ψ_θ encodes some knowledge about the underlying data distribution through its parameter vector θ, and handling nonstationarity amounts to updating this parameter vector over time. This updating procedure can consist in a complete re-estimation of θ based on the latest observations [Rashid et al., 2016, Tuor et al., 2018]. It can also be made more incremental, for instance by asking a security expert to review the alerts returned by the model and label them as true or false positives, then adapting the system to correctly predict these labels [Veeramachaneni et al., 2016, Haidar and Gaber, 2018, Siddiqui et al., 2019]. Another way to smoothly update the model is to use Bayesian filtering methods to track the evolution of θ [Lee et al., 2021] (see Chapter 5 for a more detailed description of this approach). Nonparametric Bayesian methods can also be used to update the model in a smooth and theoretically sound fashion [Heard and Rubin-Delanchy, 2016, Sanna Passino and Heard, 2019].

Finally, nonstationarity can be handled to some extent by directly modelling the temporal variations of the distribution using some training dataset. Seasonal variations, which can be expected to occur in event logs, are a good candidate for this type of approach [Turcotte et al., 2014, Sanna Passino et al., 2020]. A more generic model can also be designed to predict the expected anomaly score at time step t based on some temporal features [Gutflaish et al., 2019]. Note that these methods do implicitly assume a longer-term stationarity of the data generating process – more specifically, they assume that short-term variations remain similar over time.

Table 3.2: Selected contributions on statistical intrusion detection in event logs, grouped according to their implicit assumptions about the two aspects of the temporal dimension: absolute (stationary or not) and relative (dependent or not).

	Independent	**Dependent**
Stationary	[Legg et al., 2015]	[Neil et al., 2013a]
	[Bohara et al., 2016]	[Tuor et al., 2017]
	[Bowman et al., 2020]	[Yuan et al., 2019]
	[Powell, 2020]	[Garchery and Granitzer, 2020]
Nonstationary	[Yen et al., 2013]	[Turcotte et al., 2014]
	[Rashid et al., 2016]	[Turcotte et al., 2016b]
	[Lee et al., 2021]	[Tuor et al., 2018]
	[Siddiqui et al., 2019]	[Aldairi et al., 2019]

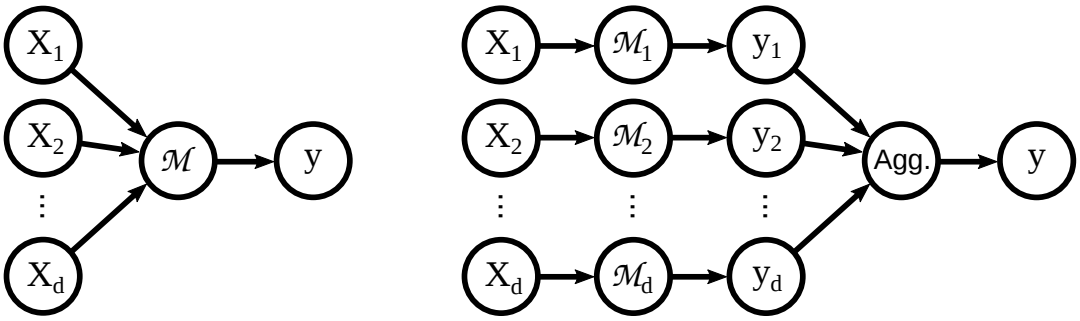
Relative component – Mutual (in)dependence of successive observations. First of all, it should be noted that statistical dependencies between successive observations are seldom considered. However, some contributions do incorporate them through explicitly Markovian models [Neil et al., 2013a, Turcotte et al., 2014, Turcotte et al., 2016b]. Others assume longer-term dependencies, typically by using recurrent neural networks of various kinds [Tuor et al., 2017, Brown et al., 2018, Tuor et al., 2018, Garchery and Granitzer, 2020]. Note that using such models does not necessarily lead to taking into account the link between consecutive observations: several contributions [Rashid et al., 2016, Wu et al., 2016a, Adilova et al., 2019, Lu and Wong, 2019] use Markovian models or recurrent neural networks to model each observation X_t (represented by a discrete sequence), but still consider X_{t+1} independent from X_t. Such distinctions highlight the insufficience of model-centric taxonomies: depending on the way it is actually applied to the data, a given algorithm can output entirely different results and insights about them.

As a side note, a less traditional approach consists in reinjecting the anomaly score given to an entity extent at time step t as an input to the anomaly scoring function at time step $t + 1$ [Aldairi et al., 2019]. While this method is unusual and only lets little information flow between successive time steps, it does effectively provide the model with a notion of dependence between consecutive observations.

3.3.3 Heterogeneous Aspect

The last of the three considered characteristics is also the most frequently overlooked. Indeed, many contributions focus on a single event type, thereby circumventing any challenge related to heterogeneity. Some event types are better suited to this approach than others: typically, remote authentications can be considered sufficient when looking for traces of lateral movement [Siadati and Memon, 2017, Wei et al., 2019, Bowman et al., 2020, Powell, 2020].

However, some detection methods do try to take advantage of the information contained in events of several different types. This can be done through two main approaches, illustrated in Figure 3.5. The first one relies on specialized models handling one event type each, and combines the outputs of these models into a single anomaly score afterwards. Statistical dependencies between events of different types are then essentially ignored, the only connection established between them being that an extent is considered more suspicious if it is anomalous with respect to several event types. In contrast, the second approach first merges events of all types belonging to each



(A) Combine, then model (B) Model, then combine

FIGURE 3.5: Illustration of the two approaches to modelling heterogeneous events: given inputs $X_1, \ldots, X_d$ representing d event types, the first approach (Figure 3.5a) directly merges $X_1, \ldots, X_d$ into a joint model $\mathcal{M}$, while the second one (Figure 3.5b) builds d specialized models $\mathcal{M}_1, \ldots, \mathcal{M}_d$, then aggregates their outputs into a unique one.

extent, then builds a single model for them. This model can then leverage complex relationships between event types.

The next paragraphs discuss these two approaches in further detail, providing concrete examples of their implementation.

Merging outputs from specialized models. The main challenge when merging outputs from specialized models is the choice of the dependence structure between them. In its simplest form, this choice could be phrased as follows: should a very high anomaly score for a single event type be considered more suspicious than moderately high scores across all types? Answering this question mainly boils down to picking a score aggregation function. While simple and generic choices such as the average exist in the literature [Hogan and Adams, 2018, Liu et al., 2018a], more subtle methods can also be used to obtain a different trade-off between the individual importance of each score and the search for simultaneous anomalies across several types. In particular, when anomaly scores are p-values, Fisher's method [Fisher, 1925] can be used to combine them [Turcotte et al., 2016a, Turcotte et al., 2016b].

These simple aggregation methods treat all event types as equivalent. From a security perspective, though, all associations of event types are not necessarily similar: some attack techniques might for instance trigger anomalies for a specific subset of event types. Taking into account such specifities leads to more complex procedures, which typically rely on predefined attack scenarios [Sexton et al., 2015, Liu et al., 2018b].

Jointly modelling different event types. Events of different types bear different meanings and contain different kinds of information, making it nontrivial to merge them into a joint model. Some type-specific details typically have to be ignored in the process, and how much of this type-specific information is accounted for by the model is the main distinctive property of the various existing methods.

The simplest way to handle heterogeneous events is to only take their types into account, discarding any further information. Some aforementioned methodologies build upon this idea: modelling event count vectors, for instance, can be seen as the simplest way to factor in statistical dependencies between event types [Gavai et al., 2015, Hu et al., 2017, Tuor et al., 2017, Liu et al., 2018a, Siddiqui et al., 2019]. Moreover, representing extents as event type sequences [Rashid et al., 2016, Wu et al.,

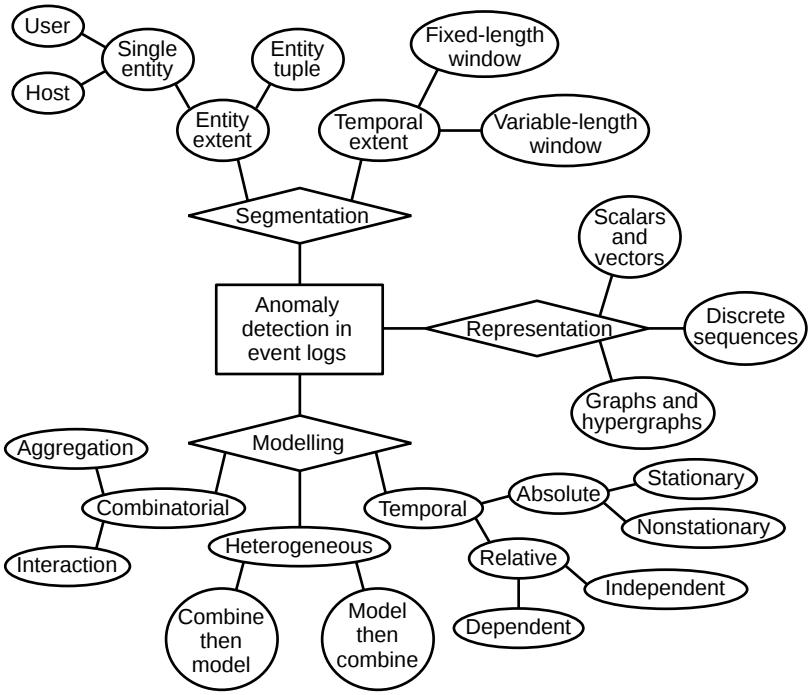


FIGURE 3.6: Taxonomy of anomaly detection methods for event logs.

2016a, Adilova et al., 2019] or event graphs [Liu et al., 2019] enables integration of more subtle relationships in the modelling step.

Other contributions include further type-specific details in the mathematical representations of the extents. They mostly rely on vectorial representations, defining a set of domain-specific features pertaining to different event types: number of exchanged packets when considering network communications [Shashanka et al., 2016], number of authentications of a specific type or with a specific authentication package when working with authentication events [Bohara et al., 2016], number and duration of accesses to each existing application and number of transitions between pairs of applications when logging application usage [Garchery and Granitzer, 2019], and so on. Including such detailed information allows anomaly detection models to identify more complex behaviors whose traces are scattered across several event types.

3.4 Conclusion

We propose a taxonomy of existing anomaly detection methods for event logs. This taxonomy, illustrated in Figure 3.6, relies on an elementary characterization of the data processing pipeline associated with event log analysis. Three main steps are identified, each of which can be thought of as defining a part of the final anomaly scoring function f_θ. First of all, the segmentation step consists in building the class of event sets Ξ on which is f_θ defined. In the representation step, a set $\mathcal{X}$ and a map $\phi : \Xi \to \mathcal{X}$ are then defined in order to transform event sets into simpler mathematical objects. Finally, an anomaly scoring function $\psi_\theta : \mathcal{X} \to \mathbb{R}$ is built in the modelling step, yielding an event log-oriented anomaly scoring function $f_\theta = \psi_\theta \circ \phi$. A simple illustration of this pipeline can be found in Figure 3.2.

The choices made at each of these three steps can be leveraged to characterize and classify existing methodologies. Although it cannot accurately describe all the subtle specificities of every contribution, our simple abstraction of the processing

pipeline allows us to highlight some key differences in terms of assumptions made about the underlying data generating process. This is enough to fulfill our goals, namely describing existing methods in a unifying framework and identifying their distinctive properties. However, extending and improving our framework in order to make it more expressive might be an interesting lead for future work: in particular, it could be a way to ease implementation of anomaly detection methods for event logs.

Having reviewed and analyzed some existing methods, we can now formulate some of the ideas explored in the next chapters. The class of event sets we use is the set of individual events, which do in fact carry enough information to enable statistical modelling and anomaly detection, as we demonstrate in the rest of this thesis. We choose to focus on interaction-based modelling, with two main motivations: first, fully leveraging the combinatorial aspect of event logs seems important when building intrusion detection algorithms. Indeed, as highlighted in Section 2.4.3, malicious events tend to involve unusual combinations of entities, suggesting that interesting anomalies should be defined from a combinatorial perspective. Secondly, this research direction raises interesting challenges: accurately modelling polyadic interactions is far from trivial. In addition, we choose to handle heterogeneity through a joint model for all event types, which also requires some contemplation. These issues are covered in Chapter 4.

As for the temporal dimension, we focus on handling the nonstationarity of the data generating process, assuming mutual independence of all observations. The reason for this is twofold: first, as stated above, malicious activity is expected to generate anomalies from a combinatorial perspective, and the sequential dimension of events seems less distinctive. In other words, since malicious events are already anomalous in and of themselves, looking for unusual sequences of normal events might actually be useless. Secondly, handling nonstationarity seems significantly more important: as evidenced in Section 2.4.2, the contents of actual event logs exhibit strong temporal variations. Therefore, a methodology which does not adequately address the need for frequent model updates may be hardly applicable in practice. This updating problem is discussed in Chapter 5.

Part II

A Statistical Model for Event Logs

Chapter 4

Anomaly Detection for Heterogeneous Polyadic Interactions

Exploration of real-world data as well as previous work on intrusion detection both suggest that explicitly modelling the combinatorial aspect of events is a promising avenue of research. In this chapter, we thus focus on statistical modelling and anomaly detection for combinatorial data. We first review some contributions focusing on homogeneous interactions (both dyadic and polyadic), highlighting the connections existing between them despite the multiplicity of mathematical objects they rely upon: graphs, hypergraphs, matrices and tensors. One of the key challenges when dealing with combinatorial data is the high dimensionality of the sample space, and the concept of dimensionality reduction thus plays a central role in this literature overview. We then factor in the existence of several event types and propose an anomaly detection model for heterogeneous events, as well as a heterogeneity-aware training procedure for this model. In particular, we use multi-task learning to optimally balance the importance of the different event types in the training criterion. The effectiveness of our method is demonstrated on the LANL dataset.

4.1 Introduction

An intruder using stolen credentials to open a remote desktop session from one host to another, or to run some reconnaissance tool on a compromised workstation, is likely to generate events involving unusual combinations of entities. More generally, many of the malicious actions an intruder would perform inside a targeted network can be seen as combinatorial anomalies: more than the number of generated events, the order in which they happen or the temporal pattern they form, the involved entity tuples are the main feature distinguishing malicious events from benign ones. This observation motivates us to look towards combinatorial anomaly detection methods when trying to detect malicious activity in event logs.

Detecting anomalies in combinatorial data is a challenging task due to the high dimensionality of the sample space. Indeed, as a basic example, drawing p elements from a set of size n yields n^p possibilities, thus naively estimating the probability of each possible combination using a finite training set quickly becomes unrealistic

as n and p grow large. This raises two important issues: first of all, estimating the support of an unknown distribution over a combinatorial space requires either a considerable amount of training samples or strong assumptions about the structure of the distribution. Secondly, the low-density regions of the sample space contain most of the possible combinations but, by definition, they should be the least represented in the training set. Thus building a useful anomaly scoring function over these regions is also difficult without additional assumptions. An intuitive translation of these issues in the case of event logs is that only a small fraction of all possible entity tuples is frequently observed, and that only a small fraction of all unlikely entity tuples is likely to be observed as a result of an intrusion. Therefore, identifying malicious events through anomaly detection is not an easy task.

An additional challenge comes from the heterogeneity of events. Indeed, we aim to build a joint statistical model for all event types, which implies that some parameters of the model should be used to compute the probability of events of different types. This can be problematic when training the model: the likelihood for all training samples of one specific event type can be thought of as an independent training objective, and jointly optimizing the objectives associated with each type then becomes a multi-objective optimization problem. One of the difficulties raised by multi-objective optimization is the possibility of a conflict between training objectives: for instance, a given parameter update might make the model better at predicting remote authentications but worse at predicting process creations. Handling such situations requires specific tools, some of which can be found in the field of multi-task learning.

Previous contributions on intrusion detection have explored the idea of treating event logs as combinatorial data. Most of them represent events as dyadic interactions, enabling the use of methods such as matrix factorization [Turcotte et al., 2016a, Tang et al., 2017] or graph embedding [Wei et al., 2019, Bowman et al., 2020]. The polyadic nature of events has also been taken into account in a few contributions, using for instance pattern mining [Siadati and Memon, 2017], recurrent neural network-based language modelling [Tuor et al., 2018], tensor decomposition [Eren et al., 2020] or representation learning [Amin et al., 2019]. However, none of the proposed methods properly handles joint modelling of heterogeneous events and the underlying multi-objective optimization problem. Interestingly, these approaches often draw inspiration from other fields of research, borrowing concepts and methods originally designed for modelling and predicting future activity rather than detecting anomalies. In particular, the field of recommender systems [Jannach et al., 2010] has given rise to a significant amount of research on statistical modelling of combinatorial data in the last 20 years, and a parallel can easily be drawn between content recommendation and user behavior monitoring in computer networks: while recommender systems try to predict which items a given user is likely to consume, intrusion detection systems aim to infer which resources a given user is supposed to access. Despite this similarity, though, the end goals of these two kinds of systems significantly differ: recommender systems aim to make relevant suggestions on average, whereas intrusion detection systems should detect relevant anomalies. More generally, while useful ideas can be drawn from existing work on modelling combinatorial data, the specificity of our use case should be kept in mind.

In this chapter, we propose an anomaly detection model for heterogeneous polyadic interactions. We use dimensionality reduction techniques and ideas from categorical anomaly detection to tackle the issues induced by the high dimensionality of the sample space. In addition, we use multi-task learning methods to handle the multiple training objectives associated with different event types. The effectiveness of our approach is demonstrated through experiments using the LANL dataset.

The rest of the chapter is organized as follows. We first review existing statistical models and anomaly detection algorithms for combinatorial data in Section 4.2, showing how some of these concepts can be leveraged to build a model for homogeneous events. Section 4.3 then generalizes this model to heterogeneous events, using multi-task learning to adapt the training procedure accordingly. Finally, we evaluate our proposed algorithm on the LANL dataset in Section 4.4, demonstrating increased detection performance with respect to existing work.

4.2 Statistical Modelling and Anomaly Detection for Combinatorial Data

We start with a simplified formulation of our problem, namely the case where all interactions are of the same type. A formal definition of the considered problem is given in Section 4.2.1. This generic setting of homogeneous interactions between entities happens to be a generalization of several specific problems, each of which has received significant attention from the research community. We thus review some relevant contributions to each of these specific cases, focusing on statistical modelling in Section 4.2.2 and anomaly detection in Section 4.2.3. Among the models and algorithms we discuss, one specific approach – namely the CADENCE model [Amin et al., 2019] – is especially relevant to anomalous event detection, and we thus present it in further detail in Section 4.2.4. Finally, drawing inspiration from some identified limitations of CADENCE, we propose our own anomalous event detection methodology in Section 4.2.5.

4.2.1 Generic Problem and Particular Cases

Consider a set $\mathcal{U}$ of m entities. We assume that the elements of $\mathcal{U}$ are numbered so that we can write $i \in \mathcal{U}$ or $i \in [m]$ indifferently to refer to entity i. Let $\Lambda \subseteq \mathcal{P}(\mathcal{U})$ be a class of subsets of $\mathcal{U}$ and $T : \Lambda \to \mathcal{Y}$ be an unknown function, which we call the interaction function (with $\mathcal{Y}$ an output space). The generic problem we consider consists in estimating T using a training set $\{(e_i, y_i)\}_{i=1}^n$, with $(e_i, y_i) \in \Lambda \times \mathcal{Y}$ for all $i \in [n]$: given a symmetric divergence function $\Delta : \mathcal{Y}^\Lambda \times \mathcal{Y}^\Lambda \to \mathbb{R}_+$, we seek an estimator $\hat{T}$ minimizing $\Delta(\hat{T}, T)$. Since the true interaction function T is unknown, a surrogate objective must be used. This objective is typically defined through a loss function $J : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, which quantifies the error made by $\hat{T}$ on a given training sample[1]. The estimator $\hat{T}$ is then found by solving

$$\hat{T} \in \operatorname*{arg\,min}_{t \in \mathcal{Y}^\Lambda} \sum_{i=1}^{n} J\big(t(e_i), y_i\big).$$

Note that depending upon the nature of the space $\mathcal{Y}$, the function T can represent different concepts: it can for instance be a probability mass function, which turns the problem into estimation of an unknown distribution over Λ. It can also take ordinal values representing the strength of interaction between given entities, or binary values representing the existence (or absence) of each possible interaction. This generic formulation thus encompasses several well-known problems, which we discuss below.

Example 4.2.1 (Collaborative filtering)**.** Let $\mathcal{U}_1 \subset \mathcal{U}$ and $\mathcal{U}_2 = \mathcal{U} \setminus \mathcal{U}_1$ be two complementary subsets of $\mathcal{U}$, and set $\Lambda = \mathcal{U}_1 \times \mathcal{U}_2$. In practice, $\mathcal{U}_1$ (resp. $\mathcal{U}_2$) typically

[1]Note that the domain of J is not always $\mathcal{Y} \times \mathcal{Y}$ in the rest of this chapter, although its role remains the same.

represents a set of users (resp. items), and the function T tells how much a given user likes a given item. The values taken by T can typically be integers between 0 and R, corresponding to ratings given by users to items. Given a set of known ratings, building the estimator $\hat{T}$ amounts to predicting how each user would rate each item.

Note that Example 4.2.1 can be interpreted as predicting the labels of unobserved edges in a labelled bipartite graph. A closely related problem is presented next.

Example 4.2.2 (Graph link prediction)**.** Let $\Lambda = (\mathcal{U} \times \mathcal{U}) \setminus \{(i,i);\, i \in \mathcal{U}\}$ be the set of pairs of distinct entities. The symmetric function $T : \Lambda \to \{0,1\}$ then defines a graph whose vertex set is $\mathcal{U}$, with an edge between i and j if and only if $T(i,j) = 1$. Let $\mathcal{E} = \{(i,j) \in \Lambda,\, T(i,j) = 1\}$. Given a subset $\mathcal{F} \subset \mathcal{E}$ of observed edges, building $\hat{T}$ amounts to recovering the full graph.

These two examples focus on dyadic interactions. They can of course be extended to the polyadic setting, as illustrated by the following example.

Example 4.2.3 (Categorical table modelling)**.** Let $L \in \{2, \dots, m\}$ be an integer, and let $\{\mathcal{U}_1, \dots, \mathcal{U}_L\}$ be a partition of $\mathcal{U}$. Setting $\Lambda = \prod_{\ell=1}^{L} \mathcal{U}_\ell$, let $T : \Lambda \to [0,1]$ be such that $\sum_{e \in \Lambda} T(e) = 1$. Building the estimator $\hat{T}$ from a set of observations $\{(e_i, N_i/n)\}_{i=1}^{n}$, where $e_i \in \Lambda$ is a tuple and $N_i \in \mathbb{N}$ is the number of occurrences of e_i, amounts to estimating an unknown probability distribution over Λ from $N = \sum_{i=1}^{n} N_i$ independent draws with replacement.

These specific cases and their practical applications, among others, have motivated extensive research on modelling combinatorial data. A brief overview of existing ideas and concepts is given in the next section.

4.2.2 Statistical Models and Dimensionality Reduction

Although the models and algorithms proposed to solve the various instances of the generic problem described above differ in some aspects, they all implement the idea of dimensionality reduction. This is an expected consequence of one of our previous statements, namely that dimensionality is the main challenge when dealing with combinatorial data. In this section, we discuss several of these models, highlighting some connections between them as well as their existing applications in event log-based intrusion detection. We start with the dyadic case, then move on to its polyadic counterpart.

Dyadic case – Matrix factorization and graph embedding. Consider the following problem: there are N users and M items, and each user has rated a small proportion of the items (this proportion can vary across users). The ratings can then be stored in a sparse matrix $\mathbf{Y} \in \mathbb{R}^{N \times M}$ (where unobserved ratings are set to 0). Having observed this matrix, how can we predict the ratings associated with the unobserved user-item pairs?

A common way to answer this question is to assume that ratings depend on latent preferences of users and latent characteristics of items, both of which can be described by low-dimensional vectors [Billsus et al., 1998]. The rating y_{ij} given by user i to item j can then be modelled as the dot product of two low-dimensional vectors $\mathbf{U}_i$ and $\mathbf{V}_j$, i.e. $y_{ij} = \mathbf{U}_i^\top \mathbf{V}_j$. The complete (and only partially observed) rating matrix $\mathbf{Y}^*$ can then be factorized, yielding

$$\mathbf{Y}^* = \mathbf{U}\mathbf{V}^\top, \quad \mathbf{U} \in \mathbb{R}^{N \times D},\, \mathbf{V} \in \mathbb{R}^{M \times D},$$

where $D \ll \min(N, M)$ is the dimension of the latent space. Inferring the unobserved ratings then amounts to estimating the latent factors using $\mathbf{Y}$, then using the product of the estimated latent factor matrices as an approximation of $\mathbf{Y}^*$. In its simplest form, this procedure can be formalized as finding

$$(\hat{\mathbf{U}}, \hat{\mathbf{V}}) \in \underset{\substack{\mathbf{U}\in\mathbb{R}^{N\times D}\\ \mathbf{V}\in\mathbb{R}^{M\times D}}}{\arg\min}\ \delta(\mathbf{U}\mathbf{V}^\top, \mathbf{Y}), \tag{4.1}$$

where $\delta : \mathbb{R}^{N\times M} \times \mathbb{R}^{N\times M} \to \mathbb{R}_+$ is a divergence function. When δ is the Frobenius norm of the difference, i.e.

$$\delta(\mathbf{A}, \mathbf{B}) = \|\mathbf{A} - \mathbf{B}\|_2, \tag{4.2}$$

the solution to this problem is obtained through the truncated singular value decomposition (SVD) of $\mathbf{Y}$ as follows [Stewart, 1993]: let $\mathbf{U}_\mathrm{s}$ (resp. $\mathbf{V}_\mathrm{s}$) be the matrix whose columns are the left-singular (resp. right-singular) vectors of $\mathbf{Y}$ in descending order of the corresponding singular values. In addition, let $\tilde{\mathbf{\Lambda}}$ be the $N \times M$ rectangular matrix whose D first diagonal coefficients are the D largest singular values of $\mathbf{Y}$ in decreasing order, and whose other coefficients are set to zero. Then $\hat{\mathbf{Y}} = \mathbf{U}_\mathrm{s}\tilde{\mathbf{\Lambda}}\mathbf{V}_\mathrm{s}^\top$ solves Equation 4.1 with δ as defined in Equation 4.2.

However, making the divergence depend on the whole matrix $\mathbf{Y}$ is not necessarily a good idea: as mentioned earlier, observed ratings actually account for a small minority of the coefficients y_{ij}. Therefore, using the SVD of $\mathbf{Y}$ gives a disproportionate importance to unavailable information: the estimated latent factors are mostly fit to set unobserved ratings to 0, leading to unaccurate predictions. To circumvent this issue, the divergence function can be modified to depend only on observed ratings, which makes the optimization problem slightly more challenging than computing the SVD [Srebro and Jaakkola, 2003]. Another issue lies in the choice of the latent space dimension K: indeed, setting K too low makes the model not expressive enough and unable to fully capture the structure of the training data, while high values of K may lead to overfitting and poor generalization. A common solution is to add a regularization term to the objective function while using a large K.

Taking these two aspects into account, Mnih and Salakhutdinov proposed a different formulation of the problem, called Probabilistic Matrix Factorization (PMF). Their approach [Mnih and Salakhutdinov, 2007] relies on solving

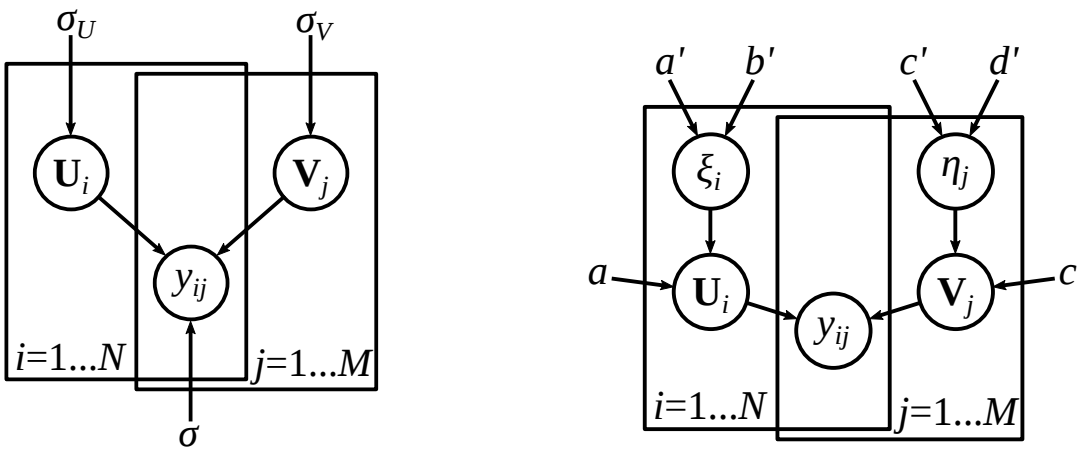
$$(\hat{\mathbf{U}}, \hat{\mathbf{V}}) \in \underset{\substack{\mathbf{U}\in\mathbb{R}^{N\times D}\\ \mathbf{V}\in\mathbb{R}^{M\times D}}}{\arg\min}\ \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\left(y_{ij} - \mathbf{U}_i\mathbf{V}_j^\top\right)^2 + \frac{\lambda_U}{2}\|\mathbf{U}\|_2^2 + \frac{\lambda_V}{2}\|\mathbf{V}\|_2^2, \tag{4.3}$$

where I_{ij} equals 1 if user i has rated item j and 0 otherwise, $\mathbf{U}_i$ (resp. $\mathbf{V}_j$) is the i-th (resp. j-th) line of $\mathbf{U}$ (resp. $\mathbf{V}$) and $\lambda_U, \lambda_V > 0$ are hyperparameters. Solving Equation 4.3 is actually equivalent to maximum a posteriori estimation under the following model: the conditional distribution of each observed rating y_{ij} is defined as

$$p\left(y_{ij} \mid \mathbf{U}_i, \mathbf{V}_j\right) = \mathcal{N}(\mathbf{U}_i\mathbf{V}_j^\top, \sigma^2),$$

with $\mathcal{N}(\mu, \sigma^2)$ denoting the normal distribution with mean μ and standard deviation σ, and the prior distribution of each latent factor is an independent zero-mean spherical Gaussian, leading to

$$p\left(\mathbf{U}_i\right) = \mathcal{N}(0, \sigma_U^2\mathbf{I}), \quad p\left(\mathbf{V}_j\right) = \mathcal{N}(0, \sigma_V^2\mathbf{I}),$$



(A) Probabilistic matrix factorization

(B) Hierarchical Poisson factorization

FIGURE 4.1: Graphical models for two matrix factorization algorithms: probabilistic matrix factorization [Mnih and Salakhutdinov, 2007] (Figure 4.1a) and hierarchical Poisson factorization [Gopalan et al., 2015] (Figure 4.1b).

where $\mathbf{I}$ is the identity matrix. The observation noise variance σ^2 and prior variances σ_U^2, σ_V^2 are related to the hyperparameters as follows: $\lambda_U = \sigma^2/\sigma_U^2$ and $\lambda_V = \sigma^2/\sigma_V^2$. See Figure 4.1a for a graphical model of PMF.

While this model is adapted to sparse rating matrices, it does not work well with so-called implicit feedback – in other words, when the only available information is whether user i has consumed item j or not. Indeed, in this case, using only observed interactions to fit the model leads to trivial solutions: setting all user and item latent factors to the same vector is enough to perfectly explain all implicit feedback. To avoid such a collapse, one possible approach is to formulate the problem as an instance of binary classification, fitting the model on both observed interactions and a subset of unobserved ones. Alternatively, Gopalan et al. [Gopalan et al., 2015] proposed another model, named Hierarchical Poisson Factorization (HPF). The main idea behind HPF is to explicitly factor in the limited budget of each user: instead of treating unobserved user-item interactions as implicit negative feedback, HPF relies on a generative process in which each user is given a finite budget of items to consume and spends it on their favorite items. This generative process, illustrated through a graphical model in Figure 4.1b, is as follows (for a set of hyperparameters $a, a', b', c, c', d' > 0$):

- For each user i, sample activity $\xi_i \sim \text{Gamma}(a', a'/b')$, then for each $d \in [D]$, sample preference $\mathbf{U}_{id} \sim \text{Gamma}(a, \xi_i)$.
- For each item j, sample popularity $\eta_j \sim \text{Gamma}(c', c'/d')$, then for each $d \in [D]$, sample attribute $\mathbf{V}_{jd} \sim \text{Gamma}(c, \eta_j)$.
- For each user-item pair (i, j), sample feedback $y_{ij} \sim \text{Poisson}\left(\mathbf{U}_i \mathbf{V}_j^\top\right)$.

In the case of implicit feedback, observed values are $y_{ij} = 1$ if user i has consumed item j and 0 otherwise. The posterior distribution of the latent factors is then approximated through a mean-field variational inference procedure [Wainwright and Jordan, 2008]. Turcotte et al. applied HPF to intrusion detection based on authentication and process creation logs, using it to spot unlikely user-host and user-process associations [Turcotte et al., 2016a]. Their work was then extended to include explicit entity attributes as well as temporal dynamics [Sanna Passino et al., 2020].

The successive improvements leading from SVD to PMF, then to HPF, exhibit a global trend towards more sophisticated relationships between the latent features of the entities and the observed interactions. Starting with a simple approximation of the rating matrix through a rank-constrained matrix, subsequent contributions gradually detached the generative model of the latent factors from the observations. Another line of work went further down this path, namely the field of graph embedding.

Simply put, graph embedding consists in building a general purpose representation of a graph's nodes in $\mathbb{R}^D$. In other words, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ (resp. $\mathcal{E}$) is the set of nodes (resp. edges) of $\mathcal{G}$, embedding $\mathcal{G}$ in $\mathbb{R}^D$ consists in defining a map $g : \mathcal{V} \to \mathbb{R}^D$ that is suited for various tasks, the most common ones being link prediction and node classification or clustering. In particular, the problem of recommendation with implicit feedback introduced above can be phrased as link prediction in a bipartite graph. As such, it can be addressed through graph embedding followed by the construction of a link prediction function $h : \mathbb{R}^D \times \mathbb{R}^D \to [0, 1]$, which takes the embeddings of two nodes as input and returns the probability of an edge connecting these nodes. Regardless of the type of prediction function used, the relevance of the predictions depends on the quality of the embedding map g.

Intuitively, one way to embed $\mathcal{G}$ in $\mathbb{R}^D$ while preserving its structure is to ensure that the distance between the embeddings of $v_1, v_2 \in \mathcal{V}$ is small if v_1 is connected to v_2 in $\mathcal{G}$ and large otherwise. This idea underpins the concept of spectral embedding: letting $\mathbf{A}$ denote the adjacency matrix of $\mathcal{G}$, we would like to solve

$$\hat{g} \in \underset{g:\mathcal{V}\to\mathbb{R}^D}{\arg\min} \sum_{i,j\in\mathcal{V}} \mathbf{A}_{ij} \left\| g(i) - g(j) \right\|_2^2$$
$$\text{subject to } \begin{cases} \forall i \in [D], \sum_{v\in\mathcal{V}} g(v)_i = 0 \\ \forall (i,j) \in [D]^2, \sum_{v\in\mathcal{V}} g(v)_i g(v)_j = \mathbb{1}_{\{i=j\}} \end{cases},$$

where $g(v)_i$ denotes the i-th component of $g(v)$. The first constraint makes the embedding centered, while the second one prevents trivial solutions mapping all the nodes to the same point. It turns out that the optimal solution to this optimization problem can be constructed using the first $D + 1$ eigenvectors of the Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$ (where $\mathbf{D}$ is the diagonal matrix whose diagonal is the vector $\mathbf{d} = \mathbf{A}\mathbf{1}$), hence the name "spectral embedding": letting $\mathbf{v}_1, \ldots, \mathbf{v}_{D+1}$ denote these eigenvectors, the $|\mathcal{V}| \times D$ matrix of optimal node embeddings is $[\mathbf{v}_2 \cdots \mathbf{v}_{D+1}]$. Modifying the second constraint to

$$\forall (i,j) \in [D]^2, \sum_{v\in\mathcal{V}} \mathbf{d}_v g(v)_i g(v)_j = \mathbb{1}_{\{i=j\}}$$

leads to a slightly different solution, commonly referred to as the Laplacian eigenmap [Belkin and Niyogi, 2003].

Popular alternatives to spectral embedding methods have been introduced in the 2010s, relying on a parallel between nodes in a random walk and words in a sentence. This approach was first introduced by Perozzi et al. under the name DeepWalk [Perozzi et al., 2014]. In practice, DeepWalk involves two main steps: first, for each node $v \in \mathcal{V}$, a fixed number of random walks starting at v are sampled from $\mathcal{G}$. A skip-gram model [Mikolov et al., 2013] is then learned from these random walks, essentially ensuring that for $v_1, v_2 \in \mathcal{V}$, $g(v_1)^\top g(v_2)$ is large if and only if v_1 and v_2 usually appear close to each other in random walks. DeepWalk inspired subsequent contributions [Tang et al., 2015b, Tang et al., 2015a], one of the most influential ones being node2vec [Grover and Leskovec, 2016]. Intuitively, the main way to improve

upon DeepWalk is to come up with more sophisticated definitions of a node's context than simple, unbiased random walks. node2vec does so by proposing a biased random walk algorithm which produces a better depiction of the local structure of $\mathcal{G}$ around each node. It was used for intrusion detection based on authentication logs, in combination with different anomaly detection approaches: Wei et al. proposed to represent each user's authentications through numeric features computed using the embeddings of the hosts visited by this user, then to detect anomalous users using these features [Wei et al., 2019]. A more traditional approach was suggested by Bowman et al., who use node embeddings to learn a link prediction function through logistic regression [Bowman et al., 2020], leading to

$$h\big(g(v_1), g(v_2)\big) = \sigma\Big(\mathbf{w}^\top \big(g(v_1) \odot g(v_2)\big)\Big),$$

where $\mathbf{w} \in \mathbb{R}^D$ is the parameter vector learned through logistic regression, $\odot$ is the element-wise product between two vectors, g and h are the fuctions introduced above (namely the embedding map and the link prediction function), and σ is the sigmoid function, defined by $\sigma(x) = \big(1 + \exp(-x)\big)^{-1}$ for $x \in \mathbb{R}$. Anomalous edges in the user-host authentication graph can then be detected using h.

Despite their greater popularity, random-walk based embeddings were actually shown to be closely related to matrix factorization and spectral embedding methods [Qiu et al., 2018], and the various existing sampling schemes can be interpreted as indirect definitions of matrices to factorize. Therefore, recent research on graph embedding has moved towards more complex models involving deep neural architectures [Hamilton, 2020]. However, these methods have not yet been widely applied to intrusion detection, thus we do not cover them here.

A few lessons can be learned from this first overview. First of all, the main intuition underpinning dimensionality reduction for combinatorial data is that the propensity of an entity to interact with others, as well as the intensity of these interactions, depend on a small number of latent attributes associated with this entity. Therefore, an interaction function T defined over a class of subsets Λ of a set $\mathcal{U}$ of m entities can be approximated using $\mathcal{O}(mD)$ parameters, where $D \ll m$ is the number of latent attributes. This indeed significantly reduces the dimensionality of the problem, in comparison with direct estimation of T for each of the $\mathcal{O}(m^2)$ elements of the class Λ in the dyadic case.

This latent factor-based approach accommodates the sparsity of actual observations (graphs or matrices in the dyadic case). However, this sparsity remains a challenge from a modelling perspective: indeed, fitting a model using a set of observed interactions boils down to determining which of the unobserved interactions are likely to happen in the future, which implies deciding how concentrated the probability mass should be. The models presented above handle this decision in different ways: through prior distributions in the case of PMF and HPF, or through the balance between positive and negative samples when training a link prediction classifier, for instance.

Finally, distinguishing the embedding of $\mathcal{U}$ in $\mathbb{R}^D$ from the estimation of the interaction function T can be a way to inject domain knowledge into the latent factors of the entities. In particular, graph embedding algorithms aim to leverage some expected properties of real-world networks, such as homophily (i.e. the tendency of nodes to be clustered in communities, with most edges connecting nodes from the same community) or structural equivalence (i.e. the existence of nodes playing the same structural role in the network without being connected). Using such properties

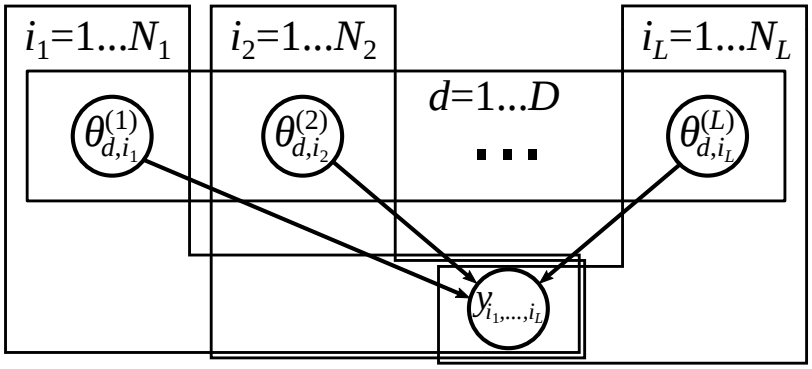


FIGURE 4.2: Graphical model for Poisson tensor factorization [Chi and Kolda, 2012].

can be an interesting way to go beyond the basic intuition that two entities which usually interact with the same other entities will keep doing so in the future.

We now move on to the polyadic setting, in which similar ideas have been applied, although with some adjustments and greater complexity.

Polyadic case – Tensor decomposition and hypergraph embedding. Similarly to the way bipartite (e.g. user-item) interactions can be represented through a rectangular matrix, fixed-size polyadic interactions involving L entities drawn from L distinct sets $\mathcal{U}_1, \ldots, \mathcal{U}_L$ can be represented through an L-way tensor $\mathcal{Y}$. The coefficient of $\mathcal{Y}$ at position $(i_1, \ldots, i_L)$ typically represents the number of observed interactions between entities $i_1 \in \mathcal{U}_1, \ldots, i_L \in \mathcal{U}_L$, or a binary indicator for the existence of at least one such interaction. Pushing the analogy further, a tensor can be approximated through a combination of lower-dimensional latent factors [Kolda and Bader, 2009]. In particular, the tensor rank decomposition [Hitchcock, 1927], also referred to as PARAFAC [Harshman, 1970] or CANDECOMP [Carroll and Chang, 1970], can be seen as a generalization of SVD to tensors. It consists in approximating a given tensor with a sum of D rank-one tensors, which can be formalized as finding

$$\left(\hat{\boldsymbol{\lambda}}, \left\{\left(\hat{\boldsymbol{\theta}}_d^{(1)}, \ldots, \hat{\boldsymbol{\theta}}_d^{(L)}\right)\right\}_{d=1}^{D}\right) \in \underset{\substack{\boldsymbol{\lambda}\in\mathbb{R}^D \\ \boldsymbol{\theta}_d^{(\ell)}\in\mathbb{S}^{N_\ell-1}}}{\arg\min} \left\| \mathcal{Y} - \sum_{d=1}^{D} \lambda_d \boldsymbol{\theta}_d^{(1)} \otimes \ldots \otimes \boldsymbol{\theta}_d^{(L)} \right\|_2, \tag{4.4}$$

where for $\ell \in [L]$, $N_\ell = |\mathcal{U}_\ell|$ is the number of entities in the ℓ-th entity set (and also the size of $\mathcal{Y}$ along the ℓ-th dimension), $\mathbb{S}^{N_\ell-1}$ denotes the unit sphere in $\mathbb{R}^{N_\ell}$, λ_d is the d-th element of $\boldsymbol{\lambda}$ and $\otimes$ denotes the outer product of two vectors. The i-th coefficient of $\boldsymbol{\theta}_d^{(\ell)}$ can then be interpreted as the d-th latent attribute of the i-th entity of type ℓ, making tensor rank decomposition akin to embedding each entity in $\mathbb{R}^D$.

Unfortunately, the least squares criterion used in Equation 4.4 is poorly suited to modelling sparse count data. Therefore, Chi and Kolda [Chi and Kolda, 2012] proposed modelling each coefficient of $\mathcal{Y}$ with a Poisson distribution instead, yielding

$$y_{i_1,\ldots,i_L} \sim \text{Poisson}\left(\sum_{d=1}^{D} \lambda_d \prod_{\ell=1}^{L} \theta_{d,i_\ell}^{(\ell)}\right),$$

where $\theta_{d,i_\ell}^{(\ell)}$ is the i_ℓ-th element of $\boldsymbol{\theta}_d^{(\ell)}$. The model, illustrated in Figure 4.2, can then be fitted through maximum likelihood estimation. This approach was applied to intrusion detection by Eren et al. [Eren et al., 2020], who represent a set of authentication

events as a binary L-way tensor. Several representations are proposed, each of them including a variable number of the following fields from each event: user, source host, destination host, hour (0 through 23), day (Monday through Sunday) and status (success or failure). The obtained tensor is then decomposed as described above, and the learned distribution over the coefficients can be used to classify subsequent events as normal or anomalous.

Tensors are a useful tool when analyzing interactions corresponding to a fixed schema – in other words, when the class Λ of possible interactions can be written as a Cartesian product of disjoint entity sets, i.e. $\Lambda = \prod_{\ell=1}^{L} \mathcal{U}_\ell$. However, they are not adapted to more general classes: in particular, variable-length interactions cannot be represented naturally through tensors. A more flexible mathematical object must then be used, and hypergraphs appear as obvious candidates.

Simply put, a hypergraph is a generalization of a graph in which an edge (also called hyperedge) can involve an arbitrary number of nodes. From now on, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denotes a hypergraph, with $\mathcal{V}$ (resp. $\mathcal{E}$) denoting its set of nodes (resp. hyperedges). Using our formalism, considering hypergraphs amounts to setting $\Lambda = \mathcal{P}(\mathcal{U}) \setminus \{\emptyset\}$. In terms of dimensionality, this entails a steep increase from the $\mathcal{O}(m^2)$ possible edges in a graph to $\mathcal{O}(2^m)$ possible hyperedges, making the need for dimensionality reduction even greater. The existence of a fair amount of contributions on hypergraph embedding thus comes as no surprise.

In the exact same manner as graph embedding, hypergraph embedding consists in building a function $g : \mathcal{V} \to \mathbb{R}^D$ which preserves useful information about the structure of $\mathcal{G}$. Some ideas introduced in the context of graph embedding have thus logically been extended to hypergraphs. Starting with the idea of spectral embedding [Zhou et al., 2006], an equivalent of the Laplacian for a hypergraph can be defined as

$$\mathbf{L} = \mathbf{I} - \mathbf{D}_\mathrm{v}^{-1/2}\mathbf{H}\mathbf{D}_\mathrm{e}^{-1}\mathbf{H}^\top\mathbf{D}_\mathrm{v}^{-1/2},$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{D}_\mathrm{v}$ is the diagonal matrix of node degrees (where the degree of $v \in \mathcal{V}$ is the number of hyperedges incident to v), $\mathbf{D}_\mathrm{e}$ is the diagonal matrix of hyperedge degrees (where the degree of $\omega \in \mathcal{E}$ is the number of nodes in ω), and $\mathbf{H}$ is the $|\mathcal{V}| \times |\mathcal{E}|$ incidence matrix of $\mathcal{G}$, whose entry at position (v, ω) equals $\mathbb{1}_{\{v \in \omega\}}$. The first D eigenvectors of $\mathbf{L}$, denoted $\mathbf{v}_1, \ldots, \mathbf{v}_D$, can then be stacked to build the node embedding matrix $[\mathbf{v}_1 \cdots \mathbf{v}_D]$.

Random walk-based hypergraph embedding methods have also been proposed. Similarly to their graph-oriented counterparts, they rely on the skip-gram model, the main difference lying in the definition of a random walk. Indeed, while an unbiased random walk in a graph can be simply defined as repeatedly sampling a neighbor of the current node uniformly at random and making it the new current node, this definition cannot be straightforwardly extended to hypergraphs. In particular, all neighbors of a given node $v \in \mathcal{V}$ should not necessarily be considered equal: some might share several incident hyperedges with v, while others may be linked to v through smaller (and thus potentially more significant) hyperedges.

A simple generalization of the notion of random walk to hypergraphs can be defined through the following transition rule: starting at node v, we first sample a hyperedge ω incident to v uniformly at random, then sample a node $v' \in \omega$ uniformly at random. Note that this is equivalent to an unbiased random walk on the star expansion of $\mathcal{G}$, which is the bipartite graph whose vertices are the nodes and hyperedges of $\mathcal{G}$, with an edge connecting each pair $(v, \omega) \in \mathcal{V} \times \mathcal{E}$ such that $v \in \omega$ [Agarwal et al., 2006]. Alternatively, each hyperedge ω can be replaced by a clique containing all nodes involved in ω, yielding the clique expansion of $\mathcal{G}$. The biased random walk

algorithm introduced by Grover and Leskovec [Grover and Leskovec, 2016] can then be run on the obtained graph to generate random walks [Huang et al., 2019a].

Using star or clique expansions to sample random walks entails the implicit assumption that all pairs of nodes sampled from a hyperedge are equally strongly connected, which is not necessarily true. In some real-world applications, a single vertex v can indeed play an essential role in the existence of a hyperedge ω, meaning that the other nodes $v' \in \omega \setminus \{v\}$ would no longer be adjacent to each other if v was removed. Huang et al. [Huang et al., 2019b] introduced the notion of indecomposability of hyperedges to describe such phenomena, along with a random walk sampling scheme making use of this concept.

Finally, Gui et al. [Gui et al., 2016, Gui et al., 2017] proposed an embedding algorithm also inspired by natural language processing and the skip-gram model, but not relying on random walks: instead of maximizing the similarity between the embeddings of nodes appearing in a fixed-length window extracted from a random walk, they simply use hyperedges themselves as sets of nodes whose embeddings should be similar. More specifically, given a hyperedge ω, its inner compatibility can be defined as

$$\kappa(\omega) = \sum_{\substack{u,v \in \omega \\ u \neq v}} g(u)^\top g(v).$$

The embedding map g can then be learned by ensuring that for each hyperedge $\omega \in \mathcal{E}$ and each node $v \in \omega$, replacing v with another node $v' \in \mathcal{V}$ such that $(\omega \setminus \{v\}) \cup \{v'\} \notin \mathcal{E}$ makes $\kappa(\omega)$ lower. This idea of inner compatibility of a polyadic interaction is also useful for anomaly detection, as we show in the next sections.

4.2.3 Anomaly Detection Methods for Polyadic Interactions

Dimensionality reduction and link prediction can be considered sufficient tools when trying to detect anomalous interactions in the dyadic case. Indeed, when only two entities are involved in each interaction, the notion of anomaly is rather univocal: entities u and v are either expected to interact or not. Similarly, there is a limited number of counterfactual explanations: interpreting an anomalous interaction between u and v amounts to asking with which other entities u (or v) was supposed to interact. Therefore, estimating a probability distribution over all possible interactions yields a reasonably good anomaly scoring function.

Things get significantly more complicated in the polyadic case. Considering an intrusion detection-related example, suppose user U remotely opens a type T session on host D from host S. Many different aspects of such an interaction might be anomalous: user U could simply not be supposed to visit host D, or to establish remote sessions from S. Going beyond pairwise associations, U might be expected to connect from S to D, but not with a type T session: consider for instance simple network logons, which can typically be triggered by fetching a file on a remote share, versus remote interactive sessions, which provide a fully-fledged graphical interface on the remote host. More generally, any of the $2^L - 1$ non-empty subsets of a set of L entities involved in a polyadic interaction could theoretically provide enough evidence to consider the whole interaction anomalous, independently of the rest of the set. Therefore, relying on dimensionality reduction and a polyadic equivalent of graph link prediction is not necessarily the best way to detect anomalous interactions in this setting.

As a consequence, factoring in the multiple intricate couplings encompassed in each single interaction is one of the main challenges in the polyadic setting. Most

contributions addressing this issue focus on fixed-length interactions defined over the Cartesian product $\Lambda = \prod_{\ell=1}^{L} \mathcal{U}_\ell$, with $\mathcal{U}_i \cap \mathcal{U}_j = \emptyset$ for $i \neq j$. In this setting, looking for anomalous associations implies studying the dependence structure of L categorical random variables. Das and Schneider [Das and Schneider, 2007] proposed doing so through direct estimation of conditional probabilities: given a tuple $\omega = (v_1, \ldots, v_L)$ and two disjoint index sets $\mathcal{I}, \mathcal{J} \in \mathcal{P}([L])$, they define the anomaly score of ω according to $(\mathcal{I}, \mathcal{J})$ as

$$r_{\mathcal{I},\mathcal{J}}(\omega) = \frac{p\left(\{v_i\}_{i\in\mathcal{I}}, \{v_j\}_{j\in\mathcal{J}}\right)}{p\left(\{v_i\}_{i\in\mathcal{I}}\right) p\left(\{v_j\}_{j\in\mathcal{J}}\right)} = \frac{p\left(\{v_i\}_{i\in\mathcal{I}} \mid \{v_j\}_{j\in\mathcal{J}}\right) p\left(\{v_j\}_{j\in\mathcal{J}} \mid \{v_i\}_{i\in\mathcal{I}}\right)}{p\left(\{v_i\}_{i\in\mathcal{I}}, \{v_j\}_{j\in\mathcal{J}}\right)},$$

with a low $r_{\mathcal{I},\mathcal{J}}$ indicating an anomalous association between $\{v_i\}_{i\in\mathcal{I}}$ and $\{v_j\}_{j\in\mathcal{J}}$. Computing this score for all possible pairs of index sets $(\mathcal{I}, \mathcal{J})$ would be both expensive and mostly useless, as one can expect most of these pairs to be uncorrelated in real-world use cases. Coming back to the remote authentication example, the (user, type) pair should probably be strongly correlated with the source host, as a given user can be expected to always perform some specific activity from the same computer. On the other hand, the correlation between the source and the destination should be much less significant. Therefore, a rare association between a (user, type) pair and a source host should be considered more anomalous than a rare association between a source and a destination. In [Das and Schneider, 2007], the most significant $(\mathcal{I}, \mathcal{J})$ pairs are thus extracted by computing the mutual information between candidate pairs on a training dataset. The marginal and conditional probabilities involved in the computations are also estimated on the training dataset through a Bayesian approach.

Aside from this explicitly probabilistic approach, Narita and Kitagawa [Narita and Kitagawa, 2008] proposed a detection algorithm relying on association rule mining, and Akoglu et al. [Akoglu et al., 2012] introduced an information-theoretic definition of anomalous entity tuples, which relies on the description length using a code built to efficiently encode normal tuples. While relying on different tools, both algorithms apply the same general idea as [Das and Schneider, 2007]: they first look for significant correlations between subsets of entities involved in normal interactions, then use these correlations to build a simple description of the data (which can be seen as a kind of dimensionality reduction). Finally, they label interactions which do not fit in this simple description as anomalous. Coming back to the remote authentication example, these methods would for instance find out that the source host can often be inferred from the (user, type) pair. They would then build a list of usual (user, type, source) tuples, then label observed tuples not appearing in this list as anomalous.

In parallel with these advances in anomalous pairing detection, some contributions also tried to directly estimate a joint probability distribution over the set of possible interactions. Silva and Willett [Silva and Willett, 2008] proposed an anomaly detection method for arbitrary hyperedges: they model the presence or absence of each entity in a hyperedge as an independent Bernoulli random variable, effectively reducing the dimensionality of the problem to make it linear in the number of entities. Going beyond this very simple model, Chen et al. [Chen et al., 2016] apply a more conventional hypergraph embedding approach, illustrated in Figure 4.3: they model the probability of a fixed-length interaction $\omega = (v_1, \ldots, v_L) \in \Lambda$ as

$$p_\theta(\omega) = \frac{\exp\left(\kappa_\theta(\omega)\right)}{\sum_{\omega'\in\Lambda} \exp\left(\kappa_\theta(\omega')\right)}, \text{ with } \kappa_\theta(\omega) = \sum_{1\leq i<j\leq L} w_{ij} g(v_i)^\top g(v_j), \tag{4.5}$$

where the parameter set θ contains the weights $\{w_{ij}\}_{1\leq i<j\leq L}$ and the embedding map

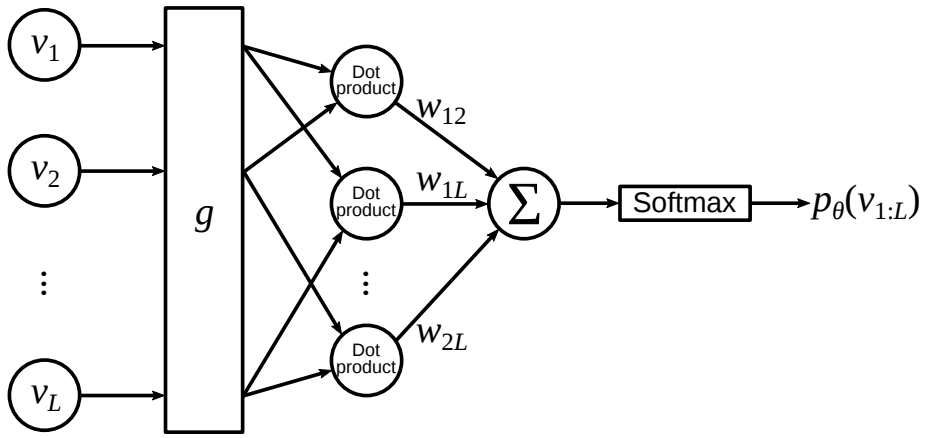


FIGURE 4.3: Illustration of the entity embedding-based model of Chen et al. [Chen et al., 2016].

$g : \bigcup_{\ell=1}^{L} \mathcal{U}_\ell \to \mathbb{R}^D$. These parameters are learned by maximizing the log-likelihood of the model over a training set of observed interactions $\mathcal{T} = \{\omega_1, \ldots, \omega_n\}$, defined as

$$\mathcal{L}(\theta; \mathcal{T}) = \sum_{i=1}^{n} \log p_\theta(\omega_i).$$

From a more machine learning-oriented perspective, this amounts to minimizing the average value of a loss function defined for each sample as

$$J_\theta(\omega) = -\log p_\theta(\omega).$$

The obvious issue with this approach is the cost of computing the partition function $Z_\theta = \sum_{\omega' \in \Lambda} \exp\big(\kappa_\theta(\omega')\big)$ in Equation 4.5. Indeed, computing a sum over the whole product space quickly becomes expensive as the number of entities grows large, and having to perform this computation after each parameter update would make the training of the model too costly. A well-known way to circumvent this issue is to use Noise Contrastive Estimation (NCE [Gutmann and Hyvärinen, 2010]).

In short, the idea of NCE is to recast the estimation of a probability distribution as a binary classification problem. Instead of directly maximizing the probability of observed samples, NCE learns to distinguish them from so-called negative samples drawn from a known noise distribution Q. The partition function Z_θ can then be treated as a parameter and learned along with the others. In [Chen et al., 2016], this leads to

$$p_\theta(\omega) = \exp\big(\kappa_\theta(\omega) + c\big),$$

with $c = -\log Z_\theta$ the parameter replacing the partition function. The surrogate loss function for each sample is then defined as

$$J_\theta^{\mathrm{NCE}}(\omega) = \log \sigma\big(\kappa_\theta(\omega) + c - \log KQ(\omega)\big) + \sum_{k=1}^{K} \log \sigma\big(\log KQ(\tilde{\omega}_k) - \kappa_\theta(\tilde{\omega}_k) - c\big),$$

where K denotes the number of negative samples drawn for each positive sample, $\tilde{\omega}_k$ is the k-th negative sample and σ is the sigmoid function. The noise distribution Q is defined through the following generative process: given a positive sample $\omega = (v_1, \ldots, v_L)$, each negative sample $\tilde{\omega} = (v_1, \ldots, v'_j, \ldots, v_L)$ is obtained by drawing an index $j \in [L]$ uniformly at random, then sampling $v'_j \in \mathcal{U}_j$ uniformly at random.

Intuitively, NCE with this noise distribution learns parameters such that local modifications of observed events decrease their inner compatibility κ_θ.

While the use of sophisticated dimensionality reduction tools makes Chen et al.'s approach appealing, making the inner compatibility $\kappa_\theta(\omega)$ depend on all pairs of entities involved in ω gives the model many degrees of freedom, which can be problematic in the context of anomaly detection. Indeed, using such a complex compatibility function can make the model fit the noise in the training data, degrading the quality of the resulting anomaly scores. In addition, designing a suitable noise distribution over the whole product space Λ is nontrivial: generating negative samples by replacing a single involved entity might be good enough, but many other – possibly better – procedures could be imagined. Addressing these two issues is the motivation behind the CADENCE model, which we discuss in the next section.

4.2.4 The CADENCE Model – Description and Limitations

The CADENCE model [Amin et al., 2019] relies on conditional anomaly detection and entity embedding methods to detect anomalous interactions in the product space Λ. Interestingly, it was explicitly proposed as a malicious behavior detection algorithm for event logs, making it all the more relevant to our work. It mainly differs from the model proposed in [Chen et al., 2016] by focusing on conditional probabilities: the product space $\Lambda = \prod_{\ell=1}^{L} \mathcal{U}_\ell$ is factorized as the Cartesian product of a context space $\mathcal{C} = \prod_{\ell=1}^{C} \mathcal{U}_\ell$ and an attribute space $\mathcal{A} = \prod_{\ell=C+1}^{L} \mathcal{U}_\ell$ (with $C \in [L-1]$), and the anomaly score of an interaction $\omega = (v_1, \dots, v_L) \in \Lambda$ is defined as the conditional probability

$$p(v_{C+1:L} \mid v_{1:C}) = \prod_{\ell=C+1}^{L} p\left(v_\ell \mid v_{1:\ell-1}\right), \tag{4.6}$$

where $v_{i:j}$ equals $\{v_i, \dots, v_j\}$ if $i \leq j$ and $\emptyset$ otherwise. Each of these conditional probabilities is estimated through an entity embedding approach: two embedding maps $g, h : \bigcup_{\ell=1}^{L} \mathcal{U}_\ell \to \mathbb{R}^D$ are learned along with weights $\{w_{ij}\}_{i,j}$ such that

$$p_\theta(v_\ell \mid v_{1:\ell-1}) = \frac{\exp\left(\kappa_\theta^\ell(v_\ell; v_{1:\ell-1})\right)}{\sum_{v' \in \mathcal{U}_\ell} \exp\left(\kappa_\theta^\ell(v'; v_{1:\ell-1})\right)},$$

with

$$\kappa_\theta^\ell(v; v_{1:\ell-1}) = \sum_{i=1}^{\ell-1} w_{i\ell} g(v_i)^\top h(v),$$

approximates $p(v_\ell \mid v_{1:\ell-1})$ for all $(v_1, \dots, v_L) \in \Lambda$ and $\ell \in \{C+1, \dots, L\}$. See Figure 4.4 for an illustration of this model. Note that in contrast with [Chen et al., 2016], CADENCE associates two separate embeddings $g(v)$ and $h(v)$ with each entity v: $h(v)$ is used when v is being predicted while $g(v)$ appears when v is used as context to predict another entity. This separation is frequent in the word embedding literature [Mikolov et al., 2013, Mnih and Kavukcuoglu, 2013].

Similarly to [Chen et al., 2016], the parameters of CADENCE are learned through NCE. However, a separate loss function is computed for each predicted entity inside a given interaction, leading to

$$J_\theta^{\mathrm{NCE}}(\omega) = \sum_{\ell=C+1}^{L} J_{\theta,\ell}^{\mathrm{NCE}}(v_\ell; v_{1:\ell-1}), \tag{4.7}$$

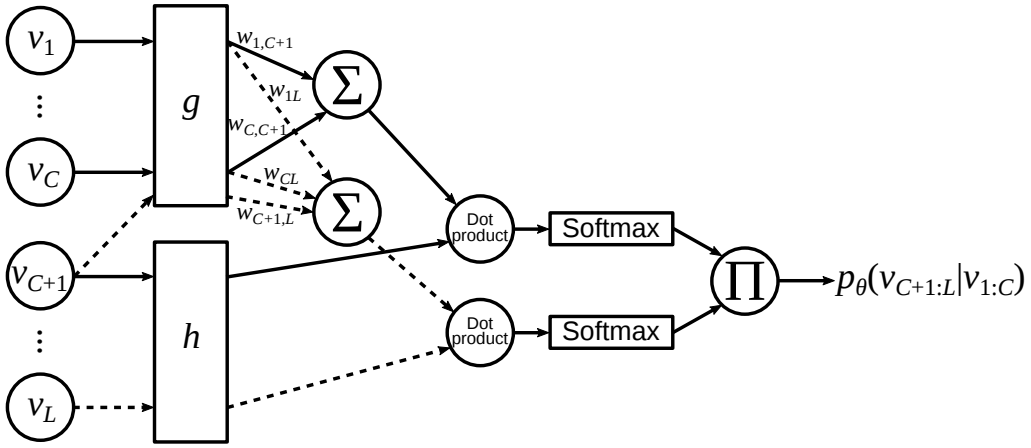


FIGURE 4.4: Illustration of the CADENCE model [Amin et al., 2019].

with

$$J^{\mathrm{NCE}}_{\theta,\ell}(v_\ell; v_{1:\ell-1}) = \log\sigma\left(\kappa^\ell_\theta(v_\ell, ; v_{1:\ell-1}) - \log KQ_\ell(v_\ell)\right) + \sum_{k=1}^{K} \log\sigma\left(\log KQ_\ell(\tilde{v}_k) - \kappa^\ell_\theta(\tilde{v}_k; v_{1:\ell-1})\right). \quad (4.8)$$

Therefore, $L - C$ distinct noise distributions $Q_{C+1}, \ldots, Q_L$ are used, each of which is defined over a single entity set $\mathcal{U}_\ell$. Since the dimension of each entity set is reasonably small, the unigram distribution can be used. This distribution gives each entity $v \in \mathcal{U}_\ell$ a probability proportional to its number of appearances in the training set, and it is frequently used to learn word embeddings. Note that the partition function is entirely omitted in Equation 4.8, which is a way to avoid learning a separate partition function for each context tuple $v_{1:\ell-1}$ (as should normally be the case with NCE). This innovation was first introduced in the context of word embeddings by Mnih and Teh, who found out that it actually did not degrade the performance of the model [Mnih and Teh, 2012].

CADENCE was shown to perform better than Chen et al.'s model at detecting malicious authentications in the LANL dataset [Amin et al., 2019]. One of the main explanations for this result is that the structure of CADENCE's anomaly scoring function is better suited to anomalous event detection: by computing the conditional probability of $v_{C+1:L}$ given $v_{1:C}$ instead of the joint probability of $v_{1:L}$, the attention of the model is more focused on learning normal pairings – and thus more able to spot anomalous ones. This can be seen as an adaptation of the ideas first introduced by Das and Schneider [Das and Schneider, 2007]. In addition, the structure of CADENCE makes predictions fairly interpretable: coming back once again to the remote authentication example, the model first asks whether the user U is supposed to open sessions of type T. It then checks whether U is expected to open type T sessions from host S. Finally, given U, T and S, CADENCE evaluates how normal destination D is. At each of these steps, counterfactual explanations can easily be built: for instance, if D happens to receive a very low predicted probability, replacing it with a more likely destination host can be expected to yield a normal event.

While its superior detection performance and interpretability make CADENCE especially interesting, some aspects of the model can still be improved. First of all, it cannot jointly model heterogeneous events, and neither does it handle nonstationary event streams. These issues are addressed in Section 4.3 and Chapter 5, respectively.

Even when considering homogeneous and identically distributed events, there remains room for improvement: first, CADENCE implicitly assumes that the entity sets $\mathcal{U}_\ell$ are disjoint. As a consequence, when considering events involving several entities of the same type (such as remote authentications), an entity appearing in several events with different roles (such as source and destination for a host) is treated as a set of separate entities – one for each role. Therefore, the model cannot simultaneously leverage different facets of an entity's behavior. Moreover, using the full conditional probability of $v_{C+1:L}$ given $v_{1:C}$ as anomaly score tends to conceal disparities between the conditional distributions of each entity: in particular, when the entity sets $\mathcal{U}_\ell$ have significantly dissimilar sizes, probability mass should tend to be more concentrated for smaller entity sets, leading to higher expected conditional probabilities. Finally, while the unigram distribution is a rather classic choice of noise distribution, it is not necessarily optimal, and other distributions could lead to better results. These aspects are further discussed in the next section.

4.2.5 An Improved Conditional Anomaly Detection Algorithm

Still focusing on homogeneous events, we now describe our conditional anomaly detection algorithm for polyadic interactions. This algorithm is largely inspired by CADENCE, and this section mostly emphasizes the aspects in which it differs. First of all, the event space we consider corresponds to an event type (e, N_e, Ω_e) as defined in Definition 2.2.1: the class Λ is the product space $\prod_{\ell\in\Omega_e} \mathcal{U}_\ell$. In particular, we do not assume that entities involved in each event come from distinct sets: an entity type can appear several times in the tuple Ω_e. However, we assume that all entities involved in an event are distinct from one another.

Letting $\mathcal{U} = \bigcup_{\ell\in\Omega_e} \mathcal{U}_\ell$ denote the whole entity set, we assign a unique embedding $\mathbf{x}_v \in \mathbb{R}^D$ to each entity $v \in \mathcal{U}$. For $\ell \in \{C+1, N_e\}$ (where $C \in [N_e - 1]$ still denotes the number of context entities), the ℓ-th inner compatibility function is then defined for all $v \in \mathcal{U}_\ell$ as

$$\kappa^\ell_\theta(v; v_{1:\ell-1}) = \mathbb{1}_{\{v\notin v_{1:\ell-1}\}} \sum_{i=1}^{\ell-1} w_{i\ell} \mathbf{x}_v^\top \mathbf{x}_{v_i}, \tag{4.9}$$

where the weights $\{w_{ij}\}_{i,j}$ are parameters of the model. The indicator function encodes the idea that each entity can appear at most once in an event. The other difference between this inner compatibility and the one associated with CADENCE is the absence of a second embedding for each entity. This choice is motivated by the same reason as the possibility of two entities of the same type appearing in an event, namely that we aim to jointly model all available information about each entity. Therefore, the same embedding should be used for every occurrence of the entity, regardless of its position in the event or whether it is currently being predicted or used to predict another entity. The conditional probability of $v_{C+1:N_e}$ given $v_{1:C}$ can then be defined through Equation 4.6. The NCE loss is also identical to the one defined in Equations 4.7 and 4.8, the only difference being the definition of the noise distributions $Q_{C+1}, \dots, Q_{N_e}$. Indeed, regardless of the type of distribution used, a slight modification results from the impossibility of an entity appearing twice, namely

$$Q_\ell(v \mid v_{1:\ell-1}) = \mathbb{1}_{\{v\notin v_{1:\ell-1}\}} \frac{\gamma_\ell(v)}{\sum_{v'\in\mathcal{U}_\ell\setminus v_{1:\ell-1}} \gamma_\ell(v')}, \tag{4.10}$$

where $\gamma_\ell : \mathcal{U}_\ell \to \mathbb{R}_+$ is any positive mass function. In the case of the unigram distribution, we have $\gamma_\ell(v) = N_\ell(v; \mathcal{T})$, with $N_\ell(v; \mathcal{T})$ the number of interactions $\omega = (v_1, \dots, v_{N_e})$ in the training set $\mathcal{T}$ such that $v_\ell = v$. However, this choice of

noise distribution is not necessarily optimal: in particular, the strongly unbalanced distribution of entity occurrence counts (see Figure 2.5) may cause the unigram distribution to concentrate too much around a small set of entities. Other entities would then almost never appear as negative samples, in turn degrading the accuracy of the model. Therefore, we also experiment with two variants of the unigram distribution, namely the log-unigram distribution, defined by $\gamma_\ell(v) = \log\big(1 + N_\ell(v; \mathcal{T})\big)$, and the power-unigram distribution, defined for $\alpha \in (0, 1)$ by $\gamma_\ell(v) = N_\ell(v; \mathcal{T})^\alpha$.

Once the model has been trained, an anomaly scoring function must be defined. As mentioned above, CADENCE directly uses the predicted probability

$$p_\theta(v_{C+1:N_e} \mid v_{1:C}) = \prod_{\ell=C+1}^{N_e} p_\theta(v_\ell \mid v_{1:\ell-1})$$

as a measure of anomalousness, which tends to conceal disparities between the conditional distributions $p_\theta(v_\ell \mid v_{1:\ell-1})$. To eliminate this drawback, we use discrete mid-p-values [Lancaster, 1952] instead: given an interaction $\omega = (v_1, \ldots, v_{N_e})$, the mid-$p$-value associated with v_ℓ is defined for $\ell \in \{C+1, \ldots, N_e\}$ as

$$p_{v_\ell | v_{1:\ell-1};\theta} = \frac{1}{2}\Bigg\{ \sum_{v \in \mathcal{U}_\ell} \mathbb{1}_{\{p_\theta(v|v_{1:\ell-1}) < p_\theta(v_\ell|v_{1:\ell-1})\}} p_\theta(v \mid v_{1:\ell-1}) + \sum_{v \in \mathcal{U}_\ell} \mathbb{1}_{\{p_\theta(v|v_{1:\ell-1}) \leq p_\theta(v_\ell|v_{1:\ell-1})\}} p_\theta(v \mid v_{1:\ell-1}) \Bigg\}.$$

Using p-values instead of probabilities allows us to actually measure how unexpected the entity v_ℓ is with respect to $v_{1:\ell-1}$: we consider v_ℓ anomalous if its predicted probability $p_\theta(v_\ell \mid v_{1:\ell-1})$ is not only low, but also lower than the predicted probability of most other candidate entities $v \in \mathcal{U}_\ell$. The use of mid-p-values instead of simple p-values is motivated by the discrete nature of the distribution considered here: in such settings, classic p-values are known to be conservative [Rubin-Delanchy et al., 2018]. Having computed a mid-p-value for each predicted entity $v_{C+1}, \ldots, v_{N_e}$, we then combine them into a unique anomaly score for the interaction ω,

$$\psi_\theta(\omega) = -\sum_{\ell=C+1}^{N_e} \log p_{v_\ell | v_{1:\ell-1};\theta}.$$

Note that this score is equal to half the test statistic associated with Fisher's method for combining independent p-values [Fisher, 1925]. The reason why we use this simple definition of ψ_θ instead of computing the p-value provided by Fisher's method is that we are only interested in ranking interactions from the most to the least anomalous. To that end, using the test statistic instead of the p-value is sufficient.

Having defined an anomaly detection algorithm for homogeneous events, we can now extend it to the heterogeneous setting. This is the goal of the next section, which mostly deals with the challenges arising when learning the model.

4.3 Modelling Heterogeneous Interactions as a Multi-Task Learning Problem

Jointly modelling different event types implies modifying our anomaly detection algorithm in several ways. First of all, the model must obviously be adapted, along

with the underlying mathematical framework. These aspects are addressed in Section 4.3.1. Secondly, these adjustments themselves raise new challenges when training the model. In particular, using shared parameters to model the distribution of several event types raises the issue of possibly conflicting training objectives associated with each type. Dealing with such challenges is the aim of a field of research called multi-task learning, which we briefly survey in Section 4.3.2. Finally, the application of a multi-task learning approach to modelling heterogeneous interactions is discussed in Section 4.3.3, which also summarizes our whole anomaly detection methodology for heterogeneous events.

4.3.1 Extending the Framework to Heterogeneous Interactions

The interactions considered in Section 4.2.5 were described by a single event type (e, N_e, Ω_e). In contrast, we now consider E different event types $\{(e, N_e, \Omega_e)\}_{e=1}^{E}$, with $\Omega_e = (\tau_1^e, \dots, \tau_{N_e}^e)$ for $e \in [E]$. The index τ_ℓ^e denotes the type of the ℓ-th entity involved in a type e event, and each event can thus be written (e, ω), with $e \in [E]$ and $\omega = (v_1, \dots, v_{N_e}) \in \prod_{\ell=1}^{N_e} \mathcal{U}_{\tau_\ell^e}$. Detecting anomalous events then implies modelling the conditional probability

$$p(v_{C_e+1:N_e} \mid e, v_{1:C_e}) = \prod_{\ell=C+1}^{N_e} p(v_\ell \mid e, v_{1:\ell-1}),$$

where $C_e \in [N_e - 1]$ denotes the number of context entities for type e events. Building upon the methodology introduced in Section 4.2.5, we model each of the conditional probabilities $p(v_\ell \mid e, v_{1:\ell-1})$ through a type-specific inner compatibility function $\kappa_\theta^{e,\ell}$, similar to the one defined in Equation 4.9.

In order to build a joint model for several event types, a distinction must be made between shared parameters and type-specific ones. The embeddings of the entities, introduced in Section 4.2.5 and denoted $\{\mathbf{x}_v\}_{v \in \mathcal{U}}$, should intuitively fall into the first category: the embedding of an entity is supposed to encode latent characteristics of its usual behavior, and all event types should contribute to the definition of these characteristics. On the other hand, the weights $\{w_{ij}\}_{i,j}$ from the inner compatibility function defined in Equation 4.9 encode the significance of each pair of involved entities in a given event, thus they should clearly be made type-specific. In addition, we define type-specific latent factors $\{\boldsymbol{\beta}_e \in \mathbb{R}^D\}_{1 \leq e \leq E}$ such that

$$\kappa_\theta^{e,\ell}(v; v_{1:\ell-1}) = \mathbb{1}_{\{v \notin v_{1:\ell-1}\}} \boldsymbol{\beta}_e^\top \left(\sum_{i=1}^{\ell-1} w_{i\ell}^e \mathbf{x}_v \odot \mathbf{x}_{v_i} \right).$$

Intuitively, these type-specific latent factors allow the model to focus on some dimensions of the latent space when predicting events of a specific type, so that two entities can be likely to appear together in an event of one specific type and not the others. To sum up, the parameters of the heterogeneous model are the shared entity embeddings $\{\mathbf{x}_v\}_{v \in \mathcal{U}}$, as well as the type-specific weights $\{w_{ij}^e\}_{i,j,e}$ and latent factors $\{\boldsymbol{\beta}_e\}_{1 \leq e \leq E}$. The respective roles of these parameters are illustrated in Figure 4.5.

Estimating these parameters from a training set $\mathcal{T} = \{(e^i, \omega^i)\}_{i=1}^n$ can still be achieved through NCE, using the procedure described in Section 4.2.5 to define the NCE loss for each event type. In particular, $N_e - C_e$ noise distributions are defined for each event type e, using either the unigram, log-unigram or power-unigram distribution. Denoting $J_\theta^{\mathrm{NCE},e}$ the NCE loss function for a type e event, the aggregate loss

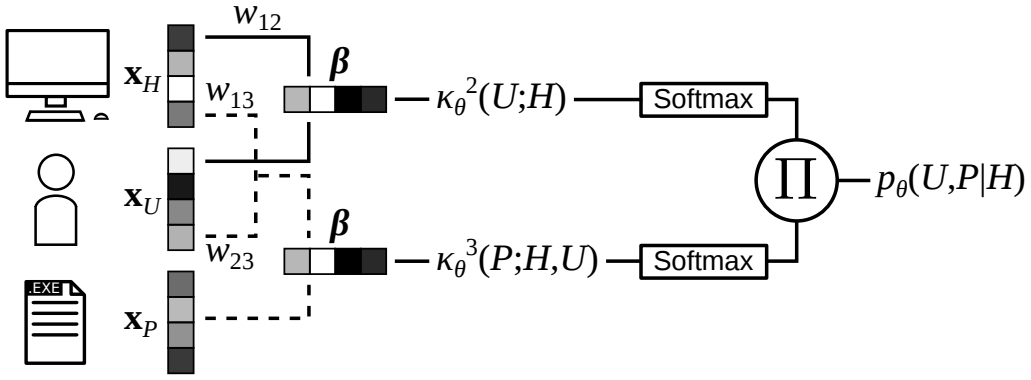


FIGURE 4.5: Computation of the estimated conditional probability $p_\theta(v_{C_e+1:N_e} \mid e, v_{1:C_e})$ for an event (here, a process creation, with $C_e = 1$ and $N_e = 3$). The involved entities are a host H, a user U and a process P.

function for the training set $\mathcal{T}$ can be naively defined as

$$J_\theta(\mathcal{T}) = \frac{1}{n} \sum_{(e,\omega)\in\mathcal{T}} \sum_{e'\in[E]} \mathbb{1}_{\{e=e'\}} J_\theta^{\mathrm{NCE},e}(\omega).$$

However, this definition gives equal importance to all samples, regardless of their type. There is no reason for this to be optimal: indeed, since the model uses some shared parameters for all event types, a given parameter update could decrease the loss for one type and increase it for another. In addition, some event types might be harder to predict on average than others, leading to tendencially higher losses. Therefore, the right balance between the loss functions corresponding to different event types should be defined through a data-driven procedure. Designing such procedures is the main goal of multi-task learning.

4.3.2 A Brief Introduction to Multi-Task Learning

Multi-task learning (MTL) was originally introduced as a way to improve generalization for machine learning algorithms by simultaneously leveraging information pertaining to several related tasks [Caruana, 1997]. Intuitively, jointly learning to perform several related tasks is a way to provide the model with a better inductive bias for each task: in the words of Caruana, *"[t]he multitask bias causes the inductive learner to prefer hypotheses that explain more than one task"*. Transposing this to event log modelling, MTL reshapes the loss function for the entity embeddings by favoring those leading to accurate predictions across all event types.

In his overview of MTL for neural networks [Ruder, 2017], Ruder distinguishes two main approaches to jointly learning multiple tasks, namely hard and soft parameter sharing. While the former uses some shared parameters for all tasks (as well as task-specific parameters), the latter defines a distinct model for each task, then adds regularization terms to the training objective in order to encourage these models to have similar parameters. Our work falls into the first category: the entity embeddings are exactly the same for all event types. Therefore, we focus on hard parameter sharing in the rest of this section.

Consider Z different tasks, each of which can be thought of as learning a mapping $f^z : \mathcal{X} \to \mathcal{Y}^z$, where $\mathcal{X}$ is a shared input space and $\mathcal{Y}^z$ is the output space for task z. Each function f^z is parameterized by a set of shared parameters θ^{sh} as well as a set of task-specific parameters θ^z. In addition, Z task-specific loss functions

$J^1, \ldots, J^Z$ are defined, which typically quantify the respective errors of $f^1, \ldots, f^Z$ on a training dataset $\mathcal{T} = \left\{(x_i, y_i^1, \ldots, y_i^z) \in \mathcal{X} \times \mathcal{Y}^1 \times \ldots \times \mathcal{Y}^Z\right\}_{i=1}^n$. Jointly learning $f^1, \ldots, f^Z$ can then be seen as a multi-objective optimization problem. While some MTL methodologies explicitly rely on multi-objective optimization concepts and tools [Li et al., 2014, Sener and Koltun, 2018], most contributions aim to design an aggregate loss function instead, thereby reverting to a single objective. This aggregate loss function is typically a weighted sum, leading to

$$\left(\theta^{*\mathrm{sh}}, \theta^{*1:Z}\right) \in \underset{(\theta^{\mathrm{sh}}, \theta^{1:Z})}{\arg\min} \sum_{z=1}^{Z} \alpha_z J^z(f^z; \mathcal{T}). \tag{4.11}$$

The main question then lies in the definition of the weights $\{\alpha_z\}_{z=1}^Z$.

A common approach consists in setting uniform weights [Collobert and Weston, 2008, Huang et al., 2014, Sermanet et al., 2014]. However, as mentioned above, there is no reason for this solution to be optimal. Therefore, data driven procedures have also been proposed: Chen et al. [Chen et al., 2018] introduced a dynamic weighting approach, which adaptively tunes $\alpha_1, \ldots, \alpha_Z$ so that the gradients of the rescaled loss functions $\alpha_1 J^1, \ldots, \alpha_Z J^Z$ roughly have the same magnitude. Intuitively, this ensures that no single task dominates the others in terms of influence on the shared parameters. Another approach was proposed by Kendall et al. [Kendall et al., 2018], which we use to train our anomalous event detection model and thus describe in further detail.

The main idea of [Kendall et al., 2018] is to give each task a weight inversely proportional to its intrinsic uncertainty. The intuition behind this heuristic is that the model should put less confidence in information obtained from intrinsically uncertain tasks. Another intuitive interpretation could be that the loss function J^z should return tendencially higher values if the corresponding input-output relationship is noisy, thus dividing it by an estimate of the noise should rescale it appropriately. More formally, we consider a classification setting, meaning that $\mathcal{Y}^z = [M_z]$ for all $z \in [Z]$, with M_z the number of existing classes for task z. For each $z \in [Z]$, the output y^z associated with input $x \in \mathcal{X}$ is then assumed to follow a Boltzmann distribution,

$$\mathbb{P}[y^z = m \mid \mathbf{g}^z(x), \sigma_z] \propto \exp\left(\frac{1}{\sigma_z^2} \mathbf{g}^z(x)_m\right), \tag{4.12}$$

where $\mathbf{g}^z : \mathcal{X} \to \mathbb{R}^{M_z}$ computes the (unscaled) logit probabilities corresponding to the M_z classes, $\mathbf{g}^z(x)_m$ is the m-th coordinate of $\mathbf{g}^z(x)$ and $\sigma_z > 0$ is the intrinsic uncertainty of task z. In other words, for a given function $\mathbf{g}^z$, the value of σ_z determines how flat the distribution $p\big(y^z \mid \mathbf{g}^z(x), \sigma_z\big)$ is. The negative log-likelihood for a given sample $(x, y^1, \ldots, y^Z)$ is then given by

$$\begin{aligned}
\mathrm{NLL} &= -\log p(y^1, \ldots, y^Z \mid x, \sigma_{1:Z}) \\
&= -\sum_{z=1}^{Z} \left\{ \frac{1}{\sigma_z^2} \mathbf{g}^z(x)_{y^z} - \log\left(\sum_{m=1}^{M_z} \exp\left(\frac{1}{\sigma_z^2} \mathbf{g}^z(x)_m\right)\right)\right\} \\
&= \sum_{z=1}^{Z} \frac{1}{\sigma_z^2} H\big(\mathbf{g}^z(x), y^z\big) + \sum_{z=1}^{Z} \log \frac{\sum_{m=1}^{M_z} \exp\left(\frac{1}{\sigma_z^2} \mathbf{g}^z(x)_m\right)}{\left(\sum_{m=1}^{M_z} \exp\left(\mathbf{g}^z(x)_m\right)\right)^{\frac{1}{\sigma_z^2}}},
\end{aligned} \tag{4.13}$$

where $H\big(\mathbf{g}^z(x), y^z\big)$ is the cross-entropy loss for the unscaled function $\mathbf{g}^z$, defined as

$$H\big(\mathbf{g}^z(x), y^z\big) = -\log \frac{\exp\big(\mathbf{g}^z(x)_{y^z}\big)}{\sum_{m=1}^{M_z} \exp\big(\mathbf{g}^z(x)_m\big)}.$$

A parallel appears between Equation 4.13 and Equation 4.11: the negative log-likelihood derived from Equation 4.12 can actually be seen as a linear combination of cross-entropy losses. This justifies using the inverse intrinsic uncertainties σ_z^{-1} as weights for the task-specific loss functions. Note that an additional term appears in Equation 4.13,

$$R(x) = \sum_{z=1}^{Z} \log \frac{\sum_{m=1}^{M_z} \exp\left(\frac{1}{\sigma_z^2}\mathbf{g}^z(x)_m\right)}{\left(\sum_{m=1}^{M_z} \exp\left(\mathbf{g}^z(x)_m\right)\right)^{\frac{1}{\sigma_z^2}}},$$

which Kendall et al. propose to replace with $\sum_{z=1}^{Z} \log \sigma_z$. The underlying approximation is

$$\frac{1}{\sigma_z} \sum_{m=1}^{M_z} \exp\left(\frac{1}{\sigma_z^2}\mathbf{g}^z(x)_m\right) \approx \left(\sum_{m=1}^{M_z} \exp\left(\mathbf{g}^z(x)_m\right)\right)^{\frac{1}{\sigma_z^2}},$$

which is valid for σ_z in the neighborhood of 1.

We now come back to the original problem, namely learning a set of classifiers $\{f^z : \mathcal{X} \to [M_z]\}_{1 \leq z \leq Z}$ using a training set $\mathcal{T} = \big\{(x_i, y_i^1, \ldots, y_i^z)\big\}_{i=1}^n$. Assuming that each classifier is defined as

$$f^z(x) = \operatorname*{arg\,max}_{m \in [M_z]} \mathbf{g}^z_{\theta^{\mathrm{sh}}, \theta^z}(x)_m,$$

with $\mathbf{g}^z_{\theta^{\mathrm{sh}}, \theta^z} : \mathcal{X} \to \mathbb{R}^{M_z}$ an auxiliary function, the aggregate loss can be defined as

$$J\left(f^{1:Z}; \mathcal{T}\right) = \sum_{i=1}^{n} \sum_{z=1}^{Z} \left\{ \frac{1}{\sigma_z^2} H\left(\mathbf{g}^z_{\theta^{\mathrm{sh}}, \theta^z}(x_i), y_i^z\right) + \log \sigma_z \right\}. \tag{4.14}$$

The task-related uncertainties $\sigma_1, \ldots, \sigma_Z$ are not known a priori. Therefore, they are treated as parameters of the model and learned along with θ^{sh} and $\theta^{1:Z}$. The additional term $\log \sigma_z$ in Equation 4.14 can then be interpreted as a regularization term preventing σ_z from going to infinity, which would lead task z to be entirely ignored. By including this regularization term, the model is thus forced to effectively learn all tasks.

This MTL methodology is both simple and well justified, which leads us to apply it to our heterogeneous event model. Practical details are described in the next section.

4.3.3 Application – Modelling Heterogeneous Interactions

Learning entity embeddings $\{\mathbf{x}_v\}_{v \in \mathcal{U}}$ as well as event type-specific latent factors $\{\boldsymbol{\beta}_e\}_{1 \leq e \leq E}$ and weights $\{w_{ij}^e\}_{i,j,e}$ through NCE can be seen as a multitask binary classification problem: for each event type e, the NCE loss $J_\theta^{\mathrm{NCE},e}$ is essentially a sum of $N_e - C_e$ binary cross-entropy losses, each of which can be thought of as a task-specific loss function. Therefore, Kendall et al.'s methodology [Kendall et al., 2018] can be applied by defining $\sigma_{e,\ell} > 0$ as the intrinsic uncertainty associated with predicting the ℓ-th entity involved in a type e event. The loss function for a type e

event can then be redefined as

$$J_\theta^{\mathrm{MTL},e}(\omega) = \sum_{\ell=C_e+1}^{N_e} \left\{ \frac{1}{\sigma_{e,\ell}^2} J_{\theta,\ell}^{\mathrm{NCE},e}(\omega) + \log \sigma_{e,\ell} \right\}.$$

In summary, our anomaly detection model for heterogeneous events is as follows. Given an event (e, ω), with $e \in [E]$ and $\omega = (v_1, \ldots, v_{N_e}) \in \prod_{\ell=1}^{N_e} \mathcal{U}_{\tau_\ell^e}$, we model the conditional probability $p(v_{C_e+1:N_e} \mid e, v_{1:C_e})$ as

$$p_\theta(v_{C_e+1:N_e} \mid e, v_{1:C_e}) = \prod_{\ell=C_e+1}^{N_e} p_\theta(v_\ell \mid e, v_{1:\ell-1}),$$

with each of the $N_e - C_e$ factors defined as

$$p_\theta(v_\ell \mid e, v_{1:\ell-1}) = \frac{\exp\left(\kappa_\theta^{e,\ell}(v_\ell; v_{1:\ell-1})\right)}{\sum_{v \in \mathcal{U}_{\tau_\ell^e}} \exp\left(\kappa_\theta^{e,\ell}(v; v_{1:\ell-1})\right)}.$$

The inner compatibility function $\kappa_\theta^{e,\ell}$ is given by

$$\kappa_\theta^{e,\ell}(v; v_{1:\ell-1}) = \mathbb{1}_{\{v \notin v_{1:\ell-1}\}} \boldsymbol{\beta}_e^\top \left(\sum_{i=1}^{\ell-1} w_{i\ell}^e \mathbf{x}_v \odot \mathbf{x}_{v_i} \right),$$

and the parameters (entity embeddings $\{\mathbf{x}_v\}_{v \in \mathcal{U}}$, event type-specific latent factors $\{\boldsymbol{\beta}_e\}_{1 \le e \le E}$ and weights $\{w_{ij}^e\}_{i,j,e}$) are learned by minimizing a loss function J_θ over a training set $\mathcal{T} = \{(e^i, \omega^i)\}_{i=1}^n$. This loss function is defined using both NCE and MTL: for each event (e, ω), the event-wise loss is given by

$$J_\theta(e, \omega) = \sum_{\ell=C_e+1}^{N_e} \left\{ \frac{1}{\sigma_{e,\ell}^2} J_{\theta,\ell}^e(v_\ell; v_{1:\ell-1}) + \log \sigma_{e,\ell} \right\},$$

with $\{\sigma_{e,\ell}\}_{e,\ell}$ a set of additional parameters of the model. Each entity-wise loss function is defined as

$$\begin{aligned} J_{\theta,\ell}^e(v_\ell; v_{1:\ell-1}) = \log \sigma \left(\kappa_\theta^{e,\ell}(v_\ell; v_{1:\ell-1}) - \log K Q_\ell^e(v_\ell \mid v_{1:\ell-1}) \right) \\ + \sum_{k=1}^{K} \log \sigma \left(\log K Q_\ell^e(\tilde{v}_k \mid v_{1:\ell-1}) - \kappa_\theta^{e,\ell}(\tilde{v}_k; v_{1:\ell-1}) \right), \end{aligned}$$

with K the number of negative samples, $\tilde{v}_k$ the k-th negative sample and Q_ℓ^e a conditional noise distribution of the form

$$Q_\ell^e(v \mid v_{1:\ell-1}) = \mathbb{1}_{\{v \notin v_{1:\ell-1}\}} \frac{\gamma_\ell^e(v)}{\sum_{v' \in \mathcal{U}_{\tau_\ell^e} \setminus v_{1:\ell-1}} \gamma_\ell^e(v')}$$

for all $v \in \mathcal{U}_{\tau_\ell^e}$ and $v_{1:\ell-1} \in \prod_{j=1}^{\ell-1} \mathcal{U}_{\tau_j^e}$, with $\gamma_\ell^e : \mathcal{U}_{\tau_\ell^e} \to \mathbb{R}_+$ some importance-quantifying function. The global loss function

$$J_\theta(\mathcal{T}) = \frac{1}{n} \sum_{i=1}^{n} J_\theta(e^i, \omega^i)$$

is minimized through stochastic gradient descent (SGD). More specifically, we use the Adam algorithm [Kingma and Ba, 2015], which is a standard choice for complex machine learning models with many parameters.

At the end of the training phase, an estimate $\hat{\theta}$ of the model's parameters is available and can be used to detect anomalous events. Given an event (e, ω), its anomaly score is defined as

$$\psi_{\hat{\theta}}(e, \omega) = -\sum_{\ell=C_e+1}^{N_e} \log p_{v_\ell|e,v_{1:\ell-1};\hat{\theta}},$$

where $p_{v_\ell|e,v_{1:\ell-1};\hat{\theta}}$ is the mid-p-value associated with the ℓ-th involved entity,

$$p_{v_\ell|e,v_{1:\ell-1};\hat{\theta}} = \frac{1}{2}\Bigg\{ \sum_{v \in \mathcal{U}_\ell} \mathbb{1}_{\{p_{\hat{\theta}}(v|e,v_{1:\ell-1}) < p_{\hat{\theta}}(v_\ell|e,v_{1:\ell-1})\}} p_{\hat{\theta}}(v \mid e, v_{1:\ell-1}) + \sum_{v \in \mathcal{U}_\ell} \mathbb{1}_{\{p_{\hat{\theta}}(v|e,v_{1:\ell-1}) \leq p_{\hat{\theta}}(v_\ell|e,v_{1:\ell-1})\}} p_{\hat{\theta}}(v \mid e, v_{1:\ell-1}) \Bigg\}.$$

Note that in order to make anomaly scores comparable across event types, we standardize them as follows: at the end of the training phase, the mean and standard deviation of $\psi_{\hat{\theta}}(e, \omega)$ are estimated on the training set for each event type $e \in [E]$. The obtained estimates are denoted $\{(\hat{\mu}_{\psi,e}, \hat{\sigma}_{\psi,e})\}_{1 \leq e \leq E}$. The standardized anomaly scoring function is then defined as

$$\tilde{\psi}_{\hat{\theta}}(e, \omega) = \frac{\psi_{\hat{\theta}}(e, \omega) - \hat{\mu}_{\psi,e}}{\hat{\sigma}_{\psi,e}}.$$

This anomaly detection methodology is empirically evaluated in the next section.

4.4 Experiments

Having defined our anomaly detection algorithm for heterogeneous events, we now evaluate its ability to detect malicious activity in real-world data. Section 4.4.1 describes our experimental setup, including the training and test datasets, the other algorithms used as baselines and the performance metrics. Results are then presented and discussed in Section 4.4.2.

4.4.1 Experimental Setup

We implement our model in Python 3.9, mostly using PyTorch [Paszke et al., 2019]. Experiments are run on a Debian 10 machine with 128GB RAM and a Tesla V100 GPU. Some hyperparameters must be tuned manually, namely the latent space dimension D and the number of negative samples K. The best detection performance is obtained for $D = 64$ and $K = 10$, and these values are thus used for all experiments unless otherwise specified. Besides, the influence of these hyperparameters on the effectiveness of the model is studied in further detail in Section 4.4.2.

As mentioned above, the Adam algorithm is used for training. We set the batch size to 5 000 samples and the number of training epochs to 30. As for the learning rate, we use the default value of 10^{-3}.

TABLE 4.1: Number of entities of each type in the reduced LANL dataset.

Name	Arity (train)		Arity (test)		Arity (all)	
	Total	Malicious	Total	Malicious	Total	Malicious
User	12 164	7	10 767	71	13 176	76
Host	12 264	27	11 943	234	13 090	255
Authentication type	23	1	21	1	24	1
Process	1566	0	1531	0	1593	0

TABLE 4.2: Number of events of each type in the reduced LANL dataset.

Event type	# Total (# malicious)	
	Train	Test
Local authentication	3 418 117 (0)	2 173 349 (0)
Remote authentication	13 198 597 (50)	8 310 963 (473)
Process creation	4 949 066 (0)	2 758 106 (0)

Dataset. The LANL dataset, described in Section 2.4, is used for our experiments. The first 8 days are used for training, and the next 5 days make up the test set. The preprocessing described in Section 2.4.1 (definition of the event types, exclusion of some events) is also applied here. In addition, we replace process names appearing less than 40 times in the whole dataset with a "Rare Process" token, as process names involved in too few events cannot be modelled reliably. Entity and event counts for this reduced dataset are given in Table 4.1 and Table 4.2, respectively.

The formal definition of the event types entails some modelling assumptions. In particular, the order in which entities are predicted should be defined based on domain-specific knowledge: intuitively, some entities are more informative than others. For instance, knowing which user is involved in a given logon event presumably restrains the number of hosts likely to appear in this event. In contrast, knowing only the authentication type leaves more possibilities open. Using such rules of thumb, we sort the involved entities for each event type as follows:

- Local authentication: user, authentication type, host.
- Remote authentication: user, authentication type, source host, destination host.
- Process creation: host, user, process name.

Besides, we set the number of context entities C_e to 1 for each event type. Note that we did not experiment with other specifications of the event types for lack of time. However, modifying the order of involved entities could maybe lead to better results, and experimenting in this direction is thus an interesting lead for future work.

Baselines. We compare our algorithm with three previously published methods. The first one is CADENCE [Amin et al., 2019], which is a rather obvious choice since our work is primarily inspired by this model. The language model of Tuor et al. [Tuor et al., 2018] (referred to as W-BEM) is also included. W-BEM represents events as discrete sequences of entities, then models them using Long Short-Term Memory (LSTM [Hochreiter and Schmidhuber, 1997]) networks. The third baseline is the graph-based model of Bowman et al. [Bowman et al., 2020], described in Section 4.2.2

and hereafter referred to as GraphAI. Note that all three baselines focus on homogeneous events. Therefore, while we train our model on both authentication and process creation events, other models are trained on authentications only (both local and remote). The open-source implementation of W-BEM is used for the experiments. We reimplemented GraphAI based on [Bowman et al., 2020] and further precisions obtained from the authors. Finally, Cadence was implemented based only on [Amin et al., 2019] since the authors did not respond to our requests for code. As for hyperparameters, we reuse those provided in the corresponding papers, except when different values give better results.

Performance metrics. Two criteria are used to evaluate detection performance. The first one is the Receiver Operating Characteristic (ROC) curve, which plots the detection rate on the whole test dataset as a function of the false positive rate (FPR) for varying detection thresholds. Due to the considerable volume of real-life event streams, only the leftmost part of the ROC curve (FPR$\leq$1%) is considered: with millions of events generated daily, a 1% false positive rate already amounts to tens of thousands of false positives to investigate every day. Therefore, detection thresholds leading to a greater FPR can be considered irrelevant. The second metric is the detection rate for a fixed daily budget B of investigated events, denoted DR-B. It gives a more realistic view of what could actually be detected when monitoring activity in an enterprise network. Note that since only authentications are labelled as benign or malicious in the LANL dataset, these two criteria are always computed only for authentication events from the test set.

Since all considered algorithms give non-deterministic results (mostly because of random parameter initialization and stochastic optimization procedures), each of them is run 20 times. We then report the mean and 95% confidence interval (obtained through a nonparametric bootstrap procedure [Efron, 1987]) for each metric.

4.4.2 Results and Discussion

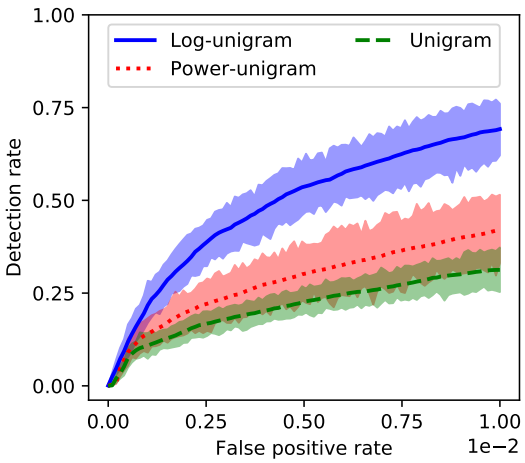
Besides comparing our methodology with previously published ones, our experiments should help us make some choices: as mentioned above, the number of negative samples K and the latent space dimension D need to be tuned in order to maximize detection performance. In addition, a type of noise distribution must be selected to perform NCE. Training our model on real-world data also allows us to check whether the obtained parameters seem plausible with respect to domain knowledge. All these matters are discussed in the next paragraphs.

Influence of the noise distribution. We start with the choice of a noise distribution for NCE. Recall that we consider distributions of the form given in Equation 4.10, defined through an importance-quantifying function $\gamma_\ell^e : \mathcal{U}_\ell \to \mathbb{R}_+$ for each event type $e \in [E]$ and predicted entity index $\ell \in \{C_e + 1, \dots, N_e\}$. In our experiments, we compare three noise distributions: the first one is the unigram distribution, defined through $\gamma_\ell^e(v) = N_\ell^e(v; \mathcal{T})$, with $N_\ell^e(v; \mathcal{T})$ the number of events of type e in the training set $\mathcal{T}$ whose ℓ-th involved entity is v. In addition to this classic choice, we consider two variants: the log-unigram distribution, defined through $\gamma_\ell^e(v) = \log\big(1 + N_\ell^e(v; \mathcal{T})\big)$, and the power-unigram distribution, corresponding to $\gamma_\ell^e(v) = N_\ell^e(v; \mathcal{T})^\alpha$, with $\alpha \in (0, 1)$. Here, we set $\alpha = \frac{3}{4}$.

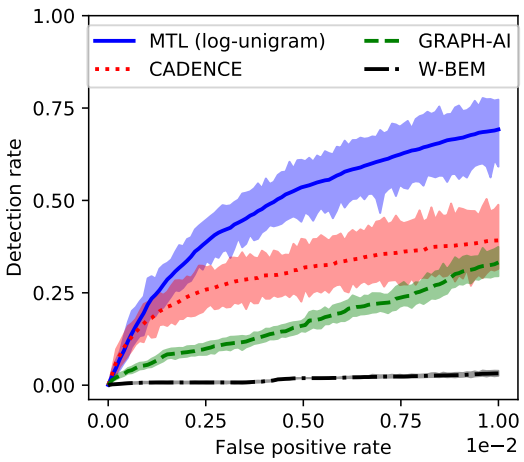
The results obtained with these three noise distributions are displayed in Figure 4.6a and Table 4.3. The log-unigram distribution globally outperforms the others, with the unigram distribution performing the worst. This confirms that the unigram

TABLE 4.3: Detection performance obtained on the LANL dataset. Our method (denoted MTL) is evaluated with three different noise distributions: unigram, log-unigram and power-unigram (with the exponent $\alpha = .75$). The best score for each metric is in bold.

Method	AUC@1%	DR-1K	DR-5K	DR-10K	DR-20K
MTL					
Unigram	.205±.034	.041±.021	.153±.029	.203±.035	.289±.044
Power-unigram	.277±.094	.053±.023	.198±.057	.282±.072	.391±.095
Log-unigram	**.478±0.082**	.081±.035	**.331±.067**	**.481±.081**	**.640±.104**
CADENCE	.291±.072	**.093±.036**	.218±.044	.276±.057	.344±.066
GRAPHAI	.170±.017	.032±.009	.084±.016	.116±.016	.218±.027
W-BEM	.016±.002	.004±.002	.009±.003	.016±.002	.025±.005



(A) Performance of our method with various noise distributions (the exponent $\alpha = .75$ is used for the power-unigram distribution).



(B) Comparison between our method (using the log-unigram noise distribution) and the three baseline algorithms.

FIGURE 4.6: Truncated ROC curves with 95% confidence intervals obtained on the LANL dataset.

distribution, which appears as the default choice, is in fact not optimal for the problem under consideration. As mentioned in Section 4.2.5, this may be explained by its strong unbalancedness when applied to event logs: since most entities rarely appear in normal events, they also rarely appear as negative samples when using the unigram distribution. Therefore, their embeddings are not as well trained as they are under a more balanced noise distribution.

As the log-unigram distribution yields the best results, it is used in all experiments from now on, unless otherwise specified.

Weights learned by the model. Having trained the model, it can be interesting to investigate the obtained parameters and what they reveal about the training dataset. In particular, the weights $\{w_{ij}^e\}$ can be straightforwardly interpreted as a measure of the correlation between pairs of involved entities: a high value of w_{ij}^e means that the i-th entity involved in a type e event is highly significant in predicting the j-th involved entity. Therefore, analyzing these weights can both help assess the relevance of the model and reveal existing correlations in the LANL dataset.

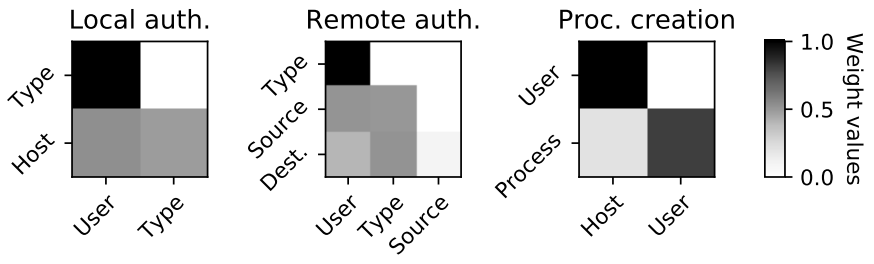
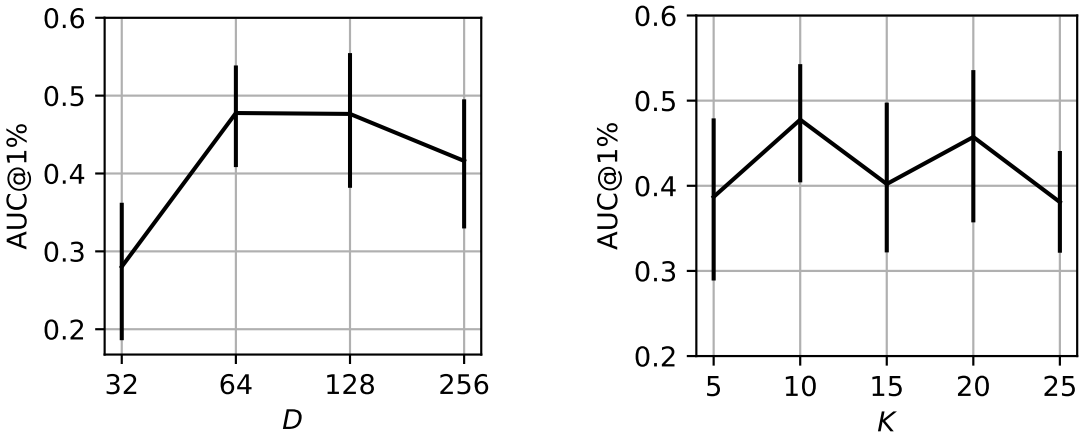


FIGURE 4.7: Weights $\{w_{ij}^e\}$ learned by our model on the LANL dataset, using the log-unigram noise distribution. Vertical labels stand for entities being predicted, while horizontal labels indicate entities being used for the prediction. For instance, when predicting the destination of a remote authentication, the model relies mostly on the authentication type, followed by the user.

The average weights obtained after training are shown in Figure 4.7. While no entity pair stands out in the case of local authentications, things are more contrasted when considering remote authentications. In particular, one rather unexpected outcome is that the destination of a remote authentication is mostly predicted through its type, while little importance is given to the user and almost none to the source. A possible explanation is that the authentication type is correlated with the functional role of the destination host: for instance, most servers should be authenticated to using Kerberos, but some of them might only support NTLM. Similarly, remote desktop sessions might be primarily used by help desk employees to connect to malfunctioning workstations, while servers might be administered using different tools. Finally, the weights associated with process creations suggest that the process name (in other words, the program being executed) is mostly correlated to the user creating the process. This hints towards the fact that each user usually executes a limited number of programs, which are related to the user's role in the organization (e.g. administrative support, information technology (IT) help desk, etc.).

Comparison with the baselines. We now evaluate the detection performance of our methodology with respect to the three baselines introduced in Section 4.4.1. Results are displayed in Figure 4.6b and Table 4.3. Our approach globally outperforms competing algorithms, with significantly higher scores for all but one performance metric, namely the detection rate for a daily investigation budget of 1 000 events. Note that these scores remain rather low: for instance, investigating 10 000 events each day is a considerable amount of work, and it only leads to a 48% detection rate with our method. However, it should be kept in mind that the aim of our algorithm is only to sort events so that the most suspicious ones rank close to the top, and further processing steps should then be applied to obtain more exploitable results. Such additional steps are discussed in Chapters 6 and 7.

As for competing methods' performance, the best contender is CADENCE, which sustains our initial intuition: handling the combinatorial aspect of events through dimensionality reduction and extraction of the most relevant pairs of involved entities is a promising approach to malicious behavior detection in event logs. Next up is GRAPHAI, which only models dyadic interactions instead of fully leveraging the combinatorics of events. This simplified description of events can thus be considered too crude, once again backing up our hypothesis regarding the importance of factoring in the complexity of the data. Finally, W-BEM performs surprisingly poorly, which can be attributed to the inadequate data specification it relies upon. Indeed, the idea behind W-BEM is to treat events as sentences, with each involved entity representing



(A) Influence of the latent space dimension D (with $K = 10$).

(B) Influence of the number of negative samples K (with $D = 64$).

FIGURE 4.8: Area under the truncated ROC curve with 95% confidence interval for several values of the latent space dimension D and the number of negative samples K.

a word. However, this representation ignores the specific structure of event logs: the number of involved entities as well as their types are fixed for each event type, making the actual sample space much smaller than the set of all sequences of entities. In other words, W-BEM gives strictly positive probability to many samples which are actually not events and can thus never happen. This can be expected to make the model less efficient at learning to predict actual events.

Sensitivity analysis. We conclude this section by looking into the impact of the hyperparameters D and K on the detection performance of our algorithm. Several values are tested: 32, 64, 128 and 256 for the latent space dimension D, and 5, 10, 15, 20 and 25 for the number of negative samples per training sample K. As mentioned above, a grid search approach reveals that $D = 64$ and $K = 10$ yield the best detection performance, quantified by the area under the truncated ROC curve. Figure 4.8 shows the variations of this metric around this optimal point.

The latent space dimension D essentially controls the number of parameters of the model. As a consequence, it should be tuned so as to avoid both underfitting and overfitting. Both of these unwanted configurations can be observed in Figure 4.8a, with the lowest and highest values of D both yielding inferior results. As for the number of negative samples K, it relates to the trade-off between the quality of the approximation of the log-likelihood gradient provided by NCE and the computational cost of training. Figure 4.8b shows that increasing K beyond 10 does not make the model better at detecting malicious events. Note that detection performance even tends to decrease for high values of K, which may result from a kind of overfitting: as K increases, the model more accurately fits the distribution of the training data. This might lead it to assign higher anomaly scores to rare but benign events, in turn yielding more false positives.

4.5 Conclusion

Building upon existing work on statistical modelling and anomaly detection for combinatorial data, we propose a new algorithm for anomaly detection in heterogeneous

event logs. We thereby address two of the three main characteristics of event logs identified in Section 2.2.3, namely the combinatorial and heterogeneous aspects.

Regarding the combinatorial aspect, two main challenges stand out: the very high dimensionality of the sample space and the many ways in which a given event can be anomalous. The former is typically addressed through dimensionality reduction techniques, building upon the intuition that the propensity of a given entity to interact with others depends on a small number of latent attributes associated with this entity. As for the latter, it can be dealt with by identifying a small number of relevant pairs of subsets of entities involved in an interaction. What makes these pairs relevant is the degree of correlation they exhibit: if a subset $\mathcal{I}$ of involved entities is usually sufficient to predict another subset $\mathcal{J}$, then looking for rare associations along the pair $(\mathcal{I}, \mathcal{J})$ can be expected to yield relevant anomalies. To the best of our knowledge, the first approach simultaneously implementing these two ideas is the CADENCE model [Amin et al., 2019]. Therefore, we build upon this model to design our own algorithm.

The main improvement we bring is the inclusion of multiple event types – in other words, we also address the heterogeneity of event logs. While this entails no major modification of the model itself, it raises new issues regarding the training procedure. Indeed, using shared parameters to model several event types amounts to jointly optimizing several functions of the same variables, namely the loss functions associated with all types. This multi-objective optimization problem is commonly referred to as multi-task learning, and it requires appropriately balancing the respective importance of each training objective. To that end, we apply an uncertainty-based weighting scheme proposed by Kendall et al. [Kendall et al., 2018].

Experiments on the LANL dataset show the effectiveness of our approach. In particular, our methodology performs better than CADENCE, demonstrating the usefulness of the improvements we propose. A more thorough investigation into the respective impact of each of these improvements would be an interesting lead for future work. However, it seems reasonable to presume that the integration of process creations along with authentications into a joint model has a positive influence on the detection performance, in accordance with previous work on multi-task learning [Caruana, 1997].

While the methodology introduced in this chapter addresses the combinatorial and heterogeneous aspects of event logs, it entirely ignores the temporal one: all events are assumed to be independently sampled from the same distribution. Chapter 5 extends our work to factor in the nonstationarity of real-world event streams. Another important issue left aside in this chapter is the fact that malicious behavior is expected to generate not one, but several anomalous events, as stated in Section 2.3.2. Therefore, computing event-wise anomaly scores is not enough: aggregating and correlating anomalous events should help distinguish truly malicious actions from rare but legitimate ones. This post-processing step is addressed in Chapters 6 and 7.

Chapter 5

Latent Space Modelling for Nonstationary Interaction Streams

Behaviors and activities observed in a computer network are likely to vary over time: from new hosts being added to the network to users taking on new roles or projects, sources of unstability abound. As they reflect these behaviors and activities, event logs exhibit temporal variations as well. Taking this temporal component into account is crucial when designing anomaly detection methods: as the underlying data distribution changes, the model underpinning the anomaly scoring function must adapt to remain relevant. Designing a sensible updating procedure for the model introduced in the previous chapter is thus our main concern in this one. To that end, we draw inspiration from existing work on Bayesian filtering, a generic framework aiming to estimate the hidden state of a dynamical system using noisy measurements. The impact of dynamic updating is assessed empirically, and the resulting anomaly detection methodology, which we call Decades *(dynamic, heterogeneous and combinatorial anomaly detection in event streams), is one of the main contributions of this thesis. Our implementation of* Decades *is openly available.*

5.1 Introduction

The anomaly detection methodology described in Chapter 4 has no temporal dimension: it builds an entity embedding-based model of heterogeneous events using some training set, then leverages this model to quantify the degree of anomalousness of all subsequent events. However, as explained in Section 2.2.3 (and further demonstrated in Section 2.4.2), event logs do exhibit a strong temporal component. This temporal component can be divided into two aspects: nonstationarity of the data generating process and statistical dependencies between successive observations. In this chapter, we focus on the former.

Using a stationary model to detect anomalies in a nonstationary data stream raises obvious reliability concerns: as time goes by, a continuously growing gap appears between the actual data distribution and its estimated counterpart. As a consequence, the notion of anomaly encoded in the model becomes less and less relevant. This phenomenon, commonly referred to as concept drift, has been taken into account by some previous contributions on intrusion detection. As explained in Section 3.3.2, various approaches have been proposed. The simplest one consists in frequently retraining the

model from scratch using only the latest data, which is clearly not optimal: indeed, the data generating process can be expected to evolve in a reasonably slow and smooth fashion. In other words, activity patterns in a computer network are not supposed to change too drastically overnight, thus data collected a few weeks or months ago may still contain some useful information when assessing the normality of current activity.

In order to avoid entirely forgetting past information when updating the model, we need a principled procedure allowing old observations to affect the model's parameters to some extent. Such a procedure should appropriately tune the importance of each observation based not only on how old, but also how reliable it is. Fortunately, adequate tools and concepts already exist: in particular, recursive Bayesian estimation [Bergman, 1999] (also known as Bayesian filtering [Särkkä, 2013]) deals with estimating the state of a dynamical system using both noisy observations and a model of the state's evolution. In our case, the state is the set of entity embeddings and each observation is the set of events happening inside a given time window. Updating the model over time then amounts to estimating the trajectory of the embeddings in the latent space given the observed events. Note that similar ideas have already been applied to anomaly detection in host communication graphs [Lee et al., 2021]. In this context, numerous contributions on dynamic graph analysis can be leveraged, including existing work on recommender systems. However, extending such methods to heterogeneous polyadic interactions is nontrivial, as the indirect statistical dependencies between entity embeddings are much more complex in that setting.

In this chapter, we refine the anomaly detection algorithm presented in Chapter 4 in order to factor in the nonstationarity of event logs. To that end, we draw inspiration from the Bayesian filtering paradigm and design an updating procedure for the entity embeddings, leaving the other parameters fixed. This procedure can be fine-tuned through hyperparameters to achieve a satisfactory trade-off between remembering past information and adapting to the latest observations. Experiments on the LANL dataset give encouraging results regarding the ability of our procedure to alleviate concept drift and decaying detection performance.

The rest of the chapter is structured as follows. We first review some useful concepts pertaining to Bayesian filtering in Section 5.2, starting with the generic framework of hidden Markov models and discussing some existing applications to nonstationary interaction streams. Section 5.3 then presents the extension of our anomaly detection algorithm with a temporal dimension, as well as our Bayesian filtering-inspired updating procedure. Finally, we display and discuss our experimental results in Section 5.4.

5.2 Preliminaries – From Hidden Markov Models to Collaborative Kalman Filters

This section briefly introduces some useful concepts related to latent space modelling for interaction streams. We first explain the core principles of Bayesian filtering, as well as the hidden Markov model framework it relies upon, in Section 5.2.1. Section 5.2.2 then highlights one of the first popular Bayesian filtering approaches to interaction stream modelling, namely the collaborative Kalman filter, and discusses its relevance to our work.

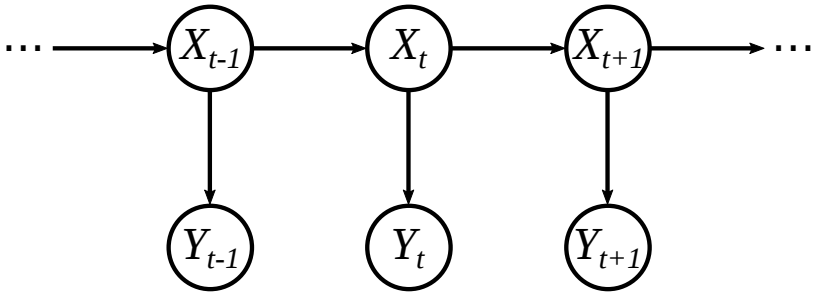


FIGURE 5.1: Graphical model for a hidden Markov model.

5.2.1 Hidden Markov Models and Bayesian Filtering

We start with a simple introduction to hidden Markov models (for a more formal and thorough presentation, see for instance [Cappé et al., 2006]). Given two sample spaces $\mathcal{X}$ and $\mathcal{Y}$, let $\{(X_t, Y_t) \in \mathcal{X} \times \mathcal{Y}\}_{t \geq 1}$ be a bivariate discrete-time stochastic process. Then the pair (X_t, Y_t) follows a hidden Markov model (HMM) if

(i) $\{X_t\}_{t \geq 1}$ is a latent Markov process;

(ii) $\forall t \geq 1$, $\forall x_{1:t} \in \mathcal{X}^t$, $\mathbb{P}\left[Y_t \in A \mid X_{1:t} = x_{1:t}\right] = \mathbb{P}\left[Y_t \in A \mid X_t = x_t\right]$, with $A \subset \mathcal{Y}$ any measurable subset.

This definition is illustrated with a graphical model in Figure 5.1. The random variables $\{X_t\}_{t \geq 1}$ are generally called hidden states, while their counterparts $\{Y_t\}_{t \geq 1}$ are referred to as observations.

A common real-world problem to which HMMs are relevant is dynamic estimation of the internal state of a system given a sequence of noisy measurements and prior knowledge about the process describing the evolution of the state. In particular, the concept of filtering refers to estimation of X_t at each time step t given the sequence of observations $Y_{1:t}$. The Bayesian formulation of this problem relies on a simple recursion: the prior conditional distribution of X_t given $Y_{1:t-1}$ can be obtained as

$$p(x_t \mid y_{1:t-1}) = \int_{x_{t-1} \in \mathcal{X}} p(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1}) \, \mathrm{d}x_{t-1}. \tag{5.1}$$

Assuming that the transition kernel $p(\cdot \mid x_{t-1})$ (or an estimate thereof) is known, this distribution can be estimated by plugging in the posterior conditional distribution of X_{t-1} given $Y_{1:t-1}$ obtained at time $t-1$. The posterior distribution of X_t can then be computed as

$$\begin{aligned} p(x_t \mid y_{1:t}) &= \frac{p(x_t, y_t, y_{1:t-1})}{p(y_{1:t})} \\ &= \frac{p(y_t \mid x_t, y_{1:t-1}) p(x_t \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})} \\ &= \frac{p(y_t \mid x_t) p(x_t \mid y_{1:t-1})}{p(y_t \mid y_{1:t-1})}. \end{aligned}$$

The denominator in the last line,

$$p(y_t \mid y_{1:t-1}) = \int_{x_t \in \mathcal{X}} p(y_t \mid x_t) p(x_t \mid y_{1:t-1}) \, \mathrm{d}x_t,$$

is constant with respect to X_t, leading to

$$p(x_t \mid y_{1:t}) \propto p(y_t \mid x_t)p(x_t \mid y_{1:t-1}). \tag{5.2}$$

Bayesian filtering relies on Equations 5.1 and 5.2 to recursively estimate the posterior distribution of X_t at each time step t. A well-known instance of the Bayesian filtering paradigm is the Kalman filter [Kalman, 1960]: setting $\mathcal{X} = \mathbb{R}^N$ and $\mathcal{Y} = \mathbb{R}^M$, consider the process defined by

$$X_t = \mathbf{A}X_{t-1} + \mathbf{R}\boldsymbol{\eta}_t,$$

$$Y_t = \mathbf{B}X_t + \mathbf{S}\boldsymbol{\nu}_t$$

for all $t \geq 2$. The two fixed matrices $\mathbf{A} \in \mathbb{R}^{N\times N}$ and $\mathbf{B} \in \mathbb{R}^{M\times N}$ are called the state transition matrix and measurement transition matrix, respectively. The randomness comes from the two sequences of i.i.d. multivariate standard centered Gaussian random variables $\left\{\boldsymbol{\eta}_t \in \mathbb{R}^N\right\}_{t\geq 2}$ and $\left\{\boldsymbol{\nu}_t \in \mathbb{R}^M\right\}_{t\geq 2}$, which are scaled through the state noise covariance matrix $\mathbf{R} \in \mathbb{R}^{N\times N}$ and measurement noise covariance matrix $\mathbf{S} \in \mathbb{R}^{M\times M}$, respectively. In this setting, the state vector X_t is Gaussian for all $t \geq 1$ (assuming that X_1 is itself Gaussian), and its mean and covariance matrix can be obtained in closed form. The popularity of the Kalman filter for modelling multivariate dynamic systems motivated its extension to the slightly more complex setting of latent space modelling for interaction streams, which we discuss in the next section.

5.2.2 Application to Interaction Streams: the Collaborative Kalman Filter

Collaborative Kalman filtering finds its roots in the connection between probabilistic matrix factorization (described in Section 4.2.2) and the Kalman filter. Indeed, the rating y_{ij} given by user i to item j can be seen as a noisy measurement of the dot product $\mathbf{U}_i^\top \mathbf{V}_j$ of the corresponding latent factors. Making these latent factors evolve in time according to some linear Gaussian model leads to

$$\mathbf{U}_{i,t} = \mathbf{A}_U\mathbf{U}_{i,t-1} + \mathbf{R}_U\boldsymbol{\eta}^U_{i,t},$$

$$\mathbf{V}_{j,t} = \mathbf{A}_V\mathbf{V}_{j,t-1} + \mathbf{R}_V\boldsymbol{\eta}^V_{j,t},$$

$$y_{ij,t} = \mathbf{U}_{i,t}^\top\mathbf{V}_{j,t} + \nu_{ij,t},$$

with notations chosen so as to highlight the analogy. The corresponding graphical model can be found in Figure 5.2.

An early version of the collaborative Kalman filter (CKF) was proposed by Lu et al. [Lu et al., 2009]. This model, however, assumes that the item latent factor matrix $\mathbf{V}$ is constant in time, allowing each user latent factor $\mathbf{U}_{i,t}$ to be estimated through standard Kalman filtering (with the item latent factors playing the role of the measurement transition matrix). In addition, it uses the identity matrix as state transition matrix for all user latent factors. Note that this amounts to placing a Brownian motion prior on the trajectory of $\mathbf{U}_{i,t}$ in the latent space. Sun et al. [Sun et al., 2012, Sun et al., 2014] then proposed a slightly more complex model, allowing the state transition matrix to differ from the identity. They learn a shared state transition matrix $\mathbf{A}$ for all users, and still treat item latent factors as a constant shared measurement transition matrix, enabling the use of Kalman filtering for inference.

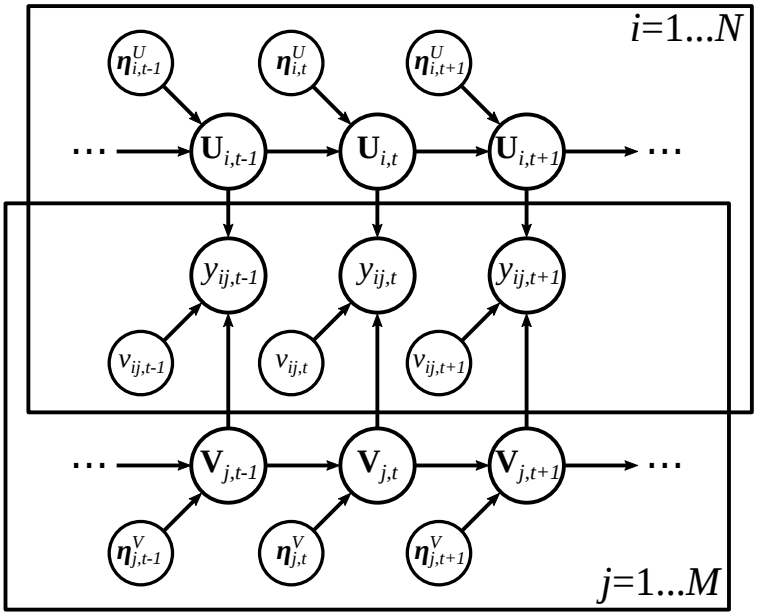


FIGURE 5.2: Graphical model for the collaborative Kalman filter.

To the best of our knowledge, the first fully dynamic model was introduced by Gultekin and Paisley [Gultekin and Paisley, 2014]. In this model, all latent factors are assumed to follow a normal distribution at each time step t, and the posterior means and covariance matrices of the latent factors are jointly estimated whenever new observed interactions are available. The state transition equations are

$$\mathbf{U}_{i,t} = \mathbf{U}_{i,t_i^*} + \alpha(t - t_i^*)\mathbf{I},$$

$$\mathbf{V}_{j,t} = \mathbf{V}_{j,t_j^*} + \alpha(t - t_j^*)\mathbf{I},$$

where t_ℓ^* denotes the timestamp of the latest interaction involving entity ℓ, and $\alpha > 0$ is a hyperparameter controlling the speed at which the latent factors are allowed to change. Given a new interaction $y_{ij,t}$, the posterior distribution of the involved latent factors can then be obtained through Bayesian filtering, yielding

$$p(\mathbf{U}_{i,t}, \mathbf{V}_{j,t} \mid y_{ij,t}) \propto$$
$$p(y_{ij,t} \mid \mathbf{U}_{i,t}, \mathbf{V}_{j,t}) \times \int p(\mathbf{U}_{i,t} \mid \mathbf{U}_{i,t_i^*}) p(\mathbf{U}_{i,t_i^*}) \, \mathrm{d}\mathbf{U}_{i,t_i^*} \times \int p(\mathbf{V}_{j,t} \mid \mathbf{V}_{j,t_j^*}) p(\mathbf{V}_{j,t_j^*}) \, \mathrm{d}\mathbf{V}_{j,t_j^*},$$

with $p(\mathbf{U}_{i,t_i^*})$ (resp. $p(\mathbf{V}_{j,t_j^*})$) denoting the posterior distribution of the latent factor associated with user i (resp. item j) at time t_i^* (resp. t_j^*). However, in contrast with the standard Kalman filter, this posterior is analytically intractable. In [Gultekin and Paisley, 2014], it is thus approximated through mean-field variational inference [Wainwright and Jordan, 2008].

Various improvements were subsequently brought to this version of the CKF: Chang et al. [Chang et al., 2017] factor in the apparition of new users (resp. items) by initializing their latent factors to the mean of existing users' (resp. existing items') latent factors. Some contributions also proposed to use different distributions than the Gaussian in order to better represent implicit feedback: with an approach similar to hierarchical Poisson factorization (see Section 4.2.2), Thac Do and Cao [Thac Do and Cao, 2018] model latent factors through Gamma distributions and ratings $y_{ij,t}$ through

TABLE 5.1: Characterization of statistical models related to collaborative Kalman filtering.

Reference	**Distribution (observations)**	**Distribution (latent factors)**	**State transition prior**
[Lu et al., 2009]	Gaussian	Gaussian (users), Dirac (items)	Isotropic zero-mean additive noise (users), no change (items)
[Sun et al., 2012]	Gaussian	Gaussian (users), Dirac (items)	Linear transformation + isotropic zero-mean additive noise (users), no change (items)
[Sun et al., 2014]	Gaussian	Gaussian (users), Dirac (items)	Linear transformation + isotropic zero-mean additive noise (users), no change (items)
[Gultekin and Paisley, 2014]	Gaussian	Gaussian	Isotropic zero-mean additive noise
[Chang et al., 2017]	Gaussian	Gaussian	Isotropic zero-mean additive noise
[Thac Do and Cao, 2018]	Poisson	Gamma	Hierarchical model
[Liu et al., 2018c]	Bernoulli/logit	Gaussian	Anisotropic zero-mean additive noise
[Song et al., 2019]	Gaussian	Gaussian	Nonlinear transformation + anisotropic zero-mean additive noise
[Lee et al., 2021]	Bernoulli/logit	Gaussian	Anisotropic zero-mean additive noise

Poisson distributions. Alternatively, Liu et al. [Liu et al., 2018c] model implicit feedback through logistic regression, besides introducing more complex (non-isotropic) priors for the trajectory of the latent factors. Finally, Song et al. [Song et al., 2019] further refine the definition of these priors, using neural networks to compute the prior mean and covariance matrix based on the previous posterior estimates.

The CKF and related models provide some insight on dynamic latent space modelling for interaction streams. Although they focus on dyadic interactions, they can thus be used as a source of inspiration. Three specific aspects are of particular interest: the choice of distributions for the latent factors and observations, the definition of the state transition priors and the trade-off between expressivity of the model and tractability of the posterior distribution.

As for the first two aspects, the options explored in the literature are summarized in Table 5.1. Note that we also include the model of Lee et al. [Lee et al., 2021], which is explicitly designed for intrusion detection. While the most frequent approach remains the use of Gaussian latent factors with Brownian motion priors (i.e. isotropic zero-mean additive noise), a trend towards more sophisticated priors can be observed in recent years. Intuitively, the underlying assumption behind such priors is that

the evolution of latent factors should follow some pattern. This assumption may be valid for recommender systems, as user preferences as well as item popularities can be expected to follow some typical trajectories. However, it seems less relevant when modelling event logs: in particular, the amount of available training data can hardly be expected to be sufficient with respect to the number of possible evolution patterns. As an illustration, consider a movie recommendation system: such a system leverages data from a considerable number of users, while the number of typical trajectories (for instance, a user discovering a genre or director and starting to watch more related movies) is arguably limited. In contrast, the number of users in a computer network is significantly lower, while the number of evolution patterns (for instance, an employee being transferred to another department) is not. Therefore, trying to learn deterministic patterns of long-term evolution in the context of computer network monitoring hardly seems relevant. Note, however, that using anisotropic noise as a prior can be considered more useful: indeed, some latent attributes of entities can be expected to change faster than others.

Regarding the expressivity/tractability trade-off, the initial CKF relying on a simple enough model so that the posterior can be obtained in closed form has quickly been replaced by more complex models. The most frequent approach is now to assume that the posterior has to be approximated, which makes expressivity the main objective when designing the model. Note, however, that posterior distributions still cannot be made arbitrarily complex: in order for inference to remain accurate, some simplifying assumptions must be made. In particular, the latent factors of the entities are always assumed to be conditionally independent from one another given the observed interactions, and often to have a diagonal covariance matrix. Under such assumptions, the posterior is then typically approximated through mean-field variational inference.

With these insights in mind, we now come back to modelling event logs and present our parameter updating procedure.

5.3 Handling the Nonstationarity of Event Logs

We first discuss in further detail the causes of real-world event logs' nonstationary nature in Section 5.3.1, distinguishing and formalizing two of them. Section 5.3.2 then presents the refinements brought to our model to factor in this nonstationarity, as well as the updating procedure we use to adapt entity embeddings to observed events. Finally, Section 5.3.3 summarizes the elements of our anomaly detection algorithm for event streams as well as the associated operating procedure.

5.3.1 Two Main Sources of Nonstationarity

In Section 2.4.2, we exhibited some tangible consequences of nonstationarity in the LANL dataset, namely the frequent appearance of previously unseen entities and events[1]. These observations can be explained by two main factors. First of all, new entities appear naturally in the everyday life of a computer network: user accounts are created for new employees, new servers are set up, new software is deployed, and so on. Secondly, existing entities can also change their behavior for legitimate reasons: for instance, users can start working on new projects, and servers can start hosting additional applications.

[1]Recall that an event (t, e, ω) is considered previously unseen if there exists no event (t', e, ω) with $t' < t$ in the available data.

More formally, we divide the time axis into windows of fixed length $\Delta T > 0$ and write

$$\mathcal{L}_T = \left\{ \left(t^k, e^k, \omega^k \right) \right\}_{(T-1)\cdot\Delta T \leq t^k < T\cdot\Delta T}$$

for the set of events observed in the T-th window (assuming that $t = 0$ corresponds to the end of the initial training phase and the beginning of the detection phase). In addition, instead of a unique entity set $\mathcal{U} = \bigcup_{\ell=1}^{L} \mathcal{U}_\ell$ (where L still denotes the number of existing entity types), we now consider a sequence

$$\left\{ \mathcal{U}^T = \bigcup_{\ell=1}^{L} \mathcal{U}_\ell^T \right\}_{T\geq 0},$$

where $\mathcal{U}_\ell^T$ denotes the set of type ℓ entities seen up to time step T. The set of new entities of type ℓ observed in the T-th time window can then be defined as

$$\tilde{\mathcal{U}}_\ell^T = \mathcal{U}_\ell^T \setminus \mathcal{U}_\ell^{T-1}.$$

As no previous activity is available for these entities, assessing the normality of their current activity is difficult. This is commonly referred to as the cold start problem in the recommender systems literature [Schafer et al., 2007]. In our case, the practical consequence of cold start is that a new entity $v \in \tilde{\mathcal{U}}^T$ (where $\tilde{\mathcal{U}}^T = \bigcup_{\ell=1}^{L} \tilde{\mathcal{U}}_\ell^T$) has no embedding $\mathbf{x}_v$ when it is first observed. Therefore, the anomaly score of an event involving v cannot be computed.

As for temporally evolving behaviors of known entities, it corresponds to the idea of concept drift mentioned above: as time passes, the conditional distributions learned during the training phase differ more and more from the true data generating process. Therefore, the parameters of the model must be dynamically adjusted to reduce this difference, which can be done through techniques presented in Section 5.2. Note that we only address long-term drift, leaving seasonal variations aside. For models including a seasonal component, see for instance [Turcotte et al., 2014, Price-Williams et al., 2018, Sanna Passino et al., 2020].

Having defined the two subproblems we aim to address, we now proceed with the definition of our updating procedure.

5.3.2 Adapting the Model through Recursive MAP Estimation

In order to make our model dynamic, we first replace the set of embeddings $\{\mathbf{x}_v\}_{v\in\mathcal{U}}$ with a sequence

$$\left\{ \{\mathbf{x}_v^T\}_{v\in\mathcal{U}^T} \right\}_{T\geq 0}.$$

The first entity set $\mathcal{U}^0$ contains all entities seen during training, and for each entity $v \in \mathcal{U}^0$, its initial embedding $\mathbf{x}_v^0$ is learned through the training procedure described in Section 4.3.3. The goal of the updating procedure is then to compute the new embeddings $\{\mathbf{x}_v^T\}_{v\in\mathcal{U}^T}$ given the old embeddings $\{\mathbf{x}_v^{T-1}\}_{v\in\mathcal{U}^{T-1}}$ and the latest event log $\mathcal{L}_T$, at each time step $T \geq 1$.

First of all, an embedding $\mathbf{x}_v^{T-1}$ must be assigned to each new entity $v \in \tilde{\mathcal{U}}^T$. Following the approach of Chang et al. for dynamic recommender systems [Chang et al., 2017], we initialize this new embedding to the average embedding of same-type

entities: letting τ denote the type of entity v, we set

$$\mathbf{x}_v^{T-1} = \frac{1}{\left|\mathcal{U}_\tau^{T-1}\right|} \sum_{u \in \mathcal{U}_\tau^{T-1}} \mathbf{x}_u^{T-1}.$$

In other words, in the absence of former activity, a new entity is assumed to behave similarly to its peers. This initial embedding is used to compute the anomaly scores of events involving v in $\mathcal{L}_T$. In addition, we use these events to fine-tune v's embedding: we place a Gaussian prior on $\mathbf{x}_v^T$,

$$\mathbf{x}_v^T \sim \mathcal{N}\left(\mathbf{x}_v^{T-1}, \sigma_0^2 \mathbf{I}\right),$$

where $\sigma_0 > 0$ is a hyperparameter. The embedding of v given $\mathcal{L}_T$ can then be obtained through maximum a posteriori (MAP) estimation. We proceed in a similar fashion to update embeddings of previously seen entities: for each entity $u \in \mathcal{U}^{T-1}$, we define the prior distribution of $\mathbf{x}_u^T$ as

$$\mathbf{x}_u^T \sim \mathcal{N}\left(\mathbf{x}_u^{T-1}, \sigma_1^2 \mathbf{I}\right),$$

with $\sigma_1 > 0$ a hyperparameter, and we infer the posterior mean of $\mathbf{x}_u^T$ through MAP estimation. Note that strictly speaking, this is not Bayesian filtering: since we only infer the posterior mean of each embedding instead of its whole posterior distribution, prior distributions at each time step are not defined in a truly recursive manner. This is done for the sake of simplicity, and our experiments show that this simple procedure already helps alleviate concept drift. However, applying an actual Bayesian filtering procedure would be an interesting lead for future work.

These priors can be integrated in our training procedure as follows. Letting $\mathfrak{X}^T$ denote the set of entity embeddings at time step T, the posterior distribution of the embeddings given $\mathcal{L}_T$ is

$$\begin{aligned} p_T\left(\mathfrak{X}^T\right) &= p\left(\mathfrak{X}^T \mid \mathcal{L}_T, \mathfrak{X}^{T-1}\right) \\ &\propto p\left(\{\omega_{C_e+1:N_e}\}_{(t,e,\omega)\in\mathcal{L}_T} \mid \{(e, \omega_{1:C_e})\}_{(t,e,\omega)\in\mathcal{L}_T}, \mathfrak{X}^T\right) p\left(\mathfrak{X}^T \mid \mathfrak{X}^{T-1}\right) \\ &= \prod_{(t,e,\omega)\in\mathcal{L}_T} p\left(\omega_{C_e+1:N_e} \mid e, \omega_{1:C_e}, \mathfrak{X}^T\right) \prod_{v\in\tilde{\mathcal{U}}^T} p\left(\mathbf{x}_v^T \mid \mathbf{x}_v^{T-1}\right) \prod_{u\in\mathcal{U}^{T-1}} p\left(\mathbf{x}_u^T \mid \mathbf{x}_u^{T-1}\right), \end{aligned}$$

where ω_ℓ denotes the ℓ-th entity in ω. The training objective can then be derived by taking the negative logarithm,

$$\begin{aligned} -\log p_T\left(\mathfrak{X}^T\right) = &- \sum_{(t,e,\omega)\in\mathcal{L}_T} \log p\left(\omega_{C_e+1:N_e} \mid e, \omega_{1:C_e}, \mathfrak{X}^T\right) \\ &+ \frac{1}{2\sigma_0^2} \sum_{v\in\tilde{\mathcal{U}}^T} \left\|\mathbf{x}_v^T - \mathbf{x}_v^{T-1}\right\|_2^2 + \frac{1}{2\sigma_1^2} \sum_{u\in\mathcal{U}^{T-1}} \left\|\mathbf{x}_u^T - \mathbf{x}_u^{T-1}\right\|_2^2 + Z, \end{aligned}$$

with Z a constant. This leads to the regularized loss function

$$J_{\mathfrak{X}^T}(\mathcal{L}_T) = -\frac{1}{|\mathcal{L}_T|} \sum_{(t,e,\omega)\in\mathcal{L}_T} \log p\left(\omega_{C_e+1:N_e} \mid e, \omega_{1:C_e}, \mathfrak{X}^T\right) + \lambda_0 R_{\text{new}}(T) + \lambda_1 R_{\text{old}}(T), \tag{5.3}$$

with $\lambda_0 = \frac{1}{2\sigma_0^2|\mathcal{L}_T|}$, $\lambda_1 = \frac{1}{2\sigma_1^2|\mathcal{L}_T|}$ and

$$R_{\text{new}}(T) = \sum_{v \in \tilde{\mathcal{U}}^T} \left\| \mathbf{x}_v^T - \mathbf{x}_v^{T-1} \right\|_2^2, \quad R_{\text{old}}(T) = \sum_{u \in \mathcal{U}^{T-1}} \left\| \mathbf{x}_u^T - \mathbf{x}_u^{T-1} \right\|_2^2.$$

The loss function defined in Equation 5.3 can be minimized through SGD: the gradient of the first term is estimated as described in Section 4.3.3, and the regularization terms can be straightforwardly differentiated. This yields the new embeddings $\boldsymbol{\mathcal{X}}^T$. Note that we do not update the other parameters of the model, as they are supposed to reflect more static properties of the data generating process.

In order to fully specify the learning algorithm, a stopping criterion must be defined. In the initial training phase described in Chapter 4, this stopping criterion was specified as a number of training epochs, which is the standard approach in deep learning. When retraining the model, however, we only fine-tune previously learned parameters. Convergence can thus be expected to happen faster. As a consequence, we define the stopping criterion in terms of variation of the total loss computed on a small validation set: at the end of each training epoch, the relative difference between the current validation loss and the previous one is computed. Retraining stops either if this relative difference is smaller than a threshold $\epsilon > 0$ or if a maximum number of epochs M_{ep} has been reached.

Once the embeddings have been updated, the distribution of the anomaly scoring function $\psi_{\hat{\theta}^T}$ on the sample space may have changed. Therefore, its mean and standard deviation are re-estimated for each event type on the set $\mathcal{L}_T$ before scoring subsequent events using the corresponding standardized anomaly scoring function $\tilde{\psi}_{\hat{\theta}^T}$.

5.3.3 Putting It All Together – The DECADES algorithm

We conclude this section with a summary of our event-wise anomaly detection methodology, which we call DECADES (dynamic, heterogeneous and combinatorial anomaly detection in event streams). Its main steps are the following:

(i) Given a training set $\mathcal{T}$, initial entity embeddings $\boldsymbol{\mathcal{X}}^0$ are learned along with event type-specific latent factors $\{\boldsymbol{\beta}_e\}_{1 \leq e \leq E}$ and weights $\{w_{ij}^e\}_{i,j,e}$ using the procedure described in Section 4.3.3. These initial parameters are denoted $\hat{\theta}^0$.

(ii) At each time step $T \geq 1$, a new event log $\mathcal{L}_T$ is received. Previously unseen entities observed in $\mathcal{L}_T$ are assigned an initial embedding, as described in Section 5.3.2. Each event $(t, e, \omega) \in \mathcal{L}_T$ is then scored using the standardized anomaly scoring function $\tilde{\psi}_{\hat{\theta}^{T-1}}$, and the most anomalous events are reviewed by an expert. If malicious events are found, they are removed from $\mathcal{L}_T$ to avoid polluting the model during the retraining phase.

(iii) New embeddings $\boldsymbol{\mathcal{X}}^T$ are then learned through MAP estimation using events from $\mathcal{L}_T$, yielding a new parameter set $\hat{\theta}^T$ and the corresponding standardized anomaly scoring function $\tilde{\psi}_{\hat{\theta}^T}$.

This procedure is summarized in Algorithm 1. Note that it aims to reduce the workload of human analysts rather than fully automate intrusion detection: as highlighted in Section 4.4.2, the output of our model remains quite noisy, with a lot of legitimate events among the most anomalous ones. However, helping analysts focus on a small fraction of all observed events is already interesting from an operational perspective. The model updating procedure presented in this chapter is essentially

Data: Event stream $\{(t^k, e^k, \omega^k)\}_{k\geq 0}$, initial entity embeddings $\mathfrak{X}^0$
Result: Sequence of anomaly scores $\{y^k\}_{k\geq 0}$
for *time step* $T \geq 1$ **do**
  $\mathcal{L}_T \leftarrow \{(t^k, e^k, \omega^k)\}_{(T-1)\cdot\Delta T \leq t^k < T\cdot\Delta T}$;
  `/* analyze incoming events */`
  for $(t^k, e^k, \omega^k) \in \mathcal{L}_T$ **do**
    `/* look for new entities */`
    for $v \in \omega^k$ **do**
      if $v \notin \mathcal{U}^{T-1}$ **then**
        $\tau \leftarrow \textsc{Type}(v)$;
        $\mathbf{x}_v^{T-1} \leftarrow \frac{1}{|\mathcal{U}_\tau^{T-1}|} \sum_{u\in\mathcal{U}_\tau^{T-1}} \mathbf{x}_u^{T-1}$;
      end
    end
    $y^k = \textsc{ComputeAnomalyScore}(\omega^k, e^k; \mathfrak{X}^{T-1})$;
  end
  remove malicious events from $\mathcal{L}_T$;
  `/* update entity embeddings */`
  $\mathfrak{X}^T \leftarrow \mathfrak{X}^{T-1}$;
  repeat
    $\mathfrak{X}^T \leftarrow \textsc{SgdEpoch}(\mathcal{L}_T, \mathfrak{X}^T)$;
  until *convergence*;
end

Algorithm 1: Analyzing a stream of events using the DECADES algorithm.

meant to make that operational gain achievable in practice: if the only way to alleviate cold start and concept drift was to frequently retrain the model from scratch, the cost-benefit balance of our method would arguably be unfavorable. The usefulness of frequent model updating is empirically demonstrated in the next section.

5.4 Experiments

While Section 4.4 aimed to justify some design choices of our model and compare its performance with state-of-the-art baselines, this section only seeks to evaluate the additional benefit of the updating procedure. To that end, we now consider the first 33 days of the LANL dataset in order to highlight the effect of time, as described in Section 5.4.1. Our results, including the impact of the regularization hyperparameters λ_0 and λ_1, are discussed in Section 5.4.2.

5.4.1 Experimental Setup

The implementation and hardware are the same as in Section 4.4. Our complete implementation of DECADES, including the model presented in Chapter 4 and its updating procedure described in this chapter, is available on GitHub[2]. As for hyperparameters, we still set the latent space dimension D to 64 and the number of negative samples K to 10. The initial training phase is also identical to the one described in Section 4.4.1. The Adam algorithm is used for retraining as well, albeit with different hyperparameters: since the training set and the modifications brought to the embeddings are both

[2]https://github.com/cl-anssi/DECADES

TABLE 5.2: Descriptive statistics on entities appearing in the days 9–33 of the LANL dataset (test set) and not in the first 8 days (training set). The proportions are defined with respect to the corresponding totals in the test set.

		Count (malicious)	**Proportion (malicious)**
Entities unseen during training	User	4 217 (9)	27.9% (9%)
	Host	2 431 (13)	16.9% (4.64%)
	Process	6 038 (0)	79.6% (-)
	Auth. type	4 (0)	14.8% (0%)
Events involving unseen entities	Local auth.	56 031 (0)	0.51% (-)
	Remote auth.	1 599 980 (53)	3.69% (8.14%)
	Proc. creation	495 183 (0)	3.41% (-)

expected to be smaller, we set the batch size to 256 and the learning rate to 10^{-4}. The relative error threshold ϵ and maximum number of epochs M_{ep} specifying the stopping criterion are set to 10^{-2} and 25, respectively.

Dataset. In order to evaluate the impact of cold start and concept drift on detection performance, as well as the effectiveness of our updating procedure at alleviating this impact, we now use the first 33 days of the LANL dataset in our experiments. This larger dataset encompasses all labelled malicious events. The first 8 days are still used for initial training, and each of the next 25 days is treated as one time step T – in other words, the window size ΔT is set to one day. The same preprocessing steps as in Section 4.4.1 are performed, including the formal definition of each event type. Relevant descriptive statistics regarding previously unseen entities in the test dataset are given in Table 5.2.

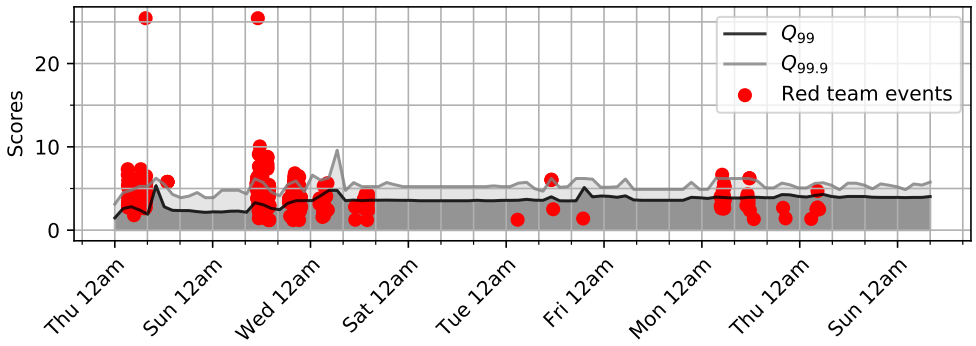
Performance metrics. Since the class imbalance is even stronger in this larger dataset than in Section 4.4, we no longer use the area under the truncated ROC curve (AUC@1%) to evaluate detection performance. Indeed, most days of the test set contain no malicious activity, thus the detection rate for a given daily investigation budget B (DR-B) is more relevant. We still report this metric along with 95% confidence intervals obtained over 20 runs of our algorithm for each evaluated setting. In addition, we track quantiles of the average distribution of anomaly scores over time and compare them with the average scores of malicious events to obtain a more detailed picture of temporal evolution. In what follows, Q_π denotes the π-th percentile of the average distribution of anomaly scores in a given time window, meaning that π% of the events observed within this time window have an average anomaly score smaller than Q_π (where the average is computed over 20 runs of the algorithm). Note that similarly to Section 4.4, all performance metrics are computed for authentication events only since process creations are not labelled.

5.4.2 Results and Discussion

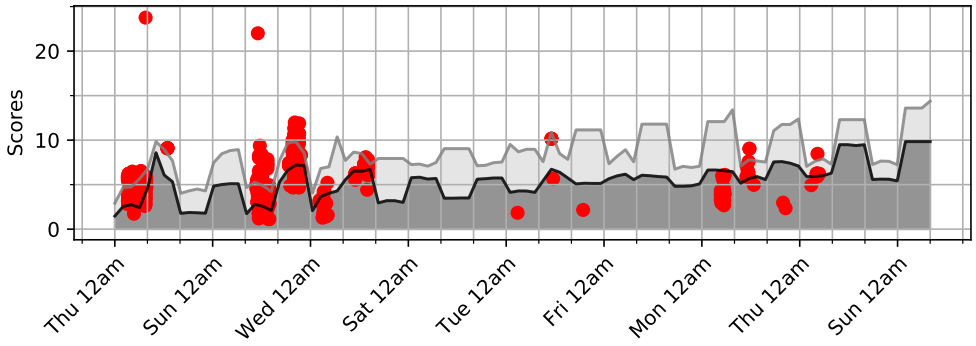
The best detection performance on the LANL dataset is obtained with regularization coefficients $\lambda_0 = 10^{-4}$ and $\lambda_1 = 10$. We first compare the results obtained with these hyperparameter values to those obtained without retraining, then investigate the influence of λ_0 and λ_1 on detection performance.

TABLE 5.3: Performance of DECADES on the days 9–33 of the LANL dataset, with and without retraining at the end of each day. Each metric is reported along with the corresponding 95% confidence interval, with the best score in bold.

	DR-1K	DR-5K	DR-10K	DR-20K
Without retraining	**.050±.027**	.198±.056	.321±.077	.460±.080
With retraining	.047±.015	**.206±.053**	**.334±.071**	**.475±.095**



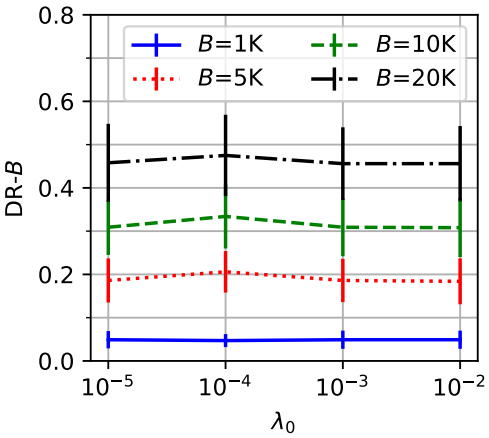
(A) Without retraining



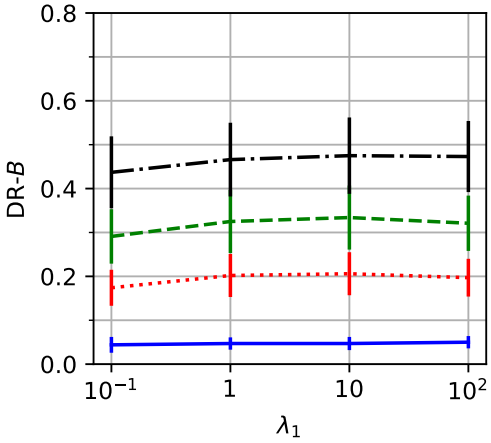
(B) With retraining

FIGURE 5.3: Temporal evolution of the 99th and 99.9th percentiles of the average anomaly score distribution, without any retraining (Figure 5.3a) and with a retraining step at the end of each day (Figure 5.3b).

Impact of frequent updating. The detection rate for an investigation budget B is reported for several values of B in Table 5.3, allowing us to compare detection performance with and without frequent parameter updates. While the difference is small, retraining does seem to slightly improve the obtained results. In order to investigate this difference in further detail, Figure 5.3 displays the temporal evolution of two high-order percentiles of the anomaly score distribution (Q_{99} and $Q_{99.9}$) along with the average scores of red team events. Three main observations can be made: first of all, while some red team events do stand out more when retraining is frequently performed, there are also some days where retraining actually makes malicious events rank lower. Secondly, malicious events are ranked fairly high even without retraining, which can explain the limited performance gain obtained through frequent updates. Note that this does not refute the importance of model updating in general: in a real-world network monitoring setting, malicious events are not guaranteed to happen right after the initial training phase. Therefore, a greater decrease in detection performance can be expected in the absence of retraining. Finally, the two tracked percentiles tend to increase in time with model updating, suggesting that the distribution of anomaly



(A) Influence of the regularization coefficient for new embeddings λ_0 (with $\lambda_1 = 10$).



(B) Influence of the regularization coefficient for old embeddings λ_1 (with $\lambda_0 = 10^{-4}$).

FIGURE 5.4: Detection rate at daily investigation budget B with 95% confidence interval for several values of the budget B and the regularization hyperparameters λ_0 and λ_1.

scores becomes long-tailed. This is an undesirable side effect as it could lead to an increased number of false positives, and it might result from overfitting. Further investigation is needed to precisely understand this temporal drift.

Sensitivity analysis. Finally, we study the impact of the regularization hyperparameters λ_0 and λ_1 on detection performance. Figure 5.4 shows the evolution of the detection rate at various investigation budgets B around what appears to be the optimal point, namely $(\lambda_0, \lambda_1) = (10^{-4}, 10)$. While variations are small and mostly not significant, the regularization coefficient for old embeddings λ_1 does seem to have a greater influence than its counterpart λ_0. In particular, detection performance starts to drop when λ_1 is small, suggesting that the model overfits recent observations and forgets too much about older events. This emphasizes the importance of not putting too much trust in the latest data, which motivates our work in this chapter.

5.5 Conclusion

We extend our event-wise anomaly detection algorithm introduced in Chapter 4 by factoring in the nonstationarity of the underlying data generating process. To that end, we draw inspiration from the Bayesian filtering paradigm, which is designed to gradually incorporate information from noisy observations into a statistical model. More specifically, we borrow from previous work on the collaborative Kalman filter, which extends latent space modelling for dyadic interactions to the dynamic setting.

Our contribution is an anomaly detection methodology for heterogeneous event streams, which we call DECADES. This method includes a model updating procedure handling both the apparition of new entities and the shifting behaviors of already known ones. For the sake of simplicity, this procedure relies on maximum a posteriori estimation at each time step rather than fully-fledged Bayesian filtering. Experiments on the LANL dataset show that even with such a simple procedure, the loss in detection performance resulting from cold start and concept drift can be slightly alleviated. This result is encouraging, and designing a more sophisticated updating procedure is thus a promising research direction.

The most obvious improvement which could be brought to our procedure would be to perform actual Bayesian filtering instead of MAP estimation. This would allow the estimated uncertainty associated with each parameter to be passed on from one time step to the next, potentially leading to a more reliable model. Further refinements include better state transition priors. Indeed, our updating procedure implicitly assumes that all entities are equally likely to change their behavior, which is arguably a crude approximation. A first step could be to use separate regularization hyperparameters λ_0 and λ_1 for each entity type. However, this would make manual tuning of each hyperparameter significantly more intensive, and a more sophisticated hyperparameter tuning procedure would thus be needed as well. Besides, allowing each latent attribute of a given entity to change at its own rate through the use of anisotropic state transition priors could also lead to better parameter updates.

Finally, some of the limitations identified in Section 4.5 still apply here. In particular, the detection performance of our methodology remains rather low, which can be partially attributed to the fact that it detects anomalies at the event granularity only. Considering each event separately from the others does not allow us to leverage a key property of malicious activity, formalized in Assumption 3: events triggered by an intrusion are connected to each other with respect to the entities they involve. It seems reasonable to presume that making use of this assumption could help distinguish malicious events from rare but legitimate ones, and this hypothesis is further explored in Chapters 6 and 7.

Part III

From Noisy Anomalies to Reliable Alerts

Chapter 6

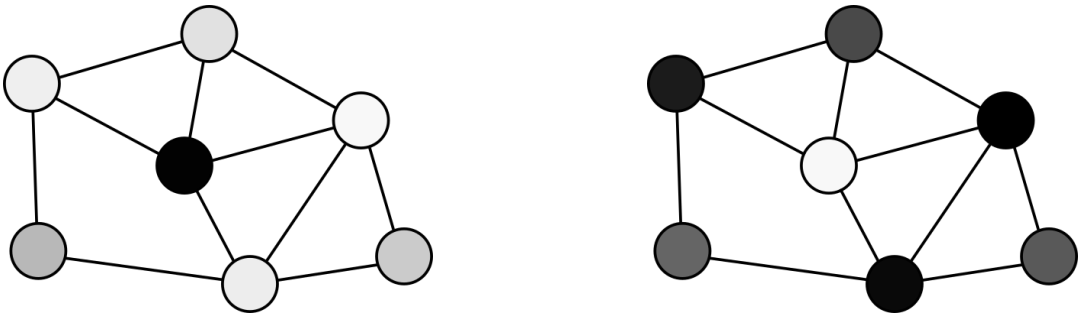
Anomaly Score Denoising through Graph Signal Processing

The statistical model described in Chapters 4 and 5 enables ranking of observed events according to their degree of anomalousness. However, experiments show that this anomaly ranking can still be improved upon: even though malicious events receive rather high anomaly scores, the number of false positives remains unacceptable. A sensible way to make anomaly scores more reliable is to leverage one of our assumptions on malicious behavior: events resulting from an intrusion should involve some shared entities. In this chapter, we introduce the concept of event graph and propose to treat event-wise anomaly scores as a graph-structured signal. Building upon this idea, we use graph signal processing tools to denoise the anomaly scores produced by any event-wise anomaly detection method, taking advantage of the connectivity of malicious event subgraphs. The gain in detection performance provided by this approach is evaluated on event graphs extracted from the LANL dataset.

6.1 Introduction

The main reason why statistical intrusion detection methods for event logs almost always rely on anomaly detection is that viable alternatives do not abound. Indeed, as explained in Section 2.3.2, the vast diversity of malicious TTPs and end goals as well as the somewhat elusive definition of malicious events themselves make direct characterization of intrusion-related activity a daunting task. While anomaly detection methods circumvent these challenges through an indirect characterization of malicious events, they also have their own drawbacks.

More specifically, our experiments in Chapters 4 and 5 show that anomaly detection algorithms for event logs exhibit a rather unfavorable detection rate/false positive rate trade-off: detecting a significant proportion of malicious events comes at the cost of numerous false positives. This low accuracy can be explained by two main causes: first of all, anomaly detection algorithms are not perfect, meaning that they cannot exactly fit the true distribution of the data. Secondly (and unfortunately), even a perfect anomaly detection algorithm would not achieve a perfect detection rate without any false positive, as all anomalous events are not malicious – in fact, not even most of them. Indeed, Assumptions 1 and 2 essentially ensure that all malicious events are anomalous, but there is no reason for the converse to be true: legitimate behavior also yields many unusual events.



(A) False positive (B) False negative

FIGURE 6.1: Event-wise anomaly scores as a graph-structured signal: each vertex stands for an event, and its color represents its anomaly score (darker vertices represent more anomalous events). The vertex at the center of the graph is either a false positive (Figure 6.1a) or a false negative (Figure 6.1b).

To alleviate this fundamental limitation of event-wise anomaly detection, we introduced Assumption 3: anomalous events actually resulting from an intrusion should involve some shared entities. Leveraging this assumption to produce better event-wise anomaly scores is the goal of this chapter. To that end, we introduce the concept of event graph, defined as a graph whose vertices are the events observed in a given time window. Edges of an event graph should encode some notion of similarity so that events generated by an intrusion form a densely connected subgraph. An adequately defined event graph can then be used to eliminate both false positives and false negatives associated with event-wise anomaly scoring, as illustrated in Figure 6.1. More specifically, the main idea explored in this chapter is to treat event-wise anomaly scores as a graph-structured signal. False positives can then be seen as noisy peaks in this signal, while false negatives can be similarly attributed to negative noise compensating an otherwise high "true" score, leading to an isolated low score. Both kinds of errors can thus be eliminated by denoising the signal using graph signal processing tools [Ortega et al., 2018]. Experiments on the LANL dataset show promising results for this approach.

Note that the event-wise anomaly score postprocessing problem addressed here bears a striking resemblance to a twenty-year-old research topic from the information security community, namely IDS alert clustering and correlation [Debar and Wespi, 2001, Valdes and Skinner, 2001]. However, a fundamental difference is that IDS alerts are binary predictions: the mere existence of an alert hints towards suspicious activity, and alert management mostly aims to find root causes and reduce investigation time. In contrast, the event graphs considered here contain a vast majority of benign events, and our goal is to denoise fuzzy suspicions so as to make them more reliable.

The rest of the chapter is structured as follows. We first review relevant contributions pertaining to IDS alert management and event graph analysis in Section 6.2, highlighting their connections to our work. Section 6.3 then describes the construction of our event graphs, and Section 6.4 presents the graph signal processing tools we propose to apply. Finally, the effectiveness of our approach is assessed through experiments on the LANL dataset in Section 6.5.

6.2 Preliminaries – Alert Postprocessing and Event Graphs

The use of graph-related tools to reduce the analyst workload associated with intrusion detection is nothing new. It traces back to several important concepts: the first one is

IDS alert clustering and correlation, which we discuss in Section 6.2.1. More recently, the idea of representing all events obtained by monitoring activity in a given perimeter as an event graph has been gaining traction, and we thus cover it in Section 6.2.2. Finally, Section 6.2.3 discusses the relevance of these two ideas to our work, as well as some other related problems.

6.2.1 Aggregating Binary Alerts: Clustering and Correlation

Most contributions on IDS alert management focus on the following setting: suppose that the traffic (both inbound and outbound) crossing the border of a given network is analyzed by an IDS, resulting in a sequence of alerts $\mathcal{O} = \left\{\mathbf{a}^k\right\}_{1\leq k\leq n}$. Each alert $\mathbf{a}^k$ is described by a timestamp t^k and m categorical attributes $a_1^k, \ldots, a_m^k$, such as the alert type or the source and destination IP addresses and ports. The two main intuitions behind alert postprocessing are the following. First of all, a given incident (such as a port scan or distributed denial of service (DDoS) attack on a specific server) can happen repeatedly, possibly originating from several distinct attackers. Alerts pertaining to these occurrences should be merged to speed up investigation. Secondly, one single incident can be expected to generate several alerts. For instance, an intruder might first launch a port scan on a vulnerable server, then run an exploit on this server and exfiltrate data from the network. Each of these steps may trigger an IDS alert, all of which are received as separate events by the analyst but should actually be investigated together. These two intuitions result in two main approaches to alert aggregation: alert clustering and alert correlation.

As for alert clustering, the central idea is to define a measure of similarity between alerts so that alerts resulting from similar incidents can be identified. Letting S denote such a measure, alert clustering can intuitively be defined as building a partition $\mathcal{A}$ of $\mathcal{O}$ into K subsets $A_1, \ldots, A_K$ such that $S(\mathbf{a}^i, \mathbf{a}^j)$ is high if $\mathbf{a}^i$ and $\mathbf{a}^j$ belong to the same subset and low otherwise. Each alert clustering method can then be characterized by two main elements, namely the similarity measure S and the procedure used to build clusters. Early contributions simply defined similarity as a binary function, each cluster then corresponding to an equivalence class [Cuppens, 2001, Debar and Wespi, 2001, Perdisci et al., 2006]. This approach was quickly overridden by slightly more sophisticated ones, which rely on various definitions of the similarity function: a classic choice is to build S as a weighted sum of attribute-specific similarities, each of which can be defined using domain knowledge [Valdes and Skinner, 2001, Lee et al., 2006, Spathoulas and Katsikas, 2013, Shittu et al., 2014, Haas and Fischer, 2018]. More complex alternatives include defining S through the likelihood of a statistical model [Hofmann and Sick, 2009] or using an expert-specified hierarchy of possible values for each attribute, such that the distance between two alerts can be derived from the position of each of their respective attributes in this hierarchy [Julisch, 2003]. Clusters can then be built through an agglomerative approach [Valdes and Skinner, 2001, Spathoulas and Katsikas, 2013] or by optimizing some global homogeneity measure [Julisch, 2003, Hofmann and Sick, 2009].

While alert clustering looks for a preferably small number of global patterns describing most elements of the alert set $\mathcal{O}$, alert correlation aims to find finer-grained relationships between alerts. This typically boils down to building directed acyclic graphs (DAGs) of alerts, with a directed edge from $\mathbf{a}^i$ to $\mathbf{a}^j$ if $\mathbf{a}^i$ precedes $\mathbf{a}^j$ and both alerts belong to the same incident. Identifying which alerts trace back to one given incident is then the main challenge. The most basic approach consists in manually defining correlation rules using the attributes of each alert [Cuppens and Miege, 2002]. For instance, $\mathbf{a}^i$ and $\mathbf{a}^j$ can be considered correlated if $|t^i - t^j|$ is below a threshold

and the same source and destination IP addresses appear in both alerts. A slightly more generic definition of correlation rules can be obtained by specifying the prerequisites and possible consequences of each alert type [Ning et al., 2002]: for instance, a port scan can help the attacker discover vulnerable services, and exploiting such a vulnerable service requires knowledge of its presence. Therefore, correlation rules can automatically be generated between port scan alerts and exploit-related ones. Finally, the next step in automating creation of correlation rules is entirely inferring them from a training set. Several statistical tools have been used to that end, including Granger causality [Qin and Lee, 2003] and diversely complex models for the conditional probability of observing a specific alert given the previous ones [Shittu et al., 2015, Lin et al., 2018].

6.2.2 Building and Analyzing Event Graphs

The main goal of the tools discussed in the previous section is to reduce the amount of work required to investigate a set of alerts. A natural extension could consist in using similar methods to enrich alerts with further knowledge, or even to generate them. This is the purpose of event graphs.

Instead of looking for similarities or correlations between security alerts only, event graphs keep track of such relationships between all sorts of events, including benign ones. Such graphs can typically be useful for forensic investigation: once a previously undetected intrusion is exposed, exploring the neighborhood of a few known malicious events can help uncover its early steps. This idea can be traced back to early work on system object graphs, where events are interpreted as dependencies between system objects and used to retrieve all entities compromised by an intruder [King and Chen, 2003, King et al., 2005]. As for event graphs, the main question is how to build them: in other words, when should two events be connected by an edge?

Not unlike alert clustering and correlation, event graph construction started with a mostly manual approach relying heavily on expert-defined rules [Pei et al., 2016, Liu et al., 2019]. Such rules specify, for each selected pair of event types, which fields of the two considered events should contain matching values. As a basic example, an outbound network connection event e_1 should be linked to an inbound network connection event e_2 if the source and destination hosts and ports are the same for e_1 and e_2, and the timestamps of e_1 and e_2 are close enough to each other. The obvious limitation of such rules is that defining them is time-consuming. In addition, since they are specific to the considered event types, new rules must be defined when considering additional data sources. Therefore, slightly more generic approaches have also been proposed, relying for instance on a notion of causality between events [Xosanavongsa et al., 2019] or the presence of shared entities involved in several events [Leichtnam et al., 2020a, Leichtnam et al., 2020b].

Having built the event graph, various tools can be used to extract relevant information from it. Anomalous events can be identified using node [Liu et al., 2019] or edge [Leichtnam et al., 2020b] embedding methodologies. The neighborhood of a suspicious event can also be retrieved in order to obtain some contextual information [Xosanavongsa et al., 2019]. A slightly more sophisticated means to that end is the use of community detection algorithms [Pei et al., 2016, Leichtnam et al., 2020a] typically relying on modularity maximization [Newman, 2006]. Interestingly, modularity-based community detection in event graphs relies on the same intuition as our approach: malicious events should be clustered in densely connected subgraphs, which are themselves sparsely connected to the rest of the event graph. However, we harness this intuition in a different way: while previous work typically assumes that

some events are known to be malicious and look for other events related to the same intrusion, we leverage the modular structure of event graphs to make noisy anomaly scores more reliable.

6.2.3 Connections with Graph-Based Anomaly Score Denoising

Beyond the aforementioned parallel between modularity-based community detection and graph-structured signal smoothing, a few interesting ideas can be drawn from the reviewed contributions. First of all, similarly to [Leichtnam et al., 2020a, Leichtnam et al., 2020b], we would like to link events based on the entities they involve. However, while Leichtnam et al. do so by building a heterogeneous graph containing both entities and events, we would rather include events only. In this regard, our approach somewhat resembles alert clustering: edges of our event graphs should represent some notion of similarity between events. To that end, building a pairwise similarity measure as a weighted sum of elementary functions [Valdes and Skinner, 2001, Lee et al., 2006, Spathoulas and Katsikas, 2013, Shittu et al., 2014, Haas and Fischer, 2018] is an interesting approach: intuitively, several factors should be taken into account, such as temporal proximity or the presence of shared entities of different types. Appropriately weighting the influence of each of these factors is then an important step in building the similarity measure.

It should be noted, however, that none of the reviewed contributions addresses the problem of noisy information (such as anomaly scores) associated with events or alerts. Ideas relevant to this setting should thus be sought in other fields. Interestingly, similar problems have been studied in the context of intrusion detection, albeit with different kinds of graphs and scores. Working with host communication graphs, Sexton et al. proposed a recursive p-value aggregation procedure to identify clusters of compromised hosts based on some individual model of normal behavior for each host [Sexton et al., 2013]. A slightly different approach was introduced by Oprea et al., who apply a belief propagation method to bipartite host-domain graphs representing DNS logs in order to iteratively identify compromised hosts [Oprea et al., 2015]. Their main intuition is that domains looked up by a compromised host may be part of a C&C infrastructure, while hosts looking up C&C domains are likely to be compromised. Finally, Roundy et al. use random walks in bipartite host-alert graphs to identify low-severity alerts frequently co-occurring with high-severity ones [Roundy et al., 2017]. This can also be seen as a kind of suspicion propagation based on an underlying graph structure.

As a side note, the study of noisy graph-structured signals has also received some attention in the bioinformatics community. More specifically, known protein-protein and protein-DNA interactions can be used to build networks of genes. Measurements of each gene's expression change under a given perturbation can then be interpreted as a noisy signal over an interaction network [Ideker et al., 2002]. Differentially expressed genes tend to form connected subgraphs in the network, motivating the use of network structure to make measurements more reliable [Chuang et al., 2007]. This approach can typically be formalized using Markov random fields (MRFs) [Wei and Li, 2007]: letting π_v denote the latent state of gene v (differentially expressed or not) and X_v denote its expression change, the joint probability of latent and observed random variables can be modelled as

$$p\left(\{X_v\}_{v\in\mathcal{V}}, \{\pi_v\}_{v\in\mathcal{V}}\right) = \prod_{v\in\mathcal{V}} p(X_v \mid \pi_v) p\left(\pi_v \mid \{\pi_u\}_{u\in N(v)}\right),$$

where $\mathcal{V}$ is the set of genes and $N(v)$ denotes the neighborhood of v in the gene network. In other words, the change of expression of gene v depends only upon its

latent state (differentially expressed or not), which itself depends on the latent states of v's neighbors. A straightforward parallel can be made with event graphs and anomaly scores: the score of an event can be seen as a noisy measurement of its latent state (benign or anomalous), which is positively correlated with the states of its neighbors. The main challenge in MRF-based methods is posterior inference of the latent states, which can for instance be done through Gibbs sampling [Sanguinetti et al., 2008] or genetic algorithms [Klammer et al., 2010].

Having reviewed relevant contributions from various research fields, we can now apply some of the interesting ideas we identified to our specific use case. The first step, discussed in the next section, is to transform event logs into event graphs.

6.3 Building the Event Graph

We first summarize the desired properties of our event graphs in Section 6.3.1. Building upon this specification, we define a pairwise event similarity function in Section 6.3.2 and use it to design our event graph construction procedure in Section 6.3.3.

6.3.1 Goals and Constraints

Let $\mathcal{V} = \{v_1, \dots, v_n\}$ be a set of vertices, with each vertex $v_k \in \mathcal{V}$ corresponding to an event (t_k, e_k, ω_k) (where t_k, e_k and ω_k are the timestamp, event type and set of involved entities, respectively). Besides, for each $k \in [n]$, let $X_k \in \mathbb{R}$ be a random variable representing the anomaly score of the corresponding event. Our goal is to build a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ such that the random vector $\mathbf{X} = (X_k)_{1 \leq k \leq n}$ can be seen as a graph-structured signal over $\mathcal{G}$.

Broadly speaking, the main property $\mathcal{G}$ should exhibit is that malicious events resulting from a given intrusion should form a densely connected cluster. More formally, letting π_k denote the latent state of vertex v_k (benign if $\pi_k = 0$ and malicious if $\pi_k = 1$), we would like π_k to be positively correlated with the latent states of v_k's neighbors. This is commonly referred to as homophily: a given vertex is more likely to connect with other vertices having the same attributes. Assumption 3 suggests that adding an edge between two events v_1 and v_2 if they involve some shared entities should produce homophily. However, an entity appearing in at least one malicious event can still be expected to be involved in a majority of benign events. In addition, assuming that v_1 and v_2 are sampled uniformly at random from $\mathcal{V}$, some entities are more likely to appear in both events than others: for instance, due to their central role in the network, domain controllers are involved in many unrelated events. Therefore, the mere existence of a shared entity should not be considered sufficient.

Such spurious edges are all the more undesirable as the set of events $\mathcal{V}$ is typically large: if the event graph represents one day's worth of data, it can be expected to contain millions of events. The density of $\mathcal{G}$ should thus be kept small to make downstream computations tractable. More generally, time and space complexity are the main constraints here: both building $\mathcal{G}$ and transforming $\mathbf{X}$ based on $\mathcal{G}$'s structure must remain computationally feasible.

6.3.2 Entity-Event Graph and Event Similarity

In order to leverage the existence of shared entities between events, we first build an intermediary graph, namely the bipartite entity-event graph. This graph, denoted

$\mathcal{H} = (\mathcal{U}, \mathcal{V}, \mathcal{E}_{\mathcal{H}})$, has two disjoint set of vertices: the entity set $\mathcal{U}$ and the event set $\mathcal{V}$. The edge set $\mathcal{E}_{\mathcal{H}}$ is then straightforwardly defined as

$$\mathcal{E}_{\mathcal{H}} = \{(u, v_k) \in \mathcal{U} \times \mathcal{V} : \, u \in \omega_k\} .$$

In words, an edge exists between entity u and event v if u is involved in v.

The entity-event graph is used to define a sensible pairwise similarity function for events. Intuitively, two events should be considered similar if they involve several shared entities, and if these entities are not involved in too many events. We formalize this intuition through the notion of random walk: letting $\Upsilon : \mathcal{U} \cup \mathcal{V} \to \mathcal{U} \cup \mathcal{V}$ denote the unbiased random walk operator on $\mathcal{H}$, which takes a vertex as input and returns one of its neighbors sampled uniformly at random, it is clear that

$$\forall (v_i, v_j) \in \mathcal{V}^2, \, \mathbb{P}\left[\Upsilon^2(v_i) = v_j\right] = \frac{1}{d(v_i)} \sum_{u \in N(v_i) \cap N(v_j)} \frac{1}{d(u)},$$

where $d(\cdot)$ denotes the degree of a vertex and $N(\cdot)$ denotes its neighborhood. We then define the (symmetric) entity-wise similarity of v_i and v_j as

$$S_0(v_i, v_j) = \frac{1}{2} \log \left(\frac{\mathbb{P}\left[\Upsilon^2(v_i) = v_j\right] \mathbb{P}\left[\Upsilon^2(v_j) = v_i\right]}{p_{\min}^2} \right),$$

where

$$p_{\min} = \frac{1}{\left(\max_{u \in \mathcal{U}} d(u)\right)\left(\max_{v \in \mathcal{V}} d(v)\right)} \leq \min_{\substack{(v_i, v_j) \in \mathcal{V}^2 \\ N(v_i) \cap N(v_j) \neq \emptyset}} \mathbb{P}\left[\Upsilon^2(v_i) = v_j\right].$$

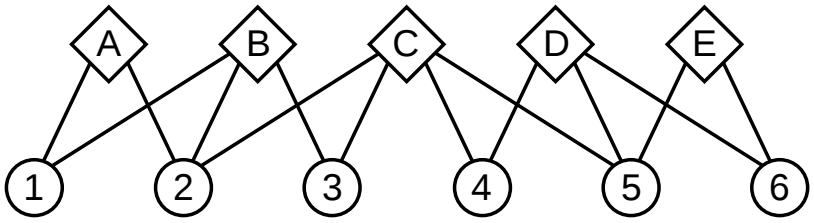
This definition allows us to assign low similarity scores to event pairs whose shared entities only result from random intersections. However, it still lacks one crucial element, namely the influence of time. Indeed, two events can naturally be thought of as more loosely connected if several hours separate them, with respect to the same events happening a few minutes apart from each other. Therefore, the final definition of our pairwise similarity function is

$$S(v_i, v_j) = S_0(v_i, v_j) \exp\left(-\frac{|t_i - t_j|}{\tau} \right), \tag{6.1}$$

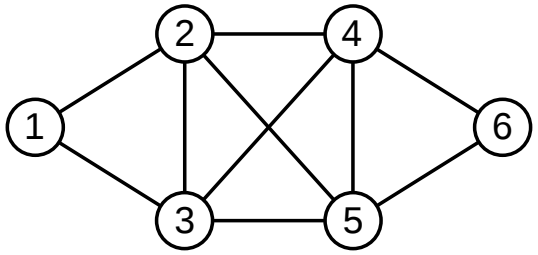
with $\tau > 0$ a hyperparameter.

If computation time and memory usage were no issues, turning $\mathcal{H}$ into an event graph $\mathcal{G}$ could be done by simply taking the unimodal projection[1] of $\mathcal{H}$ onto $\mathcal{V}$, with each edge weighted by the corresponding similarity. However, this is not a viable option: as mentioned in Section 6.3.1, the density of $\mathcal{G}$ must be kept small, which would not be the case when using the unimodal projection. Indeed, for each entity $u \in \mathcal{U}$, the set $N(u)$ of events involving u becomes a clique in the unimodal projection. The latter would thus have too many edges (see Figure 6.2 for an illustration). As a consequence, only highly similar pairs of events should be connected in the event graph. This selection process is discussed in the next section.

[1]The unimodal projection of a bipartite graph $G = (U, V, E)$ onto one of its node sets (say U) is the graph $G_U = (U, E_U)$ where for each $(u_1, u_2) \in U^2$, $(u_1, u_2) \in E_U$ if and only if u_1 and u_2 have a common neighbor in G.



(A) Bipartite entity-event graph



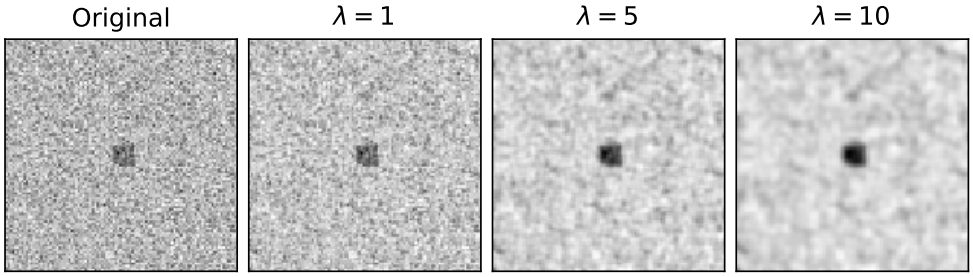
(B) Unimodal projection onto the event set

FIGURE 6.2: A bipartite entity-event graph (Figure 6.2a) and its unimodal projection onto the event set (Figure 6.2b). Entities (resp. events) are denoted A through E (resp. 1 through 6). Notice that the set of events involving one given entity forms a clique in the unimodal projection.

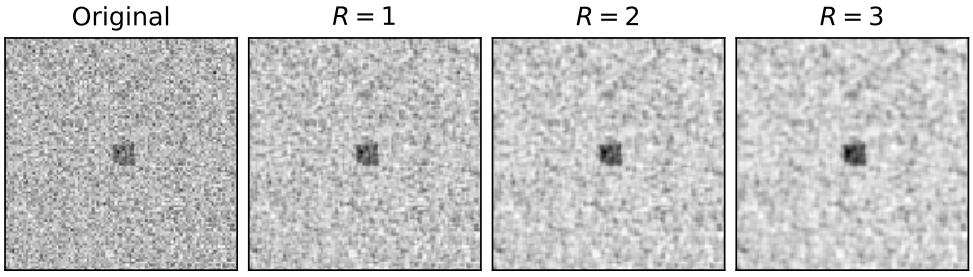
6.3.3 From Event Logs to Event Graphs

Having defined a pairwise similarity function for events, we now need a sensible criterion to decide which event pairs are similar enough to be connected in the event graph. Three possibilities typically come to mind: first, a similarity threshold ζ could be fixed, so that $(v_i, v_j) \in \mathcal{E}$ if and only if $S(v_i, v_j) \geq \zeta$. Finding an adequate threshold is nontrivial as the value of the similarity function has no direct interpretation. The second idea thus consists in fixing ζ so that the density of $\mathcal{G}$ equals some desired value. Finally, a third possibility is to define $\mathcal{G}$ as a K-nearest neighbor graph (K-NNG), meaning that each event is connected to the K most similar others. This last option is the one we implement here. This choice is once again motivated by computational issues: defining a threshold ζ so that the resulting graph has a certain density requires knowing the distribution of the similarity function. A naive approach – namely computing all pairwise similarities – would have $\mathcal{O}(n^2)$ space and time complexity, which is obviously unacceptable. Note that a more sophisticated estimation method based on subsampling could alleviate this issue, but this option is not explored here. In contrast, building a K-NNG only requires $\mathcal{O}(nK)$ memory and can easily be parallelized, while straightforwardly bounding the density of $\mathcal{G}$.

The event graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is thus defined as the weighted K-NNG associated with the similarity function S. Each edge $(v_i, v_j) \in \mathcal{E}$ is assigned a weight equal to $S(v_i, v_j)$. The transformation of an event log $\mathcal{V} = \{v_1, \ldots, v_n\}$ into an event graph $\mathcal{G}$ depends on two hyperparameters, namely the time constant $\tau > 0$ and the number of neighbors $K \in [n]$. The former can be coarsely tuned based on domain knowledge: the intuitive meaning of τ is that the similarity of v_i and v_j is divided by approximately 2.72 when the difference between t_i and t_j becomes τ seconds larger. Therefore, τ should be chosen large enough so that events resulting from a sequence of actions performed by an intruder are less than τ seconds apart from each other. As for the number of neighbors K, it is primarily chosen so as to make computations tractable. Its influence on the structure of the event graph and the effectiveness of graph-based



(A) Heat kernel



(B) Message passing

FIGURE 6.3: Effect of the two chosen denoising tools on a Gaussian graph-structured signal, with several hyperparameter values. The underlying graph is a two-dimensional square lattice. The values of the signal are independent and follow a standard centered normal distribution, except at the center of the lattice where the mean is $\mu = 2$.

anomaly score denoising methods is investigated in Section 6.5.

6.4 Smoothing Signals on the Event Graph

Once an event log and the corresponding anomaly scores have been turned into an event graph and a graph-structured signal, respectively, various tools can be used for anomaly score postprocessing. In this section, we describe two of them, namely the heat kernel (Section 6.4.1) and a message passing scheme inspired by the Weisfeiler-Lehman algorithm (Section 6.4.2). In what follows, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denotes an undirected, weighted graph with n vertices and weighted adjacency matrix $\mathbf{A}$, and $\mathbf{X} = (X_k)_{1 \leq k \leq n}$ is a signal over the vertices of $\mathcal{G}$.

6.4.1 Graph Signal Processing and the Heat Kernel

The first tool we propose to use comes from the field of graph signal processing (GSP). The main goal of GSP is to extend concepts from classical signal processing to graph-structured signals, including for instance frequencies and filters. For a thorough introduction, see for instance [Shuman et al., 2013, Ortega et al., 2018].

More specifically, we consider signal denoising on a graph using the heat kernel. This operator is defined as

$$\mathbf{H}(\lambda) = e^{-\lambda \mathbf{L}} \tag{6.2}$$

for $\lambda \geq 0$, where $\mathbf{L}$ denotes the Laplacian of the graph[2]. This definition finds its roots in the following analogy: suppose that $X_k(t)$ is the temperature at the k-th vertex at time t, and that for each $(i, j) \in [n]^2$, the coefficient of $\mathbf{A}$ at index (i, j) is positive

[2]Recall that the Laplacian of a graph is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{A}$ is the adjacency matrix of the graph and $\mathbf{D}$ is the diagonal matrix whose i-th coefficient is the degree of node i.

and represents the conductivity of the corresponding edge. Then the heat equation

$$\dot{\mathbf{X}}(t) = -\mathbf{L}\mathbf{X}(t)$$

with initial condition $\mathbf{X}(0) = \mathbf{X}$ is solved by $\mathbf{X}(t) = \mathbf{H}(t)\mathbf{X}$ for all $t \geq 0$.

The heat kernel can thus be interpreted as a diffusion operator, with the parameter λ standing for the diffusion time. In a more GSP-oriented perspective, the heat kernel can be seen as a low-pass filter: letting $\lambda_1, \ldots, \lambda_n$ denote the eigenvalues of the Laplacian in ascending order[3] and $\mathbf{e}_1, \ldots, \mathbf{e}_n$ denote the corresponding eigenvectors, the signal $\mathbf{X}$ can be decomposed as

$$\mathbf{X} = \sum_{k=1}^{n} \alpha_k \mathbf{e}_k,$$

with $\alpha_1, \ldots, \alpha_n$ some real coefficients. It follows that

$$\mathbf{H}(\lambda)\mathbf{X} = \sum_{k=1}^{n} e^{-\lambda\lambda_k} \alpha_k \mathbf{e}_k.$$

In other words, the k-th component of $\mathbf{X}$ is scaled by a factor $e^{-\lambda\lambda_k}$. Since the Laplacian is positive semi-definite [Chung, 1997], its eigenvalues are non-negative. Therefore, the k-th scaling factor is a decreasing function of λ_k, meaning that the components of $\mathbf{X}$ associated with large eigenvalues of the Laplacian are attenuated. In practice, this makes the signal smoother, as illustrated in Figure 6.3a.

By smoothing the variations of $\mathbf{X}$ over $\mathcal{G}$, the heat kernel removes isolated peaks, which can also be understood as signal denoising. This is what makes it interesting here: in a case such as those depicted in Figure 6.1, applying the heat kernel to the signal would smooth out false positives and negatives. Note that it has been used for that purpose in the context of gene expression analysis mentioned in Section 6.2.3 [Qiu et al., 2010], which makes it all the more relevant to our problem.

From a computational point of view, the main difficulty lies in the matrix exponential in Equation 6.2. Instead of computing this exponential, Hammond et al. proposed to directly approximate the matrix-vector product $\mathbf{H}(\lambda)\mathbf{X}$ through a truncated polynomial expansion [Hammond et al., 2011]. This approach takes advantage of the sparsity of real-world graphs: the desired product is approximated through repeated sparse matrix-vector multiplication, with a complexity of $\mathcal{O}(|\mathcal{E}| + n)$.

6.4.2 Message Passing and Weisfeiler-Lehman Schemes

The other approach we consider relies on a more local message passing process. It somewhat borrows from the Weisfeiler-Lehman algorithm [Weisfeiler and Lehman, 1968], which was originally proposed as a graph isomorphism test but later inspired many contributions in various graph-related research fields.

The Weisfeiler-Lehman algorithm works as follows: given two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ whose n vertices are assigned discrete labels, higher-order vertex labels are recursively created by aggregating the current label of each vertex with those of its neighbors. More formally, letting $l^{(r)}(v)$ denote the label of vertex v at step r (with $l^{(0)}(v)$ being

[3]Since the Laplacian is a real symmetric matrix, these eigenvalues are real. This also ensures that the corresponding eigenvectors form a basis.

the initially assigned label), its label at step $r+1$ is

$$l^{(r+1)}(v) = \left\{ l^{(r)}(v), \textsc{Sort}\left(\left\{ l^{(r)}(u) \right\}_{u \in N(v)} \right) \right\},$$

where Sort denotes an appropriate sorting procedure ensuring that identical neighborhoods lead to identical label multisets. At each step $r \in [n]$, the sets of order r labels corresponding to the vertices of $\mathcal{G}_1$ and $\mathcal{G}_2$ are compared; if they are different, then $\mathcal{G}_1$ and $\mathcal{G}_2$ are not isomorphic. If no difference has been observed after n steps, then the algorithm cannot tell whether the graphs are isomorphic.

It turns out that the general idea of recursively combining some value assigned to each vertex with some function of the values assigned to its neighbors has numerous applications in network data analysis. The most popular of these applications include the construction of graph kernels [Shervashidze et al., 2011] – in other words, generic pairwise similarity functions for graphs. Another well-known application of this method is Kipf and Welling's Graph Convolutional Network [Kipf and Welling, 2017]. This second example is somewhat related to our work, as Kipf and Welling draw a parallel between their Weisfeiler-Lehman diffusion scheme and linear approximation of spectral graph filters from the GSP world.

Our recursive message passing operator is simply defined as

$$\begin{cases} \mathbf{X}^{(0)} = \mathbf{X} \\ \forall r \in [R],\ \mathbf{X}^{(r)} = \frac{1}{2}\left(\mathbf{M} + \mathbf{I}\right)\mathbf{X}^{(r-1)} \end{cases},$$

where $R \geq 1$ is the total number of iterations, $\mathbf{I}$ is the identity matrix and $\mathbf{M}$ is the row-normalized version of the weighted adjacency matrix $\mathbf{A}$. As illustrated in Figure 6.3b, this transformation also denoises the signal by making it smoother. However, it is slightly less expensive than the heat kernel from a computational perspective: it only requires R sparse matrix-vector multiplications, with R typically less than 5. In contrast, the number of multiplications performed by Hammond et al.'s algorithm [Hammond et al., 2011] is typically a few dozens. Besides, the message passing approach gives more accurate control over the range of information diffusion: when performing R message passing steps, the value of the signal at vertex v cannot affect vertices located more than R hops away from v. This is more easily interpretable than the diffusion time λ associated with the heat kernel.

6.5 Experiments

In order to study the structure of the event graphs generated by the procedure described in Section 6.3 and compare the effectiveness of the denoising operators introduced in Section 6.4, we perform numerical experiments using three days of the LANL dataset, as explained in Section 6.5.1. Our results are displayed and discussed in Section 6.5.2.

6.5.1 Experimental Setup

All algorithms are implemented in Python 3.9, with some intensive parts translated into C using Cython [Behnel et al., 2010]. Regarding the polynomial approximation of the heat kernel, we use the implementation provided in the PyGSP library [Defferrard et al., 2017]. Computations are run on a Debian 10 machine with 128GB RAM and a 2.2GHz, 20-core CPU.

TABLE 6.1: Number of authentication events in the three selected days of the LANL dataset.

Day	Total number of events	Number of malicious events	Proportion δ of malicious events
9	2 593 542	261	.010%
14	2 845 961	75	.0026%
15	2 560 926	25	.00098%

Event graphs. We focus on the authentication events from three days of the LANL dataset, namely the 9th, 14th and 15th ones. These days are chosen because of the number of red team events they contain, which is high, medium and low, respectively (see Table 6.1 for a more detailed description). For each of these three days, we build the event graph for four different values of the number of nearest neighbors K (50, 60, 70 and 80). As for the time constant τ, it is set to half an hour (1 800 seconds).

Synthetic signals. We first assess the effectiveness of our approach using synthetic anomaly scores. These scores are generated as follows: for each event graph $\mathcal{G}_{d,K} = (\mathcal{V}_d, \mathcal{E}_{d,K})$ (corresponding to day $d \in \{9, 14, 15\}$ and number of neighbors $K \in \{50, 60, 70, 80\}$), we sample 100 signals $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(100)}$. For each $i \in [100]$ and $v_k \in \mathcal{V}_d$, we independently sample $X_k^{(i)} \sim \mathcal{N}(\mu_k, 1)$, with

$$\mu_k = \begin{cases} \mu > 0 & \text{if } v_k \text{ is a red team event,} \\ 0 & \text{otherwise.} \end{cases}$$

This operation is repeated for three different values of μ, namely $\mu \in \{1, 2, 3\}$.

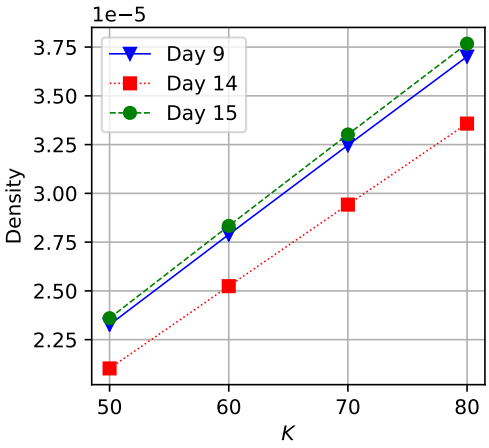
Real anomaly scores. To get a more realistic picture, we also inject real anomaly scores obtained using the DECADES model (presented in Part II) into our event graphs. The scores we use are those computed in Section 5.4 with regularization hyperparameters $\lambda_0 = 10^{-4}$ and $\lambda_1 = 10$. There are thus 20 signals for each of the selected days. Note that we still use authentication events only.

Performance metric. The effectiveness of our approach is evaluated by computing the variation of the area under the truncated ROC curve (AUC@1%) induced by denoising the anomaly scores. This variation is computed separately for each day in order to assess the impact of the number of malicious events.

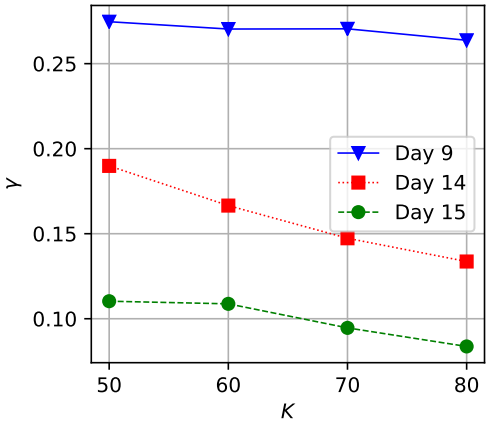
6.5.2 Results and Discussion

The effectiveness of graph-based denoising at improving anomaly scores depends on the structure of the underlying event graph, which we therefore investigate first. Our results for both synthetic and real anomaly scores are presented next.

Characteristics of the event graphs. We start with a simple description of the generated graphs, studying their density and its evolution as K varies. While defining the event graph as a K-NNG ensures that its density is at most $2K/(n-1)$, this upper bound is not necessarily attained: since two given events can both be among the K nearest neighbors of each other, some of the nK generated edges may be duplicates. In addition, some events might have strictly positive similarity with less



(A) Density of the event graph.



(B) Average proportion γ of red team events in the neighborhood of a red team event.

FIGURE 6.4: Characteristics of the event graph for each of the three selected days, with several values of K.

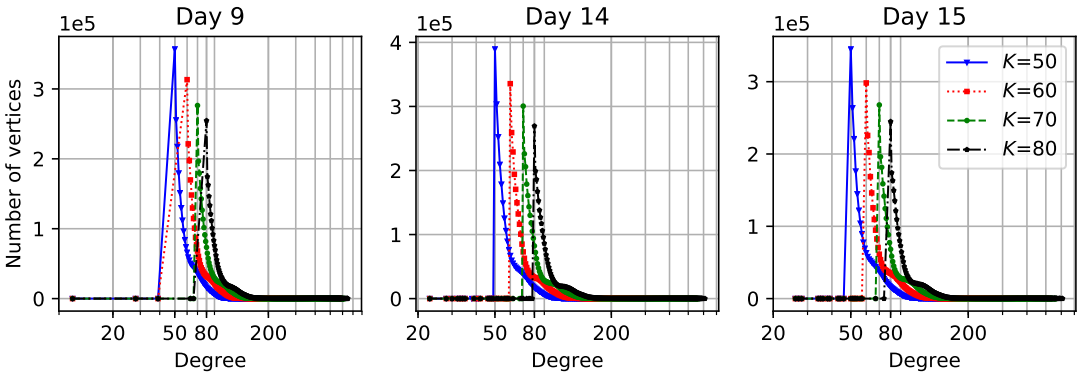


FIGURE 6.5: Degree distribution of the event graph for each of the three selected days, with several values of K.

than K other events, again leading to missing edges. This loss in density with respect to the theoretical upper bound can be observed in Figure 6.4a: for instance, the upper bounds for $K = 50$ are $3.86 \cdot 10^{-5}$, $3.51 \cdot 10^{-5}$ and $3.90 \cdot 10^{-5}$ for days 9, 14 and 15, respectively. The actual densities are smaller by approximately one third. Besides, they seem to grow linearly with K, suggesting that the proportion of missing edges does not depend on K (at least when K is small).

The degree distribution is another interesting property of the generated graphs. It can be expected to peak around K, but its global shape is less predictable: in particular, the maximum degree of an event is hard to guess. Figure 6.5 shows that degrees are actually quite concentrated around K, with extreme values not exceeding a few hundreds. A possible explanation is the influence of time in the similarity function from Equation 6.1: since pairwise similarity decreases exponentially with time, the set of events having a given event v_k among their K nearest neighbors is restricted to a small time window around t_k. As a consequence, v_k's degree cannot be arbitrarily large (unless the time constant τ is sufficiently high).

In addition to their structure, the event graphs we build are characterized by the weights associated with their edges. Figure 6.6 displays the aggregate distribution of edge weights (i.e. pairwise similarities of adjacent vertices) for the three generated

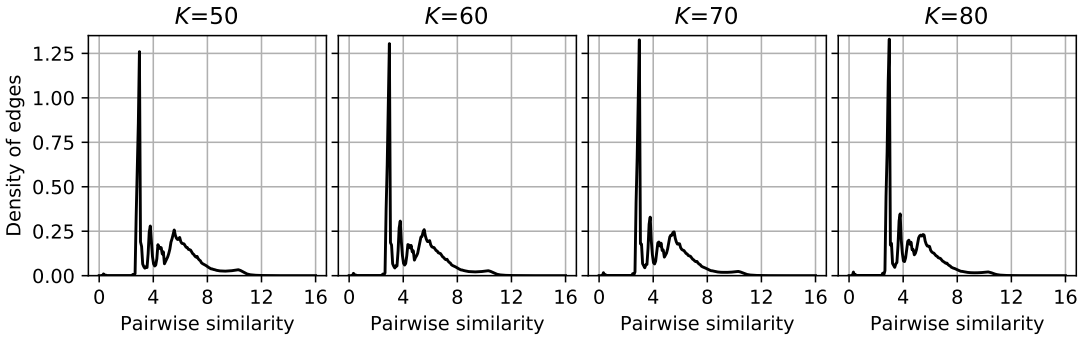
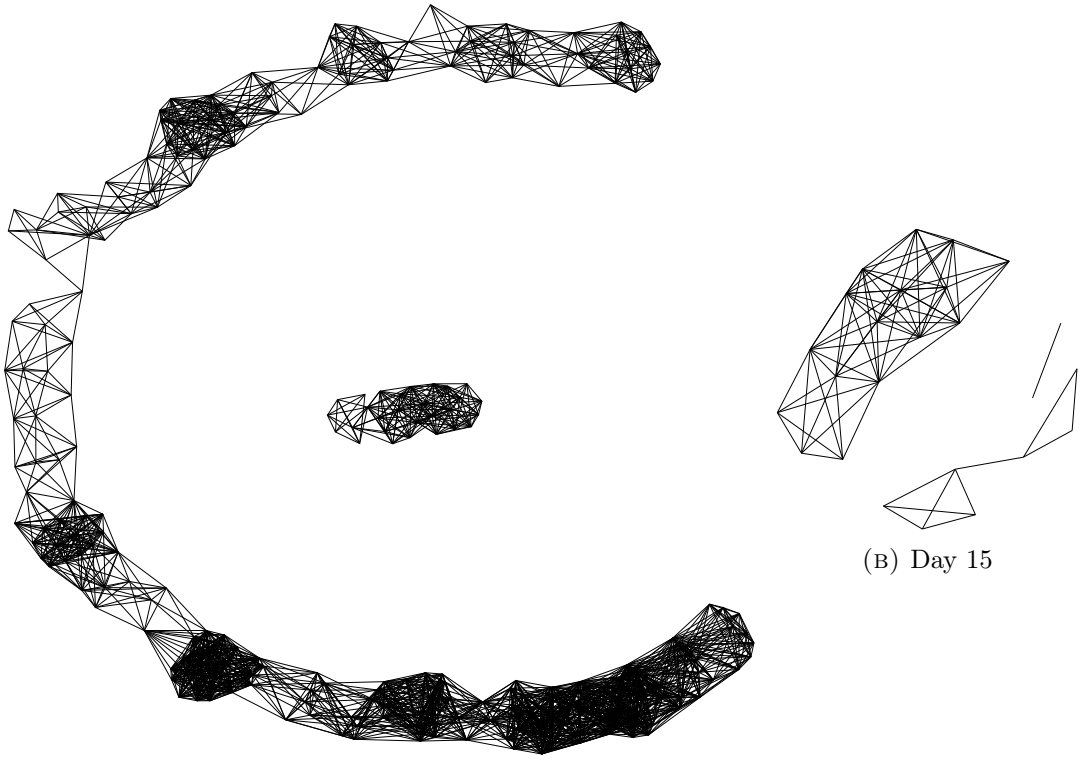


FIGURE 6.6: Empirical probability density function of the pairwise similarity function for edges of the event graph. The distribution is computed over the edges of all three considered event graphs for several values of K.

graphs. Four peaks can be distinguished, which may correspond to the possible numbers of shared entities between two events: since we consider authentication events, there are indeed at most four involved entities for each event (user, source, destination and authentication type). Note that the shape of the distribution is essentially the same regardless of the number of neighbors K. The only difference is that the first peaks, corresponding to the clusters of edges with the lowest similarity, are higher for greater values of K (which intuitively makes sense).

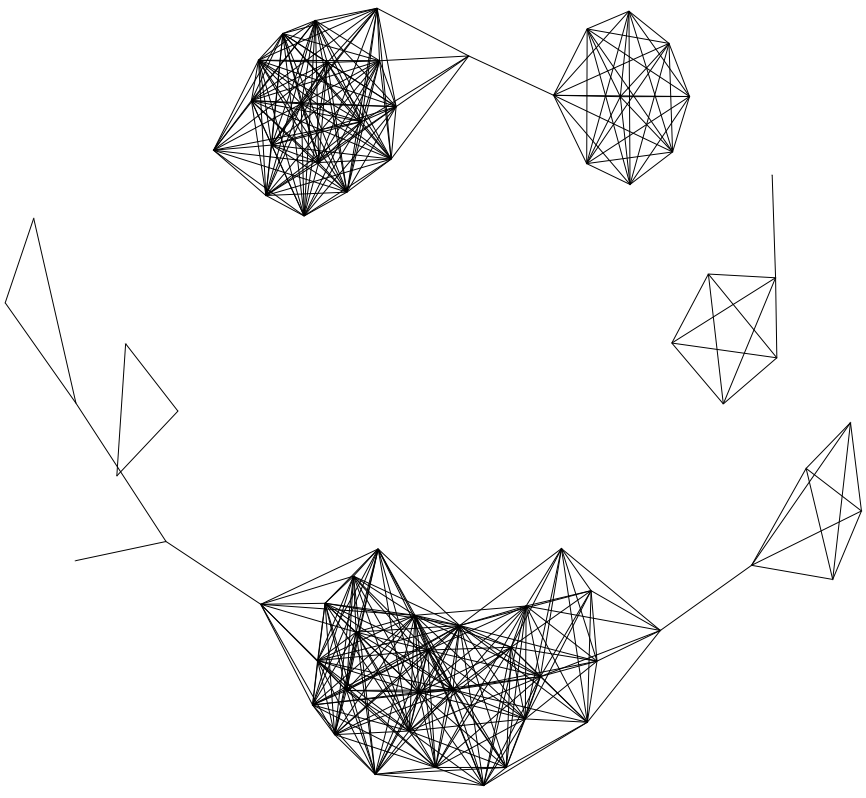
We now move on to one of the most important aspects, namely the structure of the subgraphs induced by red team events. Recall that in order for our denoising approach to actually increase detection performance, malicious events should be gathered in densely connected clusters. To assess whether this is the case in the generated event graphs, we compute the average proportion of red team events in the neighborhood of a red team event (hereafter denoted γ) for each considered day. The results are displayed in Figure 6.4b. The proportion γ appears to be positively correlated with the overall proportion of malicious events (denoted δ and given in Table 6.1), which is not much of a surprise. However, γ seems to decrease slower than a linear function of δ, which is a desirable property: it implies that the malicious subgraph remains dense even when there are few malicious events. More generally, red team events exhibit significant homophily, with γ taking values orders of magnitude larger than δ. We also observe that increasing the number of neighbors K beyond 50 leads to lower values of γ. From the perspective of anomaly score denoising, this may have a negative impact: each additional edge between a red team event and a benign one makes the anomaly score of the red team event decrease more when smoothing scores over the event graph. However, this might be made up for by a similar effect on false negatives: assuming that such events are mostly connected to benign ones, letting them form more edges makes denoising more efficient at decreasing their anomaly score.

A more detailed understanding of the structure of malicious event subgraphs can be gained through direct visualization. We thus display the subgraph induced by all red team events for each of the three considered days in Figure 6.7. Since the obtained subgraphs are roughly similar for all four values of K, we only show those extracted for $K = 50$. While each subgraph has several connected components, these connected components are especially dense. Interestingly, their structure seems to reflect the passage of time, which can be observed most clearly for day 9 (Figure 6.7a): the largest connected component is structured as a sequence of dense communities, with each of these communities being more loosely connected to the previous and next ones. Since red team events are spread out between 9:30 and 23:00, each community can intuitively be interpreted as a sequence of actions carried out over a short time window. Such



(A) Day 9

(B) Day 15



(C) Day 14

FIGURE 6.7: Subgraph induced by the red team events for each selected day (with K=50).

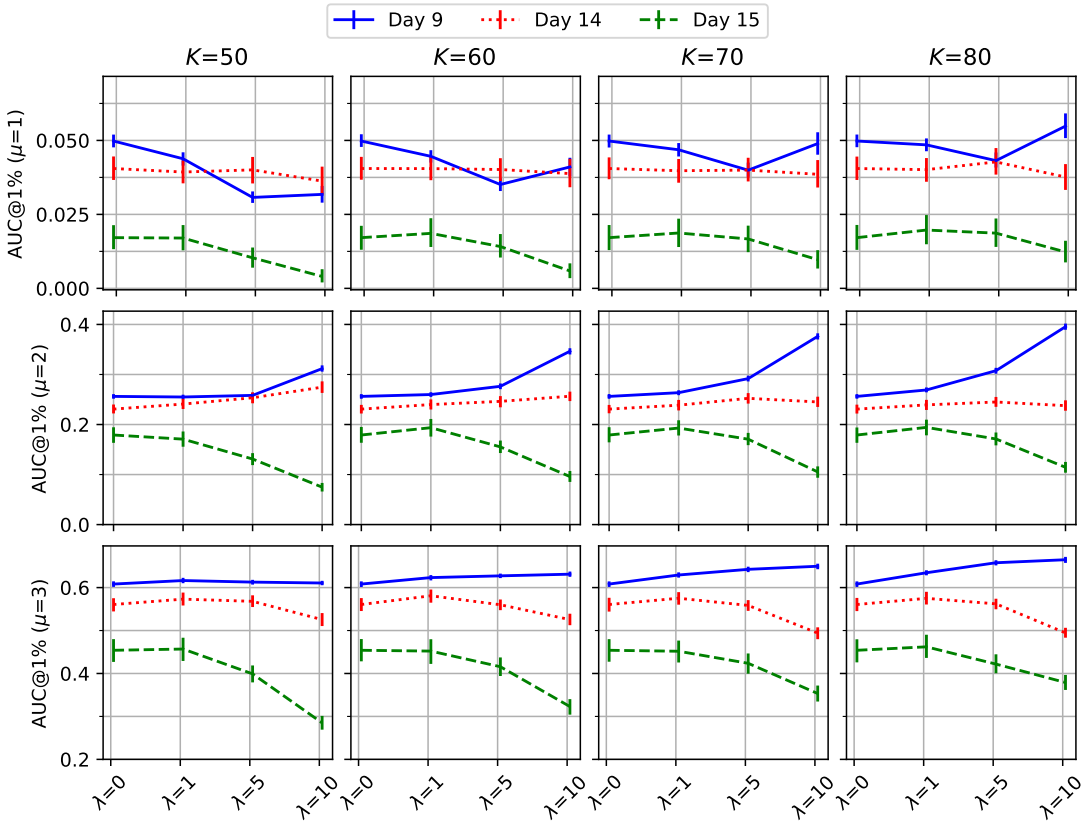
TABLE 6.2: Maximum increase in AUC@1% obtained by each method on synthetic signals for each day and each value of the signal strength μ, with 95% confidence interval. The maximum is taken over all values of K, λ and R. The score of the best-performing method for each setting is in bold.

	Method	$\mu = 1$	$\mu = 2$	$\mu = 3$
Day 9	**Heat kernel**	.005±.004	.139±.006	.057±.009
	Message passing	**.313±.010**	**.525±.012**	**.205±.024**
Day 14	**Heat kernel**	.002±.005	.044±.010	.020±.019
	Message passing	**.132±.014**	**.369±.017**	**.123±.031**
Day 15	**Heat kernel**	.003±.003	.015±.013	.008±.029
	Message passing	**.040±.012**	**.180±.026**	**.066±.042**

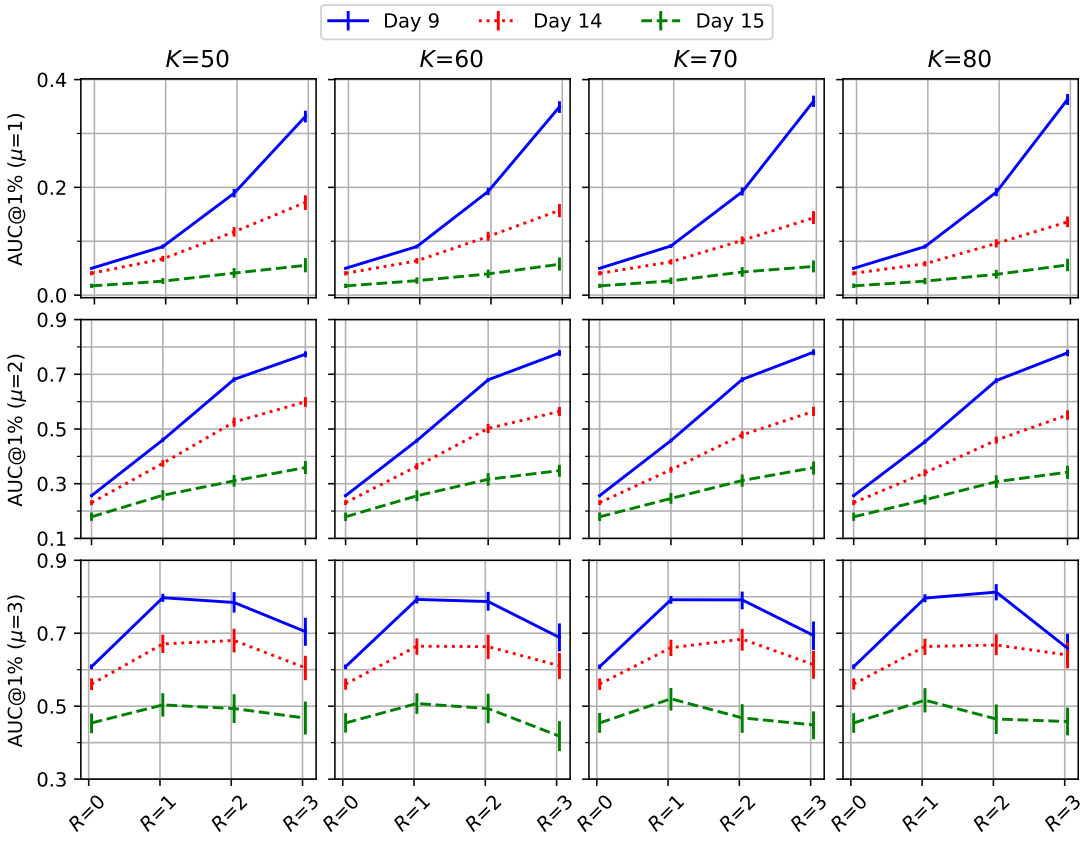
patterns suggest that our event graph construction procedure successfully encodes relevant information about the underlying event log.

Impact of denoising for synthetic scores. We now assess the impact of graph-based denoising in terms of detection performance for synthetic anomaly scores. Starting with an aggregate perspective, we compute the maximum increase in AUC@1% obtained across all hyperparameter values for each day and each value of the signal strength μ. In other words, for a given day-signal strength pair, we compute the difference between the AUC@1% before and after denoising for each combination of possible values of the number of neighbors K and the method-specific hyperparameter (either the diffusion time λ or the number of message passing iterations R). The values we consider are the following: $K \in \{50, 60, 70, 80\}$, $\lambda \in \{1, 5, 10\}$ and $R \in \{1, 2, 3\}$. Results are displayed in Table 6.2. The message passing approach consistently outperforms the heat kernel, which might be explained by the difficulty of appropriately tuning the hyperparameter λ: the values we picked are mostly arbitrary and might not be optimal. Beyond this comparison, the results obtained through message passing are encouraging: with an increase in AUC@1% of up to 0.52, graph-based denoising appears as a potentially effective way to improve detection performance.

A more detailed perspective can be found in Figure 6.8, which displays the AUC@1% for all hyperparameter combinations for both methods. Several trends can be observed. First of all, increasing K leads to tendencially better performance, although the gain is smaller for the message passing approach. This suggests that the aforementioned decrease in the proportion γ of red team neighbors is indeed counteracted by some other effect. However, the obtained performance increase is positively correlated with the proportion δ of red team events, which is itself correlated with γ. The effectiveness of denoising thus does appear to depend on γ, which is a sensible conclusion. As for the influence of the hyperparameters λ and R, they seem to have a different optimal value for each day and signal strength. This is especially evident for the message passing approach (Figure 6.8b): for each considered day, there appears to be a maximum reachable value of the AUC@1% (around 0.8, 0.68 and 0.52 for days 9, 14 and 15, respectively), which is attained for a smaller value of R when μ becomes larger. This can be explained in terms of signal-to-noise ratio (SNR): as μ grows, so does the SNR, and less denoising is thus needed to make red team events distinguishable. The fact that the AUC@1% cannot reach a perfect score of one may result from the presence of several connected components in the malicious subgraph:



(A) Heat kernel



(B) Message passing

FIGURE 6.8: AUC@1% with 95% confidence interval computed on synthetic signals for increasing values of the denoising hyperparameters λ and R, with several values of the number of neighbors K and the signal strength μ. Results for $\lambda = 0$ or $R = 0$ correspond to the absence of denoising.

TABLE 6.3: Maximum increase (or minimum decrease) in AUC@1% obtained by each method on real anomaly scores for each day, with 95% confidence interval. The maximum is taken over all values of K, λ and R. The score of the best- (or least badly-) performing method for each setting is in bold.

Method	Day 9	Day 14	Day 15
Heat kernel	.091±.044	**-.043±.011**	**-.021±.011**
Message passing	**.102±.046**	-.086±.017	-.035±.015

even when all vertices in the largest component are at the top of the anomaly ranking, smaller components have a lower SNR and thus cannot be detected as effectively.

Impact of denoising for real scores. Having investigated the impact of the two denoising methods in various settings, we now study their effectiveness for actual anomaly scores output by our event-wise anomaly detection algorithm. Once again, we first compute the best performance obtained by each method across all hyperparameter values. The results are displayed in Table 6.3. They are significantly worse than for synthetic scores: for days 14 and 15, denoising actually degrades detection performance. However, day 9 (which has the highest proportion of red team events) does allow for a slight increase, with the message passing approach performing best.

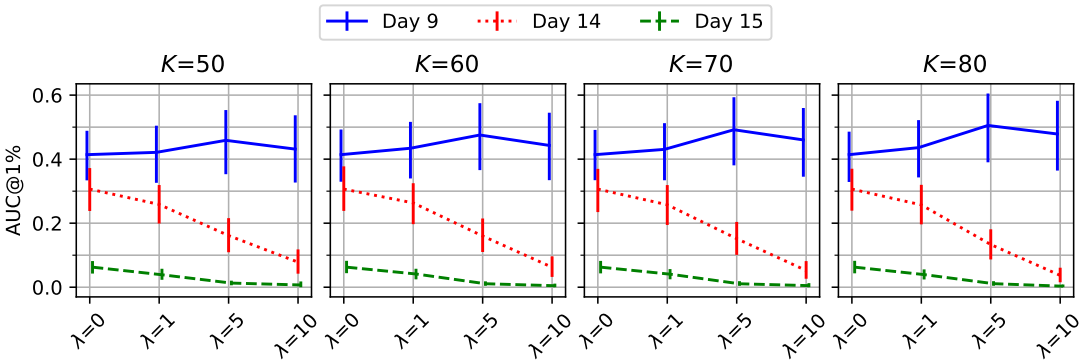
The most plausible explanation for these globally poor results is that false positives are in fact not mutually independent. Indeed, some benign but unusual behaviors are likely to generate clusters of statistically anomalous events. A typical example is an administrator deploying software on several computers: such actions do not follow any regular pattern, and they can generate many connected events (at least one for each affected computer). The reason why increased detection performance can still be obtained on day 9 could then be that the largest malicious cluster observed on that day is significantly larger than benign clusters, allowing these false positives to be smoothed out without affecting true red team events. A more in-depth investigation into the evolution of the set of highest-ranking events when denoising the signal could lead to further insight.

Finally, Figure 6.9 shows more detailed results. Besides the clear influence of the proportion of red team events, the impact of the number of neighbors K also exhibits similar trends as in the synthetic case. More specifically, increasing K still seems to yield slightly better results, especially for the heat kernel.

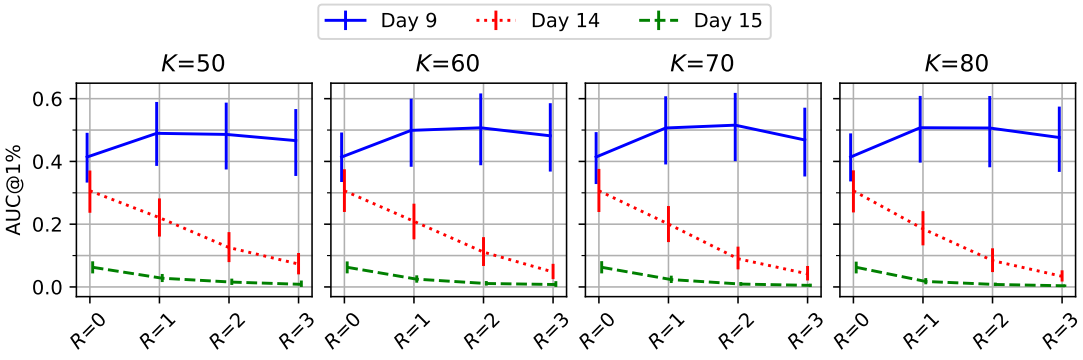
6.6 Conclusion

We propose a GSP-inspired approach to event-wise anomaly score postprocessing. Our main intuition is that anomaly scores can be seen as a noisy signal defined over an event graph. The homophily of malicious events can then be leveraged to make these scores more reliable: since localized peaks in the graph-structured signal are likely to be false positives, smoothing them out can lead to better detection performance. In practice, we use two denoising tools to perform this task: the heat kernel, which can be seen as the GSP equivalent of a low-pass filter, and a message passing scheme inspired by the Weisfeiler-Lehman algorithm. We also define an event graph construction procedure relying on a pairwise event similarity function reflecting the presence of shared entities in two events.

Experiments show that our approach can indeed increase detection performance in some cases. However, it requires a relatively high number of malicious events to make



(A) Heat kernel



(B) Message passing

FIGURE 6.9: AUC@1% with 95% confidence interval computed on real anomaly scores for increasing values of the denoising hyperparameters λ and R, with several values of the number of neighbors K. Results for $\lambda = 0$ or $R = 0$ correspond to the absence of denoising.

up for the presence of benign clusters in real-world settings. While this limitation may be partially alleviated by using better event-wise anomaly detection algorithms, the fundamental challenge of malicious behavior detection remains: from the point of view of event logs, many benign behaviors look an awful lot like malicious ones. As a consequence, even though postprocessing algorithms can be useful, they can never fully replace human reasoning and organizational procedures (such as notifying security analysts when a large-scale administrative operation takes place).

Despite this somewhat insurmountable limitation, some improvements could be brought to our approach, opening interesting research leads. First of all, our event graph construction procedure and the similarity function it relies upon could probably be refined. In particular, a more appropriate weighting scheme for the respective contributions of involved entities to the overall similarity could be designed. Secondly, hyperparameter setting is a tricky issue here: our experiments show that performance gains are especially sensitive to the diffusion time λ and the number of message passing iterations R, and picking the value that yields the best detection performance is obviously not an option in a real-world network monitoring setting. It would thus be interesting to design unsupervised criteria for hyperparameter tuning.

Finally, while the methodology described in this chapter aims to take advantage of the presence of clusters of malicious events, it does not actually detect them. This can be seen as a limitation: assuming that intrusions are characterized by the presence of a large cluster of anomalous events, being able to know whether such a cluster is present before starting any further investigation could spare some precious time. Therefore, Chapter 7 addresses the generic problem of cluster detection in a network

with node-related scalar observations.

Chapter 7

Detecting Clusters of Anomalous Events: a Percolation-Based Approach

The event graphs defined in the previous chapter allow us to harness the assumption that malicious activity generates connected subgraphs of anomalous events rather than isolated anomalies. In addition to making event-wise anomaly scores more reliable, this connectivity can be leveraged to predict the presence of malicious activity in a more global manner: indeed, while the presence of individually anomalous events can often result from rare but legitimate behaviors, connected clusters of anomalous events are more significant. The ability to detect such clusters is thus of particular interest. As a consequence, we study the generic problem of cluster detection in networks with vertex-related scalar observations. Motivated by the limited scalability of standard approaches to this problem, we build upon previous contributions highlighting the connection between cluster detection and percolation theory. We leverage this parallel to design two statistical tests and demonstrate the computational efficiency and detection performance of these tests on a synthetic benchmark dataset, as well as event graphs extracted from the LANL dataset.

7.1 Introduction

Volume is the main obstacle preventing systematic analysis of event logs generated in a computer network: with millions of events recorded every day, visualization, basic heuristics and manual inspection are simply not enough, and advanced statistical methods are needed to direct human analysts' attention towards the most suspicious sections of the event stream. In the previous chapters, we have been aiming to give each event an individual anomaly score, which then allows the analyst to sort events in descending order of anomalousness and only investigate the highest-ranking ones each day. However, in a normal operation setting, one could expect to observe no intrusion on most days. Therefore, investigating the same number of events each day clearly seems suboptimal: ideally, being able to predict the presence of malicious activity before starting to analyze the logs would significantly reduce the amount of human effort necessary to monitor the event stream.

To perform such predictions, we make use of the event graphs introduced in the previous chapter. Our intuition is as follows: since malicious activity is assumed to

generate connected subgraphs of anomalous events, detecting the existence of such activity can be formalized as a case of cluster detection in a graph. The definition of cluster detection considered here relates to the more general framework of structured elevated mean detection: given a graph $\mathcal{G}$ with a real-valued signal observed over its vertices, a cluster is defined as a connected subgraph $\mathcal{S} \subset \mathcal{G}$ inside of which the values taken by the signal are significantly higher than expected [Arias-Castro et al., 2011]. In other words, cluster detection is a hypothesis testing problem where the class of alternatives has a combinatorial structure [Addario-Berry et al., 2010].

Numerous real-world applications have motivated extensive research on practical cluster detection. These applications are often related to the field of spatial statistics and include, for instance, disease outbreak detection [Kulldorff, 1997], object detection in images [Langovoy and Wittich, 2013] or sensor network monitoring [Sharpnack et al., 2013a]. The standard approach relies on scan statistics [Glaz et al., 2001]: given a scoring function quantifying how significant a potential cluster is, the scan statistic is defined as the maximum of this scoring function over the set of potential clusters. A significant cluster can be expected to exist in the network if the scan statistic is above an appropriate threshold. Computing this statistic then becomes the main challenge, essentially reducing cluster detection to a combinatorial optimization problem.

Unfortunately, the main specificity of our use case – namely the considerable size of the ambient graph – happens to make scan statistics essentially unusable. Indeed, because of the computational cost of solving the underlying optimization problem, scan statistics-based detection does not scale well. While several contributions have sought to alleviate this issue through more efficient optimization methods (both exact and approximate), we take an orthogonal path and explore an optimization-free approach to cluster detection in a graph. This approach relies upon the following intuition: when removing all nodes except those at which the signal takes its highest values, the size of the largest remaining connected component should be small in the absence of a cluster. On the other hand, when a cluster is present, most of its nodes should remain in the thresholded graph, leading to a significantly larger connected component. We derive two testing procedures from this idea, the second one differing from the first through the adjunction of a denoising step analogous to the smoothing methodology studied in Chapter 6.

The rest of this chapter is structured as follows. We formally define the problem of cluster detection and review the main approaches to it in Section 7.2. Section 7.3 then introduces some notions of percolation theory and highlights their relevance to cluster detection. These notions are then used to define our cluster detection procedures in Section 7.4. We finally evaluate these procedures on synthetic data in Section 7.5 and on event graphs from the LANL dataset in Section 7.6.

7.2 Cluster Detection, Scan Statistics and Alternatives

We first formally define cluster detection as a hypothesis testing problem and review some related theoretical results in Section 7.2.1, then discuss practical detection methods and their limitations in Section 7.2.2.

7.2.1 Problem Statement and Theoretical Results

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected and connected graph, where $\mathcal{V} = \{v_1, \dots, v_n\}$ denotes the set of vertices of $\mathcal{G}$ and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is its edge set. Let $\mathbf{A} \in \{0,1\}^{n \times n}$ denote the adjacency matrix of $\mathcal{G}$ and $\mathbf{M} = \mathbf{D}^{-1}\mathbf{A}$ denote its row-normalized counterpart (where $\mathbf{D}$ is the diagonal matrix whose k-th diagonal coefficient is the degree of v_k). We

define Λ as the set of subsets of $\mathcal{V}$ whose induced subgraph in $\mathcal{G}$ is connected. For each vertex $v_k \in \mathcal{V}$, let X_k be a real-valued random variable attached to v_k. Similarly to Chapter 6, the random vector $\mathbf{X} = (X_k)_{1 \leq k \leq n}$ can be seen as a signal observed over the vertices of $\mathcal{G}$. Consider the following hypothesis testing problem: letting F_0 denote a probability distribution with zero mean and unit variance, the null hypothesis is defined as $H_0 : X_k \overset{\text{iid}}{\sim} F_0$. In addition, for each $\mathcal{S} \in \Lambda$,

$$H_{\mathcal{S}} : \forall v_k \in \mathcal{V},\, X_k \overset{\text{ind}}{\sim} \begin{cases} F_1 & \text{if } v_k \in \mathcal{S} \\ F_0 & \text{otherwise} \end{cases}$$

is one possible alternative, with $F_1 \neq F_0$ a probability distribution such that

$$\mathbb{E}_{X \sim F_1}[X] > 0.$$

The problem of cluster detection can then be formulated as

$$H_0 \quad \text{vs.} \quad H_1 = \bigcup_{\mathcal{S} \in \Lambda} H_{\mathcal{S}}.$$

That is, we want to know whether there exists a connected subgraph of $\mathcal{G}$ inside of which the observations X_k are drawn from an alternative distribution with elevated mean. Note that we only care about detection, leaving the reconstruction of $\mathcal{S}$ aside.

The most frequent instance of this generic problem in the literature is the Gaussian case, in which F_0 is a standard centered normal distribution and F_1 is a shifted version of F_0 with mean $\mu > 0$. Even in this standard setting, deriving theoretical results on detectability is highly nontrivial: a specific structure must be assumed for both the ambient graph $\mathcal{G}$ and the possible subgraphs $\mathcal{S}$ to make the problem tractable. Under such assumptions, asymptotic separability of H_0 and H_1 as $n \to \infty$ can be stated in the minimax sense: letting

$$\gamma_n(\xi) = \mathbb{P}[\xi(\mathbf{X}) = 1 \mid H_0] + \max_{\mathcal{S} \in \Lambda} \mathbb{P}[\xi(\mathbf{X}) = 0 \mid H_{\mathcal{S}}]$$

denote the worst-case risk of a given test $\xi : \mathbb{R}^n \to \{0, 1\}$, the minimax risk can be defined as

$$\gamma_n = \min_{\xi} \gamma_n(\xi).$$

The null and alternative hypotheses are then said to be asymptotically inseparable if

$$\lim_{n \to \infty} \gamma_n = 1.$$

Conversely, a sequence of tests $(\xi_n)_{n \geq 1}$ asymptotically separates H_0 and H_1 if

$$\lim_{n \to \infty} \gamma_n(\xi_n) = 0,$$

and the two hypotheses are separable if there exists such a sequence.

Several theoretical results on asymptotic separability of cluster detection problems can be found in the literature. Table 7.1 summarizes some of these results. However, while they provide valuable insights on the intrinsic difficulty of cluster detection, such theoretical statements are of little help when designing practical detection procedures. We thus adopt a more applied perspective in the next section.

TABLE 7.1: Theoretical results on cluster detection: conditions of asymptotic separability and inseparability in the minimax sense. The mean $\mu_{\mathcal{S},n}$ of the signal under $H_{\mathcal{S}}$ for an ambient graph with n vertices is normalized as $\mu_{\mathcal{S},n} = \mu_n|\mathcal{S}|^{-1/2}$ so that the signal strength $\mu_n > 0$ is independent of the size of $\mathcal{S}$.

Ref.	**Ambient graph**	**Class of possible clusters**	**Separable if...**	**Inseparable if...**
[Arias-Castro et al., 2005]	Line graph	Segments	$\mu_n = \sqrt{2(1+\eta)\log n}$ $(\eta > 0)$	$\mu_n = \sqrt{2(1-\eta)\log n}$ $(\eta > 0)$
	Square lattice (dimension 2)	Disks, segments, rectangles, ellipsoids	$\mu_n = \sqrt{2(1+\eta)\log(n)^2}$ $(\eta > 0)$	$\mu_n = \sqrt{2(1-\eta)\log(n)^2}$ $(\eta > 0)$
	Square lattice (dimension d)	Rectangles	$\mu_n = \sqrt{2(1+\eta)\log(n)^d}$ $(\eta > 0)$	$\mu_n = \sqrt{2(1-\eta)\log(n)^d}$ $(\eta > 0)$
[Arias-Castro et al., 2008]	Square lattice (dimension 2)	Paths	$\mu_{\mathcal{S},n}\sqrt{\log\lvert\mathcal{S}\rvert} \to \infty$	$\mu_{\mathcal{S},n}\log\lvert\mathcal{S}\rvert\sqrt{\log\log\lvert\mathcal{S}\rvert} \to 0$
	Binary tree	Paths	$\mu_{\mathcal{S},n} \geq \sqrt{2\log 2}$	$\mu_{\mathcal{S},n} \to 0$
[Arias-Castro et al., 2011]	Square lattice (dimension d)	Band of thickness h_n around a path of length ℓ_n	$\mu_n\sqrt{h_n/\ell_n} \to \infty$	$\mu_n\sqrt{h_n/\ell_n}(\log n)^{3/2} \to 0$

7.2.2 Practical Detection Methods – Scan Statistics and Beyond

Besides theoretical analysis, practical detection methods and real-world applications have received significant attention in the literature. The most common approach relies on scan statistics. Broadly speaking, this method consists in defining a scoring function $f : \mathcal{P}(\mathcal{V}) \to \mathbb{R}$, computing the test statistic

$$T = \max_{\mathcal{S}\in\Lambda} f(\mathcal{S}),$$

then rejecting H_0 if T exceeds a given threshold. This amounts to finding the best potential cluster $\mathcal{S}^*$ in Λ, and then rejecting the null hypothesis if $\mathcal{S}^*$ is significant enough. Defining f requires some assumptions on the alternative distribution F_1. For instance, when F_1 has a parametric form, $f(\mathcal{S})$ can be defined as the likelihood ratio between $H_{\mathcal{S}}$ and H_0. In the Gaussian case mentioned above, the scoring function f_{g} is classically defined as

$$f_{\mathrm{g}}(\mathcal{S}) = \frac{1}{\sqrt{|\mathcal{S}|}} \sum_{v_k\in\mathcal{S}} X_k. \tag{7.1}$$

With no more prior knowledge about F_1 than the elevated mean assumption, however, finding a suitable scoring function is nontrivial. Moreover, computing T implies maximizing f over the combinatorial class Λ, which quickly becomes computationally intensive as the ambient graph grows large. Therefore, most related work focuses on making the computation of scan statistics more efficient. Ways to achieve this include the following:

Restriction of the class Λ. The easiest way to speed up the computation is to simply reduce the size of the search space by considering only a subset of Λ. Such restriction can be based on domain-specific knowledge [Kulldorff, 1997, Priebe et al., 2005, Kulldorff et al., 2006, Neil et al., 2013a] or more general heuristics [Patil and Taillie, 2004].

Convex relaxation. Another classical approach to combinatorial optimization consists in solving a convex relaxation of the problem, and then projecting the solution back onto the original search space. This method was applied to scan statistics [Qian et al., 2014, Qian and Saligrama, 2014, Aksoylar et al., 2017], using elements of spectral graph theory [Chung, 1997] to find a relaxed form of the connectivity constraint. Similar ideas were also used in a slightly different context [Sharpnack et al., 2013a, Sharpnack et al., 2013b, Sharpnack et al., 2015], where the class Λ consists of subgraphs with low cut size rather than connected ones.

Algorithmic approaches. Finally, efficient optimization algorithms have been used to find exact or approximate values for the scan statistic, including simulated annealing [Duczmal and Assuncao, 2004, Duczmal et al., 2006], greedy algorithms [Rozenshtein et al., 2014], primal-dual algorithms [Rozenshtein et al., 2014], branch and bound algorithms [Speakman et al., 2015] and dynamic programming algorithms [Wu et al., 2016b].

Despite the popularity of scan statistics, other ideas have also been considered in the literature. We focus on one of these alternative approaches, namely the Largest Open Cluster (LOC) test, which was first studied in the context of object detection in images [Langovoy and Wittich, 2013, Langovoy et al., 2013]. The idea of this method is to represent an image as a two-dimensional lattice, each node carrying a random variable standing for the value of the associated pixel. Then, after deleting from the lattice every vertex whose pixel value is lower than a suitable threshold, the largest remaining connected component is expected to be small if there is no object in the image. On the other hand, if an object is present, an unexpectedly large connected component should remain in the thresholded lattice. The theory behind the LOC test has since been extended to regular lattices of arbitrary dimension [Arias-Castro and Grimmett, 2013], but to the best of our knowledge, the underlying idea of using percolation theory to detect anomalous connected subgraphs has not yet been applied to complex, arbitrary-shaped networks.

7.3 Percolation Theory and Its Relevance to Cluster Detection

Theoretical results underpinning the LOC test rely on percolation theory. More generally, some basic notions pertaining to this field are useful when defining and intuitively justifying the detection procedures we propose. We thus briefly introduce the general framework of percolation theory in Section 7.3.1, then more specifically highlight its connections with cluster detection in Section 7.3.2.

7.3.1 A Brief Introduction to Percolation Theory

Broadly speaking, the goal of percolation theory is to study the connectivity of a network when a given proportion of its elements are randomly deleted. The deleted

elements can be either vertices or edges, leading to the two closely related settings of site and bond percolation, respectively. Since we are interested in vertex-related signals, we here focus on the former. Note, however, that similar developments could be made about the link between bond percolation and cluster detection in edge-weighted networks. Fore a more formal and detailed introduction, see for instance [Kesten, 1982, Grimmett, 1999, Chung et al., 2009].

Still considering an undirected and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n vertices, the main focus of percolation theory is the following problem. Let each vertex of $\mathcal{G}$ be independently occupied uniformly at random with probability p or unoccupied with probability $1 - p$. In other words, each vertex $v \in \mathcal{V}$ is deleted along with all its incident edges with probability $1-p$. Then, letting $C(p)$ denote the size of the largest component of $\mathcal{G}$ at occupation probability p, what is the expected value of $C(p)$ as n becomes large? Extremal values of p yield obvious results: $C(0) = 0$ almost surely for any n and for $p = 1$, $\mathcal{G}$ almost surely has a connected component with infinitely many vertices as n goes to infinity. For intermediate values of p, however, there are two possible regimes. If p is small enough, only small connected components are present and $C(p)/n$ converges in probability to 0. On the other hand, larger values of p lead to the emergence of a giant connected component, which contains a constant fraction of the vertices. The transition between the two regimes happens for a critical value of p called the percolation threshold p_c. Note that p_c depends on the graph structure and can be vanishingly small. Although this phase transition is only well-defined in the limit of an infinite graph, a somewhat similar behavior can be observed in the finite case [Callaway et al., 2000, Karrer et al., 2014].

While most existing work on percolation adopts a static point of view by studying the distribution of $C(p)$ for a fixed value of p, we are more interested in a dynamic perspective. More specifically, we consider the canonical coupling defined as follows: for each vertex $v \in \mathcal{V}$, let U_v be an independent random variable following a uniform distribution on $[0, 1]$. Then, for all $p \in [0, 1]$, let $\mathcal{G}(p)$ be the subgraph induced by the vertex set $\{v \in \mathcal{V} : U_v \leq p\}$, and let $\mathcal{C}(p)$ be the largest connected component of $\mathcal{G}(p)$. This construction gives $|\mathcal{C}(p)|$ the same distribution as $C(p)$ for all $p \in [0, 1]$ while ensuring that $\mathcal{G}(p)$ and $|\mathcal{C}(p)|$ are both increasing in p, meaning that

$$\forall (p, q) \in [0, 1]^2,\ p \leq q \implies \mathcal{G}(p) \subseteq \mathcal{G}(q) \text{ and } |\mathcal{C}(p)| \leq |\mathcal{C}(q)|$$

for any realization of the random vector $(U_v)_{v \in \mathcal{V}}$. A tightly related process is obtained by considering the imbedded Markov chain $\{\mathcal{G}_k\}_{0 \leq k \leq n}$, where $\mathcal{G}_k$ is the subgraph induced by the vertices associated with the k smallest random variables. Letting $\mathcal{C}_k$ denote the largest connected component of $\mathcal{G}_k$, $\{|\mathcal{C}_k|\}_{0 \leq k \leq n}$ can be seen as a discretized version of $\{|\mathcal{C}(p)|\}_{0 \leq p \leq 1}$. The two cluster detection procedures introduced in Section 7.4 rely on these stochastic processes. In contrast, previous contributions on percolation-based cluster detection focused on the static point of view, as explained in the next section.

7.3.2 Application to Cluster Detection: Theory and Practice

A rather intuitive connection can be made between the concepts discussed in the previous section and the problem of cluster detection. To the best of our knowledge, this connection was first discussed by Langovoy and Wittich in the context of object detection in a two-dimensional image [Langovoy and Wittich, 2013].

Their main idea is the following: let $\mathcal{G}$ be a two-dimensional lattice, and suppose that the graph-structured signal $\mathbf{X} = (X_k)_{1 \leq k \leq n}$ represents noisy pixel values. The

coordinates of $\mathbf{X}$ are assumed to be independent and identically distributed when no object is present in the corresponding image. Given a threshold $\tau \in \mathbb{R}$, let $\mathcal{G}(\tau)$ be the subgraph induced by the vertex set $\{v_k \in \mathcal{V} : X_k \geq \tau\}$, and let $\mathcal{Q}(\tau)$ be the largest connected component of $\mathcal{G}(\tau)$. Then, if no object is present, $|\mathcal{Q}(\tau)|$ has the same distribution as $C(p_\tau)$ for some appropriately chosen p_τ, namely

$$p_\tau = \mathbb{P}[X_1 \geq \tau \mid H_0], \tag{7.2}$$

with H_0 denoting the null hypothesis (i.e. no object in the image)[1]. Therefore, choosing a threshold τ such that $p_\tau < p_c$ ensures that $|\mathcal{Q}(\tau)|$ is small under H_0. However, the threshold should also be chosen so that

$$\mathbb{P}[X_k \geq \tau \mid v_k \text{ belongs to an object}] > p_c,$$

ensuring that the presence of an object leads to the existence of a significant cluster in $\mathcal{G}(\tau)$. To be able to provide such a suitable threshold as well as theoretical guarantees on the resulting test, Langovoy and Wittich assume that $\mathcal{G}$ is a triangular lattice and that observations $\{X_k\}_{1\leq k\leq n}$ have a symmetric distribution.

The work of Langovoy and Wittich was then extended to a more generic setting by Arias-Castro and Grimmett [Arias-Castro and Grimmett, 2013], who take the d-dimensional square lattice as ambient graph. The cluster shapes considered in [Arias-Castro and Grimmett, 2013] are hypercubes and self-avoiding paths of fixed length. With these structural assumptions, the LOC test is studied for several choices of threshold τ, with three different regimes: subcritical, supercritical and critical, corresponding to p_τ smaller, greater and close to p_c, respectively. The critical regime is shown to be the most powerful one when detecting small clusters. In particular, the LOC test with a near-critical threshold τ is nearly as powerful as the scan statistic for path detection, but not for hypercube detection.

The static standpoint on percolation, which is adopted by the LOC test through the choice of a single threshold τ, can be useful when focusing on theory. Indeed, as mentioned above, most theoretical results on percolation are stated for a fixed occupation probability p. Therefore, leveraging such results to obtain theoretical guarantees for percolation-based tests necessarily leads to considering a single threshold. However, our aim in this chapter is to come up with effective detection procedures for real-world applications. To that end, studying the evolution of $|\mathcal{Q}(\tau)|$ for an increasing sequence of thresholds can be more practical. In particular, it exempts us from looking for an optimal threshold τ, which is nontrivial when considering complex networks whose structure is not known a priori.

A key observation is that the rationale behind the LOC test can be straightforwardly extended to the dynamic perspective. Indeed, the LOC test fundamentally relies on the fact that $|\mathcal{Q}(\tau)|$ has the same distribution as $|\mathcal{C}(p_\tau)|$ under H_0, and a different distribution under H_1 (for a well-chosen threshold τ). Similarly, as long as the observations $\{X_k\}_{1\leq k\leq n}$ are i.i.d. under H_0, the stochastic processes $\{|\mathcal{Q}(\tau)|\}_{\tau\in\mathbb{R}}$ and $\{|\mathcal{C}(p_\tau)|\}_{\tau\in\mathbb{R}}$ have the same joint law under H_0, but not under H_1. The same observation applies to their discretized counterparts $\{|\mathcal{Q}_k|\}_{0\leq k\leq n}$ and $\{|\mathcal{C}_k|\}_{0\leq k\leq n}$. Therefore, a statistical test for cluster detection can also be designed using entire sample paths of these processes rather than marginal distributions for a single threshold τ.

As an illustration, Figure 7.1 shows the evolution of $|\mathcal{C}(p)|$ for $p \in [0, 1]$, for three types of graphs with increasingly complex structures: a two-dimensional square

[1] Note that since the coordinates of $\mathbf{X}$ are i.i.d. under H_0, any one of them could be used instead of X_1 in Equation 7.2.

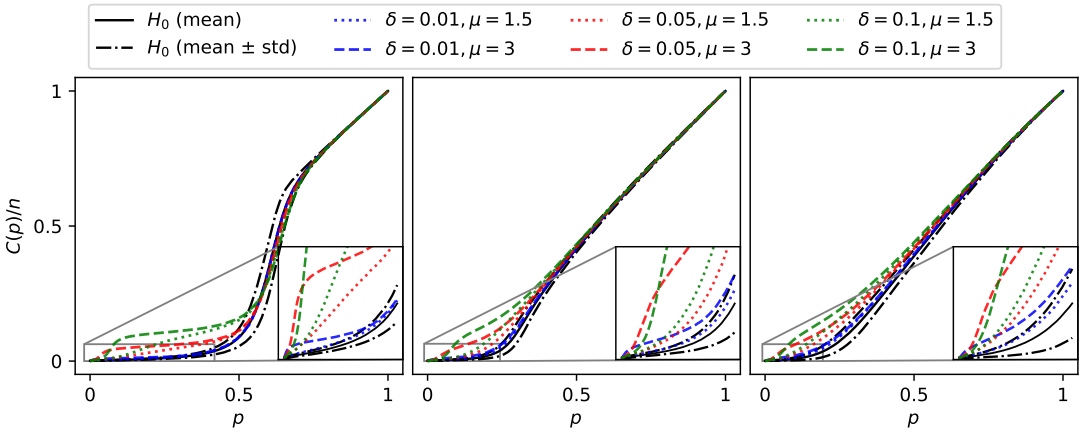


FIGURE 7.1: Evolution of the fraction of vertices in the largest connected component as p varies from 0 to 1, under H_0 and various alternatives, for three kinds of graphs: a two-dimensional square lattice (left), an Erdős-Rényi random graph (center) and a Barabási-Albert preferential attachment graph (right).

lattice, an Erdős-Rényi random graph [Erdős and Rényi, 1960] and a Barabási-Albert preferential attachment graph [Barabási and Albert, 1999]. For each model, a graph with 1 024 vertices and approximately 2 000 edges is generated, and the mean and standard deviation of $|\mathcal{C}(p)|$ for each value of p is estimated using 10 000 Monte Carlo simulations. Then, for each graph, we generate a subgraph $\mathcal{S}$ containing a fraction δ of the vertices, assign to each vertex v_k an independent Gaussian random variable

$$X_k \sim \mathcal{N}\left(\mu \mathbb{1}_{\{v_k \in \mathcal{S}\}}, 1\right)$$

and compute the associated sample path of the stochastic process $\{|\mathcal{Q}(\tau)|\}_{\tau \in \mathbb{R}}$. This experiment is repeated 1 000 times for each graph, and the mean value of $|\mathcal{Q}(\tau)|/n$ is plotted as a function of the fraction of occupied vertices at threshold τ for several values of δ and μ. The two regimes of the percolation process can be observed, and the shape and location of the phase transition both clearly depend on the graph model. While the separation between the two regimes is quite clear for the lattice and the Erdős-Rényi graph, it is much blurrier for the Barabási-Albert model, which yields more complex structures – most interestingly, heavy-tailed degree distributions. Despite these disparities, the trajectory of $\{|\mathcal{Q}(\tau)|\}_{\tau \in \mathbb{R}}$ is clearly different in the presence of an injected cluster for the three types of graphs. More specifically, $|\mathcal{Q}(\tau)|$ is consistently higher under H_1 before the phase transition, which underpins the two detection procedures presented in the next section.

7.4 Two Percolation-Based Tests

We now introduce our cluster detection procedures. The first one, described in Section 7.4.1, relies only on the trajectory of the process $\{|\mathcal{Q}(\tau)|\}_{\tau \in \mathbb{R}}$. Building upon the graph signal processing tools studied in Chapter 6, we then extend this procedure by prepending a denoising step, as explained in Section 7.4.2.

7.4.1 Looking for Deviations of the Percolation Process

Our main detection procedure relies on the dynamic extension of the LOC test introduced in Section 7.3.2. More specifically, given an observed signal $\mathbf{X} = (X_k)_{1 \leq k \leq n}$, let $v_{(1)}, \ldots, v_{(k)}$ be the vertices of $\mathcal{G}$ sorted in descending order of their signal values,

i.e. $X_{(1)} \geq \ldots \geq X_{(n)}$. Then, let $\mathcal{H}_k \subseteq \mathcal{G}$ be the subgraph induced by $v_{(1)}, \ldots, v_{(k)}$ (for $k \geq 1$), and let $\mathcal{Q}_k$ denote its largest connected component. As mentioned above, the size of $\mathcal{Q}_k$ is expected to be larger under H_1 for small values of k – namely, for k smaller than a critical value K_c corresponding to the phase transition under H_0.

In practice, a frequent heuristic for finite and arbitrary-shaped networks locates the onset of the phase transition at the smallest occupation probability p_0 such that the expected value of $C(p_0)$ is greater than $\sqrt{n}$. Therefore, we set

$$K_c = \min\left\{k \geq 2,\, \mathbb{E}_0\left[|\mathcal{Q}_k|\right] \geq \sqrt{n}\right\}, \tag{7.3}$$

where $\mathbb{E}_0[\,\cdot\,]$ denotes the expected value under H_0. Then the test statistic is

$$T_{\mathcal{G}}(\mathbf{X}) = \frac{1}{n(K_c - 1)} \sum_{k=2}^{K_c} \frac{|\mathcal{Q}_k| - \mathbb{E}_0\left[|\mathcal{Q}_k|\right]}{\mathbb{V}_0\left[|\mathcal{Q}_k|\right]^{1/2}}, \tag{7.4}$$

where $\mathbb{V}_0[\,\cdot\,]$ denotes the variance under H_0.

Two problems remain: first, the statistic $T_{\mathcal{G}}(\mathbf{X})$ depends on a priori unknown expected values and variances. Secondly, deciding whether to reject H_0 based on $T_{\mathcal{G}}(\mathbf{X})$ requires a calibration step: how likely is the test statistic to be at least as high under the null hypothesis? Both issues are addressed through simple Monte Carlo simulations. Indeed, as explained in Section 7.3.2, assuming that the observations $\{X_k\}_{1 \leq k \leq n}$ are i.i.d. under H_0 implies that $\{|\mathcal{Q}_k|\}_{1 \leq k \leq n}$ follows the same joint law as $\{|\mathcal{C}_k|\}_{1 \leq k \leq n}$. Therefore, a set of $B \geq 1$ sample paths

$$\left\{ \left\{ \left| \mathcal{Q}_k^{(b)} \right| \right\}_{1 \leq k \leq n} \right\}_{1 \leq b \leq B}$$

can straightforwardly be obtained by uniformly sampling B random orderings of the vertices, even when the null distribution F_0 is unknown. We then compute estimates for the unknown moments,

$$\forall k \in \{2, \ldots, n\},\ \hat{\mu}_k = \frac{1}{B} \sum_{b=1}^{B} \left| \mathcal{Q}_k^{(b)} \right| \text{ and } \hat{\sigma}_k = \sqrt{\frac{1}{B-1} \sum_{b=1}^{B} \left(\left| \mathcal{Q}_k^{(b)} \right| - \hat{\mu}_k \right)^2},$$

which are plugged into Equation 7.3 and Equation 7.4, leading to

$$\hat{K}_c = \min\left\{k \geq 2,\, \hat{\mu}_k \geq \sqrt{n}\right\}, \qquad \hat{T}_{\mathcal{G}}(\mathbf{X}) = \frac{1}{n\left(\hat{K}_c - 1\right)} \sum_{k=2}^{\hat{K}_c} \frac{|\mathcal{Q}_k| - \hat{\mu}_k}{\hat{\sigma}_k}.$$

The estimated test statistic is finally used to compute the empirical p-value

$$\hat{p} = \frac{1}{B} \sum_{b=1}^{B} \mathbb{1}_{\left\{\hat{T}_{\mathcal{G}}^{(b)} \geq \hat{T}_{\mathcal{G}}(X)\right\}}, \text{ where } \hat{T}_{\mathcal{G}}^{(b)} = \frac{1}{n\left(\hat{K}_c - 1\right)} \sum_{k=2}^{\hat{K}_c} \frac{\left| \mathcal{Q}_k^{(b)} \right| - \hat{\mu}_k}{\hat{\sigma}_k}.$$

In terms of computational cost, the sequence $\mathcal{Q}_1, \ldots, \mathcal{Q}_n$ (as well as the B additional samples) can be obtained in $\mathcal{O}(n)$ operations using the Newman-Ziff algorithm [Newman and Ziff, 2001]. Besides, the observations $X_1, \ldots, X_n$ need to be sorted beforehand, leading to $\mathcal{O}(n(B + \log n))$ overall complexity.

7.4.2 Adding a Denoising Step – The Diffusion-Percolation Test

In Chapter 6, we showed that the structure of event graphs could be leveraged to denoise event-wise anomaly scores and make them more reliable. This idea also applies to cluster detection: by smoothing the signal over $\mathcal{G}$, one could expect to erase noisy peaks while preserving true clusters. Such denoising may help make potential clusters stand out, motivating us to propose a slightly more sophisticated detection procedure using signal smoothing as a preprocessing step.

In practice, this denoising step relies on the message passing approach described in Section 6.4.2: given an observed signal $\mathbf{X}$, its smoothed counterpart is defined as

$$\tilde{\mathbf{X}} = \frac{1}{2}(\mathbf{M} + \mathbf{I})\mathbf{X}, \tag{7.5}$$

where $\mathbf{I}$ denotes the $n \times n$ identity matrix. The message passing approach is mainly chosen because of its relatively low computational cost, as explained in further detail below. The sequence of largest connected component sizes $|\mathcal{Q}_1|, \dots, |\mathcal{Q}_n|$ can then be computed as in the previous section. However, generating samples from the null distribution is not as simple anymore.

Indeed, since the smoothed observations $\tilde{X}_1, \dots, \tilde{X}_n$ are no longer independent, the denoising step breaks the equivalence between the two processes $\{|\mathcal{Q}_k|\}_{1 \leq k \leq n}$ and $\{|\mathcal{C}_k|\}_{1 \leq k \leq n}$. Therefore, we now need to sample normal observations from F_0, which is still assumed to be unknown. To circumvent this issue, we resort to a bootstrap procedure [Efron, 1979]: for each $b \in [B]$, the signal $\mathbf{X}^{(b)}$ is obtained by uniformly sampling $X_1^{(b)}, \dots, X_n^{(b)}$ with replacement from $X_1, \dots, X_n$. It is then denoised using Equation 7.5, yielding a smoothed signal $\tilde{\mathbf{X}}^{(b)}$ from which we derive a sequence $\mathcal{Q}_1^{(b)}, \dots, \mathcal{Q}_n^{(b)}$ of largest connected components. The rest of the procedure (computation of the unknown moments, the test statistic and the empirical p-value) is then identical to the one described in the previous section.

As for complexity, Equation 7.5 can be computed in $\mathcal{O}(|\mathcal{E}|)$ operations, and the rest of the procedure (sorting and Newman-Ziff algorithm) still has $\mathcal{O}(n \log n)$ complexity. However, the whole computation (including sorting and denoising) must now be performed $B + 1$ times, leading to $\mathcal{O}(B(n \log n + |\mathcal{E}|))$ overall complexity. These $B + 1$ iterations justify using the cheapest possible smoothing operator: more sophisticated graph signal processing tools applied at each of these iterations would make the computational cost of the whole procedure too high.

7.5 Experiments on Synthetic Data

Before evaluating our detection procedures on actual event graphs, we study them in a more general setting. To that end, we generate a synthetic benchmark dataset and compare our tests with previously published methods on this dataset, as explained in Section 7.5.1. Our results are presented and discussed in Section 7.5.2.

7.5.1 Experimental Setup

Our procedures, which we call Percolation Only (PO) and Diffusion-Percolation (DP), respectively, are implemented in Python 3.9. The most intensive parts (essentially the Newman-Ziff algorithm) are translated into C using Cython. Computations are run on a Debian 10 machine with 128GB RAM and a 2.2GHz, 20-core CPU.

Dataset. The benchmark dataset is generated as follows: first, 50 random graphs of various sizes are sampled using the Kronecker graph model [Leskovec et al., 2010]. More specifically, a single generator matrix Θ =[0.9 0.5; 0.5 0.3] is combined with 5 different numbers of Kronecker product iterations ($i \in \{10, 11, 12, 13, 14\}$) to generate 10 graphs for each value of i. Only the largest connected component of each graph is kept, yielding a connected graph with approximately 2^i vertices. Given two parameters $\delta, \mu > 0$, we then generate 50 normal signals and 50 anomalous signals for each graph: normal observations are independent standard centered Gaussians, while anomalous signals have mean μ on a random connected subgraph containing a proportion δ of the vertices. In summary, for each parameter triple (i, δ, μ), 1 000 signals are generated, half of which contain an anomalous cluster.

Baselines. We compare our procedures with two baselines: the first one is the Upper Level Set scan statistic (ULS [Patil and Taillie, 2004]), which is an approximation of the scan statistic relying on a reduction of the search space. More specifically, ULS also extracts the subgraphs $\mathcal{H}_k$ induced by the highest-scoring vertices. The approximated scan statistic is then the maximum of the scoring function defined in Equation 7.1 over the set of connected components of all subgraphs $\mathcal{H}_k$ for $k \in [n]$. We implement this method in Python using our Cython implementation of the Newman-Ziff algorithm. We also include the adaptive Graph Fourier Scan Statistic (GFSS [Sharpnack et al., 2015]), which approximates the scan statistic through the eigendecomposition of the Laplacian. The open source Python implementation provided by the authors is used for the experiments. Each test is calibrated using 1 000 simulations (for GFSS and ULS, these simulations consist in recomputing the test statistic after randomly permuting the observations assigned to the vertices).

Performance metrics. Two main criteria are used to evaluate the considered algorithms: effectiveness and efficiency – in other words, the ability to detect clusters and the computational cost of doing so. As for the first quality, the area under the ROC curve is computed for each parameter triple (i, μ, δ), highlighting the influence of these parameters on the difficulty of the detection problem. Regarding the second one, we evaluate the running time of each algorithm as a function of the graph size, quantified by the number of Kronecker product iterations i.

7.5.2 Results and Discussion

Figure 7.2 shows the area under the ROC curve for each evaluated method and parameter triple (i, μ, δ), and Figure 7.3 displays the mean computation time for each method as a function of i.

Detection performance. DP and PO perform best overall, with DP doing slightly better for low δ and high μ. This suggests that the diffusion step is especially useful when looking for small but strong clusters. An intuitive explanation is that since PO only leverages the values of the signal through their ranking, it undergoes some kind of saturation. Indeed, assuming that a cluster $\mathcal{S}$ is present, what makes it detectable through PO is that each vertex $v \in \mathcal{S}$ is associated with one of the highest observations $X_1, \ldots, X_n$. As long as this condition is fulfilled, further increasing the signal strength μ has no impact on the test statistic. In contrast, the diffusion step extends the cluster to all vertices $v' \notin \mathcal{S}$ having at least one neighbor in $\mathcal{S}$, which does increase the test statistic. Note that this gain comes with a moderate loss in performance for low μ and high δ. However, ULS tends to perform best in this setting anyway.

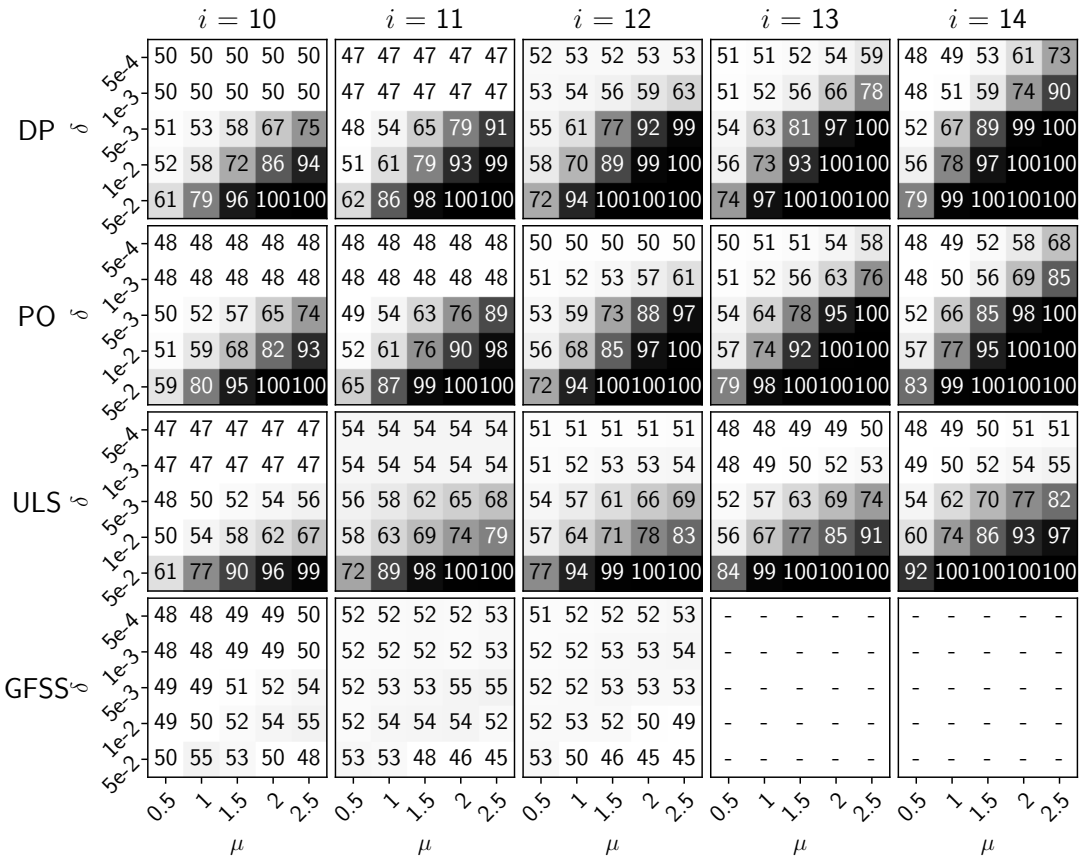


FIGURE 7.2: Area under the ROC curve for each evaluated method, with different combinations of values of i, δ and μ. Dashes indicate unavailable results due to excessive computation times.

As for the GFSS, its surprisingly low performance may be explained by the type of clusters it was designed to detect. Indeed, the underlying assumption of the GFSS is that clusters should have many internal edges and be loosely connected to the rest of the network. While many real-world clusters do exhibit this property (in addition to being connected), the random subgraphs generated for our experiments do not.

Computational cost. Regarding computation times, Figure 7.3 shows that our procedures are both rather efficient, with an expected increase due to the denoising step for DP. In particular, PO is comparable to ULS in terms of computational cost while exhibiting significantly better detection performance.

As a side note, the significantly higher computation times of the GFSS should be taken with a grain of salt since they mostly result from implementation choices. More specifically, the open source implementation we used performs a full diagonalization of the Laplacian of the ambient graph to compute the GFSS, which is typically done through $\mathcal{O}(n^3)$ complexity algorithms [Watkins, 2007]. This considerable overhead could probably be avoided by using the graph signal processing tools introduced in Section 6.4.1 [Hammond et al., 2011].

7.6 Application to Event Graphs

Having assessed the relevance of our detection procedures with respect to the state of the art in a generic setting, we now evaluate their detection performance in our

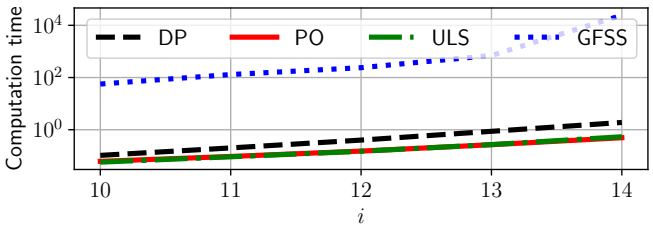


FIGURE 7.3: Mean computation time (in seconds) for each evaluated method.

TABLE 7.2: Definition of the subsets of days from the LANL dataset based on the proportion δ of red team events, and size of each subset.

Value of δ	**Number of days**
$\delta = 0$	40
$\delta \in (0, 5 \cdot 10^{-6}]$	10
$\delta \in (5 \cdot 10^{-6}, 2 \cdot 10^{-5}]$	5
$\delta \in (2 \cdot 10^{-5}, 1]$	3

actual use case – namely malicious behavior detection in event graphs. Section 7.6.1 describes the experiment we carried out (to wit, injection of synthetic anomaly scores), and our results are displayed and discussed in Section 7.6.2.

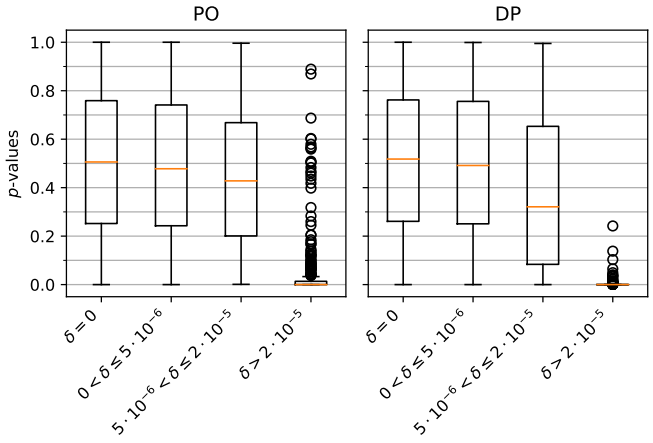
7.6.1 Experimental Setup

Our goal is to evaluate the feasibility of malicious event cluster detection in real-world event graphs. To that end, we apply our detection procedures to synthetic signals sampled over event graphs extracted from the LANL dataset.

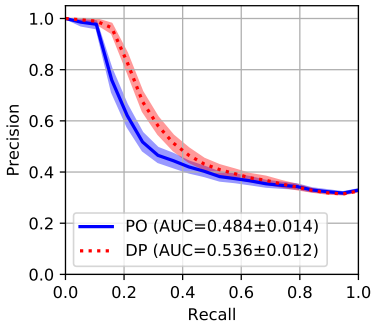
Event graphs. We formulate cluster detection as a binary classification problem: given all authentication events collected on a given day as well as the corresponding anomaly scores, we aim to predict whether red team events happened on this day. Therefore, we use the procedure defined in Section 6.3 to build the graph of authentication events for each day of the LANL dataset. We set the number of neighbors K to 50 and the time constant τ to half an hour. Among the 58 event graphs we obtain, 18 contain red team events. We denote δ the proportion of red team events for a given day. In order to study the impact of δ on detection performance, we divide the set of event graphs into four subsets, each of which corresponds to an interval for δ. More details can be found in Table 7.2.

Generation of synthetic signals. For each day d, we sample 100 Gaussian signals $\mathbf{X}_d^{(1)}, \ldots, \mathbf{X}_d^{(100)}$ with a similar procedure as in Section 7.5.1. The main difference is that the anomalous cluster $\mathcal{S}$ is now defined as the set of red team events. Note that as shown in Section 6.5.2, it actually has several connected components for some days. However, the intuition behind our tests extends to this setting as long as these connected components are large enough and there are not too many of them. The signal strength μ is set to 2.

Performance metrics. Since there are more days without red team events than the opposite, we use the precision-recall curve to evaluate detection performance. More specifically, for each $i \in [100]$, we run our detection procedures on the signals



(A) p-value distribution for different values of the proportion of red team events δ.



(B) Average precision-recall curve with 95% confidence interval for each evaluated test.

FIGURE 7.4: Red team activity detection – Results obtained with our procedures on the LANL event graphs, with synthetic anomaly scores.

$\left\{\mathbf{X}_d^{(i)}\right\}_{d=1}^{58}$ and compute the corresponding precision-recall curves. We then report the average curves and AUCs along with 95% confidence intervals. In addition, we investigate the overall p-value distribution for each of the four subsets defined in Table 7.2. This gives us a finer-grained perspective on the influence of the proportion of red team events.

7.6.2 Results and Discussion

Our results are displayed in Figure 7.4. Starting with the p-value distributions (Figure 7.4a), our procedures behave as expected: in the absence of red team events, the empirical p-value follows a uniform distribution on $[0, 1]$ as it is supposed to. More interestingly, the distribution sensibly changes for $\delta > 5 \cdot 10^{-6}$, and it is well concentrated and close to zero for $\delta > 2 \cdot 10^{-5}$. This suggests that the detectability threshold for $\mu = 2$ is somewhere between these values of δ for both tests. This first result means that detecting the presence of malicious activity through cluster detection in the event graph might be possible: even for remarkably small values of δ, our tests can predict the presence of red team events. This statement should obviously be nuanced: as evidenced in Section 6.5.2, applying graph-based postprocessing methods to real anomaly scores can yield significantly worse results than in the synthetic case. However, our results remain encouraging.

As a side note, DP seems to perform slightly better than PO for $\delta > 5 \cdot 10^{-6}$. Two main explanations can be considered: first, we observed in Section 7.5.2 that DP tends to achieve better performance when looking for small clusters with strong signals, which corresponds rather well to the task at hand here. Secondly, clusters of malicious events are densely connected. This makes graph-based denoising more effective at making them distinguishable, as evidenced in Section 6.5.2.

Finally, the precision-recall curve (Figure 7.4b) confirms our previous observations. More specifically, both detection procedures perform fairly well at detecting days containing red team events – or at least days containing enough of them. In addition, DP globally outperforms PO, further demonstrating the usefulness of the denoising step for the specific case of event graphs.

7.7 Conclusion

Motivated by the use case of malicious event cluster detection in event graphs, we study practical approaches to the more general problem of cluster detection in a network. The specificity of our application (in particular, the size of real-world event graphs) makes traditional scan statistics-based methods hardly usable, leading us to look for more scalable alternatives. The one we choose is the LOC test, which relies on a parallel between cluster detection and percolation theory. By extending this parallel to a dynamic perspective on percolation, we come up with a more practical detection procedure. Moreover, we derive a second test by adding a denoising step to the first one, reusing one of the graph-based denoising tools introduced in Chapter 6. Experiments on a synthetic benchmark dataset show that our tests perform better than other existing methods while keeping computation times reasonable.

We also provide a first assessment of the effectiveness of our tests at detecting malicious activity in event graphs using noisy event-wise anomaly scores. Although we have not experimented with real anomaly scores yet, our initial results are encouraging. Indeed, even with a very low proportion of malicious events and a reasonably small signal strength, our detection procedures successfully detect the presence of red team activity in the LANL event graphs. Experimenting with different values of the signal strength μ as well as real anomaly scores would be a natural lead for future work.

Other interesting directions include a systematic assessment of the detectability of anomalous subgraphs having several connected components. More generally, characterizing detectability using finer-grained notions than the size of the anomalous cluster and the signal strength could be a fruitful direction. In particular, the structure of both the ambient graph and the anomalous cluster obviously have an impact on detectability, and studying this impact (both from a theoretical and empirical perspective) could yield precious insights. Regarding the theoretical point of view, one of the main challenges is that existing work on percolation theory mostly deals with the static setting. Studying the sample paths obtained through the canonical coupling described in Section 7.3.1 would be both a necessary preamble to theoretical analysis of our tests and an interesting problem in and of itself.

Chapter 8

Conclusion

8.1 Summary

Intrusion detection in event logs could roughly be defined as looking for complex anomalies in complex data. As a typical big data-era problem, it calls for carefully tailored solutions, namely a mathematical representation framework, a statistical model and a distinctive anomaly detection procedure. This thesis aims to address these three aspects. Part I formally defines some essential notions and reviews previous work in order to identify the key characteristics of statistical intrusion detection methods for event logs. This systematic approach allows us to rationalize our statistical modelling choices, leading to the DECADES model presented in Part II. Making this model reflect the multiple facets of event logs requires simultaneous use of concepts from various fields, and merging these concepts into a consistent framework is thus the main challenge. Finally, while DECADES shows competitive detection performance with respect to the state of the art, its event-wise predictions do not take advantage of the multi-step nature of malicious activity. Therefore, Part III deals with graph-based postprocessing of event-wise anomaly scores. Using both novel and existing tools, we leverage graph-structured representations of event logs to enhance noisy scores at the event granularity and detect clusters of malicious activity.

Throughout this thesis, experiments on the LANL dataset allow us to evaluate some of our hypotheses on event logs and intrusion-related event sets. We now summarize the outcomes of these experiments.

Modelling the multiple facets of event logs. In Part I, we identify three main characteristics of event logs: they are combinatorial, heterogeneous and temporal data. Experiments with various detection algorithms on the LANL dataset yield basic insights on the respective importance of each of these aspects.

First of all, comparing the static part of DECADES with several baselines in Chapter 4 puts the focus on the combinatorial and heterogeneous aspects. The key importance of the former tends to be confirmed by our results: the best-performing algorithms are those relying on the most accurate description of the combinatorics of events. As for heterogeneity, the fact that DECADES outperforms CADENCE suggests that factoring it in through a joint model for all event types leads to better results. However, the exact cause of this difference in detection performance cannot be reliably isolated based solely on our experiments.

Secondly, the experiments presented in Chapter 5 emphasize the temporal dimension of the data. The absence of significant impact of retraining on detection performance is a somewhat interesting outcome: it suggests that despite the non-stationarity of real-world event streams, the data generating process changes slowly enough so that a static detection model can remain relevant for at least a few weeks.

However, experiments on a longer time range would be needed to get a more accurate picture of the adequate retraining frequency.

Reliably detecting malicious event sets. The existence of intersections between the sets of involved entities corresponding to intrusion-related events, formalized in Assumption 3, is an important specificity of intrusion detection. Indeed, leveraging the peculiar structure of malicious event sets appears as a promising way to make anomaly detection in event logs more reliable. Our experiments in Chapter 6 show that building event graphs reflecting the presence of shared entities can successfully transcribe these structures through the graph-theoretical notion of densely connected cluster. However, we have not yet come up with a fully satisfactory characterization of malicious clusters. In particular, the existence of legitimate activities generating equally dense and suspicious clusters remains an important challenge.

8.2 Perspectives

We conclude this thesis by sketching what we think to be some of the most interesting leads for future work.

Systematically characterizing intrusion detection workflows. The taxonomy of statistical intrusion detection methods presented in Chapter 3 relies on a somewhat basic description of the processing pipeline leading from event logs to security alerts. Refining this description to formally define the basic blocks of intrusion detection workflows for event logs could lead to a higher-level perspective on designing new statistical methodologies. This approach has a long history in the bioinformatics community, with frameworks such as Taverna [Oinn et al., 2004] or Snakemake [Köster and Rahmann, 2012] letting researchers easily reuse and recombine elements of previously published data analysis pipelines. However, to the best of our knowledge, no such framework exists for event log analysis. What comes closest is the anomaly detection language of Memory et al. [Memory et al., 2013], but we are not aware of any widely used implementation of their work.

Redesigning Decades in a more consistent fashion. As mentioned above, the main challenge in building the Decades model was to merge concepts from different fields in order to address the various facets of event logs. As a result, Decades somewhat lacks consistency: the initial training algorithm has a rather frequentist flavour, while the parameter updating procedure relies upon a Bayesian formulation. We think it would be preferable to redesign the whole inference procedure with a fully Bayesian approach, similarly to the work of Lee et al. for host communication graphs [Lee et al., 2021]. Note that this might also require bringing some modifications to the underlying model.

Making anomaly detection practical and explainable. Finally, one aspect we mostly left aside throughout this thesis is the practical usability of our contributions for security experts. This general problem encompasses several issues. The first one pertains to human-machine interfaces and visualization: how can we efficiently display the results of an algorithm such as Decades, or those of the cluster detection tests described in Chapter 7? Moreover, how can we make these results explainable? Regarding this second question, the notion of counterfactual explanation (which we briefly mention in Chapter 4) may be a promising lead: while explaining what makes

a given event anomalous is not easy, building counterfactuals can be an effective way to show how it differs from normal activity patterns. As for the construction of a practical interface, the best solution may be to start from already existing software, such as Security Information and Event Management (SIEM) systems. While our work does not deal with these challenges, addressing them is no less important than building accurate and reliable anomaly detection algorithms in the more general effort of extracting actionable information from large-scale event logs.

Appendix A

Résumé des contributions

A.1 Supervision des réseaux informatiques et données massives

Quel est le point commun entre les journaux Security Auditing de Microsoft Windows, NetFlow et syslog ? Une réponse possible pourrait être la suivante : bien qu'aucun d'entre eux n'ait été initialement conçu pour la détection d'intrusion par l'application à grande échelle d'outils statistiques, ils ont tous fini par être utilisés dans ce but.

Le protocole syslog a été développé dans les années 1980 pour permettre à toutes sortes de terminaux au sein d'un réseau informatique de remonter des événements à un serveur de journalisation central [Gerhards et al., 2009]. Son principal objectif était de distinguer le programme générant les événements de ceux chargés de les remonter et de les enregistrer, facilitant ainsi pour les développeurs l'intégration de fonctions de journalisation dans leurs programmes. Il n'avait donc pas pour vocation de permettre la détection d'intrusion, ni de garantir que les événements générés seraient adaptés à la modélisation statistique. Cependant, une journalisation modulaire et centralisée est un élément essentiel de la supervision de réseaux informatiques à grande échelle. De la même manière, le protocole NetFlow fut introduit dans les années 1990 pour permettre la supervision du trafic au niveau de la couche réseau[1] pour des applications telles que la facturation ou l'identification de dysfonctionnements [Claise et al., 2004]. Même si ses créateurs n'avaient probablement pas cette fin à l'esprit, l'analyse statistique de données NetFlow à grande échelle devint rapidement un axe de recherche important dans le domaine de la sécurité des réseaux [Lakhina et al., 2004]. Enfin, le journal Security Auditing de Windows apparut pour la première fois en 1996 lorsque la notion de source des événements fut introduite dans Windows NT 4.0, rendant plus nette la distinction entre les événements liés à la sécurité (tels que les ouvertures de sessions ou les modifications de fichiers système) et d'autres fonctions de la journalisation, comme l'identification de dysfonctionnements. Tout comme les données NetFlow, les journaux d'événements Windows n'ont pas été initialement conçus pour l'analyse statistique à grande échelle, mais la grande variété d'informations qu'ils contiennent en a fait une proie de choix pour la communauté grandissante des *data miners*.

D'après Efron et Hastie, *"quelque chose d'important a changé dans le monde des statistiques en ce nouveau millénaire"* [Efron and Hastie, 2016][2]. Auparavant, un problème typique d'analyse de données était l'essai clinique : étant donné une question précise (à savoir, ce médicament produit-il les effets attendus ?), le statisticien la formalisait en termes mathématiques, concevait une procédure méticuleuse pour collecter précisément la quantité de données nécessaire selon la théorie, et utilisait

[1] Dans le modèle Open Systems Interconnection à sept couches, la couche réseau est la troisième. Elle gère l'adressage et le routage entre différents réseaux locaux. Le protocole Internet (IP) est un célèbre protocole de couche réseau.

[2] Nous traduisons cette citation et les suivantes.

finalement ces observations pour répondre à la question initiale avec le niveau de confiance désiré. C'était l'époque de la rareté, tant des données que de la puissance de calcul – Efron et Hastie de nouveau :

> Avant l'ère de l'ordinateur, il y a eu celle de la calculatrice, et avant le *big data*, il y avait de petits jeux de données, comportant souvent quelques centaines de nombres ou moins, laborieusement collectés par des scientifiques travaillant seuls et avec des contraintes expérimentales drastiques.

Il semble superflu de préciser que la détection de comportements anormaux par l'analyse statistique à grande échelle de journaux NetFlow ne correspond pas tout à fait à cette description. Au-delà de la différence évidente en matière de volumes de données, cette approche ressemble en effet à l'exact opposé de l'essai clinique d'un point de vue méthodologique : la collecte de données est ici la première étape, et c'est seulement dans un second temps que des statisticiens ont commencé à réfléchir à ce qu'ils pourraient bien faire de cette nouvelle source d'information. D'où une question naturelle : comment le domaine de la statistique s'est-il retrouvé sens dessus dessous ? La réponse d'Efron et Hastie pourrait se résumer en un mot : ordinateur. Ou, de manière moins succincte :

> La technologie informatique permet aux scientifiques de collecter d'immenses jeux de données, d'une taille de plusieurs ordres de grandeur supérieure à celles pour lesquelles la théorie statistique classique a été conçue ; les données massives requièrent de nouvelles méthodes, et ce besoin est satisfait par une vague d'algorithmes statistiques innovants reposant sur le calcul par ordinateur.

Il est plutôt aisé de comprendre comment l'avènement des ordinateurs permet la généralisation de méthodes que les statisticiens de l'ère de la règle à calcul ne pouvaient pas même imaginer. La mécanique par laquelle les ordinateurs fournissent une quantité quasiment infinie de données à analyser mérite cependant une étude plus poussée. En effet, le terme approprié pourrait ici être "informatisation" plutôt qu'"ordinateur".

On définit ici l'informatisation comme une démarche globale consistant à remodeler toutes sortes d'activités par l'utilisation d'ordinateurs (ce dernier terme étant ici à prendre dans son acception étendue, incluant notamment les ordiphones). Écouter de la musique par le biais d'un service de *streaming* constitue ainsi un exemple d'informatisation, au même titre que l'achat de disques sur Internet plutôt que chez un disquaire. L'aspect important est ici le suivant : dès lors qu'un nombre croissant d'activités et autres aspects de la vie quotidienne passent par la médiation d'outils informatiques, de plus en plus de données liées à une variété toujours plus grande de sujets deviennent disponibles. En effet, du fait de la capacité innée des ordinateurs à enregistrer tout ce qui leur arrive, collecter des données sur n'importe quel phénomène devient trivial une fois ce dernier informatisé : plus besoin de dispositifs expérimentaux coûteux ou d'intenses réflexions préalables concernant le choix des données à collecter, toute l'information que l'on pourrait souhaiter avoir est déjà là. Naturellement, le fait d'enregistrer toutes sortes de données avant même de réfléchir à leurs possibles usages soulève de nouveaux problèmes [Jagadish et al., 2014] : la volumétrie, bien sûr, mais aussi l'hétérogénéité des données, leur format parfois peu pratique, et l'omniprésence des erreurs et autres valeurs manquantes.

Les journaux d'événements informatiques, tels que les enregistrements Windows Security Auditing ou NetFlow évoqués ci-dessus, appartiennent à une espèce particulière de coproduits de l'informatisation : encore mieux que des activités médiées par ordinateur, ils décrivent la vie des ordinateurs eux-mêmes. Tout comme n'importe

quelle source de données de l'ère de l'informatique, ils contiennent une certaine quantité d'informations précieuses, dissimulées par leur volume et leur format extravagants. Il est donc tentant de leur appliquer quelque algorithme statistique récent et de voir ce qui en ressort – mais dans quel but, au fait ?

Il se trouve que la réponse à cette question provient également d'une informatisation globale et pas toujours parfaitement réfléchie. En effet, la transhumance généralisée vers Internet a eu pour effet secondaire d'exposer une part toujours croissante de nos vies à la menace également croissante des cyberattaques. La diffusion mondiale de vers informatiques comme ILOVEYOU [Knight, 2000] ou Slammer [Moore et al., 2003] avait déjà laissé entrevoir le potentiel de déstabilisation d'un usage malveillant de l'informatique, mais ce n'était que le début. Très vite, divers acteurs commencèrent à cibler des victimes tout aussi diverses pour toute une variété de raisons. Des attaquants techniquement sophistiqués et soupçonnés d'agir pour le compte d'États furent repérés en train de dérober de la propriété intellectuelle et d'autres données sensibles à des entreprises et des gouvernements [Mandiant, 2013, FireEye, 2015], tandis que d'autres mettaient leur savoir-faire au service d'actions de sabotage [Kerr et al., 2010, Park and Walstrom, 2017] ou d'attaques motivées par des gains financiers [Trautman and Ormerod, 2018]. Les criminels trouvèrent également d'alléchantes opportunités dans le cyberespace, avec pour conséquence des opérations de vol de données ou d'extorsion menant à d'immenses fuites de données privées [NCSC, 2017], des pertes financières colossales [IBM, 2020] et, par ricochet, des décès [Wetsman, 2020].

Face à une menace aussi remuante, les défenseurs des systèmes d'information n'ont guère d'autre choix que d'accroître leurs capacités. Si le durcissement de leurs réseaux et systèmes constitue une protection nécessaire, celle-ci ne saurait être suffisante : avec suffisamment de savoir-faire et de détermination, un attaquant parviendra toujours à contourner ces défenses. Il est donc indispensable de pouvoir détecter les intrusions le plus vite possible, et idéalement dès leur commencement. Les journaux d'événements semblent alors connaître leur moment de gloire : puisqu'ils enregistrent tout ce qui se passe à l'intérieur du réseau supervisé, ils devraient en particulier contenir des traces de toute forme d'activité malveillante. Hélas, ils contiennent aussi (et surtout) une quantité écrasante d'informations sans intérêt – une telle quantité que dans bien des situations, personne n'essaie réellement d'examiner le contenu des journaux. Par conséquent, les intrus parviennent à éviter la détection pendant de longues périodes malgré l'existence de signes indiscutables de leur présence dans les journaux : 207 jours en moyenne d'après l'édition 2020 de l'étude annuelle Cost of a Data Breach réalisée par IBM [IBM, 2020].

Cette situation suscite un besoin d'outils plus sophistiqués pour l'analyse de journaux d'événements, et les méthodes statistiques développées dans les dernières décennies en réponse à la disponibilité croissante de jeux de données volumineux et diversifiés apparaissent comme des candidats idéaux. En un sens, la boucle est ainsi bouclée : après des années d'effervescence durant lesquelles l'informatisation a démultiplié les capacités de collecte et d'analyse de données, favorisant ainsi l'innovation dans le domaine statistique, les fruits de cette innovation sont à présent mis à profit pour combattre certains des effets secondaires indésirables de cette même informatisation. Cette thèse est notre modeste contribution à ce vaste effort. S'inspirant de divers domaines de recherche tels que les systèmes de recommandation, la bioinformatique, la vision par ordinateur ou encore les systèmes dynamiques, elle s'inscrit dans la volonté de construire des outils efficaces pour la détection de comportements malveillants dans les journaux d'événements. Cette tâche n'est assurément pas aisée, et elle recouvre en réalité plusieurs sous-problèmes que nous présentons dans les sections suivantes. Tout d'abord, appliquer un modèle statistique aux journaux d'événements est plus

ardu qu'il n'y paraît : ces données sont en fait particulièrement complexes, ce que nous illustrons dans la Section A.2. De plus, il n'est pas non plus évident de définir un comportement malveillant du point de vue des données, sans même parler de le détecter. Les méthodes statistiques pour la détection d'intrusion contournent typiquement ce problème en recourant à la détection d'anomalies, et ce lien est souligné dans la Section A.3. Nos contributions, résumées dans la Section A.4, couvrent chacun de ces deux aspects.

A.2 Représentation et modélisation de données complexes et multi-facettes

Deux principaux défis compliquent la construction d'un modèle statistique pour les journaux d'événements. Tout d'abord, la complexité des données rend délicat le choix d'une représentation mathématique appropriée. En effet, aucun des objets mathématiques usuellement étudiés en statistique et en analyse de données ne retranscrit pleinement les multiples facettes des journaux d'événements, que nous présentons brièvement dans la Section A.2.1. De plus, ayant conçu une représentation idoine, la construction d'un modèle statistique adéquat demande également une réflexion poussée : comme nous l'expliquons dans la Section A.2.2, prendre en compte toutes les caractéristiques des données nécessite un modèle ad hoc, suscitant lui-même le besoin d'une procédure d'inférence appropriée.

A.2.1 Construction d'une représentation abstraite pour des données complexes

La notion de journal d'événement considérée dans cette thèse recouvre diverses sources de données concrètes, parmi lesquelles on retrouve les journaux Windows Security Auditing et NetFlow évoqués ci-dessus. Bien que chacune de ces variantes possède ses traits distinctifs, nous nous focalisons sur ce qu'elles ont en commun afin de ne pas restreindre les cas d'application de nos contributions. En particulier, la principale spécificité de toutes les sources de données que nous considérons est qu'elles ne s'insèrent naturellement dans aucun formalisme mathématique usuel.

Les journaux d'événements sont multi-facettes. Les journaux d'événements sont en premier lieu des données combinatoires, en ce que chaque événement est défini par un nombre fini de champs contenant en majorité des valeurs discrètes : noms d'utilisateurs et d'hôtes, adresses IP, etc. Cependant, ils sont aussi intrinsèquement hétérogènes : les entités désignées par chacun de ces champs, de même que les événements eux-mêmes, peuvent être de différents types. En d'autres termes, un utilisateur n'est pas équivalent à une adresse IP, et une création de processus n'a pas la même signification qu'un enregistrement NetFlow d'une communication entre deux hôtes. Enfin, chaque événement est horodaté, et la probabilité d'observer un certain événement n'est pas nécessairement constante dans le temps : elle peut en effet montrer des variations saisonnières ou une évolution de plus long terme. De plus, on peut s'attendre à des dépendances statistiques entre des événements successifs : par exemple, la probabilité d'une création de processus impliquant l'utilisateur U et l'hôte H devrait intuitivement être plus élevée si U a récemment ouvert une session sur H. Ces deux propriétés dotent les journaux d'événements d'une dimension temporelle.

Bien qu'aucune de ces trois caractéristiques ne soit inconnue du statisticien expérimenté, il est plus rare de les voir toutes trois réunies. En conséquence, aucun des

objets mathématiques habituellement étudiés ne représente parfaitement ces multiples facettes : les matrices et les tenseurs peuvent par exemple traduire adéquatement des données combinatoires, mais leur usage devient inadapté lorsque des événements de différents types décrits par un nombre variable de champs sont considérés. De même, les processus ponctuels et les séries temporelles peuvent restituer pleinement la dimension temporelle des séquences d'événements, mais inclure un caractère combinatoire dans l'un ou l'autre de ces formalismes n'a rien d'immédiat.

Deux directions opposées. Puisque les journaux d'événements ne peuvent être parfaitement représentés par les objets mathématiques usuels, les outils statistiques existants ne peuvent pas être appliqués directement : l'important fossé sémantique séparant les données du modèle doit être comblé. La majorité des travaux existants résout ce problème en adaptant les données au modèle, simplifiant suffisamment celles-ci pour les rendre compatibles avec un cadre théorique classique [Yen et al., 2013, Legg et al., 2015, Hu et al., 2017]. Cela induit une perte significative d'information, limitant en fin de compte les performances de la procédure de détection dans son ensemble : une fois les aspects importants des données effacés, même le plus sophistiqué des algorithmes est incapable de les retrouver. Par conséquent, des contributions récentes ont adopté une approche diamétralement opposée, proposant des modèles mieux adaptés plutôt que de simplifier à outrance les données [Tuor et al., 2018, Amin et al., 2019, Leichtnam et al., 2020b]. En particulier, des avancées importantes ont été réalisées concernant la modélisation de la combinatoire des événements. Nous appuyant sur ces percées récentes, nous proposons une représentation abstraite des journaux d'événements tournant autour de la notion d'interaction.

Une approche centrée sur les interactions. L'intuition fondamentale qui sous-tend cette thèse est que les événements devraient être considérés comme des interactions entre entités. En plus de mettre en exergue leur caractère combinatoire, cette description suggère également que les événements résultent de l'intrication de comportements individuels. En d'autres termes, elle repose sur une dualité entité-événement : les événements passés nous permettent d'inférer le rôle et le comportement habituel de chaque entité (utilisateur, hôte, processus, etc.), et les événements futurs peuvent alors être prédits en s'appuyant sur cette connaissance. La principale motivation de cette approche réside dans le constat que les événements malveillants sont souvent des interactions rares. Par exemple, un attaquant utilisant des identifiants volés pour explorer un réseau compromis devrait générer plusieurs interactions inhabituelles entre le compte utilisateur correspondant, un hôte source et diverses destinations. Il semble donc pertinent de se focaliser sur les interactions dans le cadre de la détection d'intrusion. Cependant, une telle approche soulève de nouvelles difficultés en matière de modélisation statistique.

A.2.2 Modélisation statistique de données non-numériques

La modélisation à grande échelle d'interactions entre entités est un sujet relativement récent en statistique – typiquement l'un de ceux dont la popularité a été accrue par l'informatisation. Au-delà des problématiques liées à la nature combinatoire de telles données, il faut également tenir compte des deux caractéristiques supplémentaires identifiées dans la section précédente, c'est-à-dire les aspects hétérogène et temporel des journaux d'événements. Cela rend particulièrement délicate la conception d'un modèle adéquat.

Modélisation de données combinatoires. D'une manière générale, une variable aléatoire combinatoire est une variable aléatoire discrète dont les réalisations sont issues d'un ensemble combinatoire, comme par exemple l'ensemble des parties d'un ensemble fini [Bekkerman et al., 2006]. Le principal défi lié à la modélisation de telles variables provient de la grande dimension de l'espace des observations. Un exemple lié aux journaux d'événements peut être formulé comme suit : soit $\mathcal{U}$ un ensemble d'utilisateurs, $\mathcal{S}$ un ensemble d'hôtes sources et $\mathcal{D}$ un ensemble d'hôtes destinations. Un événement d'ouverture de session distante peut alors être vu comme un triplet $T = (U, S, D) \in \mathcal{U} \times \mathcal{S} \times \mathcal{D}$ indiquant que l'utilisateur U s'est connecté depuis S vers D. On se donne ici pour but d'estimer la distribution de T à partir d'un ensemble de n événements passés, noté $\{(u_i, s_i, d_i)\}_{i=1}^{n}$. Dans un réseau informatique réel, chacun des ensembles $\mathcal{U}$, $\mathcal{S}$ et $\mathcal{D}$ peut typiquement contenir des milliers, voire des dizaines de milliers d'éléments. Le nombre de valeurs possibles pour T est donc compris entre 10^9 et 10^{12}. Sans hypothèse supplémentaire sur la distribution sous-jacente, le nombre d'observations n doit ainsi être immense pour réaliser une quelconque inférence de manière raisonnablement fiable. Pire encore, la taille de l'espace des observations croît exponentiellement avec le nombre d'entités impliquées dans les événements considérés, et ce dernier n'a aucune raison d'être limité à trois.

La construction d'un modèle fiable pour des données combinatoires passe donc nécessairement par une forme de réduction dimensionnelle. Nous nous appuyons pour cela sur le concept de modèle à espace d'états [Turnbull, 2020]. L'intuition essentielle qui sous-tend cette approche est que la propension d'une entité donnée à interagir avec les autres dépend d'un petit nombre d d'attributs latents de cette entité. En considérant à nouveau l'exemple des ouvertures de session, soit $g : \mathcal{U} \cup \mathcal{S} \cup \mathcal{D} \to \mathbb{R}^d$ la fonction inconnue associant à chaque entité son vecteur d'attributs. Un modèle raisonnable pour le triplet aléatoire T peut alors être défini par

$$\forall (u, s, d) \in \mathcal{U} \times \mathcal{S} \times \mathcal{D},\ \mathbb{P}\big[T = (u, s, d)\big] = f\big(g(u), g(s), g(d)\big), \tag{A.1}$$

avec $f : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$ une fonction d'affinité connue. L'inférence de la distribution complète de T se ramène alors à l'estimation des d attributs latents de chaque entité, soit $d(|\mathcal{U}| + |\mathcal{S}| + |\mathcal{D}|)$ paramètres à apprendre – une réduction significative par rapport aux $|\mathcal{U}| \cdot |\mathcal{S}| \cdot |\mathcal{D}|$ paramètres correspondant à l'estimation naïve de la fonction de masse de probabilité.

Des modèles à espace d'états ont été proposés pour diverses sortes d'interactions, notamment certaines liées aux journaux d'événements et à la détection d'intrusion [Turcotte et al., 2016a, Amin et al., 2019, Lee et al., 2021, Sanna Passino et al., 2020]. Cependant, ces contributions étudient principalement des interactions dyadiques, ce qui s'avère insuffisant dans le cas des journaux d'événements : comme le montre l'exemple des ouvertures de session mentionné ci-dessus, supposer que les événements n'impliquent jamais plus de deux entités semble excessivement réducteur. Par conséquent, une partie de nos travaux (présentée dans le Chapitre 4) consiste en la conception d'un modèle à espace d'états adapté aux interactions polyadiques.

Prise en compte de l'hétérogénéité. Un autre trait spécifique aux journaux d'événements par rapport à d'autres types d'interactions est leur hétérogénéité, qui se manifeste essentiellement par l'existence de plusieurs types d'événements. Cette spécificité ne peut être ignorée dans la conception d'un modèle statistique : le fait qu'un utilisateur U s'authentifie usuellement auprès d'un serveur S ne devrait ainsi clairement pas mener à penser que U est censé modifier régulièrement des fichiers système sur S. Par conséquent, notre modèle à espace d'états doit tenir compte des

différences entre types d'événements. En reprenant les notations de l'Équation A.1, il semble raisonnable d'intégrer cette distinction en rendant la fonction d'affinité f spécifique à un type d'événement, tout en réutilisant la même fonction de représentation g pour tous les types. Cette intuition peut être rattachée au champ de l'apprentissage multitâche [Caruana, 1997] : il a ainsi été montré que l'apprentissage simultané de plusieurs tâches liées les unes aux autres donnait de meilleures performances que l'apprentissage indépendant de chacune des tâches. Une interprétation répandue de ce phénomène consiste à voir les tâches auxiliaires comme une source de biais inductif. En d'autres termes, en cherchant un modèle performant pour toutes les tâches, les algorithmes d'apprentissage multitâche restreignent la classe des modèles possibles pour chacune des tâches, rendant ainsi l'apprentissage plus efficace. Appliqué aux journaux d'événements et aux modèles à espace d'états, ce constat implique que la recherche d'attributs latents expliquant le comportement d'une entité pour tous les types d'événements est susceptible d'aboutir à un modèle plus réaliste.

Le principal défi de l'apprentissage multitâche consiste à trouver un équilibre adéquat entre les différentes tâches. En effet, minimiser conjointement l'erreur commise par le modèle pour chacune des tâches se ramène à un problème d'optimisation multi-objectif, ce qui est significativement plus complexe que le problème usuel de l'apprentissage d'une seule tâche. Par exemple, une certaine mise à jour des attributs latents de l'utilisateur U pourrait mener à de meilleures prédictions pour les authentifications effectuées par U tout en dégradant les prédictions liées aux créations de processus impliquant U. Que faire dans une telle situation ? Afin de répondre à cette question et de proposer une procédure d'apprentissage adaptée pour notre modèle, nous nous inspirons de travaux antérieurs dans le domaine de l'apprentissage multitâche [Kendall et al., 2018].

Adaptation à des flux de données non-stationnaires. Enfin, la dimension temporelle des journaux d'événements doit être prise en compte dans la construction d'un modèle statistique. En particulier, les modes d'interaction habituels dans un réseau supervisé ne peuvent pas raisonnablement être supposés constants dans le temps : les utilisateurs endossent de nouveaux rôles organisationnels ou commencent à travailler sur de nouveaux projets, tandis que les serveurs se mettent à héberger de nouvelles applications, et ainsi de suite. De plus, de nouvelles entités apparaissent fréquemment (comptes utilisateurs pour des nouveaux venus ou nouvelles machines, par exemple). Par conséquent, un modèle statique risque de devenir rapidement obsolète dans un contexte réaliste de supervision de sécurité. Heureusement, les modèles à espace d'états peuvent être naturellement étendus à la modélisation de flux d'interactions non-stationnaires.

En effet, puisque la probabilité d'une interaction donnée ne dépend que des attributs latents des entités impliquées, maintenir le modèle à jour ne requiert qu'un suivi de l'évolution de ces attributs latents. En réutilisant encore une fois les notations de l'Équation A.1, cela revient à remplacer la fonction g par une suite $\{g_t\}_{t\geq 0}$, où chaque indice $t \geq 0$ représente un pas temporel. Cela peut se comprendre dans le contexte des journaux d'événements : les changements dans la structure des interactions observées résultent de modifications liées aux entités, comme l'apparition d'un nouvel hôte dans le réseau ou un changement dans les habitudes d'un utilisateur.

Du point de vue de l'inférence, une analogie peut être tracée entre l'utilisation d'événements observés pour suivre l'évolution de la fonction latente g et le cadre théorique plus général des modèles de Markov cachés. En effet, les événements observés entre les pas temporels $t-1$ et t peuvent être vus comme des échantillons d'une loi d'émission Q_t dépendant de la fonction de représentation g_t. De même, la

séquence $\{g_t\}_{t\geq 0}$ peut être vue comme un processus markovien latent caractérisé par un noyau de transition P. L'inférence de la fonction de représentation g_t à partir des événements observés jusqu'à l'instant t se ramène alors à un problème de filtrage. Nous nous appuyons sur ce parallèle dans le Chapitre 5 pour concevoir une procédure de mise à jour de notre modèle à espace d'états.

A.3 Détection robuste d'anomalies pertinentes

La construction d'un modèle statistique pour les journaux d'événements est un premier pas vers leur exploitation à des fins de sécurité. Cependant, elle ne permet pas directement la détection d'intrusion : des efforts supplémentaires sont nécessaires pour repérer des comportements malveillants à l'aide de ce modèle. L'idée centrale est que les événements découlant d'une intrusion devraient être anormaux et, par conséquent, improbables d'après le modèle, comme expliqué dans la Section A.3.1. Cependant, de nombreux événements inhabituels sont constamment observés dans un réseau informatique réel. De ce fait, la simple recherche d'événements improbables génère plus de faux positifs que de réelles détections, et la Section A.3.2 présente les améliorations que nous proposons pour dépasser ce problème.

A.3.1 Les événements malveillants sont anormaux...

Si la détection d'intrusion est un problème remarquablement difficile, c'est parce que les comportements malveillants ne peuvent être caractérisés d'une manière à la fois générique et fiable : leur seul trait commun est que l'administrateur légitime du réseau ciblé les considère comme indésirables, et cette propriété n'apparaît évidemment pas explicitement dans les journaux. Seules deux approches sont donc envisageables : compiler des descriptions explicites du plus grand nombre possible d'actions malveillantes spécifiques, ou tenter de définir une caractérisation indirecte mais plus générale. C'est dans cette seconde approche que s'inscrit cette thèse.

Une caractérisation indirecte. L'ensemble des événements qui pourraient théoriquement résulter d'un comportement malveillant est à la fois immense et complexe. Considérons par exemple un attaquant tentant de s'introduire dans le réseau ciblé en collectant des identifiants par le biais d'une campagne de hameçonnage. Ayant passé avec succès cette première étape, l'attaquant se connectera typiquement à un point d'accès de réseau privé virtuel (VPN) avec les identifiants volés, puis il utilisera ces mêmes identifiants pour se latéraliser au sein du réseau et collecter des informations. Du point de vue des journaux d'événements, on peut attendre d'un tel comportement qu'il génère toutes sortes d'authentifications distantes : les identifiants volés peuvent appartenir à n'importe quel utilisateur, et les hôtes sources et destinations peuvent également correspondre à n'importe quelle machine du réseau interne. De plus, aucune spécificité ne distingue les événements malveillants des autres : pris indépendamment, chaque champ de chaque événement lié à l'intrusion semble en tout point légitime. Comme évoqué plus haut (et développé plus amplement dans le Chapitre 2), la seule caractéristique que l'on peut raisonnablement s'attendre à trouver chez les événements malveillants est qu'ils impliquent des entités qui n'interagissent pas ensemble d'habitude. Par conséquent, plutôt que d'essayer de décrire l'ensemble des authentifications distantes potentiellement malveillantes, une approche plus prosaïque consiste à caractériser les motifs d'interaction correspondant aux comportements légitimes

habituels. Un événement futur peut alors être considéré comme suspect s'il ne correspond pas à cette caractérisation. Cette approche est communément appelée détection d'anomalies.

Détection d'anomalies combinatoires. Si la détection d'anomalies est un sujet largement étudié [Chandola et al., 2009], son application à des données combinatoires soulève des problématiques spécifiques. En effet, une anomalie se définit intuitivement comme une observation de faible probabilité, et la détection de telles observations se ramène ainsi à la construction d'un modèle statistique à partir de données normales, puis au calcul d'une probabilité estimée pour chaque nouvelle observation à l'aide de ce modèle. Cependant, une loi conjointe sur un espace combinatoire n'est en réalité que peu informative. Supposons par exemple que l'on observe une authentification distante $(u, s, d) \in \mathcal{U} \times \mathcal{S} \times \mathcal{D}$ avec une faible probabilité estimée. Plusieurs explications peuvent en fait être avancées pour cette faible probabilité : u pourrait par exemple être censé se connecter à d, mais pas depuis s. On peut également penser que u n'est censé interagir ni avec s, ni avec d. Plus généralement, toute partie non-vide de l'ensemble des entités impliquées dans une interaction donnée peut fournir une raison suffisante pour considérer l'interaction tout entière comme suspecte du point de vue de la sécurité. Les algorithmes de détection d'anomalies combinatoires doivent donc reposer sur une description plus fine de la distribution. Ce problème a été abordé dans la littérature liée à la fouille de données et à l'apprentissage statistique [Das and Schneider, 2007, Akoglu et al., 2012, Amin et al., 2019], et nous nous inspirons de ces contributions passées pour concevoir notre algorithme de détection d'événements anormaux dans le Chapitre 4.

A.3.2 ...mais tous les événements anormaux ne sont pas malveillants

En supposant que les événements résultant d'une intrusion sont anormaux, être en capacité de détecter des interactions anormales entre entités est une condition nécessaire de la détection d'intrusion. Celle-ci n'est cependant pas suffisante : de nombreux événements légitimes sont également anormaux au sens statistique, et distinguer ces événements de ceux que l'on souhaite réellement détecter requiert un nouveau traitement, ainsi que de nouvelles hypothèses.

Des anomalies partout. À chaque fois qu'un salarié quelconque clique au mauvais endroit et ouvre le mauvais partage réseau dans son interface graphique, un événement anormal est généré. Cependant, on se convaincra aisément que l'administrateur du réseau ne souhaite pas recevoir de notification dans un tel cas. Plus généralement, de nombreuses activités légitimes se produisent de manière irrégulière – on pensera par exemple au déploiement d'applications ou à d'autres tâches administratives. Du fait de cette irrégularité, ces activités seront probablement déclarées anormales par les modèles statistiques. Il est important de comprendre qu'il ne s'agit pas là d'une simple imperfection du modèle, que l'on pourrait pallier avec plus de données d'entraînement ou de meilleurs algorithmes : il existe tout simplement un pan entier du comportement humain qui ne se prête pas à la prédiction. Par conséquent, il n'est pas suffisant de détecter des événements anormaux : des étapes de traitement supplémentaires doivent être mises en place pour éviter un taux de faux positifs inacceptable. Dans cette thèse, ces étapes supplémentaires reposent sur une hypothèse fondamentale : les événements liés à une intrusion sont non seulement anormaux, mais aussi reliés les uns aux autres par les entités qu'ils impliquent.

Exploitation des liens entre événements. Revenons à l'exemple du hameçonnage introduit plus haut. Les événements d'authentification distante découlant de l'exploration du réseau par l'attaquant devraient théoriquement impliquer certaines entités communes : en effet, chaque nouveau mouvement latéral effectué par l'attaquant doit partir d'un hôte déjà compromis, et des identifiants volés pourraient également être utilisés plus d'une fois. Plus généralement, chaque action effectuée par un attaquant devrait impliquer au moins une entité préalablement compromise, ce qui génère des intersections entre les événements sous-jacents.

Ces relations peuvent typiquement être résumées par un graphe dont les sommets représentent les événements. Une arête entre deux événements traduit alors une forme de similarité, qui doit être définie de telle manière que les événements liés à une intrusion forment un sous-graphe densément connecté [Pei et al., 2016, Leichtnam et al., 2020a]. Un tel graphe ayant été construit, les scores d'anormalité associés aux événements peuvent être vus comme un signal bruité structuré en graphe. Les événements rares mais légitimes devraient alors correspondre à des pics isolés dans ce signal, tandis que les grappes d'événements anormaux devraient apparaître sous la forme de plateaux hauts. Nous déduisons deux méthodes de post-traitement de cette intuition. Tout d'abord, comme nous l'expliquons dans le Chapitre 6, des outils de traitement de signal sur graphe peuvent être employés pour débruiter les scores d'anormalité, effaçant les faux positifs tout en préservant les scores élevés des événements malveillants. Par ailleurs, les sous-graphes connexes d'événements anormaux peuvent être détectés automatiquement avant de déclencher une quelconque investigation manuelle, aidant ainsi les analystes à concentrer leurs efforts sur les parties les plus suspectes des journaux. Des méthodes visant à détecter de tels sous-graphes sont présentées dans le Chapitre 7.

A.4 Contributions de cette thèse

En résumé, cette thèse aborde trois défis liés à l'analyse statistique de journaux d'événements et à la détection d'intrusion : la représentation et la modélisation des données, la détection d'événements anormaux et le post-traitement de scores d'anormalité dans le but de classer les anomalies comme bénignes ou malveillantes.

Principales contributions. Notre première contribution est une définition rigoureuse de la détection d'intrusion dans les journaux d'événements ainsi qu'une classification détaillée des méthodes existantes. Cette formalisation du problème nous permet d'identifier les caractéristiques les plus importantes des données, et nous nous appuyons sur cette connaissance pour concevoir un algorithme de détection d'événements anormaux reposant sur la modélisation par espace d'états et l'apprentissage multitâche. Nous comparons cet algorithme avec des approches concurrentes issues de la littérature et obtenons de meilleures performances de détection sur un jeu de données réel. Notre implémentation du modèle proposé est publiquement accessible. Enfin, nous proposons une approche reposant sur les graphes d'événements pour le post-traitement de scores d'anormalité. Notre méthodologie comprend deux aspects : nous expérimentons tout d'abord l'utilisation de deux outils de traitement de signal sur graphe pour débruiter les scores d'anormalité en s'appuyant sur la structure du graphe des événements. Nous étudions ensuite le problème de la détection de sous-graphe anormal dans un graphe dont les sommets portent des observations scalaires. Dans ce cadre, nous proposons deux nouvelles procédures de détection et les évaluons

contre des méthodes existantes sur un jeu de données synthétique. Nos procédures sont également appliquées à des graphes d'événements réels.

Sur l'évaluation des méthodes de détection. Il n'est pas simple d'évaluer la pertinence et l'utilité de nos contributions. En effet, les modèles et algorithmes que nous proposons sont conçus sur mesure pour un cas d'usage spécifique, ce qui les rend peu adaptés à l'analyse théorique. De plus, l'évaluation empirique s'avère également complexe : du fait de la grande diversité des réseaux informatiques et des comportements malveillants, les résultats obtenus sur un seul jeu de données ne donnent qu'une vision partielle. Cependant, en l'absence d'un meilleur critère, nous évaluons l'applicabilité et l'efficacité de nos algorithmes sur un jeu de données réel publié par le Los Alamos National Laboratory, intitulé "Comprehensive, Multi-Source Cyber-Security Events" [Kent, 2015a, Kent, 2015b]. L'intérêt de ce jeu de données réside en grande partie dans la présence de traces annotées d'un test d'intrusion (autrement dit, une intrusion réalisée par des experts en sécurité pour évaluer le niveau de sécurité du réseau). Par conséquent, la capacité de nos algorithmes à détecter de l'activité malveillante peut être estimée à l'aide de ces annotations. Nous décrivons le jeu de données plus en détail dans le Chapitre 2.

Structure de la thèse. La suite de cette thèse est divisée en trois parties, chacune d'entre elles détaillant l'une de nos trois principales contributions : formalisation et classification de la littérature, détection d'événements anormaux, et post-traitement de scores d'anormalité à l'aide de graphes d'événements. Chacun de ces sujets peut être associé à un maillon d'une chaîne globale de traitement des journaux d'événements, illustrée par la Figure A.1.

La Partie I pose tout d'abord le décor en définissant formellement le problème considéré et en passant en revue les travaux existants. Nous introduisons quelques notions élémentaires dans le Chapitre 2, en particulier celles d'événement, de journal et d'intrusion. En partant de ces définitions, nous proposons une formulation générique de la détection d'intrusion dans des journaux d'événements et mettons en évidence certaines des difficultés liées à ce problème. Nous nous appuyons ensuite sur ce formalisme pour décrire et comparer les algorithmes de détection publiés précédemment, ce qui aboutit à une taxinomie des approches existantes dans le Chapitre 3. Cette taxinomie met l'accent sur les choix de représentation ainsi que les hypothèses de modélisation caractérisant chacun des travaux considérés, et nous relions ces choix et hypothèses aux diverses facettes des données. Une partie des travaux présentés dans cette partie a été publiée à la *Conference on Artificial Intelligence for Defense* (CAID 2020) [Larroche et al., 2020b].

La Partie II présente ensuite notre algorithme de détection d'événements anormaux, ainsi que le modèle statistique et la procédure d'inférence sous-jacents. Le but de notre algorithme est de tenir compte des trois principaux aspects des journaux d'événements, à savoir les dimensions combinatoire, hétérogène et temporelle. Les deux premiers aspects sont traités dans le Chapitre 4, qui décrit notre modèle à espace d'états pour les interactions polyadiques hétérogènes. Nous détaillons également l'algorithme d'apprentissage associé, en mettant en évidence les défis supplémentaires liés à l'existence de plusieurs types d'événements. Le Chapitre 5 ajoute ensuite l'aspect temporel, tirant parti du parallèle entre les modèles à espace d'états et le filtrage bayésien pour concevoir une procédure de mise à jour des paramètres de notre modèle. Des parties de ces deux chapitres ont été prépubliées sans revue par les pairs [Larroche et al., 2021b].

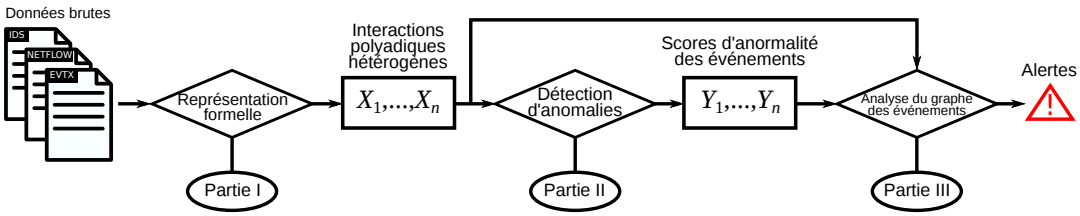


FIGURE A.1: Chaîne de traitement des journaux d'événements et parties correspondantes de la thèse.

Enfin, la Partie III a pour thème le post-traitement de scores d'anormalité à l'aide de graphes d'événements. Le Chapitre 6 décrit notre procédure de construction des graphes d'événements et présente des outils de traitement de signal sur graphe pouvant être utilisés pour débruiter les scores. Nous étudions ensuite la détection de sous-graphe anormal dans le Chapitre 7, cette tâche étant définie de manière générique comme un test d'hypothèse combinatoire. Afin de pallier le coût élevé des méthodes de détection existantes, nous proposons deux tests peu onéreux reposant sur la théorie de la percolation. Ces tests sont évalués contre des méthodes précédemment publiées sur un jeu de données synthétique, et nous proposons également une première estimation de leur efficacité pour des graphes d'événements réels. Plusieurs parties du contenu du Chapitre 7 ont été publiées au *Symposium on Intelligent Data Analysis* (IDA 2020) [Larroche et al., 2020a] et au *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* (ESANN 2021) [Larroche et al., 2021a].

Bibliography

[Addario-Berry et al., 2010] Addario-Berry, L., Broutin, N., Devroye, L., and Lugosi, G. (2010). On combinatorial testing problems. *Ann. Stat.*, 38(5).

[Adilova et al., 2019] Adilova, L., Natious, L., Chen, S., Thonnard, O., and Kamp, M. (2019). System misuse detection via informed behavior clustering and modeling. In *DSN-W*.

[Agarwal et al., 2006] Agarwal, S., Branson, K., and Belongie, S. (2006). Higher order learning with graphs. In *ICML*.

[Akoglu et al., 2012] Akoglu, L., Tong, H., Vreeken, J., and Faloutsos, C. (2012). Fast and reliable anomaly detection in categorical data. In *CIKM*.

[Aksoy et al., 2019] Aksoy, S. G., Nowak, K. E., Purvine, E., and Young, S. J. (2019). Relative hausdorff distance for network analysis. *Appl. Netw. Sci.*, 4(1):80.

[Aksoylar et al., 2017] Aksoylar, C., Orecchia, L., and Saligrama, V. (2017). Connected subgraph detection with mirror descent on sdps. In *ICML*.

[Aldairi et al., 2019] Aldairi, M., Karimi, L., and Joshi, J. (2019). A trust aware unsupervised learning approach for insider threat detection. In *IRI*.

[Amin et al., 2019] Amin, M. R., Garg, P., and Coskun, B. (2019). Cadence: Conditional anomaly detection for events using noise-contrastive estimation. In *AISec*.

[Arias-Castro et al., 2011] Arias-Castro, E., Candes, E. J., and Durand, A. (2011). Detection of an anomalous cluster in a network. *Ann. Stat.*, 39(1).

[Arias-Castro et al., 2008] Arias-Castro, E., Candès, E. J., Helgason, H., Zeitouni, O., et al. (2008). Searching for a trail of evidence in a maze. *Ann. Stat.*, 36(4).

[Arias-Castro et al., 2005] Arias-Castro, E., Donoho, D. L., Huo, X., et al. (2005). Near-optimal detection of geometric objects by fast multiscale methods. *IEEE Trans. Inf. Theory*, 51(7).

[Arias-Castro and Grimmett, 2013] Arias-Castro, E. and Grimmett, G. R. (2013). Cluster detection in networks using percolation. *Bernoulli*, 19(2).

[Barabási and Albert, 1999] Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286(5439).

[Behnel et al., 2010] Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., and Smith, K. (2010). Cython: The best of both worlds. *Comput. Sci. Eng.*, 13(2):31–39.

[Bekkerman et al., 2006] Bekkerman, R., Sahami, M., and Learned-Miller, E. (2006). Combinatorial markov random fields. In *ECML*.

[Belkin and Niyogi, 2003] Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396.

[Bergman, 1999] Bergman, N. (1999). *Recursive Bayesian estimation: Navigation and tracking applications.* PhD thesis, Linköping University.

[Bhattacharjee et al., 2017] Bhattacharjee, S. D., Yuan, J., Jiaqi, Z., and Tan, Y.-P. (2017). Context-aware graph-based analysis for detecting anomalous activities. In *ICME*.

[Billsus et al., 1998] Billsus, D., Pazzani, M. J., et al. (1998). Learning collaborative information filters. In *ICML*.

[Bohara et al., 2017] Bohara, A., Noureddine, M. A., Fawaz, A., and Sanders, W. H. (2017). An unsupervised multi-detector approach for identifying malicious lateral movement. In *SRDS*.

[Bohara et al., 2016] Bohara, A., Thakore, U., and Sanders, W. H. (2016). Intrusion detection in enterprise systems by combining and clustering diverse monitor data. In *HotSoS*.

[Böse et al., 2017] Böse, B., Avasarala, B., Tirthapura, S., Chung, Y.-Y., and Steiner, D. (2017). Detecting insider threats using radish: a system for real-time anomaly detection in heterogeneous data streams. *IEEE Syst. J.*, 11(2):471–482.

[Bowman et al., 2020] Bowman, B., Laprade, C., Ji, Y., and Huang, H. H. (2020). Detecting lateral movement in enterprise computer networks with unsupervised graph AI. In *RAID*.

[Breunig et al., 2000] Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). Lof: identifying density-based local outliers. In *SIGMOD*.

[Brown et al., 2018] Brown, A., Tuor, A., Hutchinson, B., and Nichols, N. (2018). Recurrent neural network attention mechanisms for interpretable system log anomaly detection. In *MLCS*.

[Buczak and Guven, 2015] Buczak, A. L. and Guven, E. (2015). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun. Surveys Tuts.*, 18(2):1153–1176.

[Callaway et al., 2000] Callaway, D. S., Newman, M. E., Strogatz, S. H., and Watts, D. J. (2000). Network robustness and fragility: Percolation on random graphs. *Phys. Rev. Lett.*, 85(25).

[Cappé et al., 2006] Cappé, O., Moulines, E., and Rydén, T. (2006). *Inference in hidden Markov models.* Springer.

[Carroll and Chang, 1970] Carroll, J. D. and Chang, J.-J. (1970). Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319.

[Caruana, 1997] Caruana, R. (1997). Multitask learning. *Mach. Learn.*, 28(1):41–75.

[Chandola et al., 2009] Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):1–58.

[Chandola et al., 2010] Chandola, V., Banerjee, A., and Kumar, V. (2010). Anomaly detection for discrete sequences: A survey. *IEEE Trans. Knowl. Data Eng.*, 24(5):823–839.

[Chang et al., 2017] Chang, S., Zhang, Y., Tang, J., Yin, D., Chang, Y., Hasegawa-Johnson, M. A., and Huang, T. S. (2017). Streaming recommender systems. In *WWW*.

[Chattopadhyay et al., 2018] Chattopadhyay, P., Wang, L., and Tan, Y.-P. (2018). Scenario-based insider threat detection from cyber activities. *IEEE Trans. Comput. Soc. Syst.*, 5(3):660–675.

[Chen et al., 2016] Chen, T., Tang, L.-A., Sun, Y., Chen, Z., and Zhang, K. (2016). Entity embedding-based anomaly detection for heterogeneous categorical events. In *IJCAI*.

[Chen and Malin, 2011] Chen, Y. and Malin, B. (2011). Detection of anomalous insiders in collaborative environments via relational analysis of access logs. In *CODASPY*.

[Chen et al., 2018] Chen, Z., Badrinarayanan, V., Lee, C.-Y., and Rabinovich, A. (2018). Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*.

[Chi and Kolda, 2012] Chi, E. C. and Kolda, T. G. (2012). On tensors, sparsity, and nonnegative factorizations. *SIAM J. Matrix Anal. Appl.*, 33(4):1272–1299.

[Chuang et al., 2007] Chuang, H.-Y., Lee, E., Liu, Y.-T., Lee, D., and Ideker, T. (2007). Network-based classification of breast cancer metastasis. *Mol. Syst. Biol.*, 3(1):140.

[Chung, 1997] Chung, F. (1997). *Spectral graph theory*. American Mathematical Soc.

[Chung et al., 2009] Chung, F., Horn, P., and Lu, L. (2009). Percolation in general graphs. *Internet Mathematics*, 6(3).

[Claise et al., 2004] Claise, B., Sadasivan, G., Valluri, V., and Djernaes, M. (2004). Cisco systems netflow services export version 9. Technical report, RFC 3954, October.

[Collobert and Weston, 2008] Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*.

[Cuppens, 2001] Cuppens, F. (2001). Managing alerts in a multi-intrusion detection environment. In *ACSAC*.

[Cuppens and Miege, 2002] Cuppens, F. and Miege, A. (2002). Alert correlation in a cooperative intrusion detection framework. In *S&P*.

[Das and Schneider, 2007] Das, K. and Schneider, J. (2007). Detecting anomalous records in categorical datasets. In *KDD*.

[Debar et al., 1992] Debar, H., Becker, M., and Siboni, D. (1992). A neural network component for an intrusion detection system. In *S&P*.

[Debar and Wespi, 2001] Debar, H. and Wespi, A. (2001). Aggregation and correlation of intrusion-detection alerts. In *RAID*.

[Defferrard et al., 2017] Defferrard, M., Martin, L., Pena, R., and Perraudin, N. (2017). Pygsp: Graph signal processing in python.

[Denning, 1987] Denning, D. E. (1987). An intrusion detection model. *IEEE Trans. Software Eng.*, (2):222–232.

[Du et al., 2017] Du, M., Li, F., Zheng, G., and Srikumar, V. (2017). Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *CCS*.

[Duczmal and Assuncao, 2004] Duczmal, L. and Assuncao, R. (2004). A simulated annealing strategy for the detection of arbitrarily shaped spatial clusters. *Comput. Stat. Data Anal.*, 45(2).

[Duczmal et al., 2006] Duczmal, L., Kulldorff, M., and Huang, L. (2006). Evaluation of spatial scan statistics for irregularly shaped clusters. *J. Comput. Graph. Stat.*, 15(2).

[Efron, 1979] Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *Ann. Stat.*, 7(1):1–26.

[Efron, 1987] Efron, B. (1987). Better bootstrap confidence intervals. *J. Am. Stat. Assoc.*, 82(397):171–185.

[Efron and Hastie, 2016] Efron, B. and Hastie, T. (2016). *Computer age statistical inference*, volume 5. Cambridge University Press.

[Eldardiry et al., 2013] Eldardiry, H., Bart, E., Liu, J., Hanley, J., Price, B., and Brdiczka, O. (2013). Multi-domain information fusion for insider threat detection. In *S&P Workshops*.

[Erdős and Rényi, 1960] Erdős, P. and Rényi, A. (1960). On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci*, 5.

[Eren et al., 2020] Eren, M. E., Moore, J. S., and Alexandro, B. S. (2020). Multi-dimensional anomalous entity detection via poisson tensor factorization. In *ISI*.

[FireEye, 2015] FireEye (2015). Apt30: The mechanics behind a decade long cyber espionage operation. https://www.fireeye.com/blog/threat-research/2015/04/apt_30_and_the_mecha.html.

[Fisher, 1925] Fisher, R. A. (1925). *Statistical methods for research workers.* Oliver and Boyd.

[Forrest et al., 1996] Forrest, S., Hofmeyr, S. A., Somayaji, A., and Longstaff, T. A. (1996). A sense of self for unix processes. In *S&P*.

[Garchery and Granitzer, 2019] Garchery, M. and Granitzer, M. (2019). Identifying and clustering users for unsupervised intrusion detection in corporate audit sessions. In *ICCC*.

[Garchery and Granitzer, 2020] Garchery, M. and Granitzer, M. (2020). Adsage: Anomaly detection in sequences of attributed graph edges applied to insider threat detection at fine-grained level. *arXiv preprint arXiv:2007.06985*.

[Garcia-Teodoro et al., 2009] Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., and Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Comput. Secur.*, 28(1-2):18–28.

[Gavai et al., 2015] Gavai, G., Sricharan, K., Gunning, D., Rolleston, R., Hanley, J., and Singhal, M. (2015). Detecting insider threat from enterprise social and online activity data. In *MIST*.

[Gerhards et al., 2009] Gerhards, R. et al. (2009). The syslog protocol. Technical report, RFC 5424, March.

[Glaz et al., 2001] Glaz, J., Naus, J., and Wallenstein, S. (2001). *Scan Statistics*. Springer.

[Gonçalves et al., 2015] Gonçalves, D., Bota, J., and Correia, M. (2015). Big data analytics for detecting host misbehavior in large logs. In *TrustCom*.

[Gopalan et al., 2015] Gopalan, P., Hofman, J. M., and Blei, D. M. (2015). Scalable recommendation with hierarchical poisson factorization. In *UAI*.

[Grimmett, 1999] Grimmett, G. R. (1999). *Percolation*. Springer.

[Grover and Leskovec, 2016] Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *KDD*.

[Gui et al., 2016] Gui, H., Liu, J., Tao, F., Jiang, M., Norick, B., and Han, J. (2016). Large-scale embedding learning in heterogeneous event data. In *ICDM*.

[Gui et al., 2017] Gui, H., Liu, J., Tao, F., Jiang, M., Norick, B., Kaplan, L., and Han, J. (2017). Embedding learning with events in heterogeneous information networks. *IEEE Trans. Knowl. Data Eng.*, 29(11):2428–2441.

[Gultekin and Paisley, 2014] Gultekin, S. and Paisley, J. (2014). A collaborative kalman filter for time-evolving dyadic processes. In *ICDM*.

[Gutflaish et al., 2019] Gutflaish, E., Kontorovich, A., Sabato, S., Biller, O., and Sofer, O. (2019). Temporal anomaly detection: calibrating the surprise. In *AAAI*.

[Gutmann and Hyvärinen, 2010] Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*.

[Haas and Fischer, 2018] Haas, S. and Fischer, M. (2018). Gac: graph-based alert correlation for the detection of distributed multi-step attacks. In *SAC*.

[Haidar and Gaber, 2018] Haidar, D. and Gaber, M. M. (2018). Adaptive one-class ensemble-based anomaly detection: an application to insider threats. In *IJCNN*.

[Hamilton, 2020] Hamilton, W. L. (2020). Graph representation learning. *Synth. Lect. Artif. Intell. Mach. Learn.*, 14(3):1–159.

[Hammond et al., 2011] Hammond, D. K., Vandergheynst, P., and Gribonval, R. (2011). Wavelets on graphs via spectral graph theory. *Appl. Comput. Harmon. Anal.*, 30(2):129–150.

[Harshman, 1970] Harshman, R. A. (1970). Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84.

[Heard and Rubin-Delanchy, 2016] Heard, N. and Rubin-Delanchy, P. (2016). Network-wide anomaly detection via the dirichlet process. In *ISI*.

[Heymann and Le Grand, 2013] Heymann, S. and Le Grand, B. (2013). Monitoring user-system interactions through graph-based intrinsic dynamics analysis. In *RCIS*.

[Hitchcock, 1927] Hitchcock, F. L. (1927). The expression of a tensor or a polyadic as a sum of products. *J. Math. Phys.*, 6(1-4):164–189.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

[Hofmann and Sick, 2009] Hofmann, A. and Sick, B. (2009). Online intrusion alert aggregation with generative data stream modeling. *IEEE Trans. Dependable Secure Comput.*, 8(2):282–294.

[Hogan and Adams, 2018] Hogan, J. and Adams, N. M. (2018). A study of data fusion for predicting novel activity in enterprise cyber-security. In *ISI*.

[Hu et al., 2017] Hu, Q., Tang, B., and Lin, D. (2017). Anomalous user activity detection in enterprise multi-source logs. In *ICDM Workshops*.

[Huang et al., 2019a] Huang, J., Chen, C., Ye, F., Wu, J., Zheng, Z., and Ling, G. (2019a). Hyper2vec: Biased random walk for hyper-network embedding. In *DASFAA*.

[Huang et al., 2019b] Huang, J., Liu, X., and Song, Y. (2019b). Hyper-path-based representation learning for hyper-networks. In *CIKM*.

[Huang et al., 2014] Huang, W., Song, G., Hong, H., and Xie, K. (2014). Deep architecture for traffic flow prediction: deep belief networks with multitask learning. *IEEE Trans. Intell. Transp. Syst.*, 15(5):2191–2201.

[Hutchins et al., 2011] Hutchins, E. M., Cloppert, M. J., Amin, R. M., et al. (2011). Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80.

[IBM, 2020] IBM (2020). Cost of a data breach report 2020. https://www.ibm.com/security/digital-assets/cost-data-breach-report/1Cost%20of%20a%20Data%20Breach%20Report%202020.pdf.

[Ideker et al., 2002] Ideker, T., Ozier, O., Schwikowski, B., and Siegel, A. F. (2002). Discovering regulatory and signalling circuits in molecular interaction networks. *Bioinformatics*, 18(suppl_1):S233–S240.

[Jagadish et al., 2014] Jagadish, H. V., Gehrke, J., Labrinidis, A., Papakonstantinou, Y., Patel, J. M., Ramakrishnan, R., and Shahabi, C. (2014). Big data and its technical challenges. *Commun. ACM*, 57(7):86–94.

[Jannach et al., 2010] Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). *Recommender systems: an introduction*. Cambridge University Press.

[Julisch, 2003] Julisch, K. (2003). Clustering intrusion detection alarms to support root cause analysis. *TISSEC*, 6(4):443–471.

[Kalman, 1960] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *J. Basic Eng.*, 82(1):35–45.

[Karrer et al., 2014] Karrer, B., Newman, M. E., and Zdeborová, L. (2014). Percolation on sparse networks. *Phys. Rev. Lett.*, 113(20).

[Kendall et al., 2018] Kendall, A., Gal, Y., and Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*.

[Kent, 2015a] Kent, A. D. (2015a). Comprehensive, multi-source cyber-security events. Los Alamos National Laboratory.

[Kent, 2015b] Kent, A. D. (2015b). Cybersecurity data sources for dynamic network research. In *Dynamic Networks in Cybersecurity*. Imperial College Press.

[Kent and Liebrock, 2013] Kent, A. D. and Liebrock, L. M. (2013). Differentiating user authentication graphs. In *S&P Workshops*.

[Kent et al., 2015] Kent, A. D., Liebrock, L. M., and Neil, J. C. (2015). Authentication graphs: Analyzing user behavior within an enterprise network. *Comput. Secur.*, 48:150–166.

[Kerr et al., 2010] Kerr, P. K., Rollins, J., and Theohary, C. A. (2010). *The stuxnet computer worm: Harbinger of an emerging warfare capability*. Congressional Research Service Washington, DC.

[Kesten, 1982] Kesten, H. (1982). *Percolation theory for mathematicians*. Springer.

[King and Chen, 2003] King, S. T. and Chen, P. M. (2003). Backtracking intrusions. In *SOSP*.

[King et al., 2005] King, S. T., Mao, Z. M., Lucchetti, D. G., and Chen, P. M. (2005). Enriching intrusion alerts through multi-host causality. In *NDSS*.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR*.

[Kipf and Welling, 2017] Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *ICLR*.

[Klammer et al., 2010] Klammer, M., Godl, K., Tebbe, A., and Schaab, C. (2010). Identifying differentially regulated subnetworks from phosphoproteomic data. *BMC Bioinform.*, 11(1):351.

[Knight, 2000] Knight, P. (2000). Iloveyou: Viruses, paranoia, and the environment of risk. *Sociol. Rev.*, 48(2_suppl):17–30.

[Kolda and Bader, 2009] Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500.

[Köster and Rahmann, 2012] Köster, J. and Rahmann, S. (2012). Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522.

[Kulldorff, 1997] Kulldorff, M. (1997). A spatial scan statistic. *Commun. Stat. Theory Methods*, 26(6).

[Kulldorff et al., 2006] Kulldorff, M., Huang, L., Pickle, L., and Duczmal, L. (2006). An elliptic spatial scan statistic. *Stat. Med.*, 25(22).

[Lakhina et al., 2004] Lakhina, A., Crovella, M., and Diot, C. (2004). Characterization of network-wide anomalies in traffic flows. In *IMC*.

[Lancaster, 1952] Lancaster, H. (1952). Statistical control of counting experiments. *Biometrika*, 39(3/4):419–422.

[Langovoy et al., 2013] Langovoy, M., Habeck, M., and Schölkopf, B. (2013). Spatial statistics, image analysis and percolation theory. *arXiv preprint arXiv:1310.8574*.

[Langovoy and Wittich, 2013] Langovoy, M. and Wittich, O. (2013). Robust nonparametric detection of objects in noisy images. *J. Nonparametr. Stat.*, 25(2).

[Larroche et al., 2020a] Larroche, C., Mazel, J., and Clémençon, S. (2020a). Percolation-based detection of anomalous subgraphs in complex networks. In *IDA*.

[Larroche et al., 2020b] Larroche, C., Mazel, J., and Clémençon, S. (2020b). Recent trends in statistical analysis of event logs for network-wide intrusion detection. In *CAID*.

[Larroche et al., 2021a] Larroche, C., Mazel, J., and Clémençon, S. (2021a). Anomalous cluster detection in large networks with diffusion-percolation testing. In *ESANN*.

[Larroche et al., 2021b] Larroche, C., Mazel, J., and Clémençon, S. (2021b). Dynamically modelling heterogeneous higher-order interactions for malicious behavior detection in event logs. *arXiv preprint arXiv:2103.15708*.

[Lee et al., 2006] Lee, S., Chung, B., Kim, H., Lee, Y., Park, C., and Yoon, H. (2006). Real-time analysis of intrusion detection alerts via correlation. *Comput. Secur.*, 25(3):169–183.

[Lee et al., 2021] Lee, W., McCormick, T. H., Neil, J., Sodja, C., and Cui, Y. (2021). Anomaly detection in large scale networks with latent space models. *Technometrics*, (just-accepted):1–23.

[Lee and Stolfo, 1998] Lee, W. and Stolfo, S. (1998). Data mining approaches for intrusion detection. In *USENIX Security*.

[Lee et al., 1997] Lee, W., Stolfo, S. J., and Chan, P. K. (1997). Learning patterns from unix process execution traces for intrusion detection. In *AAAI Workshops*.

[Legg et al., 2015] Legg, P. A., Buckley, O., Goldsmith, M., and Creese, S. (2015). Automated insider threat detection system using user and role-based profile assessment. *IEEE Syst. J.*, 11(2):503–512.

[Leichtnam et al., 2020a] Leichtnam, L., Totel, E., Prigent, N., and Mé, L. (2020a). Forensic analysis of network attacks: Restructuring security events as graphs and identifying strongly connected sub-graphs. In *EuroS&P Workshops*.

[Leichtnam et al., 2020b] Leichtnam, L., Totel, E., Prigent, N., and Mé, L. (2020b). Sec2graph: Network attack detection based on novelty detection on graph structured data. In *DIMVA*.

[Leskovec et al., 2010] Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., and Ghahramani, Z. (2010). Kronecker graphs: An approach to modeling networks. *J. Mach. Learn. Res.*, 11(2).

[Li et al., 2014] Li, C., Georgiopoulos, M., and Anagnostopoulos, G. C. (2014). Pareto-path multitask multiple kernel learning. *IEEE Trans. Neural Netw. Learn. Syst.*, 26(1):51–61.

[Lin et al., 2018] Lin, Y., Chen, Z., Cao, C., Tang, L.-A., Zhang, K., Cheng, W., and Li, Z. (2018). Collaborative alert ranking for anomaly detection. In *CIKM*.

[Liu et al., 2019] Liu, F., Wen, Y., Zhang, D., Jiang, X., Xing, X., and Meng, D. (2019). Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise. In *CCS*.

[Liu et al., 2008] Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *ICDM*.

[Liu et al., 2018a] Liu, L., De Vel, O., Chen, C., Zhang, J., and Xiang, Y. (2018a). Anomaly-based insider threat detection using deep autoencoders. In *ICDM Workshops*.

[Liu et al., 2018b] Liu, Q., Stokes, J. W., Mead, R., Burrell, T., Hellen, I., Lambert, J., Marochko, A., and Cui, W. (2018b). Latte: Large-scale lateral movement detection. In *MILCOM*.

[Liu et al., 2018c] Liu, Y., Zhao, L., Liu, G., Lu, X., Gao, P., Li, X.-L., and Jin, Z. (2018c). Dynamic bayesian logistic matrix factorization for recommendation with implicit feedback. In *IJCAI*.

[Lu and Wong, 2019] Lu, J. and Wong, R. K. (2019). Insider threat detection with long short-term memory. In *ACSW*.

[Lu et al., 2009] Lu, Z., Agarwal, D., and Dhillon, I. S. (2009). A spatio-temporal approach to collaborative filtering. In *RecSys*.

[Mandiant, 2013] Mandiant (2013). Apt1: Exposing one of china's cyber espionage units. https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf.

[Memory et al., 2013] Memory, A., Goldberg, H. G., and Senator, T. E. (2013). Context-aware insider threat detection. In *AAAI Workshops*.

[Metelli and Heard, 2019] Metelli, S. and Heard, N. (2019). On bayesian new edge prediction and anomaly detection in computer networks. *Ann. Appl. Stat.*, 13(4):2586–2610.

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

[Mnih and Kavukcuoglu, 2013] Mnih, A. and Kavukcuoglu, K. (2013). Learning word embeddings efficiently with noise-contrastive estimation. In *NeurIPS*.

[Mnih and Salakhutdinov, 2007] Mnih, A. and Salakhutdinov, R. R. (2007). Probabilistic matrix factorization. In *NeurIPS*.

[Mnih and Teh, 2012] Mnih, A. and Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. In *ICML*.

[Moore et al., 2003] Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., and Weaver, N. (2003). Inside the slammer worm. *IEEE Secur. Priv.*, 1(4):33–39.

[Moriano et al., 2017] Moriano, P., Pendleton, J., Rich, S., and Camp, L. J. (2017). Insider threat event detection in user-system interactions. In *MIST*.

[Narita and Kitagawa, 2008] Narita, K. and Kitagawa, H. (2008). Detecting outliers in categorical record databases based on attribute associations. In *APWeb*.

[NCSC, 2017] NCSC (2017). Linkedin 2012 hack: what you need to know. https://www.ncsc.gov.uk/blog-post/linkedin-2012-hack-what-you-need-know.

[Neil et al., 2013a] Neil, J., Hash, C., Brugh, A., Fisk, M., and Storlie, C. B. (2013a). Scan statistics for the online detection of locally anomalous subgraphs. *Technometrics*, 55(4).

[Neil et al., 2013b] Neil, J., Uphoff, B., Hash, C., and Storlie, C. (2013b). Towards improved detection of attackers in computer networks: New edges, fast updating, and host agents. In *ISRCS*.

[Newman, 2006] Newman, M. E. (2006). Modularity and community structure in networks. *Proc. Natl. Acad. Sci. U.S.A.*, 103(23):8577–8582.

[Newman and Ziff, 2001] Newman, M. E. and Ziff, R. M. (2001). Fast monte carlo algorithm for site or bond percolation. *Phys. Rev. E*, 64(1).

[Ning et al., 2002] Ning, P., Cui, Y., and Reeves, D. S. (2002). Constructing attack scenarios through correlation of intrusion alerts. In *CCS*.

[Oinn et al., 2004] Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M. R., Wipat, A., et al. (2004). Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054.

[Oprea et al., 2015] Oprea, A., Li, Z., Yen, T.-F., Chin, S. H., and Alrwais, S. (2015). Detection of early-stage enterprise infection by mining large-scale log data. In *DSN*.

[Ortega et al., 2018] Ortega, A., Frossard, P., Kovačević, J., Moura, J. M., and Vandergheynst, P. (2018). Graph signal processing: Overview, challenges, and applications. *Proc. IEEE*, 106(5):808–828.

[Park and Walstrom, 2017] Park, D. and Walstrom, M. (2017). Cyberattack on critical infrastructure: Russia and the ukrainian power grid attacks. https://jsis.washington.edu/news/cyberattack-critical-infrastructure-russia-ukrainian-power-grid-attacks/.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

[Patcha and Park, 2007] Patcha, A. and Park, J.-M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Netw.*, 51(12):3448–3470.

[Patil and Taillie, 2004] Patil, G. P. and Taillie, C. (2004). Upper level set scan statistic for detecting arbitrarily shaped hotspots. *Environ. Ecol. Stat.*, 11(2):183–197.

[Paxson, 1999] Paxson, V. (1999). Bro: A system for detecting network intruders in real-time. *Comput. Netw.*, 31(23-24):2435–2463.

[Pei et al., 2016] Pei, K., Gu, Z., Saltaformaggio, B., Ma, S., Wang, F., Zhang, Z., Si, L., Zhang, X., and Xu, D. (2016). Hercule: Attack story reconstruction via community discovery on correlated log graph. In *ACSAC*.

[Perdisci et al., 2006] Perdisci, R., Giacinto, G., and Roli, F. (2006). Alarm clustering for intrusion detection systems in computer networks. *Eng. Appl. Artif. Intell.*, 19(4):429–438.

[Perozzi et al., 2014] Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *KDD*.

[Portnoy, 2000] Portnoy, L. (2000). *Intrusion detection with unlabeled data using clustering*. PhD thesis, Columbia University.

[Powell, 2020] Powell, B. A. (2020). Detecting malicious logins as graph anomalies. *J. Inf. Secur. Appl.*, 54:102557.

[Price-Williams and Heard, 2020] Price-Williams, M. and Heard, N. A. (2020). Nonparametric self-exciting models for computer network traffic. *Stat. Comput.*, 30(2):209–220.

[Price-Williams et al., 2018] Price-Williams, M., Turcotte, M., and Heard, N. (2018). Time of day anomaly detection. In *EISIC*.

[Priebe et al., 2005] Priebe, C. E., Conroy, J. M., Marchette, D. J., and Park, Y. (2005). Scan statistics on enron graphs. *Comput. Math. Organ. Theory*, 11(3).

[Qian and Saligrama, 2014] Qian, J. and Saligrama, V. (2014). Efficient minimax signal detection on graphs. In *NeurIPS*.

[Qian et al., 2014] Qian, J., Saligrama, V., and Chen, Y. (2014). Connected sub-graph detection. In *AISTATS*.

[Qin and Lee, 2003] Qin, X. and Lee, W. (2003). Statistical causality analysis of infosec alert data. In *RAID*.

[Qiu et al., 2018] Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., and Tang, J. (2018). Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*.

[Qiu et al., 2010] Qiu, Y.-Q., Zhang, S., Zhang, X.-S., and Chen, L. (2010). Detecting disease associated modules and prioritizing active genes based on high throughput data. *BMC Bioinform.*, 11(1):26.

[Rashid et al., 2016] Rashid, T., Agrafiotis, I., and Nurse, J. R. (2016). A new take on detecting insider threats: exploring the use of hidden markov models. In *MIST*.

[Roundy et al., 2017] Roundy, K. A., Tamersoy, A., Spertus, M., Hart, M., Kats, D., Dell'Amico, M., and Scott, R. (2017). Smoke detector: cross-product intrusion detection with weak indicators. In *ACSAC*.

[Rozenshtein et al., 2014] Rozenshtein, P., Anagnostopoulos, A., Gionis, A., and Tatti, N. (2014). Event detection in activity networks. In *KDD*.

[Rubin-Delanchy et al., 2018] Rubin-Delanchy, P., Heard, N. A., and Lawson, D. J. (2018). Meta-analysis of mid-p-values: some new results based on the convex order. *J. Am. Stat. Assoc.*

[Ruder, 2017] Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

[Sanguinetti et al., 2008] Sanguinetti, G., Noirel, J., and Wright, P. C. (2008). Mmg: a probabilistic tool to identify submodules of metabolic pathways. *Bioinformatics*, 24(8):1078–1084.

[Sanna Passino and Heard, 2019] Sanna Passino, F. and Heard, N. A. (2019). Modelling dynamic network evolution as a pitman-yor process. *Found. Data Sci.*, 1(3):293.

[Sanna Passino et al., 2020] Sanna Passino, F., Turcotte, M. J., and Heard, N. A. (2020). Graph link prediction in computer networks using poisson matrix factorisation. *arXiv preprint arXiv:2001.09456*.

[Sapegin et al., 2015] Sapegin, A., Amirkhanyan, A., Gawron, M., Cheng, F., and Meinel, C. (2015). Poisson-based anomaly detection for identifying malicious user behaviour. In *MSPN*.

[Särkkä, 2013] Särkkä, S. (2013). *Bayesian filtering and smoothing*. Number 3. Cambridge University Press.

[Schafer et al., 2007] Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. (2007). Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer.

[Schölkopf et al., 2001] Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., and Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7):1443–1471.

[Schon et al., 2017] Schon, C., Adams, N., and Evangelou, M. (2017). Clustering and monitoring edge behaviour in enterprise network traffic. In *ISI*.

[Scott and Nowak, 2006] Scott, C. D. and Nowak, R. D. (2006). Learning minimum volume sets. *J. Mach. Learn. Res.*, 7:665–704.

[Sener and Koltun, 2018] Sener, O. and Koltun, V. (2018). Multi-task learning as multi-objective optimization. In *NeurIPS*.

[Sermanet et al., 2014] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*.

[Sexton et al., 2015] Sexton, J., Storlie, C., and Neil, J. (2015). Attack chain detection. *Stat. Anal. Data Min.*, 8(5-6):353–363.

[Sexton et al., 2013] Sexton, J., Storlie, C., Neil, J., and Kent, A. (2013). Intruder detection based on graph structured hypothesis testing. In *ISRCS*.

[Sharpnack et al., 2015] Sharpnack, J., Rinaldo, A., and Singh, A. (2015). Detecting anomalous activity on networks with the graph fourier scan statistic. *IEEE Trans. Signal Process.*, 64(2).

[Sharpnack et al., 2013a] Sharpnack, J., Singh, A., and Rinaldo, A. (2013a). Change-point detection over graphs with the spectral scan statistic. In *AISTATS*.

[Sharpnack et al., 2013b] Sharpnack, J. L., Krishnamurthy, A., and Singh, A. (2013b). Near-optimal anomaly detection in graphs using lovasz extended scan statistic. In *NeurIPS*.

[Shashanka et al., 2016] Shashanka, M., Shen, M.-Y., and Wang, J. (2016). User and entity behavior analytics for enterprise security. In *BigData*.

[Shervashidze et al., 2011] Shervashidze, N., Schweitzer, P., Leeuwen, E. J. v., Mehlhorn, K., and Borgwardt, K. M. (2011). Weisfeiler-lehman graph kernels. *J. Mach. Learn. Res.*, 12(Sep):2539–2561.

[Shittu et al., 2014] Shittu, R., Healing, A., Ghanea-Hercock, R., Bloomfield, R., and Muttukrishnan, R. (2014). Outmet: A new metric for prioritising intrusion alerts using correlation and outlier analysis. In *LCN*.

[Shittu et al., 2015] Shittu, R., Healing, A., Ghanea-Hercock, R., Bloomfield, R., and Rajarajan, M. (2015). Intrusion alert prioritisation and attack detection using post-correlation analysis. *Comput. Secur.*, 50:1–15.

[Shuman et al., 2013] Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Process. Mag.*, 30(3):83–98.

[Siadati and Memon, 2017] Siadati, H. and Memon, N. (2017). Detecting structurally anomalous logins within enterprise networks. In *CCS*.

[Siddiqui et al., 2019] Siddiqui, M. A., Stokes, J. W., Seifert, C., Argyle, E., McCann, R., Neil, J., and Carroll, J. (2019). Detecting cyber attacks using anomaly detection with explanations and expert feedback. In *ICASSP*.

[Silva and Willett, 2008] Silva, J. and Willett, R. (2008). Hypergraph-based anomaly detection of high-dimensional co-occurrences. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 563–569.

[Song et al., 2019] Song, Q., Chang, S., and Hu, X. (2019). Coupled variational recurrent collaborative filtering. In *KDD*.

[Sood and Enbody, 2012] Sood, A. K. and Enbody, R. J. (2012). Targeted cyberattacks: a superset of advanced persistent threats. *IEEE Secur. Priv.*, 11(1):54–61.

[Spathoulas and Katsikas, 2013] Spathoulas, G. P. and Katsikas, S. K. (2013). Enhancing ids performance through comprehensive alert post-processing. *Comput. Secur.*, 37:176–196.

[Speakman et al., 2015] Speakman, S., McFowland III, E., and Neill, D. B. (2015). Scalable detection of anomalous patterns with connectivity constraints. *J. Comput. Graph. Stat.*, 24(4).

[Srebro and Jaakkola, 2003] Srebro, N. and Jaakkola, T. (2003). Weighted low-rank approximations. In *ICML*.

[Stewart, 1993] Stewart, G. W. (1993). On the early history of the singular value decomposition. *SIAM Rev.*, 35(4):551–566.

[Strom et al., 2018] Strom, B. E., Applebaum, A., Miller, D. P., Nickels, K. C., Pennington, A. G., and Thomas, C. B. (2018). Mitre att&ck: Design and philosophy. *Technical report*.

[Sun et al., 2014] Sun, J. Z., Parthasarathy, D., and Varshney, K. R. (2014). Collaborative kalman filtering for dynamic matrix factorization. *IEEE Trans. Signal Process.*, 62(14):3499–3509.

[Sun et al., 2012] Sun, J. Z., Varshney, K. R., and Subbian, K. (2012). Dynamic matrix factorization: A state space approach. In *ICASSP*.

[Tang et al., 2017] Tang, B., Hu, Q., and Lin, D. (2017). Reducing false positives of user-to-entity first-access alerts for user behavior analytics. In *ICDM Workshops*.

[Tang et al., 2015a] Tang, J., Qu, M., and Mei, Q. (2015a). Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD*.

[Tang et al., 2015b] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015b). Line: Large-scale information network embedding. In *WWW*.

[Tankard, 2011] Tankard, C. (2011). Advanced persistent threats and how to monitor and deter them. *Netw. Secur.*, 2011(8):16–19.

[Thac Do and Cao, 2018] Thac Do, T. and Cao, L. (2018). Gamma-poisson dynamic matrix factorization embedded with metadata influence. In *NeurIPS*.

[Trautman and Ormerod, 2018] Trautman, L. J. and Ormerod, P. C. (2018). Wannacry, ransomware, and the emerging threat to corporations. *Tenn. L. Rev.*, 86:503.

[Tsai et al., 2009] Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., and Lin, W.-Y. (2009). Intrusion detection by machine learning: A review. *Expert Syst. Appl.*, 36(10):11994–12000.

[Tuor et al., 2017] Tuor, A., Kaplan, S., Hutchinson, B., Nichols, N., and Robinson, S. (2017). Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In *AAAI Workshops*.

[Tuor et al., 2018] Tuor, A. R., Baerwolf, R., Knowles, N., Hutchinson, B., Nichols, N., and Jasper, R. (2018). Recurrent neural network language models for open vocabulary event-level cyber anomaly detection. In *AAAI Workshops*.

[Turcotte et al., 2014] Turcotte, M., Heard, N., and Neil, J. (2014). Detecting localised anomalous behaviour in a computer network. In *IDA*.

[Turcotte et al., 2016a] Turcotte, M., Moore, J., Heard, N., and McPhall, A. (2016a). Poisson factorization for peer-based anomaly detection. In *ISI*.

[Turcotte et al., 2016b] Turcotte, M. J., Heard, N. A., and Kent, A. D. (2016b). Modelling user behaviour in a network using computer event logs. In *Dynamic Networks and Cyber-Security*, pages 67–87. World Scientific.

[Turnbull, 2020] Turnbull, K. R. (2020). *Advancements in latent space network modelling*. PhD thesis, Lancaster University.

[Valdes and Skinner, 2001] Valdes, A. and Skinner, K. (2001). Probabilistic alert correlation. In *RAID*.

[Veeramachaneni et al., 2016] Veeramachaneni, K., Arnaldo, I., Korrapati, V., Bassias, C., and Li, K. (2016). ai^2: training a big data machine to defend. In *BigDataSecurity*.

[Wainwright and Jordan, 2008] Wainwright, M. J. and Jordan, M. I. (2008). *Graphical models, exponential families, and variational inference*. Now Publishers Inc.

[Watkins, 2007] Watkins, D. S. (2007). *The matrix eigenvalue problem: GR and Krylov subspace methods*. SIAM.

[Wei et al., 2019] Wei, R., Cai, L., Yu, A., and Meng, D. (2019). Age: Authentication graph embedding for detecting anomalous login activities. In *ICICS*.

[Wei and Li, 2007] Wei, Z. and Li, H. (2007). A markov random field model for network-based analysis of genomic data. *Bioinformatics*, 23(12):1537–1544.

[Weisfeiler and Lehman, 1968] Weisfeiler, B. and Lehman, A. A. (1968). A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya*, 2(9):12–16.

[Wetsman, 2020] Wetsman, N. (2020). Woman dies during a ransomware attack on a german hospital. The Verge, https://www.theverge.com/2020/9/17/21443851/death-ransomware-attack-hospital-germany-cybersecurity.

[Whitehouse et al., 2016] Whitehouse, M., Evangelou, M., and Adams, N. M. (2016). Activity-based temporal anomaly detection in enterprise-cyber security. In *ISI*.

[Wu et al., 2016a] Wu, J.-S., Lee, Y.-J., Wei, T.-E., Hsieh, C.-H., and Lai, C.-M. (2016a). Chainspot: mining service logs for cyber security threat detection. In *TrustCom*.

[Wu et al., 2016b] Wu, N., Chen, F., Li, J., Zhou, B., and Ramakrishnan, N. (2016b). Efficient nonparametric subgraph detection using tree shaped priors. In *AAAI*.

[Xosanavongsa et al., 2019] Xosanavongsa, C., Totel, E., and Bettan, O. (2019). Discovering correlations: A formal definition of causal dependency among heterogeneous events. In *EuroS&P*.

[Yen et al., 2013] Yen, T.-F., Oprea, A., Onarlioglu, K., Leetham, T., Robertson, W., Juels, A., and Kirda, E. (2013). Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In *ACSAC*.

[Yuan et al., 2019] Yuan, S., Zheng, P., Wu, X., and Li, Q. (2019). Insider threat detection via hierarchical neural temporal point processes. In *BigData*.

[Zhou et al., 2006] Zhou, D., Huang, J., and Schölkopf, B. (2006). Learning with hypergraphs: Clustering, classification, and embedding. In *NeurIPS*.

ECOLE
DOCTORALE
DE MATHEMATIQUES
HADAMARD

Titre : Détection d'intrusion dans un réseau informatique par l'analyse statistique de journaux d'événements : une approche centrée sur les interactions

Mots clés : Détection d'intrusion, Détection d'anomalies, Données massives, Analyse de réseaux

Résumé : Les journaux d'événements sont des données structurées décrivant toutes sortes d'activités au sein d'un réseau informatique. En particulier, les comportements malveillants adoptés par d'éventuels attaquants sont susceptibles de laisser une trace dans ces journaux, rendant ces derniers utiles pour la supervision et la détection d'intrusion. Cependant, le volume considérable des journaux d'événements générés en production en rend l'analyse difficile. Cette problématique a suscité de nombreux travaux de recherche sur l'analyse statistique de journaux d'événements pour la détection d'intrusion. Cette thèse étudie certaines des principales difficultés rendant actuellement peu aisé le déploiement de telles approches.

Tout d'abord, il n'est pas évident de construire une représentation abstraite des journaux d'événements : ces données sont complexes et peuvent être abordées sous de multiples perspectives, et il est donc difficile d'en capturer tout le sens dans un objet mathématique simple. Nous choisissons une approche centrée sur la notion d'interaction, motivée par l'idée que de nombreux événements malveillants peuvent être vus comme des interactions inattendues entre des entités (utilisateurs, hôtes, etc.). Tout en préservant les informations les plus cruciales, cette représentation rend cependant la modélisation statistique ardue. Nous proposons donc un modèle ad hoc ainsi que la procédure d'inférence associée, en nous inspirant de concepts tels que les modèles à espace d'états, le filtrage bayésien et l'apprentissage multitâche.

Une autre caractéristique des journaux d'événements est qu'ils contiennent une large majorité d'événements bénins, dont certains sont incongrus bien que légitimes. Il n'est donc pas suffisant de détecter des événements anormaux, et nous étudions également la détection de clusters d'événements potentiellement malveillants. Nous nous appuyons pour cela sur la notion de graphe d'événements afin de redéfinir les scores d'anormalité associés aux événements comme un signal structuré en graphe. Cela permet l'usage d'outils de traitement du signal afin de débruiter les scores d'anormalité produits par un modèle statistique. Enfin, nous proposons des méthodes efficaces pour la détection de cluster anormal dans un graphe de grande taille dont les sommets portent des observations scalaires.

Title : Network-Wide Intrusion Detection through Statistical Analysis of Event Logs: an Interaction-Centric Approach

Keywords : Intrusion Detection, Anomaly Detection, Big Data, Network Analysis

Abstract : Event logs are structured records of all kinds of activities taking place in a computer network. In particular, malicious actions taken by intruders are likely to leave a trace in the logs, making this data source useful for security monitoring and intrusion detection. However, the considerable volume of real-world event logs makes them difficult to analyze. This limitation has motivated a fair amount of research on malicious behavior detection through statistical methods. This thesis addresses some of the challenges that currently hinder the use of this approach in realistic settings.

First of all, building an abstract representation of the data is nontrivial: event logs are complex and multi-faceted, making it difficult to capture all the relevant information they contain in a simple mathematical object. We take an interaction-centric approach to event log representation, motivated by the intuition that malicious events can often be seen as unexpected interactions between entities (users, hosts, etc.). While this representation preserves critical information, it also makes statistical modelling difficult. We thus build an ad hoc model and design a suitable inference procedure, using elements of latent space modelling, Bayesian filtering and multi-task learning.

Another key challenge in event log analysis is that benign events account for a vast majority of the data, including a lot of unusual albeit legitimate events. Detecting individually anomalous events is thus not enough, and we also deal with spotting clusters of potentially malicious events. To that end, we leverage the concept of event graph and recast event-wise anomaly scores as a noisy graph-structured signal. This allows us to use graph signal processing tools to improve anomaly scores provided by statistical models. Finally, we propose scalable methods for anomalous cluster detection in node-valued signals defined over large graphs.

Institut Polytechnique de Paris
91120 Palaiseau, France