Contributions to representation learning of multivariate time series and graphs

Edouard Pineau

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2020IPPAT037

Thèse de doctorat

TELECOM Paris

IP PARIS

Contributions to representation learning of multivariate time series and graphs

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

Ecole doctorale de l'Institut Polytechnique de Paris, n°626 (ED IP Paris)
Spécialité de doctorat : Informatique, données et intelligence artificielle

Thèse présentée et soutenue à Paris, le 07/12/2020, par

EDOUARD PINEAU

Composition du Jury :

Florence d'Alché-Buc Professeur, Télécom Paris	Présidente
Ahlame Douzal Professeur associé, Université Joseph Fourier Grenoble	Rapporteur
Thierry Artières Professeur, Ecole Centrale Marseille	Rapporteur
Ludovic Denoyer Professeur / ingénieur de recherche, Université Paris 6 / Facebook	Examinateur
Fabien Moutarde Professeur, Ecole Nationale Supérieure des Mines de Paris	Examinateur
Thomas Bonald Professeur, Télécom Paris	Directeur de thèse
Sébastien Razakarivony Ingénieur de recherche, Safran	Encadrant industriel

Remerciements

Quelques remerciements. Les noms qui n'y apparaissent pas sont peut-être entre les lignes, lisez attentivement.

Un premier remerciement aux encadrants de ma thèse, Thomas et Sébastien. Leur temps et leur énergie consacrés à suivre et orienter mes idées et mon cap ont, à différents niveaux, contribué à la réussite de ma thèse. Le côté humain a aussi une importance que l'intensité des échanges scientifiques n'aurait pu combler et, en cela, je les remercie également, ils ont été des compagnons de voyage en plus d'être des encadrants de recherche.

Par ailleurs, je remercie mes parents, pour leur soutien inconditionnel tout au long de ma vie. J'étends ces remerciements à toute ma famille qui a toujours été à mes côtés.

A Anne-Cécile, pour sa présence indéfectible, pour son soutien si important.

A mes nouveaux amis, ceux que j'ai rencontrés sur le bord du chemin. Les membres du LINCS, les collègues de Safran. Ils sont nombreux, ceux que j'aurai grand plaisir à revoir. Certains ont particulièrement compté, ils se reconnaîtront.

A ceux qui ne rentrent pas dans les précédents paragraphes et qui sont concernés par ces remerciement, je pense à mes amis d'autrefois qui le sont encore aujourd'hui, je vous salue.

Résumé

Les algorithmes d'apprentissage automatique permettent d'apprendre, à partir de données, une capacité à prendre des décisions ou faire des prédictions, sur un large panel de tâches comme la classification de données (par exemples des images, des graphes, etc.) ou la surveillance de systèmes mécaniques. Le modèle appris par l'algorithme est une approximation statistique d'un modèle optimal inconnu. L'efficacité d'un algorithme d'apprentissage dépend d'un équilibre entre la complexité de la distribution des données, la richesse du modèle statistique utilisé et la complexité de la tâche (décision/prédiction) à effectuer sur de nouvelles données issues de la même distribution que les données d'apprentissage.

Néanmoins, il est fréquent d'avoir besoin d'hypothèses simplificatrices sur les données (par exemple la séparabilité linéaire, l'indépendance des variables, etc.). Mais quand la distribution est complexe (par exemple en grande dimension avec des interactions entre les variables), les hypothèses simplificatrices sont souvent fausses et donc contre-productives. Dans cette situation, une solution consiste à ne pas donner directement les données brutes à l'algorithme d'apprentissage mais une représentation alternative de ces données. L'objectif de cette représentation est de séparer les informations pertinentes du bruit (qui sera filtré de la représentation), en particulier quand l'information pertinente est latente (cachée), afin d'aider le modèle statistique de décision.

Jusqu'à récemment, une majorité des représentations standards utilisées dans les algorithmes d'apprentissage étaient construites par des experts, à la main, sur la base de connaissances métier et d'intuitions empiriques. Récemment, une branche de l'apprentissage automatique appelée apprentissage profond a changé le paradigme. L'apprentissage profond est essentiellement basé sur les réseaux de neurones, une riche famille de fonctions paramétriques, qui apprennent des représentations latentes des données directement à partir des données, avec peu de connaissances d'experts. Ces récentes avancées ont surpassé la plupart des méthodes standards dans des domaines comme la vision par ordinateur, l'analyse de la parole ou de textes. Dans cette thèse, nous nous sommes intéressés à l'apprentissage de représentation de deux types de données : les séries temporelles multivariées et les graphes. Les séries temporelles et les graphes sont des objets complexes qui ont des caractéristiques les rendant difficilement traitables par des algorithmes standards de machine learning. Par exemple, ils peuvent avoir des tailles variables et ont des alignements non-triviaux, qui empêchent l'utilisation de métriques standards pour les comparer entre eux (e.g. norme Euclidienne). Il est alors nécessaire de trouver pour les échantillons observés (séries temporelles multivariées ou graphes) une représentation alternative qui les rend comparables par le biais de métriques et hypothèses standards, et donc utilisables dans des algorithmes adoptant les hypothèses simplificatrices citées ci-avant.

Les contributions de ma thèse sont un ensemble d'analyses, d'approches pratiques et de résultats théoriques présentant des méthodes d'apprentissage de représentation de STM et de graphes.

Deux méthodes de représentation de séries temporelles multivariées sont dédiées au suivi d'état caché de systèmes mécaniques. La première propose une représentation "model-based" appelée Sequence-to-graph (Seq2Graph). Seq2Graph se base sur l'hypothèse que les données observées ont été générées par un modèle causal simple, dont l'espace des paramètres sert d'espace de représentation. Cette méthode a été testée pour différents objectifs : détection de causalité dans des échantillons courts de séries temporelles et

détection de dérive par rapport à un modèle moyen. La seconde méthode propose une méthode générique de détection de tendances dans des séries temporelles, appelée Contrastive Trend Estimation (CTE), qui fait l'hypothèse que le vieillissement d'un système mécanique est monotone. Une preuve d'identifiabilité et une extension à des problèmes d'analyse de survie rendent cette approche puissante pour le suivi d'état de système mécaniques. En parallèle de l'étude théorique d'identifiabilité de la tendance dans le modèle CTE, nous avons étudié l'identifiabilité plus générale des problèmes de séparation de sources indépendantes dans des séries temporelles multivariées.

Deux méthodes de représentation de graphes pour la classification sont aussi proposées. Une difficulté avec les graphes réside dans le fait que l'objet graphe est invariant (en propriété) à la permutation des indices des nœuds. Il faut donc trouver une représentation des graphes qui soit invariante à cette permutation. Une première propose de voir les graphes comme des séquences de nœuds et donc de les traiter avec un outil standard de représentation de séquences : un réseau de neurones récurrent. Le passage répété sur chaque graphe vu dans des ordres de nœuds différents permet de forcer le réseau de neurones à apprendre cette invariance. Une seconde méthode propose une analyse théorique et pratique du spectre du Laplacien pour la classification de graphes, qui lui est intrinsèquement invariant à la permutation des nœuds du graphe. Cette dernière méthode, simple, permet d'établir une baseline solide pour la classification de graphes.

Contents

List of Symbols

$\mathcal{X}$	Input space
$\mathcal{Z}$	Latent space of representation
$\mathcal{Y}$	Output space
X	Sample from input space
Z	Sample from latent space
y^X	Variable y (e.g. label) associated to sample X
$F : \mathcal{X} \longrightarrow \mathcal{Z}$	Representation function (also called *encoder*)
$D : \mathcal{Z} \longrightarrow \mathcal{X}$	Generator (also called *decoder*)
$F_\phi : \mathcal{X} \longrightarrow \mathcal{Z}$	Neural representation function with parameters ϕ
$D_\psi : \mathcal{Z} \longrightarrow \mathcal{X}$	Neural generator with parameters ψ
$A^\top$	Transpose of matrix A
$\mathbb{E}_\mathcal{X}$	Expectation computed over $\mathcal{X}$
ΔX	Temporal differential operator for temporal signal
$\langle X \rangle$	Mean with respect to time if X is a time series: $\langle X \rangle = \frac{1}{T}\int_0^T X_t dt$
$\lambda(M)$	Eigenvalues of a matrix M in ascending order
$\mathcal{P}(n)$	Set of all $n \times n$ permutation matrices
$\mathbf{1}_n$	n-dimensional vector of ones
$\mathbb{1}_E$	Indicator of event E: equals 1 if E, 0 otherwise

Index

AE	Autoencoder
BSS	Blind source separation
CLT	Central limit theorem
CNN	Convolutional neural network
CTE	Contrastive trend extraction
DGI	Divergence to graph isomorphism
DL	Deep learning
DRM	Dimensionality reduction model
FBM	Flow-based model
FC	Fully-connected
GLS	Graph Laplacian spectrum
GNN	Graph neural network
GRU	Gated recurrent unit
HM	Health monitoring
HMM	Hidden Markov Model
ICA	Independent component analysis
INN	Invertible neural network
LRM	Latent representation model
LVM	Latent variable model
ML	Machine learning
MLP	Multi-layer perceptron
MTS	Multivariate time series
NN	Neural network
PCA	Principal component analysis
RAE	Recurrent autoencoder
RNN	Recurrent neural network
UTS	Univariate time series
VAE	Variational autoencoder

Chapter 1

Context, outline and contributions

1.1 Context of the thesis

1.1.1 Machine learning context

Machine learning (ML) algorithms are designed to *learn* from data the ability to take decisions or make predictions, in a large panel of tasks like classification of images or monitoring of mechanical systems. In general, the learned *model* is a statistical approximation of the true/optimal unknown decision/prediction process.

The efficiency of an ML algorithm depends on an equilibrium between model richness, complexity of the data distribution and complexity of the task. Nevertheless, for computational convenience, the statistical model may adopt simplifying assumptions about the data (linear separability, independence of the observed variables, etc.). Yet, when data distribution is complex (e.g. high-dimensional with nonlinear interactions between observed variables), the simplifying assumptions are counterproductive. In this situation, a solution is to feed the model with an alternative *representation* of the data. The objective of the data representation is to separate the relevant information from the noise, in particular if the relevant information is hidden (latent), in order to help the simple statistical model.

Until recently and the rise of modern ML, many standard representations consisted in an expert-based handcrafted preprocessing of data, to filter the unnecessary information before feeding the ML model. Recently, a branch of ML called *deep learning* (DL) completely shifted the paradigm. DL uses neural networks (NNs) [Goodfellow et al., 2016], a family of powerful parametric functions, as learning data representation pipelines. Thanks to cheap high-capacity hardware (e.g. GPUs), many NN-based solutions were developed to *learn* to extract relevant representations of *different types* of data (images, vectors, graphs, time series, etc.). These recent advances outperformed most of the handcrafted data features [Nanni et al., 2017] in many domains, like computer vision, speech analysis and text processing.

1.1.2 Industrial context

In this context, Safran, a major aeronautic industrial actor, deployed financial and scientific means in data-driven activities, like non-destructive inspection, predictive maintenance, aerial object detection, au-

tonomous vehicles or design, to propose new tools to engineers.

In particular, my thesis, presented in this document, is undertaken in the Maths and Algorithms for Temporal Data (MATD) team from the Signal and Information Technologies (SIT) research group. The purpose of MATD is to propose methods for health monitoring (HM) of Safran products (mechanical systems, landing gears, etc.) based on data recorded by sensors. The products from which data is recorded/generated will be generically called *mechanical systems* .

Data-based HM is the set of operations implemented to assess the state of a system from data, along its life (see an illustration in Figure 1.1). It is the intermediate phase between data recording and the decision of performing a maintenance or not. HM from data is an important topic and a major need for the future of aeronautics business models for two main reasons. First, monitoring the state of a mechanical system enhances its safety, which is a major concern in aeronautics. Second, besides additional safety, being more accurate and specific on monitoring would generate cost savings by, for example, optimizing inter-maintenance time-lapse.

When data is recorded while the studied system is running in real-life situation, several limits occur in terms of data analysis. First, our data comes from aircraft that landed safely: there is no example of in-use failures, hence no explicit notion of *impairing* system. For this limit, test bench may be set to reach rupture point. However, such data is expensive with a limited representation of real life usage. We note that in certain situations, simulations of failing mechanical systems can also be used to test the HM methods. Second, there is generally no *data-driven* definition of the state of a mechanical system. Some indicators exist, based on mechanical signals (for example exhaust gas temperature margins (EGT) [Ackert, 2015]). Yet, in certain situations, they are not fully satisfying, for example because of confusing effects (a growing EGT margin can both signify improvement or deterioration of an engine, depending on other information). A supplementary richer monitoring indicator may bring additional meaningful information. Finally, unobserved variables can influence the system. They can be endogenous (e.g. unrecorded part of the studies mechanical system) or exogenous (e.g. weather, load). These unobserved variables can sometimes mislead the standard indicators.

In order to ease the problem analysis, we may split HM for in-use mechanical systems in four steps: ① recording the data with sensors, ② representing data with an appropriate representation inference model and/or modeling the studied system (with or without a priori knowledge) with appropriate physical and/or statistical model, ③ extracting the meaningful information hidden in the representation and/or the modeling, ④ defining and monitoring the mechanical system's state from the steps ② and ③.

We mainly focus on the steps ② and ③, i.e. representation and information extraction.

1.1.3 The thesis

In this thesis we are interested in extracting meaningful information from two types of data: multivariate time series (MTS) and graphs. The time series part is directly related to the industrial context, presented above, since the sensor data we treat at Safran are MTS (one time series by sensor). The graph part is a contribution to a standard problem in ML community: the classification of graphs.

The interest of studying these types of data is the following: MTS and graphs are particular objects that do not directly match standard requirements of ML algorithms. For example, they can have variable size and non-trivial alignment, such that comparing two MTS or two graphs with standard metrics is generally

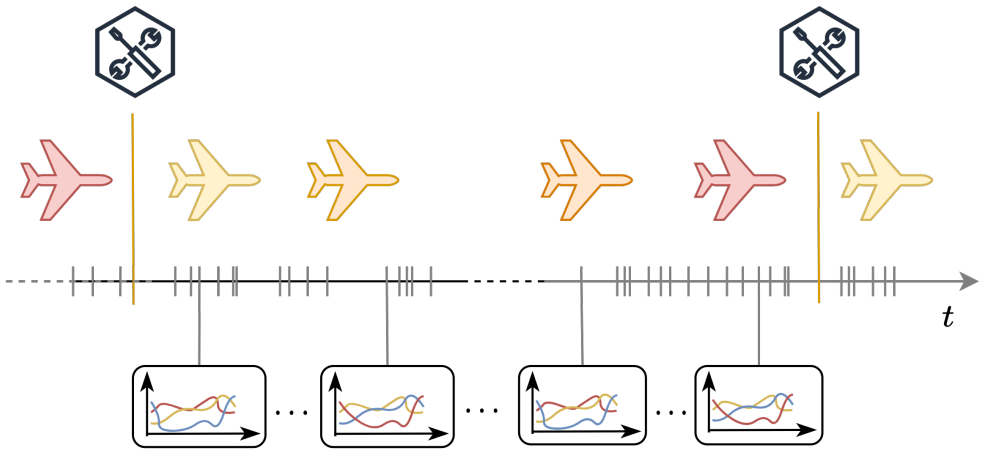


Figure 1.1 – A mechanical system (here a plane) effectuates several flights (gray irregular graduations) during which data are recorded from sensors. The state of the plane varies with time. Maintenances are carried out regularly to reset the state of the system. The objective is to learn the unknown state of the system from the recorded MTS, eventually without the supervision of any technical report that explicitly describes the state of the system at a certain moment of its life.

not relevant. The objective of this thesis is therefore to study and propose solutions to help ML decision process with appropriate representation of MTS and graphs.

1.2 Thesis outline

The thesis is divided into seven chapters, among which two are dedicated to the introduction (including the current), four to original contributions and one last to the conclusion. Chapters 3 and 4 are dedicated to the contributions to MTS representation learning, in particular for HM of mechanical systems. Chapters 5 and 6 are dedicated to the contributions to graph representation and classification. We propose below a chapter-wise outline.

Chapter 2 introduces the main concepts required to fully understand the current thesis: representation learning, multivariate time series and graphs.

Chapter 3 presents a new encoder-decoder framework that consists in training an MTS embedding function that maps MTS samples to model-based Granger causality graphs. We claim and show that such framework gives relevant MTS representation for mechanical system state monitoring.

Chapter 4 presents a new simple yet powerful method to extract trend signal underlying temporal data, based on the application of a principle recently widely used in the ML community: the *contrastive learning*.

Chapter 5 leverages the high expressiveness of recurrent neural networks, coupled with assumptions from both sequence and graph mining, for the classification of graphs.

Chapter 6 proposes and analyzes a simple graph spectral feature that constitutes a strong baseline for graph classification.

Chapter 7 concludes the thesis.

An illustration of the thesis outline is given in Figure 1.2.

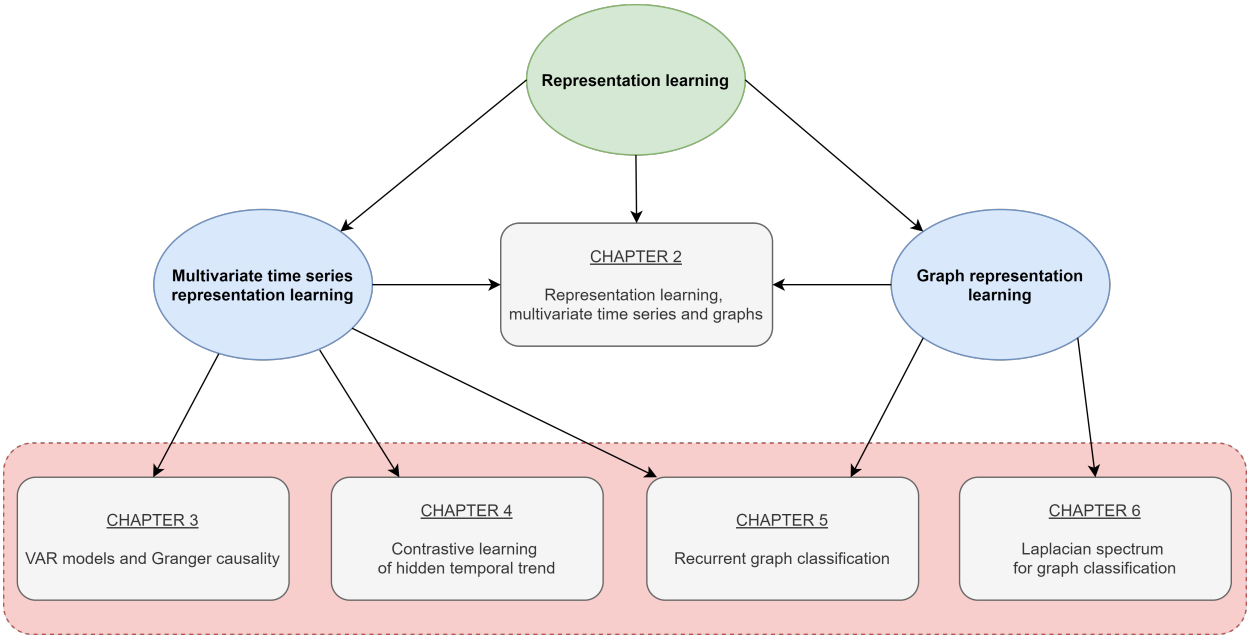


Figure 1.2 – Structure of the thesis

1.3 Contributions

The work done during the thesis resulted in several contributions. In chronological order:

- [Pineau and Lelarge, 2018] Pineau, E. and Lelarge, M. (2018). Infocatvae: Representation learning with categorical variational autoencoders. *Preprint*.
- [de Lara and Pineau, 2018] de Lara, N. and Pineau, E. (2018). A simple baseline algorithm for graph classification. *Workshop on Relational Representation Learning, at Advances in Neural Information Processing Systems (R2L, NIPS 2018)*.
- [Pineau, 2019] Pineau, E. (2019). Using Laplacian Spectrum as Graph Feature Representation. *Preprint*.
- [Pineau and de Lara, 2019] Pineau, E. and de Lara, N. (2019). Variational recurrent neural networks for graph classification. *Workshop on Representation Learning on Graphs and Manifolds, at International Conference on Learning Representation (RLGM, ICLR 2019)*.
- [Pineau et al., 2019] Pineau, E., Razakarivony, S., and Bonald, T. (2019). Seq2var: multivariate time series representation with relational neural networks and linear autoregressive model. *Advanced Analysis and Learning on Temporal Data, pages 126–140, in Lecture Notes on Artificial Intelligence*, Springer.
- Pineau, E., Razakarivony, S., and Bonald, T. (2020). French patent (information, included title, are confidential until the end of the publication delay in January 2022).

- [Pineau et al., 2020a] Pineau, E., Razakarivony, S., and Bonald, T. (2020). Time series source separation with slow flows. *Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models, at International Conference of Machine Learning (INNF+, ICML 2020)*.

- [Pineau et al., 2020b] Pineau, E., Razakarivony, S., and Bonald, T. (2020). Unsupervised ageing detection of mechanical systems on a causality graph. *International Conference on Machine Learning and Applications (2020)*. Oral.

Chapter 2

Introduction to representation learning, multivariate time series and graphs

Abstract In this chapter, we introduce the different concepts required to understand what is *representation learning*, in particular of *multivariate time series* and *graphs*.

2.1 Overview

This section introduces key concepts of representation learning.

2.1.1 Data representation

Data representation is a generic expression that points out a pipeline preceding decision-making model. Its role is to extract and organize relevant information hidden and entangled within data, so that a human or an algorithm can take decisions from data. The *representation inference mechanism* can be seen as a general *embedding function* that takes data as input and projects it within a *representation space* (also called *embedding space*, *feature space* or *latent space* depending on the context) where relevant information is *disentangled*, i.e. the properties of interest are clearly separated from each other and from irrelevant information (considered as *noise*). The performance of the downstream task is generally highly dependent on the quality and the relevance of data representation, hence on the quality and relevance of the representation pipeline.

Notations We note X a data sample, $\mathcal{X}$ a set of data samples (dataset), $\mathcal{Z}$ the representation space (generally lower dimensional than the observation space $\mathcal{X}$) and $F : \mathcal{X} \to \mathcal{Z}$ the representation pipeline/embedding function. The objective is to build F and $\mathcal{Z}$ with appropriate properties such that relevant information about X can be extracted from $F(X)$, $\forall X \in \mathcal{X}$. The size of the dataset is not specified. Except when explicitly stated, the order (and index) of the samples in the dataset is not relevant.

There exist many possible constructions of function F that we can split into two groups: manufactured or learned. In the former case, the function F is built by experts based on a strong (human) *a priori*

knowledge about data properties. In the latter case, an algorithm learns function F that extract knowledge from raw data. It is called *representation learning* and is the focus of this thesis. Hence, the representation learning consists in building not the function F but the learning algorithm that learns F.

2.1.2 Inductive bias for representation learning

A list of specifications that define what is a *good* data representation is proposed in [Bengio et al., 2013]. In this list, we find for example natural clustering, hierarchies, independent directions, sparsity, spatial and temporal consistency. All these specifications correspond to properties *hidden* and *entangled* within data, and a good representation function F must unveil and disentangle these properties.

These specifications are rarely found by chance in data. To find relevant representation of data, it is standard to use a priori knowledge and/or assumptions on the structure of the data and on the objective of the representation (the downstream tasks). When the a priori is integrated into the representation function F and the learning procedure, it is called an *inductive bias*.

Definition 1. *The **inductive bias** of a learning algorithm, also known as learning bias, is the set of assumptions used to predict relevant outputs from data (expressiveness), even on data not used during training (generalization). The inductive bias can be applied on the representation function F and/or on the learning procedure and/or on the representation objective.*

If the objective of the representation is known, the bias will lean towards the objective; otherwise, the bias will lean towards preserving and organizing information contained within data. If the representation of data has no objective nor inductive bias, any embedding function F that preserves all data information is acceptable: it can be for example any invertible function. In this situation, $F(X)$ has few chance to be useful or relevant, since it was not learned to separate and organize the part of the data that is considered as *structure* from the part the is considered as *noise*. Hence, before creating a data representation pipeline, we check the following points:

1. It is important to understand what is the data structure and what information we want to extract from data before learning the embedding function. The more precise the idea, the more possibly relevant the representation. The most precise idea is a knowledge about the data structure and the existence of exact target values to be predicted from the representation that drives the representation of data
2. Otherwise, a stronger set of assumptions about the data, in the shape of an *inductive bias*, is required

Then, the inductive bias is the manifestation of the a priori we have on data properties and structure, which we want to transfer into the embedding function to obtain relevant data representation. An objective of this thesis is to analyze different inductive biases for the representation of multivariate time series and graphs.

2.1.3 The generative assumption behind representation learning

A way to introduce inductive bias in a representation model is the *generative assumption*, described below.

It is always possible to assume that there exist parameters associated with a generator that generated the data. A common assumption for representation learning is that the number of relevant generative

parameters is low. In this case, the generative parameters are a low-dimensional representation of data, with respect to the generator. We use the following generic example to illustrate the *generative assumption*.

We assume that the dataset $\mathcal{X}$ has been generated from model $D(p, \mu) = p\mathcal{N}(\mu, 1) + (1-p)\mathcal{N}(-\mu, 1)$ with parameters $(p, \mu) \in \{0, 1\} \times \mathbb{R}^+$ (e.g. a Gaussian mixture). We note $y^X = (p^X, \mu^X)$ the class and mean from which sample X has been generated. The generative assumption stipulates that these parameters are a representation of the data with respect to the generator D. In this situation, a relevant representation inference function F is a substitute to the inverse generator, i.e. $F(X) = y^X$.

Yet, we will see in Section 2.2.5 that, unlike the true shape of the model is known (here a Gaussian mixture) or unlike we observe the true generative parameters and learn the mapping, it is rare to find them. In general, the best we can do is to find a latent representation $Z = F(X)$ such that there exists a function g such that $g(Z) = y^X$.

The generative assumption is related to a larger family of *generative models*, called *latent variable models* (LVMs) and presented in Section 2.2, to obtain meaningful data representation.

Remark 1. *When we observe the parameters y^X, the generative assumption is not required (yet still true), since we can directly learn the function F to map X to y^X. Such models are called discriminative.*

2.1.4 Different objectives to learn the embedding function

As mentioned above, another way to introduce inductive bias is to insert it into the objective of the representation. We split the representation objectives into four families that depends on the quantity of information we have about data.

Supervised learning The existence of a target y^X for each sample $X \in \mathcal{X}$ enables a *supervised* representation, where data is represented such that we can simply infer the target values with an operator $\mathcal{T}$ (usually linear). Hence, the representation problem is:

$$\min_{F,\mathcal{T}} \mathbb{E}_{X \sim \mathcal{X}} \left[d\left(\mathcal{T} \circ F(X), y^X\right) \right]$$

where d is a divergence. For example, each data sample X is associated to a binary label $y^X \in \{0, 1\}$ and we want to estimate with $\mathcal{T} \circ F(X)$ the probability $p(y^X = 1|X)$. These models are called *discriminative*. If $\mathcal{T}$ is linear, we learn a representation function F such that data with the same label are represented next to each other in $\mathcal{Z}$ and conversely, forming dense linearly separable clusters.

In Chapter 5, we use supervised learning to learn a graph representation function.

Unsupervised learning The representation learning without supervision is called *unsupervised representation learning*. It is a set of methods used to extract relevant features from data without target values. It can be used to find natural clusters [Xu and Tian, 2015], to compress the data in a low-dimensional space [Uthayakumar et al., 2018] (where information needs to be better structured), to create relative distance between samples [Wang et al., 2019], or more generally to find the hidden generative factors of variations [Fabrigar and Wegener, 2011] (the generative assumption 2.1.3) or to estimate the distribution of the data. The objective function for unsupervised representation is as follows:

$$\max_F \mathbb{E}_{X\sim\mathcal{X}}\left[q\left(X, F(X)\right)\right]$$

where $q\left(X, F(X)\right)$ is a quantifier of the information about X included in $F(X)$. It can be for example the mutual information between X and $F(X)$ [Linsker, 1989] (or a lower bound) or the joint likelihood [MacKay, 1996] of X and $F(X)$ (or a lower bound). The latter is related to the generative assumption cited above.

Remark 2. *It is possible to add* $\min_q$ *in the problem to have* q *learned to force* F *to be the best representation in the worst situation: it is the principle of adversarial learning [Goodfellow et al., 2014].*

We propose an unsupervised representation learning of multivariate time series in Chapter 3. We also propose to study a simple baseline unsupervised graph representation in Chapter 6.

Remark 3. *If we have access to labels for certain data samples, a task can be to learn a latent-to-labels mapping for the annotated data (supervised learning) in addition to the unsupervised representation learning on all samples. This mix is called semi-supervised classification [Kingma et al., 2014], i.e. learning labels from incompletely annotated data.*

Self-supervised learning More recently, new methods have extended or completed the definition of unsupervised learning by proposing *self-supervision* [Weng, 2019]. Self-supervised representation learning is the set of methods used to find data representations through self-generated tasks. The self-generated tasks, also called *pretext tasks*, can be for example auto-completion (hide random part of each data and learn to predict it from the remaining part), denoising (add random noise to the data and learn to denoise it), inverse transformation learning (transform the data with random invertible transformations like rotation or dimension randomization and learn to find the original data). Let $\mathcal{T}_1 \dots \mathcal{T}_N$ be N self-generative tasks, such that for $X \in \mathcal{X}$ we have $\left(x^{(i)}, y^{(i)}\right) = \mathcal{T}_i(X)$, with $x^{(i)}$ of the same size than X and $y^{(i)}$ a target. We can define a set of operators $\{T^{(i)}\}_{i=1}^N$ and a set of divergences $\{d_i\}_{i=1}^N$ such that the problem is:

$$\min_{F, D^{(1)}\dots D^{(N)}} \mathbb{E}_{X\sim\mathcal{X}}\left[\sum_{i=1}^{N} d_i\left(y^{(i)}, T^{(i)} \circ F\left(x^{(i)}\right)\right) \mathbb{1}_{\left(x^{(i)}, y^{(i)}\right)=\mathcal{T}_i(X)}\right]$$

F is shared between all pretext tasks, such that if the function F is sufficiently powerful and well trained, it becomes an expressive representation inference pipeline. We note that the choice of the pretext tasks is an inductive bias.

Contrastive learning Related to both supervised and self-supervised learning, *contrastive learning* (CL) is a type of learning procedure that consists in predicting the relative divergence between inputs. It is also called *metric learning*. It consists in selecting similar (dissimilar) samples, with respect to a predetermined similarity measure $s^{\mathcal{X}}$ in the observation space, and learning an embedding function that represents them closely (distantly) with respect to a divergence $s^{\mathcal{Z}}$ in the representation space. The value of the latent representation $F(X)$ of samples X is not important, only is the relative distance. This idea is standard in dimensionality reduction models [Bar-Hillel et al., 2003, Hadsell et al., 2006].

Similarity $s^{\mathcal{X}}$ is usually based on auxiliary information: labels [Hadsell et al., 2006, Chechik et al., 2010], indices of sequential data [Wang and Gupta, 2015], neighboring in graphical data [Lelarge, 2018]. Using the notations above, the objective of the representation learning with CL is:

$$\min_{F} \mathbb{E}_{(X,Y)\in\mathcal{X}^2} \left[s^{\mathcal{Z}} \left(F(X), F(Y), s^{\mathcal{X}}(X,Y) \right) \right] \tag{2.1}$$

A standard CL loss is the *pairwise* contrastive loss, where $s^{\mathcal{X}}(X,Y) \in \{-1,1\}$ (dissimilar or similar) and $s^{\mathcal{Z}}\left(F(X),F(Y),s^{\mathcal{X}}(X,Y)\right) := s^{\mathcal{X}}(X,Y)\, d(F(X),F(Y))$ with d any divergence on $\mathcal{Z}$. Richer versions of contrastive loss that match (2.1) exist (e.g. triplet loss), but the purpose is the same: placing closer similar samples and further dissimilar samples, for a given definition of the samples similarity.

A complementary note on CL is given in Appendix B.5. We use CL in Chapter 4, applied to trend extraction from multivariate time series.

Remark 4. *The similarity $s^{\mathcal{X}}$ contains the main inductive bias of the CL, since it represents an a priori on the relative position of samples. This similarity drives the properties of the learned representation and consequently is usefulness to downstream tasks.*

The four types of learning objectives will be tackled in this thesis. As in Remark 3, the learning objectives can be mixed to improve the expressiveness of the learned representation function. In particular, enriching the representation function of a weakly supervised task with self-supervised or contrastive tasks is common [Gidaris et al., 2019, Arora et al., 2019]. For example, in Chapter 5, we propose to improve the supervised representation of graphs with self-supervised node prediction task.

We have introduced the basis of representation learning required to understand the content of the thesis: embedding functions, latent space and representation, inductive bias and generative assumption. A remaining problem is the construction of the function F. In fact, the problem of finding a representation function in a general function space is not tractable. Assumptions must be set about the function shape. In standard ML, linear assumption is mainly used since it is tractable and theoretically backed. Yet, in many situations, nonlinear representation is required. In certain situations, for example in physics, explicit models can be proposed to fit data (e.g. equations of motion). When the knowledge is limited about the true function, it is required to have universal approximation functions. In the next section, we introduce a generic family of tractable universal approximation functions that we use in this thesis: the *neural networks* (NNs).

2.1.5 Neural embedding functions

Recently, advances in NNs and deep learning (DL) methods [Goodfellow et al., 2016] favored the representation learning against handcrafted representation, particularly for high-dimensional large data with complex structure.

The NNs compose a rich family of parametric approximation functions. A NN is a nested composition of simple operations (linear operator, concatenations, reshaping), simple non-linear activations (sigmoid, rectified linear unit, etc.) and simple regularizing units (normalization, dropout, etc.). The stacked set of operations that compose NNs is called *architecture*. NNs architecture can be adapted depending on the type and complexity of the data. Their simple structure enables efficient (though, ecologically and

memory expensive) learning procedures using gradient descent methods [Rumelhart et al., 1986]. NNs have demonstrated high flexibility and expressiveness for data representation. In particular, NNs are universal approximation functions [Csáji et al., 2001, Lu et al., 2017] such that, when learned with enough data, they have excellent fit and generalization capacities [Neyshabur et al., 2017]. The latter implies that the learned NNs gives relevant representation of new data from the same distribution than training data. Finally, NNs and related learning methods can manage and learn the structure of various types of data (images, speaks, physics, language, graphs) with variable complexity, on large datasets.

We note F_ϕ the NN used as embedding function F, with parameters $\phi \in \Phi$. We can replace embedding F by a neural network F_ϕ in the objectives presented in Section 2.1.4. The optimization on function space is now limited to an optimization on the set of feasible parameters Φ. In general, the set of feasible parameters is given for one NN architecture, such that we only train the values of the parameters, not the architecture of the NN. This architecture is predefined and is part of the inductive bias.

The inductive biases of neural networks architectures We can introduce many inductive biases in the basis elements of neural networks. For example, specific units have been developed to learn representation of vectors, images, sequences or graphs with respectively fully-connected (FC) multi-layer perceptron [Van Der Malsburg, 1986] (MLP), convolutional neural networks [LeCun et al., 1989] (CNN), recurrent neural networks [Hochreiter and Schmidhuber, 1997] (RNN) and graph neural networks [Scarselli et al., 2009] (GNN). These units leverage specific properties of the data. CNN, with local convolution and weight sharing across space, brings translation invariance, a fundamental property of images. RNN deals with sequential data by recording in a memory cell the past information. GNN represents all nodes of a graph by representing together node attributes, information of neighboring nodes and the attributes of the connecting edges. With this panel of neural units and their intrinsic properties, many data types and problems can be efficiently handled. Otherwise, inappropriate choices for the construction and learning of the embedding NNs may induce irrelevant hence useless data representation.

We can also easily use the auxiliary knowledge in the objective function used to train the NNs, which are the same than those in Section 2.1.4, where $\min_F$ is replaced by $\min_\phi$.

We decided to let the details about the NNs architectures in Appendix A to not cut the reading flow. The important knowledge to have is that NNs are easily adaptable to leverage different data structures and properties, and have high representation and generalization power when the right architecture and training procedure are chosen. That being said, we still encourage the reader to refer to the appendix for a better understanding of NNs if needed.

We have introduced the main concepts underlying the representation learning problem. In the next section, we introduce data models based on the generative assumption (see paragraph 2.1.3) widely used for representation learning: the *latent variable models* (LVMs), and their neural extension.

2.2 Latent variable models

2.2.1 Principle of LVMs

When $\mathcal{X}$ is large or high-dimensional, directly modeling distribution $p(X)$ or capturing useful information about data from $\mathcal{X}$ is challenging. A latent variable $Z \in \mathcal{Z}$ is introduced, generally low-dimensional (compared to X) and with a simple distribution (e.g. Gaussian), under the assumption that Z generated X (generative assumption 2.1.3).

A likelihood of the data $p(X|Z)$ is then defined conditionally to the latent variable Z with distribution $p(Z)$. The marginal likelihood can be retrieved by marginalizing over the latent:

$$p(X) = \int_{\mathcal{Z}} p(X, Z)dZ = \int_{\mathcal{Z}} p(X|Z)p(Z)dZ \tag{2.2}$$

For each data sample X, at maximum likelihood (i.e. $Z = \arg\max_{z \in \mathcal{Z}} p(X|z)$), variable Z contains the maximal information about X given distribution p. Z is therefore interpreted as the most likely variable from which data has been generated and can be also considered as the most likely representation of X.

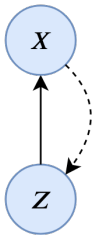


Figure 2.1 – Graphical representation of a generative (solid arrow) and inference (broken arrow) latent variable model.

Using Bayes' rule, we can express the *posterior* distribution of the latent representation conditionally to the data:

$$p(Z|X) = \frac{p(X|Z)p(Z)}{p(X)} \tag{2.3}$$

Hence, sampling from posterior distribution gives likely representations of X. The problem is then to find the distributions $p(Z|X)$, $p(X|Z)$ and $p(Z)$ to infer meaningful data representation $Z \sim p(Z|X)$ and/or generate new data from latent space $X \sim p(X|Z)$ with $Z \sim p(Z)$. We note that X is always assumed to be uniformly sampled on $\mathcal{X}$, such that we have the following relation between inference and generation:

$$p(Z|X) \propto p(X, Z) = p(X|Z)p(Z) \tag{2.4}$$

Remark 5. *Equation (2.4) gives insight about the strong link between discriminative and generative models. As an example, different works [Ng and Jordan, 2002, Grathwohl et al., 2019] have confronted generative and discriminative models in classification tasks, with tight advantage of discriminative models in fully supervised problems; same observation for regression problems. Yet, the generative models gives better results*

in robust classification/regression and semi-supervised classification [Grathwohl et al., 2019], besides their standard application to data modeling, out-of-distribution detection, or unsupervised representation learning.

Latent variable models with auxiliary variables It is possible to enrich the latent variable models with observed auxiliary variables U, one for each sample X. Hence, the posterior becomes

$$p(Z|X,U) = \frac{p(X|Z)p(Z|U)}{p(X|U)} \tag{2.5}$$

illustrated in Figure 2.2. The auxiliary variable can be for example categorical (e.g. a class), discrete (e.g. a time index) or continuous (e.g. a control variable). We note that the prior $p(Z|U)$ also depends on U: if each sample X is associated with a label $y^X \in [\![1, L]\!]$, we may choose a mixture of L unimodal distributions $\sum_{l=1}^{L} p(Z|l)$ as prior $p(Z)$.

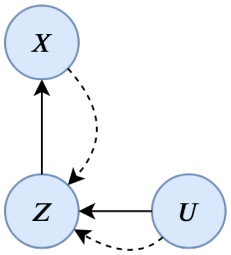


Figure 2.2 – Graphical representation of a generative (solid arrow) and inference (broken arrow) latent variable model with observed auxiliary variable. We note that U is both used as condition for generation and for inference.

Remark 6. *Non-constant auxiliary variables break the independence identically distributed (i.i.d) assumption generally made for the data distribution $p(X)$; only $X|U$ is i.i.d.. In particular, we will see in Section 2.2.5 that non-constant auxiliary variable is a necessary condition for the identification of the generative factors underlying the data.*

2.2.2 Embedding functions and LVMs

Posterior estimation problem When we assume the generative model $p(X|Z)$, the main challenge is to estimate the posterior distribution $p(Z|X)$, whose objective is to sample good representations $Z \sim p(Z|X)$ of sample X.

First, we need to choose an *expressive prior*. Intuitively, the prior is the human knowledge from which the estimation of the posterior goes. Hence, it is better when the prior distribution $p(Z)$ have interesting properties such that the posterior can be easily induced from prior $p(Z)$ and data $\mathcal{X}$ (for example auxiliary variables or the assumption of natural clusters cited above). When the representation Z of X sampled from $p(Z|X)$ contains relevant information about X, we say that posterior is also expressive.

Second, we need to find the posterior distribution, given a prior. There are three main families of posterior estimation methods. First, when the likelihood and prior distributions are conjugate, we can obtain a closed-form posterior distribution. This case is rarely achieved for complex data. Second, Markov chain

Mont Carlo (MCMC) methods [Robert and Casella, 2013]. MCMC are approximate inference algorithms with asymptotic guarantees. They estimate posterior distribution by iterative sampling, such that the chain of samples is a Markov chain whose stationary distribution is the true posterior distribution. MCMC methods, although theoretically perfect estimators, suffer from expensive computation and intractability for large data. Third, variational inference (VI) substitutes the unknown posterior $p(Z|X)$ by a surrogate posterior $q(Z|X)$ chosen in a simpler *variational family of distributions* with the objective to minimize $KL(q(Z|X)||p(Z|X))$, with KL the Kullback-Leibler divergence (see details on VI are given in Appendix B.1). Despite the fact that, unlike sampling-based methods, it will almost never find the exact solution (optimize lower bound of the true data likelihood), VI is generally chosen since cheap and amenable to modern ML techniques (e.g. parallel computing, GPU acceleration, stochastic gradient descent).

Posterior estimation with an embedding function We can use an embedding function F to create a surrogate posterior distribution $q_F(Z|X)$, given a prior distribution and a likelihood, such that $\mathbb{E}_{X\sim\mathcal{X}}[q_F(Z|X)] = p(Z)$, and such that $\tilde{X} \approx X$ for $\tilde{X} \sim p(X|\tilde{Z})$ and $\tilde{Z} \sim q_F(Z|X)$. For example, if $p(Z)$ is $\mathcal{N}(0, I)$, q_F can be a Gaussian density such that $F(X)$ the mean of the Gaussian, i.e. $Z = F(X) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, I)$.

The likelihood distribution $p(X|Z)$ can also be estimated using an (approximate) inverse of the embedding function F noted D ($D = F^{-1}$ if F invertible). For example $p_D(X|Z) = \mathcal{N}(D(Z), I)$. The objective with embedding functions (and its inverse) is therefore to match the joint distributions $q_F(X, Z) = q_F(Z|X)q_{\mathcal{X}}(X)$ and $p_D(X, Z) = p_D(X|Z)p(Z)$.

We note $p_{(F,D)}(X, Z)$ the estimated joint distribution that satisfies the matching.

Remark 7. *If we have access to auxiliary variables, we can choose $F : \mathcal{X} \times \mathcal{U} \to \mathcal{Z}$ and $D : \mathcal{Z} \times \mathcal{U} \to \mathcal{X}$.*

LVMs additional objectives We have seen four families of representation objectives in Section 2.1.4. In the unsupervised case, with q the joint likelihood of (X, Z), we find back the LVM problem described above. More generally, if we remark that the deterministic case is a particular case of the LVM approach, with $p(Z|X) \propto \mathbb{1}_{Z=F(X)}$ and $p(Z) = \mathcal{U}(\mathcal{Z})$, then the representation objectives can then be used in the LVM framework by replacing $\mathbb{E}_{X\sim\mathcal{X}}$ and $F(X)$ by $\mathbb{E}_{(X,Z)\sim q_F(X,Z)}$ and Z to enrich the estimated surrogate posterior distribution $q_F(Z|X)$.

Neural LVMs As expected, we can replace embedding function F by a neural network F_ϕ in the LVMs. The posterior $p(Z|X)$ is estimated with a neural network $F_\phi : \mathcal{X} \to \mathcal{Z}$, with parameters by ϕ usually called *encoder*. The likelihood $p(X|Z)$ is estimated with a neural network $D_\psi : \mathcal{Z} \to \mathcal{X}$, with parameters ψ usually called decoder. We note respectively $q_\phi(Z|X)$ and $p_\psi(X|Z)$ these neural based distributions. We assume for simplicity that hyperparameters ϕ and ψ contain both information on neural network architectures and weight values (yet only the values are learned, the architecture being fixed).

2.2.3 Identifiability of the latent generative factors

A high-valued concept in data representation, and more generally in statistical inference, is the *identifiability*. It concerns parametric models and is defined as follows:

Definition 2. *If $\mathcal{P} = \{p_\theta : \theta \in \Theta\}$ is a parametric statistical model with $\theta \mapsto p_\theta$ bijective, then $\mathcal{P}$ is identifiable if $\{p_{\theta_1} = p_{\theta_2}\} \Rightarrow \{\theta_1 = \theta_2\}$ $\forall(\theta_1, \theta_2) \in \Theta^2$. A model that do not have such property is called unidentifiable.*

It implies that if $\mathcal{P}$ contains the true (unknown) model, we can estimate/learn the true (unknown) values of its parameters (generally from an infinite number of observations). In LVMs, we are interested in finding the true latent factors Z that generated the data, considered as generative parameters of an unknown generator $p(X|Z)$. We give an adaptation/extension of the identifiability for LVMs in the following definition:

Definition 3. *A LVM $p(X, Z)$ is identifiable if $\forall(Z, Z') \in \mathcal{Z}^2$:*

$$\{p(X|Z) = p(X|Z')\} \Longrightarrow \{Z = Z'\} \tag{2.6}$$

We can adapt the definition to LVMs with embedding functions, inspired from [Khemakhem et al., 2019]. We say that a LVM is weakly identifiable if, given two embedding functions F and F' (with respective surrogate inverse D and D') learned on a dataset $\mathcal{X}$, $\exists A$ full-rank such that

$$\{p_{F,D}(X, Z) = p_{F',D'}(X, Z')\} \Longrightarrow \{F(X) = AF'(X)\} \tag{2.7}$$

If A is a weighted permutation, then the LVM is identifiable.

Having an identifiable LVM implies that: if the true generative model can be substituted by a LVM $p_{F^*}(X, Z)$ (i.e. $p_{F^*}(X, Z) = p(X, Z)$), then in the limit of infinite data, any learned embedding function F is an estimate of the true function F^*, up to linear transform A.

Remark 8. *Identifying the true generative factors underlying data is called blind source separation (BSS) [Jutten and Karhunen, 2003].*

We note that, without assumption about the shape of the true LVM, the identifiability is only reachable with universal approximation functions, justifying again the usage of NNs as embedding function for the posterior distribution.

2.2.4 Examples of latent variable models

We present several standard LVMs for illustration and better understanding of the introduced concepts.

Probabilistic principal components analysis First, we introduce principal component analysis (PCA). PCA is a *dimensionality reduction method* (DRM). It consists in linearly projecting the sample X onto the $k < d$ dominant eigenvectors $W_{:k}$ (i.e. corresponding to the highest eigenvalues) of the eigendecomposition $\hat{\Sigma} = W\Lambda W^T$ of the empirical covariance matrix $\hat{\Sigma}$ (estimated from the whole dataset $\mathcal{X}$). With this procedure, PCA maximizes the variance of the projected dataset in each latent direction. Equivalently, PCA minimizes the reconstruction error $\sum_{X \in \mathcal{X}} \|X - XW_{:k}W^T\|_2^2$.

In [Tipping and Bishop, 1999], they derive a probabilistic framework for PCA by adding a standard Gaussian prior on latent variables. It is called probabilistic PCA (PPCA) and is a LVM. It is defined as

$$X = W_{:k}Z + \epsilon + \mu \tag{2.8}$$

with $Z \sim \mathcal{N}(0, I)$ and $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. Thanks to linear and Gaussian assumptions, the likelihood and the posterior have closed form.

Remark 9. *PPCA is a good illustration of the interest of using a generative model over a dimensionality reduction model. First, since we have a probabilistic model, parameters W and σ can be inferred by maximum-likelihood estimation, which is interesting for modeling and unsupervised learning problems (see Section 2.1.4). Second, it is more robust to outliers since PPCA explicitly models the low-density representations. Third, it can manage missing data since it fills the latent space with a distribution. Fourth, related to third advantage, it can be used to generate new data (for PPCA, from prior distribution $\mathcal{N}(0, I)$ we have on Z).*

Independent component analysis Independent component analysis (ICA) is a general family of LVMs, where the latent variables are assumed independent, but not necessarily Gaussian. The most known ICA assumes linear model $X = AZ$ [Hyvärinen, 1997]. In particular, linear ICA solves BSS (see 2.2.3) if at most one dimension of Z is Gaussian (from central limit theorem, any linear mixture of independent sources is more Gaussian than the original sources; hence two mixed Gaussian sources cannot be discriminated from their mixture). The non-Gaussianity assumption is the most used inference method of ICA [Hyvärinen and Oja, 2000] since it is not particularly restrictive. Yet, is also possible to maximize any independence criteria between the dimensions of a linear transformation of the data. The used independence criteria are usually the mutual information [Hyvärinen, 1997, Stögbauer et al., 2004, Chen, 2006, Taleb and Jutten, 1999], kernel-based independence criterion [Bach and Jordan, 2002, Gretton et al., 2005, Shen et al., 2007] or high-order statistics [Cardoso, 1999]. By giving an explicit (non-Gaussian) probabilistic model to the latent variables, we can also solve the ICA using maximum-likelihood approach.

Variational autoencoders First, we introduce *autoencoders* (AEs). AEs are neural DRMs (not proper LVMs, like PCA). They are composed with an encoder F_ϕ and a decoder D_ψ with parameters ϕ and ψ optimized to minimize reconstruction error, i.e. $\phi, \psi = \arg\min \mathbb{E}_{X \in \mathcal{X}} \|X - D_\psi \circ F_\phi(X)\|_2^2$. The encoder and the decoder can have any form and properties, provided that they serve the autoencoding objective (i.e. $D_\psi \approx F_\phi^+$).

Remark 10. *AE with one hidden layer (i.e. F_ϕ and D_ψ linear) finds encoder F_ϕ that spans the same subspace as the subspace spanned by the weights W of a PCA [Baldi and Hornik, 1989], even though $F_\phi \neq W$. In [Plaut, 2018], they propose a method to recover PCA weights from AE linear encoder.*

Several extensions of AE have been proposed. Regularized AEs (RegAE) are trained with a penalization $\Omega(F_\phi(X))$, for example a l^1 norm for sparse latent representations [Ng et al., 2011] or l^2 norm for centered spherical representations [Ghosh et al., 2019]. Denoising AEs (DAEs) [Vincent et al., 2010] consist in adding noise to the input and learning how to reconstruct the clean version of the data. Contractive AEs [Rifai et al., 2011] (CAEs) use the Frobenius norm of the Jacobian of the encoder's activation with respect to inputs as penalty. It forces the AE to learn features that are robust to input changes, hence finding a

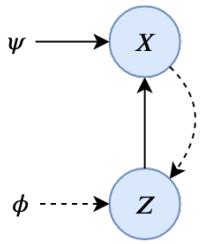


Figure 2.3 – Graphical representation of a VAE. Broken arrows are inference (approximate posterior). Plain arrows are generation. We note that the parameters ϕ and ψ are learned from the whole dataset, such that we can add an arrow from $\mathcal{X}$ to (ϕ, ψ).

smooth representation manifold. They show in the paper that CAEs can be related to RegAEs and DAEs. Finally, as already mentioned for PCA, a way to regularize and/or enrich a DRM is to add a probabilistic model to the latent representation.

In [Kingma and Welling, 2013], they derive a probabilistic framework for AE and train it with variational inference (see Appendix B.1): it is the *variational autoencoder* (VAE) [Kingma and Welling, 2013]. Like PPCA, latent prior $p(Z)$ is conventionally $\mathcal{N}(0, I)$. Under Gaussian likelihood distribution, with Gaussian prior, the *variational posterior* $q(Z|X)$ is also Gaussian [Raiffa and Schlaifer, 1961]. Hence, VAE uses neural networks to find the parameters of the variational posterior $q_\phi(Z|X) = \mathcal{N}(\mu^X, \sigma^X)$ where $\mu^X, \sigma^X = F_\phi(X)$. The likelihood distributions is also parametrized by neural networks, i.e. and $p_\psi(X|Z) = \mathcal{N}(D_\psi(Z), I)$. VAE is illustrated in Figure 2.3.

Remark 11. *In VAEs, the embedding Z is stochastic, i.e. randomly sampled from the estimated distribution $q_\phi(Z|X)$ before decoding. The stochasticity of VAE latent representation, beyond the learning of a generative model, has an interesting regularizing effect. It forces the representation to be consistent (i.e. close samples are close in the representation space). Let consider that the VAE is well trained. If $Z_1 \dots Z_n$ are sampled from $q_\phi(Z|X)$, VAE requires that $D_\psi(Z_i) \approx X \ \forall i \in [\![1, n]\!]$. Hence, the factors learned by VAE must be slowly consistent. Moving slowly in the latent space must implies that the generated data change consistently, and conversely. It is a very powerful inductive bias for representation learning.*

Yet, stochasticity can also be a drawback. In fact, since an infinity of points in the latent space corresponds to the same input X, the representation is uncertain and so is the generation. This phenomenon in image representation with VAE is called blurriness, illustrated in Figure 2.4. Recent works showed that the blurriness can be limited by using appropriate representation inference neural network architectures [Vahdat and Kautz, 2020]. Other works proposed to create generative models with deterministic AE instead of stochastic. In [Makhzani et al., 2015] they regularize the distribution of the dataset's latent representation to match known representation. In [Ghosh et al., 2019], they propose to train a RegAE and to fit ex-post a density $p(Z)$ to the set of data representation $\{F_\phi(X)|X \in \mathcal{X}\}$.

The variational family of distributions can be extended to more general family, as soon as the density function is differentiable with respect to the inferred parameters, since neural networks are trained with gradient descent methods. For example, all distributions of the exponential family and all the distributions that can be related to exponential family through smooth transformation [Kingma and Welling, 2013]. We can also enriched the latent space distribution with mixture of simple distributions, for example to represent

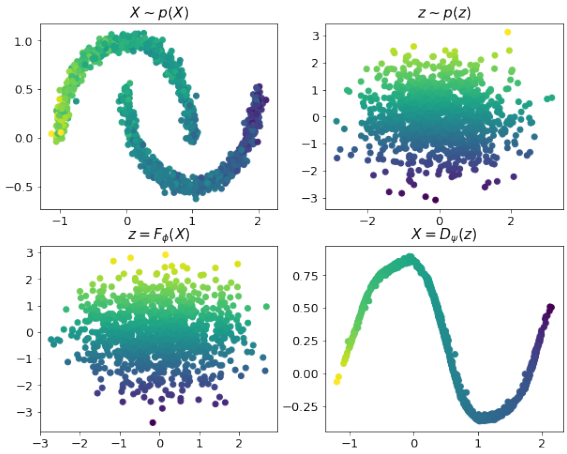


Figure 2.4 – VAE based representation learning and generation, to illustrate the impact of stochasticity in VAE-based representation and the sensitivity of VAE to the importance of the KL in the loss. **Top left**: data $X \sim p(X)$. **Bottom left**: representation $Z = F_\phi(X)$ as a Gaussian distribution. **Top right**: Gaussian sampling from prior $p(Z)$. **Bottom right**: generation from $p_\psi(X|Z)$. We see that the learned representation does include out-of-distribution points because of the stochastic aspect of VAE: the central extremity of the two moons has been averaged.

data in a naturally clustering latent space [Kingma et al., 2014]. It is also possible to change the VAE loss to induce more expressive posterior approximate.

VAE is a very flexible and powerful representation learning tool. A large review of recent advances in VAE-based representation learning is presented in [Tschannen et al., 2018].

Remark 12. *A thesis preliminary work done to grasp the important concepts underlying representation learning, in particular with VAEs, appears in the preprint paper [Pineau and Lelarge, 2018]. It consists in clustering data with VAE using Gaussian mixture latent space and adding information-based regularization. We show that it improves the quality of both the clustering and the generation in a latent space whose shape is inspired by subspace clustering. This work is not presented in this report.*

Flow-based models A recent alternative family of approximate posterior models are the *flow-based models* (FBMs). FBMs are exact likelihood models. To achieve this property, they lean on *normalizing flows* (NFs) [Papamakarios et al., 2019]. A NF is a chain of invertible transformations $\{f_r\}_{r=1}^R$ that enables to pass from a simple distribution $p(Z)$ (e.g. Gaussian) to a complex distribution $q_\mathcal{X}(X)$ (e.g. data) via the change of variable $X = f_1 \circ \cdots \circ f_R(Z) = F^{-1}(Z)$ (see Appendix B.2):

$$q_\mathcal{X}(X) = p\left(F(X)\right) \prod_{r=1}^{R} \left| \det \frac{\partial f_r(x)}{\partial x}\bigg|_{x=f_r \circ F(X)} \right|^{-1} \tag{2.9}$$

The transformations f_r can be arbitrarily complex as soon as they are invertible and we can compute the Jacobian. We see from (2.9) that a NF requires to be easily invertible and to have Jacobian determinant easy to compute. Recent works in [Dinh et al., 2014, Dinh et al., 2016] propose to use *invertible neural networks* (INNs) $F_\phi^{-1} = \{f_{\phi_r}\}_{r=1}^R$ with parameters $\phi = \{\phi_r\}_{r=1}^R$ to enhance the flexibility and the representation

capacity of the normalizing flow. Since the NFs are *deterministic*, we have $q_\phi(Z|X) = \delta(Z - F_\phi(X))$. We show an illustration of a flow-based representation and generation in Figure 2.5 to compare with VAE.

Yet, the INNs with closed-form Jacobian have limited expressive power and flexibility; hence achieving good results for complex data may require large INNs and deep NFs (large R). Different works have subsequently proposed new expressive families of NFs with tractable Jabocian [Papamakarios et al., 2019]. In [Rezende and Mohamed, 2015], they propose to use NF to improve the posterior estimation of VAE, taking best of both worlds. They expand the Gaussian variational posterior estimation through normalizing flows in order to obtain non-Gaussian posterior estimation. They derive a new loss for this flow-based VAE and show that this method enables to achieve highly expressive posterior distributions.

We note that a current thesis work, whose preliminary results are presented in [Pineau et al., 2020a], uses INNs for BSS of time series data. This work is presented and used in the experiments of Chapter 4.

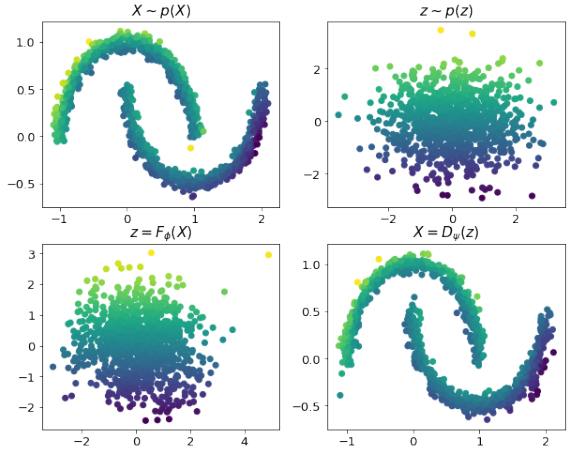


Figure 2.5 – Flow-based representation learning and generation, using RealNVP model [Dinh et al., 2016], to compare with VAE-based representation in Figure 2.4. **Top left**: data $X \sim p(X)$. **Bottom left**: representation $Z = F_\phi(X)$ as a Gaussian distribution. **Top right**: Gaussian sampling from prior $p(Z)$. **Bottom right**: generation from $p_\psi(X|Z)$. We see that, compared to VAE based representation in Figure 2.4, the two moons are perfectly separated, thanks to the deterministic invertible mapping F_ϕ that favors the conservation of the data structure.

2.2.5 Identifiability in neural representations

We have seen in Section 2.2.3 the concepts of *identifiability* and blind source separation (BSS). In Section 2.2.4 we have seen that linear ICA solves the linear BSS problem under simple assumptions on the latent factors Z. Yet, in general nonlinear settings, additional assumptions are required on the sources to obtain identifiability, since it was shown that an infinite set of nonlinear mappings could infer arbitrary set of latent variables that still fulfill assumptions like independence [Hyvärinen and Pajunen, 1999].

In recent works from Aapo Hyvarïnen [Hyvarinen and Morioka, 2016, Hyvarinen and Morioka, 2017, Hyvarinen et al., 2018], they use neural networks F_ϕ as universal approximation for the nonlinear mapping (assumed invertible, for example INNs). Then, they obtain identifiability by relaxing the i.i.d. assumption usually made about the observed variables. To do so, they use auxiliary variables U associated to samples X.

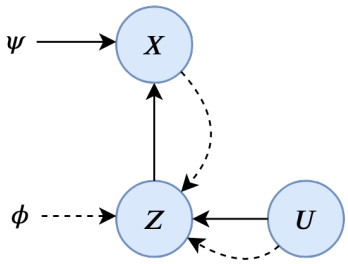


Figure 2.6 – Graphical representation of a VAE with auxiliary variable U. Broken arrows are inference (approximate posterior). Solid arrows are generation. The usage of an appropriate auxiliary variable is a condition of identifiability of the true factors Z.

Contrary to standard approaches where the sources Z are simply assumed to be statistically independent, here the factors Z are assumed statistically dependent and independent conditionally to U. They prove in different settings that it is a sufficient condition for identifiability (in the presence of infinite data and universal approximating neural networks, up to simple transformations).

A first setting uses the neural LVMs framework, i.e. with a NN F_ϕ that estimates the posterior distribution of the latent representation whose parameters ϕ are learned by maximum-likelihood (see paragraph **Neural LVMs** 2.2.2). In [Khemakhem et al., 2019], they introduce auxiliary variables U in VAE framework, on which latent variables Z are explicitly conditioned, to break the i.i.d. assumption. The latent prior $p(Z|U)$ is factorial, but not necessarily $p(Z)$. They show that it is a condition of latent variable identifiability in VAE. These results are extended to flow-based LVMs (Section 2.2.4) in [Sorrenson et al., 2020].

A second setting presented in [Hyvarinen et al., 2018] is based on (binary) contrastive learning (see Section 2.1.4). Parameters ϕ are learned such that a classifier can distinguish between $(F_\phi(X), U)$ and $(F_\phi(X), U^*)$, where U^* is randomly sampled from the distribution of auxiliary variables U.

In this section, we have seen several important concepts and models for the representation learning of data. In the following sections, we focus on the main topic of this thesis: learning representations of multivariate time series and graphs.

2.3 Multivariate time series

We first introduce key concepts attached to multivariate time series (MTS) data. These concepts are the basis of many time series analysis methods. Then, using the concepts presented in Section 2.1, we introduce MTS representation learning.

2.3.1 Basic multivariate time series concepts

Notations We note $X \in \mathbb{R}^{d \times T}$ a d-dimensional time series of length T: X is a matrix whose columns are indexed by a time index $t \in [\![1, T]\!]$. X is a univariate time series (UTS) if $d = 1$ and is a multivariate time series (MTS) if $d > 1$.

The time index indicates a sampling ordering, i.e. X_t has been sampled after X_{t-1}. The ordering of the index is important and is a particularity of the time series data. Each X_t is a d-dimensional vector. Each

variable $X^{(i)} \in \mathbb{R}^T$, $i = 1 \dots d$, is a UTS. In this thesis, we study real valued MTS with discrete time index.

The important point is the following: MTS have two dimensions d and T. While the spatial dimension d is standard (like static vectors), the temporal dimension requires specific representation tools.

We note $\mathcal{X} \subseteq \mathbb{R}^{d \times T}$ the dataset containing the observed data.

Moments and stationarity Time series data are a set of vectors with particular order. As for any set of vectors, a simple representation of a set of vectors is its set of estimated moments. We can infer statistical moments from time series X at different time indices t. These moments are basic features of time series, from which we can define the notion of *stationarity*.

The mean of time series X at time t is the expected value of the vector X_t

$$\mathbb{E}\left[X_t\right] = \mu_t$$

We say that the time series is *mean-stationary* if the mean is constant over time, i.e. $\forall t \in [\![1, T]\!]$, $\mu_t = \mu$. The notion can be extended to any moment of order n $\mathbb{E}[X_t^n]$ or central moment of order n $\mathbb{E}[(X_t - \mu_t)^n]$, where the monomial is applied per dimension.

Another feature in multivariate analysis lives in the interactions between variables. The linear interactions are a moment-like feature, given by the covariance matrix

$$\mathbb{E}\left[(X_t - \mu_t)(X_t - \mu_t)^T\right] = \Sigma_t$$

The diagonal of Σ_t contains the central moments of order 2 of the d variables. For $h \in [\![0, T]\!]$, if the time series is mean-stationary in time range $[\![t - h : t]\!]$ for $t \in [\![h + 1, T]\!]$, we can define the lag-h covariance matrices

$$\mathbb{E}\left[(X_t - \mu_t)(X_{t-h} - \mu_t)^T\right] = \Sigma_{t,h} \tag{2.10}$$

The diagonal of $\Sigma_{t,h}$ contains the lag-h auto-covariances of the d variables. If $\Sigma_{t,h}$ is not time-dependent, i.e. $\forall t, h$ we have $\Sigma_{t,h} = \Sigma_h$, we say that time series X is *auto-covariance-stationary*. In this situation, we remark that $\Sigma_h = \Sigma_{-h}^T$.

Remark 13. *Covariance and auto-covariance represent linear (non-directed) statistical interactions between variables. Nonlinear interactions can be captured with other indicators like mutual information, kernel-based discrepancy, etc., from which we can build a similarity matrix. The notion of stationarity with respect to lagged-covariance matrices can be extended to any lagged-similarity matrices based on these indicators.*

Finally, when moments and interaction are not representative enough, a broader definition of stationarity is the following:

Definition 4. *We note F^X the cumulative distribution function of any subsample of the time series X. X is stationary if for all $\tau \in [\![1, T]\!]$ and all $t_1, t_2 \in [\![1, T - \tau]\!]$ such that $t_1 < t_2$, $F^X(X_{t_1 \dots t_2}) = F^X(X_{t_1+\tau \dots t_2+\tau})$. It means that the data generating process is not time-dependent.*

Remark 14. *We find back the notion of data generation we presented in Section 2.1.3.*

In this thesis, we will be particularly interested in the nonstationaries hidden in time series data. In fact, a key concept related to health monitoring is the *drift* underlying the properties of the monitored system, which can be interpreted as *ageing*.

Autoregressive modeling The modeling of time series generally takes into account the time ordering. In particular, at each time step t, X_t is conditioned on past information $X_1 \dots X_{t-1}$, i.e.:

$$p(X) = \prod_{t=1}^{T} p(X_t | X_1 \dots X_{t-1}) \tag{2.11}$$

It is the *autoregressive* assumption, illustrated in Figure 2.7.

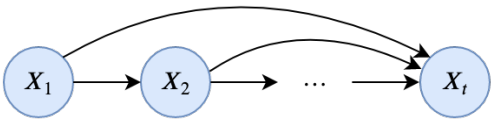


Figure 2.7 – Graphical representation of a generative autoregressive model.

Yet, keeping all past information at each time step is not tractable. Two modeling solutions are generally adopted: the *Markovian assumption* and the *latent variable modeling*, presented in subsequent sections.

The Markovian assumption It is known that in many cases, keeping all past information at each time step t is not useful: the memory is finite. For example, in the linear case introduced above, there exists $K \in \mathbb{N}^*$ such that $\forall h > K$, $\|\Sigma_{t,h}\| = 0$. The finite memory of order K extends to nonlinear general case, such that $p(X) = \prod_{t=1}^{T} p(X_t | X_1 \dots X_{t-K})$. We call this the *Markovian assumption of order* K. It is illustrated in Figure 2.8.

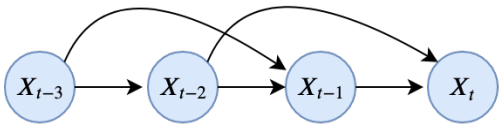


Figure 2.8 – Graphical representation of a generative Markovian autoregressive model of order 2.

VAR model The most famous and used MTS model using Markovian assumption is the vector autoregressive model (VAR). It consists in a linear regression of some past observations on present observations, i.e. for d-dimensional time series X, we have a tensor $W \in \mathbb{R}^{K \times d \times d}$ and a bias $b \in \mathbb{R}^d$ such that $\forall t \in [\![K, T]\!]$

$$X_t = \sum_{k=1}^{K} W_k X_{t-k} + b. \tag{2.12}$$

More precisely, model (2.12) is a d-dimensional linear VAR model of order K, noted VAR(d, K). VAR model is originally used in the field of econometrics: it helps describing the behavior of a economic system's

variables. It describes the dynamical interactions between the variables and improve forecasting models by taking into account the *causal influences* between them. This model will be used for time series representation learning in Chapter 3.

Causality in multivariate time series Causality is type of interaction between variables of a time series. The notion of causality is a large concept that can be summarized as the study of cause and consequences. For a good overview of causality in statistical models, see [Pearl et al., 2009] and [Peters et al., 2017]. In particular, for MTS analysis, the most used notion of causality is the Granger causality [Granger, 1969, Diks and Panchenko, 2006], whose general definition is:

Definition 5. *Let X be a multivariate time series. We say that variable $X^{(i)}$ Granger causes variable $X^{(j)}$ if $p\left(X_t^{(j)}|X_{<t}^{(j)}\right) \neq p\left(X_t^{(j)}|X_{<t}^{(j)}, X_{<t}^{(i)}\right)$, $t \in [\![1, T]\!]$. Causality is therefore a probabilistic dependency graph. We note $X^{(i)} \overset{GC}{\Longrightarrow} X^{(j)}$ the Granger causality from $X^{(i)}$ to $X^{(j)}$.*

Causality in autoregressive models is illustrated in Figure 2.9.

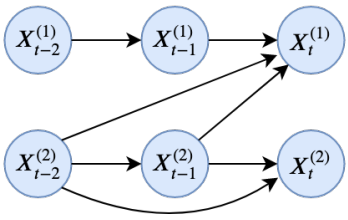


Figure 2.9 – Graphical representation of a Markovian autoregressive model of order 2, with $d = 2$. The arrows are the causalities. Here $X^{(2)} \overset{\text{GC}}{\Longrightarrow} X^{(1)}$. We note that the edges can have features W.

Hence, Granger causality is a lagged directed dependency. There exists different approaches to extract Granger causality graph from data, with two main families of methods. First, causality extraction is simply processed as a variable selection in a regression of the present on the past, in linear [Granger, 1969, Eichler and Didelez, 2012] and nonlinear [Haufe et al., 2010, Davis et al., 2016, Tank et al., 2018] settings. It is the definition we use in Chapters 3. Second, causality is extracted using dependency metrics between lagged variables, using kernel methods [Ancona et al., 2004], information-theory [Solo, 2008] or probabilistic approaches [Hu and Liang, 2014].

Causality and sparse VAR models VAR model (2.12), like common linear regression, can suffer from correlation between variables. A regularization of the model at training time is generally mandatory. In particular, a L_0 regularization of the weights of the linear model provides the elimination of the variables that contain redundant information about the target values (next time step in VAR). The importance of such regularization is illustrated in Figure 2.10.
In VAR(d, K) (3.1), the sparsity is directly related to the notion of Granger causality [Eichler, 2001, Davis et al., 2016] (Definition 5 in Section 2.3.1). If for $(i, j) \in [\![1, d]\!]^2$, there exists $k \in [\![1, K]\!]$ such that $W_{k,i,j} \neq 0$, then $X^{(i)} \overset{\text{GC}}{\Longrightarrow} X^{(j)}$. The zero values of W define the non-causalities in X.

The latent variable modeling Instead of explicitly carrying the whole information about the past time-steps along time axis or limiting the memory like in the Markovian assumption, it is possible to assume that

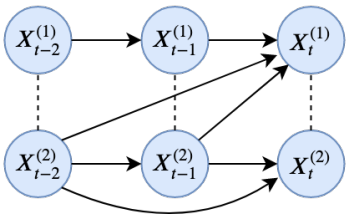


Figure 2.10 – Graphical model of a VAR$(2, 2)$. The dotted lines are the correlations, the arrows are the causalities. In this example, the absence of sparsity inducing regularization will probably let the model detect $X^{(1)}_{t-1} \to X^{(2)}_{t}$ and $X^{(1)}_{t-2} \to X^{(2)}_{t}$ besides the actually existing causalities $X^{(2)}_{t-1} \to X^{(1)}_{t}$ and $X^{(2)}_{t-2} \to X^{(1)}_{t}$, because of the correlation between variables $X^{(1)}$ and $X^{(2)}$. We note that the edges can be weighted with weights W, and that the set of edge's weights can change from one MTS sample X to the other.

a memory cell contains and carries an embedding of the past information at each time step, as illustrated in Figure 2.11.

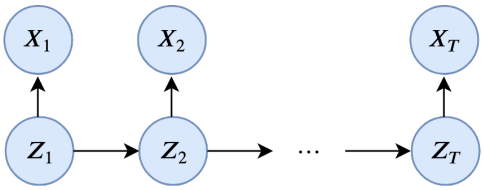


Figure 2.11 – Graphical representation of a generative latent variables autoregressive model. The vertical arrow from Z to X is called *emission* or *generation*. The horizontal arrows from Z_{t-1} to Z_t are called *transition*.

In these models, the size of the memory is implicit and may be theoretically infinite.

We remark that it is a *generative* LVM for X, given a prior on Z_1: $p(X) = p(Z_1) \prod_{t=2}^{T} p(X_t|Z_t)p(Z_t|Z_{t-1})$. LVMs for time series representation learning will be more exhaustively presented in Section 2.3.

Temporal multi-scaling in time series In many time series, there exist several temporal resolutions that contain relevant information. For example, in economic time series, days, weeks, trimesters, years and cycles, are temporal resolutions for which a model is needed. Each resolution is the aggregation of higher resolution plus some additional specific information. Each resolution can be modeled by its own autoregressive model based on the aggregate information of higher dimension, like illustrated in Figure 2.12. The time series can be represented as a hierarchy of statistical representations [Akintayo and Sarkar, 2015]. This multi-resolution with hierarchy of resolution is the principle of deep learning.

We henceforth need to name the two representation scales for time series. First, we will refer to the representation of time series at the scale of time index as *time series decomposition*, as we decompose the samples into temporal explanatory factors (Z^1 in Figure 2.12). Second, the representation of the time series sample viewed as a whole (multiple consecutive time steps), which we will refer to as *time series embedding*, as we embed the whole sample (e.g. each Z^2_i in Figure 2.12 is an embedding of a T_i-long time series $X_{T_i+1} \dots X_{T_{i+1}}$).

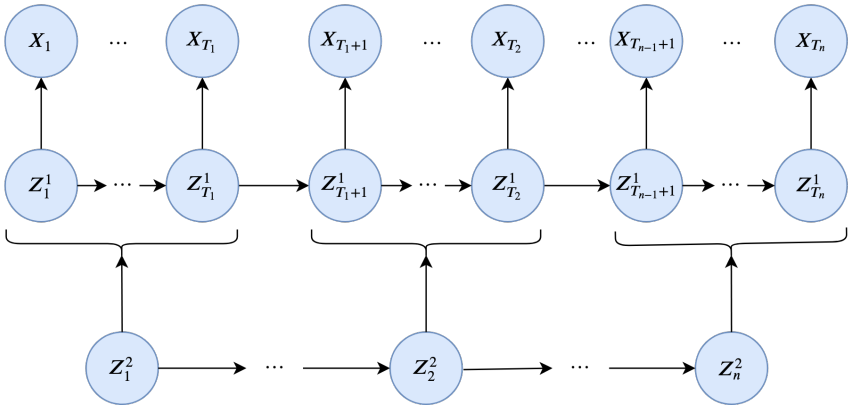


Figure 2.12 – Graphical representation of a generative multiscaled autoregressive model. Z^1 has a low scale autoregressive model; Z^2 has a large-scale autoregressive model. $\{T_i\}_{i=1}^n$ is a set of time index that indicates the change of large-scale generative factor Z^2. We note that the horizontal arrows can be kept only on one stage: for example, if we only want to model a high-level temporal consistency, we may only keep the temporal flow between variables Z^2.

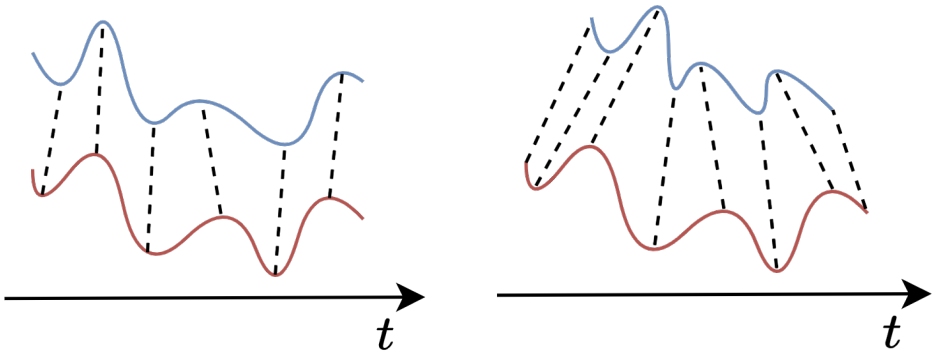


Figure 2.13 – Alignment and size problem for time series comparison. **Left**: dynamic time warping for equal sized UTS. **Right**: dynamic time warping for unequal sized UTS.

2.3.2 Standard MTS representations

Time series data present three interesting challenges. First time series have *variable size*. Second, even in the case they have equal size, time series data have *no trivial alignment*. Third, in multivariate case, many complex interactions between variables generally prevent the use of standard (even warped) metrics to compare MTS data. With these limits, ML on time series data requires special metrics or/and a preceding representation step that aligns the data samples.

The most common distance between time series data, that deals with the first two challenges, is the *dynamic time warping* (DTW) distance. It aligns unaligned curves by warping the temporal axis to find the minimal distance between them, as illustrated in Figure 2.13.

In this thesis, instead of working on a rich metric to compare time series data and apply ML on it, we instead are interested in MTS representation. The representation, as mentioned above, enables to apply

simple metrics of a representation space to compare MTS samples.

Remark 15. *Enriching the metric to compare unaligned samples or finding a representation of data that aligns them before using a simple metric are two related concepts that treat the same problem with a different approach.*

In [Lin et al., 2007], they remind that time series representation and associated similarity measures were initially studied and build for specific tasks: database indexing (finding the most similar time series in a database given a query and a similarity measure) [Chakrabarti et al., 2002], time series classification (assigning predefined class to unlabeled time series) [Esmael et al., 2012], motif discovery (summarizing a time series by extracting its relevant patterns) [Ge and Smyth, 2000] or anomaly detection (finding the abnormal time indices in a long time series, or samples in a dataset, given a normality model) [Keogh et al., 2005]. Depending on the task, a decomposition and/or an embedding of the time series is required. The jungle of time series representation methods is traditionally split into three main families [Esling and Agon, 2012], presented below along with examples of standard models.

Non data-adaptative representations It is the set of methods where the parameters of the representation are not data-dependent, no matter the nature of the time series. We find for example discrete Fourier transform [Faloutsos et al., 1994] or discrete wavelet transform [Chan and Fu, 1999]. The set of coefficients of the decomposition is the time series representation (embedding). We also find piecewise aggregate approximation (PAA), which reduces temporal dimensionality by replacing sub-sequences of the time series by their mean [Keogh et al., 2001] (e.g. $Z_n^2 = \frac{1}{T_n - T_{n-1}} \sum_{t=T_{n-1}+1}^{T_n} Z_t^1$ in Figure 2.12).

Data-adaptive representations Contrary to non data-adaptative methods, data-adaptative representation adapts the parameters of the transformation to the nature of the studies time series. For example, the singular value decomposition (SVD) of the observed set of time series creates a data-dependent representation space on which data is projected. The piecewise linear approximation (PLA) segments time series into linear pieces [Zhu et al., 2007] whose slope summarizes the segments (size of the segments are data-dependent). With PLA, the time series is represented as a sequence of slopes. It can be extended to piecewise polynomial approximation. In [Lin et al., 2007], a symbolic aggregate approximation (SAX) extends PAA/PLA by adding a data-dependent discretization of the sub-sequences mean/slopes to produce sequence of symbols. SAX transforms the time series into strings. Hence, algorithms for text data can be used: for example, segmentation into patterns [Cohen et al., 2007]. We note that we can further transform the sequence of slopes/symbols into histogram of patterns to embed the time series samples.

Symbolic representation is popular, with many different usages. For example, in [Badiane et al., 2018], they propose to use SAX for classification and regression tasks on time series data, in a kernel support vector machine (SVM) framework. In particular, they use the edit distance (distance that quantifies how dissimilar strings are by computing the minimal number of operations required to transform one into the other) in a radial-basis function (RBF) kernel to compute samples similarity matrix. In the same paper, they compare SAX-RBF kernel with other common kernels applicable for time series data. In [Rao et al., 2009], they do

not symbolize the time series itself but the sequence of wavelet coefficients of the time series (more robust to noise).

A related time series embedding method are the shapelet-based methods. It consists of creating a dictionary of maximally informative sub-sequences [Ye and Keogh, 2009] and embedding time series as a set of frequencies of learned dictionary entries. The shapelets have initially been developed for supervised time series classification. SAX representation is a way to simplify the construction of the dictionary with sequences of discrete values from a finite alphabet.

In this thesis, we focus on MTS representation. SAX for MTS analysis is proposed for example in [Ordonez et al., 2011], where they assign a frequency to each word of d symbols (for d channels composing the MTS). In [McGovern et al., 2011], they search discriminative string patterns in each variable for MTS classification. These two examples show that at most, SAX exploits the co-occurence of patterns, but no other more complex variables interaction. And it is the same for other approaches, like shapelet-based MTS embedding for classification in [Bostrom and Bagnall, 2017, Yazdi et al., 2018]. Yet, we want to take into account the multivariate characteristics and in particular the interactions between variables. For example, in [Chouakria-Douzal, 2003], they propose a method to reduce the temporal dimensionality of MTS while preserving the correlation between the variables.

Model-based representations They assume that the data was *generated* from a model, with or without latent variables. We find back the generative assumption of representation learning exposed in Section 2.1.3. If we do not use latent variables, it is usual to choose a parametric model whose parameters are a representation of the data. For example autoregressive models like VAR models (Section 2.3.1) [Xiong and Yeung, 2002, Prado et al., 2006]: we can compare time series by comparing their VAR parameters. If we assume the existence of latent variables (Figure 2.11), we can work in the latent space of representation. For example, a hidden Markov models (HMM) [Rabiner, 1989] can segment a time series into a sequence latent discrete states. Also, two HMMs (hence two time series representations) can be compared (we remind that a HMM is characterized by its initial state, its state transition matrix and the observation emission probability) to compare the time series on which they have been fitted [Antonucci et al., 2015]. We note that it is possible to enhance model-based representation with NNs as in [Kuehne et al., 2018].

Neural representation The application of NNs helps to represent complex MTS. In particular, we know that recurrent neural networks (RNNs, see Appendix A.1.4) have been developed to model and represent time series data. They transform each time steps X_t into a representation Z_t by sequentially embedding the information contained in $X_{1:t-1}$. In this situation, each Z_t is a representation of X_t with respect to past information. Depending on the objective of the representation, Z_t can contain different information about the sample X.

For forecasting, the parameters of the RNN are learned such that $X_{t+1} = D_\psi(Z_t)$ with D_ψ a MLP with parameters ψ. In this situation, we remark that the RNN and the function D_ψ form a latent variable model that can generate the full time series X. For classification, we learn the RNN parameters to sequentially embed the information in Z such that Z_T contains class information about the sample X. We also can embed X without supervision. The Recurrent Autoencoder (RAE) [Cho et al., 2014b, Malhotra et al., 2017] is the

most popular adaptation of AE (see Section 2.2.4) for time series. It consists in using RNNs as encoder and decoder. Like AE, RAE can be enriched to meet interesting properties. For example, in [Lei et al., 2017], they propose an RAE such that the distance between the learned representations of the samples is the dynamic time warping (DTW) distance (they want to learn pattern consistency in the latent space). In [Ienco and Interdonato, 2020], they propose a deep time series embedding clustering model using RAE with an attention mechanism, a binary gate to obtain sparse embedding and a refinement step that stretches the learned representation toward clusters centroids (they want to find natural clustering in the latent space).

Other interesting unsupervised time series embedding: in [Franceschi et al., 2019], they adapt the self-supervised (see Section 2.4.3) and the contrastive learning (see Section 2.4.3) to create an unsupervised scalable time series representation (USTR) model. To learn a meaningful embedding function, they use the following assumption: a MTS sample is closer to one of its subsamples (*positive* sampling) than to a randomly chosen sample of the dataset (*negative* sampling): they can learn their representation model with a triplet loss. The learned unsupervised representations feeds a linear classifier independently trained. It achieves good results in classification downstream task.

More generally, methods developed for static data and presented in Section 2.1 can be adapted for MTS data, in particular thanks to the high flexibility and power of NNs that adapts to many types of data.

In this thesis, we are interested in neural representation under model-based generative assumption. In particular, model-based representation are directly related to the LVMs presented in Section 2.2. In the next section, we introduce a LVM framework for time series, called *state-space models* (SSMs), and its neural extension.

2.4 State-space models

In this section, we present a latent representation learning framework for MTS data: the *state-space models* (SSM), a LVM for sequential data.

2.4.1 Introduction to state-space model

Let $X \in \mathcal{X}$ be a time series of length T, Z a T-long sequence of hidden states and U a T-long sequence of inputs (control commands, auxiliary information, etc.). As already mentioned, a time series X is both a matrix and a ordered set of vector observations X_t, $t \in [\![1, T]\!]$, it can therefore be decomposed and embedded. The two representation scales can be treated with a general framework: the state-space models (SSM).

SSMs were first developed to model physical systems from sequential observations. Like LVM, the SSM represents the distribution of the data with hidden states and auxiliary variables as follows:

$$p(X_{1:T}|Z_{1:T}, U_{1:T}) = p(X_{1:T}|Z_{1:T})p(Z_{1:T}|U_{1:T}) = \prod_{t=1}^{T} p(X_t|Z_t)p(Z_t|Z_{t-1}, U_t) \tag{2.13}$$

The prior $p(Z_t|Z_{t-1}, U_t)$ brings *temporal consistency* to the model. See an illustration in Figure 2.14.

Remark 16. *The relation with static LVM is clear. At each time step t, data X_t, auxiliary U_t and latent variables Z_{t-1} and Z_t form a LVM. The previous time-step latent representation Z_{t-1} is an additional*

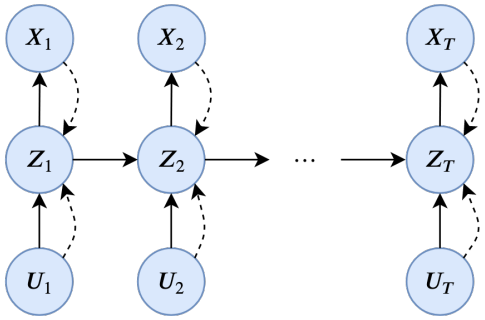


Figure 2.14 – Graphical representation of a generative state-space model. $U_t \to Z_t$ is called *control*. $Z_{t-1} \to Z_t$ is called *transition*, $Z_t \to X_t$ is called *emission*. Solid arrows are the generative model. Broken arrows are the inference model.

auxiliary variable. This addition is the propagation of information along the time axis, which enriches the posterior estimation with dynamical information.

The SSM is therefore a general but simple time series modeling tool. In particular, from (2.13) we can build the following MTS representations:

1. $Z_{1:T}$ as a decomposition of X
2. Each Z_t as a an embedding of X_t or an embedding of $X_{1:t}$

Depending on the task, we will refer to one of the items above.

Like in static models, the nature of the latent variable can change depending on what we seek in MTS data. If we want to segment the time series, we may use categorical $Z_t|Z_{t-1}$ (see HMM cited above). If we want to model the time series with simple distribution, we can use Gaussian distribution for $Z_t|Z_{t-1}$. If we want both, we can use a Gaussian mixture. If we want to reduce the temporal dimensionality, we can use deterministic LVMs for $Z_t|Z_{t-1}$ and only keep Z_T (see RAE cited above). We have the high flexibility of static LVMs in SSMs.

Remark 17. *Different natural auxiliary variables U_t exist in time series. For example, we can use $U_t = X_{t-1}$, $U_t = X_{1:t-1}$ or $U_t = t$.*

Most of the tools and concepts presented for LVMs are applicable in SSM framework. We can use functions to model the distributions (static case in Section 2.2.2): embedding function F, emission function D (inverse or approximated inverse of F) and transition functions P which model respectively the posterior $p(Z_t|X_t, Z_{t-1}, U_t)$, the likelihood $p(X_t|Z_t)$ and the prior $p(Z_t|Z_{t-1}, U_t)$ distributions involved in SSMs. For example:

$$\begin{cases} Z_t = P(Z_{t-1}, U_t) + \epsilon_t, \epsilon_t \sim \mathcal{N}(0, Q_t) \\ X_t = D(Z_t, U_t) + v_t, v_t \sim \mathcal{N}(0, R_t) \end{cases} \tag{2.14}$$

Neural SSM As for LVMs, the model (2.14) can be estimated with NNs (and adapted learning procedures) [Krishnan et al., 2017]. F_ϕ models approximate posterior $q(Z_t|Z_{t-1}, X_t, U_t)$ and D_ψ for likelihood $p(X_t|Z_t)$. We note respectively $q_\phi(Z_t|Z_{t-1}, X_t, U_t)$ and $p_\psi(X_t|Z_t)$ the neural posterior and likelihood distributions.

We give noteworthy examples of SSMs in the following section, for a better understanding of the introduced concepts.

2.4.2 Examples of SSMs

Kalman filter Initially, state-space models assumed linear relations and Gaussian transition [Kalman, 1960]:

$$\begin{cases} Z_t = P_t[Z_{t-1}, U_t] + \epsilon_t, \epsilon_t \sim \mathcal{N}(0, Q_t) \\ X_t = D_t[Z_t, U_t] + v_t, v_t \sim \mathcal{N}(0, R_t) \end{cases} \tag{2.15}$$

with P_t the transition, Q_t the transition's covariance, D_t the emission, v_t the observation noise with covariance R_t. In the general case, parameters $\theta_t := [P_t, B_t, Q_t, D_t, R_t]$ is time-dependent. In many cases, the model is *stationary*, i.e. $\theta_t = \theta, \forall t \in [\![1, T]\!]$, or *piecewise stationary* i.e. $\theta_t = \theta_{t-1}$ for almost all $t \in [\![2, T]\!]$ [Xiong and Zhou, 2013].

Linear Gaussian SSM, thanks to the parsimonious dependency structure (see Figure 2.14) and Gaussian assumption (implying Gaussian marginal, conditional and joint distributions), has closed-form likelihood from which we can learn the parameters and/or the hidden states Z. The pure parameter learning can be done with any maximimum likelihood algorithm. The simultaneous parameters and state inference can be done with expectation-maximization (EM) algorithm [Ghahramani and Hinton, 1996]. Exhaustive treatment and use of standard linear SSM can be found in [Casals et al., 2018].

Remark 18. *In (2.14), when P and D are smooth enough, i.e. well approximated by first-order Taylor expansion, extended Kalman filter [Smith et al., 1962] can be used. When the functions P and G are highly non-linear, difficult to estimate, when the problem is high-dimensional such that the computation of the Jacobian is intractable, or simply when we do not want assume too much about the distribution in the modeling problem, NNs are a generic and flexible solution to estimate the emission and transition functions.*

Slow LVM *Slowness* is a common temporal structure used in time series decomposition. It represents the fact that two consecutive time-steps in a time series have close values. The extraction of slow features from time series data is called *Slow feature analysis* (SFA) [Wiskott and Sejnowski, 2002]. The slowness is generally modeled as the fact that temporal increments of a time series have low variance. The SFA problem is then defined as follows:

$$\min_{F:Z=F(X)} \sum_{i=1}^{d} \langle \Delta Z^{(i)^2} \rangle \text{ s.t. } \langle Z \rangle = 0, ZZ^T = I \tag{2.16}$$

with $\Delta Z = \left[\Delta Z^{(1)} \dots \Delta Z^{(d)}\right]^T \in \mathbb{R}^{T \times d}$ the set of incrementz and $Z_0 = 0$ (without loss of generality) such that $\Delta Z_1 = Z_1$ is defined. Constraints avoid trivial constant solutions and information redundancy between the latent factors. It is also standard but not required to add an ordering constraint for the slowness, i.e. $\langle \Delta Z^{(i)^2} \rangle < \langle \Delta Z^{(j)^2} \rangle$ if $i < j$.

Remark 19. *The increasing slowness inductive bias is the direct counterpart of decreasing explained variance inductive bias of PCA. Both are powerful inductive biases for dimensionality reduction and representation learning.*

In [Turner and Sahani, 2007], the SFA is expressed as a latent variable model. Constraints can be replaced by $Z_t \sim \mathcal{N}(0, I)\ \forall t \in [\![1, T]\!]$. Prior on $Z = F(X)$ (2.16) is:

$$Z_t | Z_{t-1} \sim \mathcal{N}(Z_{t-1}, \Sigma) \tag{2.17}$$

and the posterior distribution is

$$Z_t | X_t \sim \mathcal{N}(F(X_t), \sigma_z^2 I) \tag{2.18}$$

such that the function F is optimized by maximum-likelihood. We note that the slowness can be adapted to many time series representation learning situations, since the temporal consistency is often a good a priori knowledge.

Time-based independent components analysis In linear ICA, we have seen that a common assumption is the non-Gaussianity of the sources [Hyvärinen and Oja, 2000]. When data is time series, an alternative assumption is the slowness [Hyvärinen et al., 2001], presented above. In fact, if the latent Z are slow uncorrelated features then $\forall t, i,\ Z_t^{(i)} \approx Z_{t-1}^{(i)}$. Hence, the covariance between $\langle Z_{1:T-1}^{(i)} Z_{2:T}^{(j)} \rangle \approx \langle Z^{(i)} Z^{(j)} \rangle = \delta_{ij}$. It is shown in [Tong et al., 1991] that having diagonal instantaneous and lagged covariance is a sufficient condition for independence.

In [Blaschke et al., 2007], they propose a nonlinear BSS method by coupling linear ICA with SFA. In [Sprekeler et al., 2014], they enrich the SFA-based nonlinear ICA with an iterative extraction of the sources. The latter showed that if the sources have different autocorrelation functions, then the true sources are identifiable. It constitutes the first theoretically grounded nonlinear identifiable nonlinear ICA. Additional results on nonlinear BSS with NNs will be introduced in Section 2.4.4 after the introduction of neural LVMs for time series data. A complete review of time-based ICA can be found in [Hyvärinen et al., 2001].

Structural independent components analysis A particular case of temporal ICA assumes the existence of *structural latent components* that are supposed to be meaningful. *Structural signal extraction* hence refers to the definition and extraction of these structural components that compose the time series. The standard decomposition in the literature consists in the combination of *structural components* that are the trends τ, the cycle c, the seasonality s and the irregularity ϵ. The trend is a monotone long-term signal. The cycle is a signal that affects the observations at random frequencies. Seasonality is a signal that affects the time series at fixed frequency. The irregularity is the rest, that mainly models exogenous factors and noise. The exogenous input can be a control variable, an environment factor or some random noise. The structural components τ, c, s, ϵ are always assumed to be independent, for identifiability reasons (see ICA in Section 2.2). This decomposition is illustrated in Figure 2.15. In this model, $Z_t = [\tau_t, c_t, s_t, \epsilon_t]\ \forall t \in [\![1, T]\!]$.

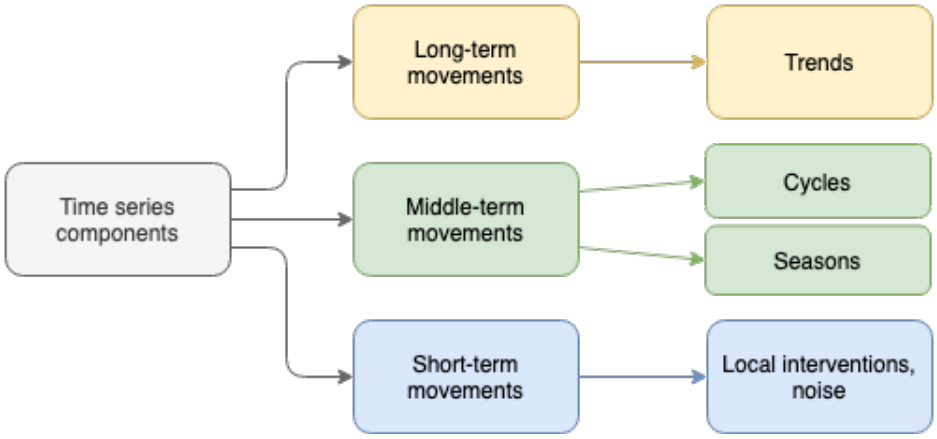


Figure 2.15 – Common latent variable modeling of time series data

Variational state-space model We can adapt the VAE to SSM framework, illustrated in Figure 2.16. Since SSM is a directed graph of static LVM connected by their latent variable, VSSM is a directed graph of VAE. The VAE share their parameter, such that the inference mechanism model all time series time-steps. The loss for variational SSM (VSSM) directly stems from VAE's loss [Fraccaro, 2018].

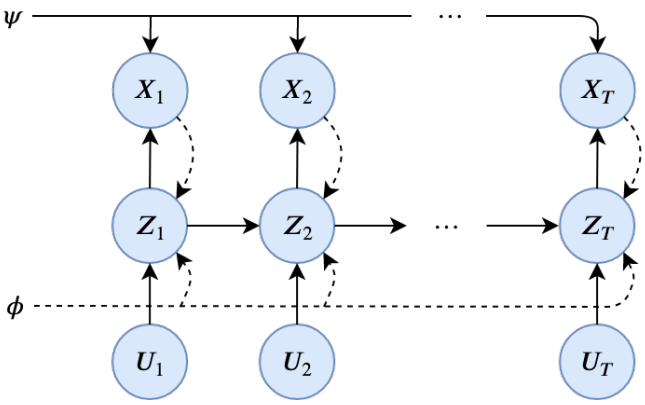


Figure 2.16 – Graphical representation of a VSSM. Broken arrows are inference (posterior), emission and control (prior). Plain arrows are generation. Parameters are shared between all time-steps.

Like for the static VAE, variational family of distribution is usually Gaussian, with for example prior $p(Z_t|Z_{t-1}, U_t) = \mathcal{N}(z_{t-1}, \sigma(U_t))$, σ being a function of the control to adapt the uncertainty of the transition depending on control actions.

Many derivations of the VSSM has been used in the literature, depending on the studied case studied. On overview of standard VSSMs and their applications is given in [Fraccaro, 2018].

2.4.3 Designing expressive SSM with additional objectives

As in the static case, we can enrich the estimation of the data representation in SSM framework using additional objectives.

Designing expressive SSM posterior distributions with self-supervised learning Like for static data, it is possible to enhance the expressiveness of the representation mechanism (posterior distribution) with self-supervised learning, with standard pretext tasks used for static data, like auto-completion or denoising [Romeu et al., 2015]. In particular, SSM are inherently self-supervising the time series representation. In fact, the propagation of information through time with latent variable Z (see illustration in Figure 2.14) can be seen as an auto-completion of the information: $Z_t|X_t$ does not contain all the information about X_t, since Z_t is also conditioned on Z_{t-1}. This one-step information completion through time is called *temporal consistency*, and contains the notion of *slowness* previously cited.

Designing expressive SSM posterior distributions with contrastive learning The difficulty of CL lies in the definition of a similarity in the observation space. When applied for time series representation, the time index can be leveraged: the closer the time index of samples, the closer they are (in term of properties or values). This CL based on time-index is called *time-contrastive learning* (TCL) and has many variants.

In [Sermanet et al., 2018], they reconcile in the latent space different viewpoints from the same scene in consecutive frames of a video. In [Wang and Gupta, 2015], they assume that close frames of a video should be represented closer than distant frames and propose a loss that directly optimizes this assumption. In [Franceschi et al., 2019], a method called Unsupervised Scalable Time Series Representation (USTR) assumes that a MTS sample is closer to one of its subsamples (*positive* sampling) than to a randomly chosen sample of the dataset (*negative* sampling). The embedding $F_\phi(X)$ is trained with a *triplet loss* to reconcile the reference sample and the positive sample, and ward the negative sample off. In [Banville et al., 2019], they use related approach to learn EEG (long time series) representation, by learning to reconcile temporally close and to discriminate temporally far MTS subsamples.

2.4.4 Identifiability of time series neural representations

We have presented recent identifiability results in Section 2.2.5. These results apply to time series representation, in the three previously presented settings, as follows.

A first setting uses the neural SSMs framework, i.e. with a NN F_ϕ that estimates the posterior distribution of the latent representation whose parameters ϕ are learned by maximum-likelihood. In [Khemakhem et al., 2019], they introduce auxiliary variables U_t in VAE framework, on which latent variables Z are explicitly conditioned, to break the i.i.d. assumption. The latent prior $p(Z_t|U_t)$ is factorial, but not necessarily $p(Z_t)$. They show that it is a condition of latent variable identifiability in VAE. We note that if U_t is constant for all samples, then we find back the standard SSM framework. U_t can be the time index t, the past observation X_{t-1} or any other label (not necessarily related to time).

A second setting is based on supervised learning. In [Hyvarinen and Morioka, 2016], for a time series X, assumed to be nonstationary (underlying properties are time-dependent), they learn a latent representation of time-steps $F_\phi(X_t)$ such that a linear classifier can determine its time index t (or at least the index of a time-window that contains t, assuming that the series is stationary-by-part). They show that, if the stationarity (by-part) condition is respected, then under simple assumptions on the true generative factors Z of X, then the latent nonlinear variable model is identifiable (up to a linear transform). By applying a linear ICA on the TCL-based representation, we then obtain true independent generative factors, up to

scaling and indexing.

A third setting is based on (binary) contrastive learning (see Section 2.1.4). In [Hyvarinen et al., 2018], a logistic neural regression learns to discriminate between $[F_\phi(X_t), U_t]$ (label 1) and $[F_\phi(X_t), U_{t^*}]$ (label 0), where U_t is an auxiliary variable for X_t and t^* a random time index.

A particular case of contrastive learning for hidden trend identification is developed in Chapter 4.

2.4.5 LVMs for time series embedding

We have mentioned above that the last latent variable of an RNN can represent the whole time series X. More generally, several extensions of the previously presented models can be adapted to *time series embedding*.

Variational recurrent autoencoder In Section 2.2.4, we have introduced AEs for dimensionality reduction and VAEs its common LVM extension. We have also introduced the RAE in paragraph **Neural representation** in Section 2.3.2. Like AE, RAE can be extended with a probabilistic model to create a *variational recurrent autoencoder* (VRAE) [Fabius and van Amersfoort, 2014]. We note h^e the latent variables of the RNN encoder and h^D the latent variables of the RNN decoder. The associated objective is similar to VAE's (see Appendix B.1 for details on the VAE loss):

$$\phi, \psi = \arg\min \mathbb{E}_{X \in \mathcal{X}} \left[-\mathbb{E}_{Z \sim q_\phi(Z|X,U)} \left[\log p_\psi(X|Z,U) \right] + KL\left(q_\phi(Z|X,U) || p(Z|U) \right) \right] \tag{2.19}$$

where $p_\psi(X|Z,U) = \prod_{t=1}^{T} p_\psi(X_t|h_t^d) \mathbb{1}_{h_t^d = D_\psi(h_{t-1}^d)}$, D_ψ is a MLP with parameters ψ, $q_\phi(Z|X,U) = q_\phi(Z|h_T^e)$, $h_t^e = F_\phi(h_{t-1}^e, X_t, U_t)$, F_ϕ is a MLP with parameters ϕ, $h_t^d = D_\psi(h_{t-1}^d, U_t)$ $\forall t \in [\![1, T]\!]$. Like for VAE, the prior on Z is generally $\mathcal{N}(0, I)$. The model is illustrated in Figure 2.17.

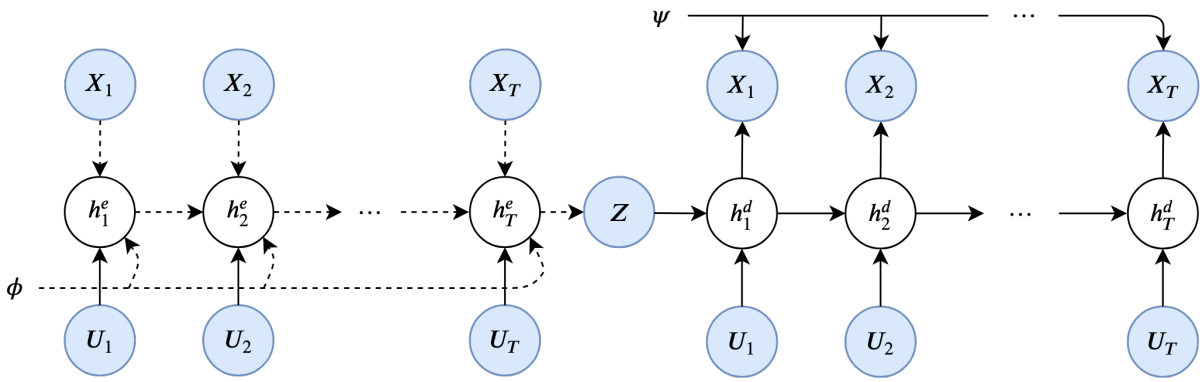


Figure 2.17 – Graphical representation of a whole sequence representation learning framework. h^e and h^d are respectively encoding and decoding RNN latent variables (see Appendix A). Broken arrows are inference (posterior), emission and control (prior). Plain arrows are generation. Parameters ϕ and ψ are shared between all time-steps. Blank cells are deterministic.

Drawbacks of VRAE for MTS representation Yet, in [Bowman et al., 2016] they raise the difficulty of learning good representation with VRAE.

Let's consider that autoregressive decoder (h^d, D_ψ) is powerful enough to model the data. We remind that the joint distribution $p(X|U)$ can be decomposed using a chain rule (2.11); hence, a sufficiently large RNN can theoretically model it. We note $p_\psi(X|U)$ the learned model of the data, such that $p_\psi(X|U) = p(X|U)$. Hence the left-hand term (negative log-likelihood) of (2.19) is minimal, no matter the learned approximate posterior distribution $q_\phi(Z|X,U)$. De facto, $q_\phi(Z|h_T^e)$ can match any prior, i.e. $KL\left(q_\phi(Z|h_T^e)||p(Z|U)\right) = 0$, minimizing the right-hand term of (2.19) without affecting the left-hand term. We have minimal loss but no relevant information in the latent space. All the information is in the decoder.

Remark 20. *This problem extends to all high-capacity decoders, i.e. decoders that inherently can model the studied data distribution.*

Trick to learn whole-MTS latent representations To avoid this drawback of VAE-based representation learning and obtain expressive posterior distributions, several solutions have been proposed.

- Control the convergence: in [Bowman et al., 2016] they control the convergence of the KL by annealing it with a coefficient β, first set at zero and then progressively increased to 1.
- Enrich the prior: in [Tomczak and Welling, 2018], to prevent the posterior to ignore a too simple (hence irrelevant) latent space distribution, they learn a richer prior than standard normal prior, by learning auxiliary meta-variables that condition a multi-model prior. It empowers the over-regularized standard VAE using a too simple (inexpressive) latent distribution $\mathcal{N}(0, I)$.
- Enforce information: several papers proposed to use the mutual information (MI) between data and latent to induce good representation. In [Alemi et al., 2018] they recycle the *information bottleneck method* [Tishby et al., 2000] stating that latent variable representation is a trade-off between distortion (reconstruction) and complexity (rate of compression). Using the MI maximization as an objective had many iterations, from blind-source identification [Bell and Sejnowski, 1995] to VAE, with implicit [Zhao et al., 2017] or explicit [Alemi et al., 2018] MI in the objective.
- Change the VRAE: another way to limit the vanishing of latent information is to modify the framework by introducing inductive bias, for example by modifying the shapes and/or properties of the encoder-decoder or by adapting and using the previously introduced posterior enriching methods. For example, in [Fortuin et al., 2018], they enrich a VRAE with several features. They jointly learn a discrete VAE [van den Oord et al., 2017] for the embedding, a self-organizing map [Kohonen, 1982] to find a neighborhood structure in the latent space and a Markov transition model [Geyer, 1992] for the dynamics. The objective is to learn a discrete representation of trajectories. Other example, important for the contribution of Chapter 3: in [Kipf et al., 2018], they propose the neural relational inference (NRI), a VAE that transforms time series into a binary relational graph, trained as a variable selection method for neural time series model. To do so, they adapt relation neural network (RelNN) and graph neural networks (GNN, see Appendix A) to embed time series data. In Chapters 3, we use similar encoder to infer causality in MTS samples.

Finally, the solutions to enrich the RAE and described in paragraph **Neural representation** in Section 2.3.2 can also be used.

Remark 21. *About the last cited paper [Kipf et al., 2018], using the relational inductive bias has become standard [Battaglia et al., 2018] since it is common to have semantic information that depends on the relation between objects in data. In MTS, many data are intrinsically relational, for example in action recognition from body sensors [Shi et al., 2019, Asadi-Aghbolaghi et al., 2017] where the graphical structure that describes the relation between the sensors is the true skeleton.*

These different examples of NN-based representation learning methods for MTS data illustrate an important point: it is simple to adapt the framework with inductive bias when using neural networks, thanks to their high flexibility and the simplicity of the associated losses.

We have presented in this section the fundamental elements of MTS representation learning, on which we build the contribution of the thesis.

2.5 Graph representation

In the contributions of the thesis, we also treat the problem of the classification of graphs, in chapters 5 and 6. The generic concepts about representation learning presented in Section 2.1 remain valuable for the graph representation problem. Yet, like for MTS data, we need to introduce specific knowledge on graphs.

2.5.1 Definitions and notations

A graph G is a representation of the relation between objects called *vertices* or *nodes*. We note n_G the number of nodes. The relation between the nodes is represented by *edges*. Illustration of several graphs is given in Figure 2.18.

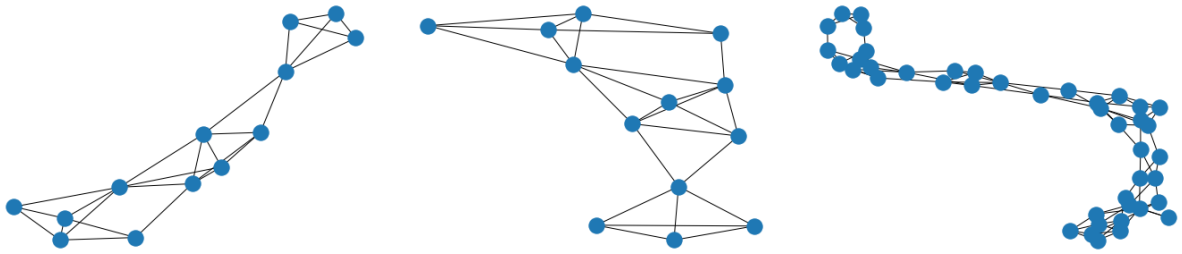


Figure 2.18 – Drawing of three graphs. The first two are the same graph with different views. The last is a longer graph. Difficulties are: finding a representation that is invariant with respect to the ordering of the nodes (to align the two equals) and creating an embedding function that deals with variable sized inputs.

More formally, we note $G = (V, E, W)$ a graph where V is the set of vertices (also names nodes) and $E \subset V \times V$ is the set of edges. The properties of a graph are characterized by its structure (i.e. the set of edges), generally represented by a binary *adjacency* matrix noted $A \in \mathbb{R}^{n_G \times n_G}$ such that $A_{ij} = \mathbb{1}_{(i,j)\in E}$. The entries of the adjacency matrix can also be real (generally in $[0, 1]$), defining a weighted adjacency matrix noted W.

When W (or A) is symmetric (i.e. $W = W^T$), we say that the graph is *undirected*. It means that $\{(i, j) \in E\} \equiv \{(j, i) \in E\}$. It is common to have undirected relations between objects; for example the

covalent bonds between the atoms of the molecules or the friendship relations in social networks. In the graph chapters 5 and 6, we only treat undirected graphs.

Additional vertex or edge features can be added upon G, yet in this thesis we only treat graphs without auxiliary features, with the objective to understand how to leverage the information contained only in the structure of graphs.

Isomorphism-invariance A particularity of graphs is that they have no natural ordering of the nodes. In fact, a graph is a set of nodes and edges that can be jointly permuted. For example, if $V = \{0, 1, 2\}$ and $E = \{(0, 1), (0, 2), (1, 0), (2, 0)\}$, and if π is a bijective index permutation such that $\pi(i) = 2 - i$, then $G^\pi = (V^\pi, E^\pi, W^\pi)$ with $V^\pi = \{0, 1, 2\}$ and $E^\pi = \{(2, 1), (2, 0), (1, 2), (0, 2)\}$ is equal to G. Yet, the adjacency W^π of G^π is different from W but is equal up to a permutation matrix Π corresponding to permutation operator π, i.e. $W^\pi = \Pi W \Pi^T$. The equality of two graphs up-to permutation of the node index is called *isomorphism*.

Remark 22. *The isomorphism problem is the problem of finding if two graphs are isomorphic. This problem is not known to be NP-complete nor solvable in polynomial time. Yet, it is a NP-hard problem.*

When looking at graph representation $F(G)$ for graph-level tasks (e.g. classification), we need *isomorphism-invariance*, i.e. invariance to node indexing:

$$F(G) = F(G^\pi).$$

Laplacian A standard object associated to graphs is the Laplacian matrix, computed as $L = D - W$, where $D = diag(W\mathbf{1})$ is the degree matrix. The degree of a node, in the undirected case, is the number of edges that are incident to the node. We note that degrees are intrinsic node features. The vector of degrees is computed as $W\mathbf{1}_{n_G}$, where $\mathbf{1}_{n_G}$ is the vector of n_G ones. Laplacian matrix is a known and commonly used representation of graphs [Merris, 1994]. Its spectrum will be the subjects of Chapter 6.

2.5.2 Graph representation for whole-graph classification

The embedding of graphs consists in extracting information that is hidden in the graphical structure. Yet, the same way different tasks at different scales exist for time series representation, *graphs representation* or *graphs embedding* [Cai et al., 2018] incorporates several concepts:

- *Node embedding* consists in mapping each node of a graph to a vector that lives generally in a low-dimensional vector space. The embedding are built to preserve topological properties of the graphs in the representation space, for example the similarity between nodes (i.e. the similarity of their neighborhood structure)
- *Node classification* consists in learning a node-to-label assignment from ground-truth node-labelling
- *Node clustering* maps each node into a cluster/group such that nodes in the same clusters correspond to nodes living in a densely connected area of the original graph

- *Whole-graph embedding* consists in mapping the graph to a vector that lives in a low-dimensional vector space

It is possible to obtain an isomorphism-invariant whole-graph embedding from node-embedding by using an aggregation of the nodes embedding that is invariant with respect to a permutation of the nodes index.

Graph embedding benefits in a wide set of tasks: community detection [Hollocou et al., 2018], recommendation [Zhou et al., 2017], link prediction [Wei et al., 2017], graph classification [Zhang et al., 2018], etc. All these methods are based on a representation of the nodes and/or the graph where the graph structure is preserved. If the task is at node-level, preservation of the neighboring in the embedding is generally sought. If the task is at graph-level, the graph-level similarity should be preserved. Since we are only interested in *whole-graph classification* in Chapters 5 and 6, we propose to only introduce whole-graph representation methods that have applied to this task.

Remark 23. *We note that the notion of similarity between whole-graphs is not obvious since graphs have variable size, no trivial alignment and are invariant to permutation of the node indices. We note that we find the first two drawbacks in time series analysis (see Section 2.3.2). The third drawback is kind of opposite: in MTS representation, the order of the observation must be preserved.*

For better understanding, we divide them into three categories: graph kernel methods, feature-based methods and deep learning.

Graph kernel methods Informally, a kernel is a function that computes the similarity between two objects. Formally, it is the inner product of a nonlinear projection of the two objects in a high/infinite-dimensional space. The *kernel trick* [Shawe-Taylor et al., 2004] avoids to compute explicitly the coordinates in the high-dimensional space, only the inner product between all pairs of data: it is an *implicit embedding methods*. The graph-kernel methods perform pairwise comparisons between atomic substructures of the graphs. The similarity between two graphs is the number of matching pairs of substructures. These substructures can be graphlets [Shervashidze and Borgwardt, 2009], subtree patterns [Shervashidze et al., 2009], random walks [Vishwanathan et al., 2010] or paths [Borgwardt and Kriegel, 2005]. Yet, the problem with these graph kernels is that they scale poorly with respect to graph size or dataset size. In [Shervashidze et al., 2011], they propose a general efficient kernel the encompasses aforementioned families of graph kernels and scale to large graphs, based on Weisfeiler-Lehman (WL) test of isomorphism (an effective test that enable to identify and distinguish a large broad of types of graphs [Weisfeiler and Lehman, 1968]). The main difficulty lives in the choice of appropriate algorithm and kernel that accept graphs with variable size while remaining computationally efficient (kernel methods, more generally than their application on graphs, can be computationally expensive but techniques like the Nyström algorithm [Williams and Seeger, 2001] allow to lower the number of comparison with a low rank approximation of the similarity matrix). In [Nikolentzos et al., 2017], they mention the fact that graph kernels mostly focus on local properties of graphs whereas we want to compare whole-graphs. They then propose two kernel methods based on global properties of the graphs, which match eigenvectors of the adjacency matrix (spectral embedding of the nodes) for all pairs of graphs in a dataset. First algorithm is an instance of the Earth Mover's Distance (EMD) [Rubner et al., 2000]. A second is based on Pyramid Macthing (PM) [Grauman and Darrell, 2007], which maps unordered feature sets (set embedding of nodes) to histograms and compute histogram intersection.

Feature-based methods Feature-based (FB) representation methods [Barnett et al., 2016] represent each graph as a concatenation of relevant features. In particular, for graph classification, the features must be discriminative. The feature-based representation can offer a certain degree of interpretability and transparency since the features are chosen and built/learned with a strong inductive bias. The most basic ones are the number of nodes or edges, the histogram of node degrees. These simple graph-level features offers by construction the sought isormorphism-invariance but suffer from low expressiveness. More sophisticated algorithms consider dynamic features (DyF) based on attributes of random walks on the graph, for example correlation patterns of node attributes seen by a random walker at different instants [Gómez and Delvenne,]. Others are based on graphlets [Kondor et al., 2009]. In this situation, the embedding of a graph is then the number of occurrences of these substructures within it. [Kondor and Borgwardt, 2008] explicitly built permutation-invariant features by mapping the adjacency matrix to a function on the symmetric group. [Verma and Zhang, 2017] proposed a family of graph spectral distances (FGSD) to build graph features, based on functions of the eigendecomposition of the graph Laplacian.

Deep learning based methods The deep learning on graphs is mainly treated with graph neural networks (GNNs) that are formally presented in Appendix A.1.1. A survey on GNNs can be found in [Wu et al., 2020]. Here we only give the interesting bibliography regarding the graph classification problem treated in Chapters 5 and 6.

GNNs learn representation of nodes of a graph by leveraging together their attributes (features attached to the nodes), information on neighboring nodes and the attributes of the connecting edges. When graphs have no vertex features, the node degrees are used instead. To create graph-level representation instead of node representation, nodes embedding are pooled by a permutation invariant readout function like summation or more sophisticated information preserving ones [Ying et al., 2018, Zhang et al., 2018].

The most known DL-based graph embedding methods are the *graph convolutional neural networks* (GCNNs). They use convolutions on graphs, considering like for image that relevant information is multi-scaled and that the multi-scaling can be achieved with hierarchical localized receptive fields. The GCNNs fall into two families of models: the spectral GCNNs and the spatial GCNNs.

Spectral GCNNs assume that the node features are a signal living on the graph. In [Bruna et al., 2013], they define the convolutions as noise-filters for the signal, in the spectral domain, using a graph Fourier transform [Shuman et al., 2013]. An approximation of the spectral GCNNs is given in [Defferrard et al., 2016], where they propose an efficient localized filter (i.e. that does note depend on the graph's size). A main limit of spectral-based lies in the fact that they are designed to process several signals on one graph. Yet, for graph classification, we are interested in processing zero or one signal on many graphs. In fact, contrary to spectral methods that operate on the Laplacian eigenspace, the spatial GCNNs operate in the graph domain to propagate a signal along the edges of the graphs. In [Niepert et al., 2016], they apply convolutional filters on locally connected region of arbitrary graphs. In [Atwood and Towsley, 2016] they see convolution as a diffusion process and propose a Diffusion Convolutional Neural Network (DCNN). Then, different methods and architectures were proposed to enrich the signal propagation/message-passing process of [Kipf and Welling, 2016], mainly by changing the functions that aggregate neighborhood information. For example, in [Hamilton et al., 2017], they propose a node embedding that is based on random Graph SAmpling and aGgrEgation (GraphSAGE) of neighbor's features to help the GCNN better generalize on unseen

nodes.

Yet, most of the existing graph-level neural networks are based on GCNNs that first embed nodes, not graphs. Hence, a dimensionality reduction strategy is required. Two main families of graph dimensionality reduction methods exist. First, we can apply pooling (max, mean) after the message-passing steps [Duvenaud et al., 2015, Li et al., 2015]. Second, we can cluster node representation, hierarchically coarsening the graph [Defferrard et al., 2016, Niepert et al., 2016, Monti et al., 2017]. For example, in [Zhang et al., 2018] they propose a pooling where the nodes are sorted and only the first are kept. They call this end-to-end graph classification method Deep Graph Convolutional Neutal Network (DGCNN). In [Ying et al., 2018], they propose a Differentiable Pooling (DiffPool), where they learn the hierarchy through trainable (hence differentiable) cluster assignment, defining a method that can be applied to any message-passing GNN. In [Luzhnica et al., 2019], they base the coarsening on clique (subset of nodes where all nodes are adjacent) aggregation. Recently, [Xinyi and Chen, 2018] leveraged capsule networks [Sabour et al., 2017] (CapsGNN), neural units designed to better preserve information at pooling time. In [Xu et al., 2018], they propose the Graph Isomorphism Network (GIN) based on Weisfeiler-Lehman (WL) test that achieves WL discriminative power and which is supposed to be intrinsically invariant to isomorphism of the graphs. This last model is built to be the most expressive for graph-level tasks like graph classification, thanks to its intrinsic isomorphism-invariance.

Many other works about graph embedding have been proposed, based on these recent advances. In particular, we can cite the theoretical works [Maron et al., 2018, Maron et al., 2019, Azizian and Lelarge, 2020] done in the continuity of [Xu et al., 2018], which try to define what a GNN can extract from graphs with new isomorphism-invariant architectures.

In this introduction, we have outlined the main concepts of representation learning, and their extension to the time series representation and whole-graph classification. We gave required knowledge to understand the contributions of the thesis exposed in the rest of the document.

Chapter 3

VAR models and Granger causality

Abstract In this chapter, we provide and experiment a new model-based multivariate time series (MTS) representation learning method using causality graphs, built as an encoder-decoder. In particular, we show in experiments that our model is appropriate to monitor ageing mechanical systems. This chapter covers two publications:

- Pineau, E., Razakarivony, S., and Bonald, T. (2019). Seq2var: multivariate time series representation with relational neural networks and linear autoregressive model. *Advanced Analysis and Learning on Temporal Data, pages 126–140, in Lecture Notes on Artificial Intelligence*, Springer.
- Pineau, E., Razakarivony, S., and Bonald, T. (2020). Unsupervised ageing detection of mechanical systems on a causality graph. *International Conference on Machine Learning and Applications*. Oral.

3.1 Seq2VAR: efficient VAR parameters inference

3.1.1 Problem setup

Let $\mathcal{X} \subseteq \mathbb{R}^{d \times T}$ be a finite set of d-dimensional multivariate time series (MTS), each indexed over the discrete time range $t = 1, \dots, T$. We first assume that each sample X follows a VAR model, i.e. for each time series X there exists a tensor $W^X \in \mathbb{R}^{K \times d \times d}$ such that $\forall t \in [\![K, T]\!]$

$$X_t = \sum_{k=1}^{K} W_k^X X_{t-k}. \tag{3.1}$$

Model (3.1) is a d-dimensional linear vector autoregressive model of order K, noted VAR(d, K), with *sample-wise parameter* W^X. Under the VAR assumption (3.1), couple (W^X, X_0) is an exhaustive representation of time series X. Hence, comparing the representation (W^X, X_0) of the time series $X \in \mathcal{X}$ is a way to compare the samples X. The initialization X_0 being observed, the difficulty of the problem is the inference of the W^X for each $X \in \mathcal{X}$.

In standard approaches, the parameters W^X are estimated by maximum likelihood, in closed form (under certain assumptions) or with a likelihood function estimated from data. This likelihood function can for

example be built for example, in the linear case, with a Kalman filter (see background in appendix). Each W^X must be estimated individually, for each sample X. There are two problems with this approach. First, the standard estimation of the linear parameter for a given sample does not help for the estimation of the parameter of another sample. There is not capitalization of the knowledge contained in the whole dataset. Second, when the dataset is large, when the data are high-dimensional or when W^X has constraints (sparsity, symmetry, etc.), estimating W^X for all samples is not tractable. We need an efficient inference function.

In this chapter, we propose to consider the VAR inference problem as a latent variable representation learning problem, illustrated in Figure 3.1. As for autoencoder based LVM, we substitute the true posterior distribution $p(W^X|X)$ by a neural network based posterior p_ϕ that maps each sample X into VAR parameter space using a learned embedding neural network F_ϕ with parameters ϕ. We call the model *sequence-to-VAR* (Seq2VAR).

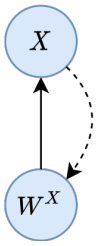


Figure 3.1 – Graphical model of the sample-wise VAR tensor inference problem.

3.1.2 Seq2VAR: an encoder-decoder for efficient VAR parameters inference

Our model belongs to *encoder-decoder* framework. We remind that an encoder-decoder consists of an encoder F_ϕ that takes a time series $X \in \mathcal{X}$ as input and outputs representation $Z = F_\phi(X)$ of X such that X can be reconstructed from Z using a decoder D, i.e. $D(Z) \approx X$. More details are given in the Chapter 2 Section 2.2.

In our approach, we train $F_\phi(X)$ to be an estimation $\hat{W}^X$ of the tensor parameter W^X defined in (3.1). In this situation, the decoder is then simply the autoregressive model (3.1). It has no parameters to learn.

We remind that the causality is a dynamical *interaction* between the variables of a multivariate time series, which is naturally represented in sparse VAR models (see Section 2.3.1). Hence, an appropriate neural inference function F_ϕ is a neural network that explicitly represent the interactions between variables. In [Santoro et al., 2017], they propose a type of neural network, called *relational neural network (RelNN)*, that embeds the interactions between objects in images for relational reasoning. Details about this particular neural network architecture is given in Appendix A.1.5.

In our case, the objects are the d individual time series composing our samples. A version of a RelNN for time series has been proposed in [Kipf et al., 2018]. There are two differences with the standard RelNN proposed in [Santoro et al., 2017]. First, the embedding function, noted f_{var}, first contracts the time dimension of the MTS samples to obtain a vectorial representation of dimension d of each time series. Second, in [Kipf et al., 2018], the RelNN is expanded with a graph neural network (GNN, see Appendix A) that takes as input the created tensor containing the embedding of each pairwise relation which can be seen as fully-connected graph with d^2 nodes whose edge features are the relations embedding. This GNN layer en-

ables the model to explicitly take into account the fact that each pairwise interaction $X^{(i)} \leftrightarrow X^{(j)}$ between variables also depends on the respective interactions of i and j with the other variables $\{1, \dots, d\} \setminus \{i, j\}$ (several pairwise interactions per variable). In Seq2VAR we use the expanded version of [Kipf et al., 2018] and specialize it for the inference of the VAR parameters, as illustrated in Figure A.5. The difference with [Kipf et al., 2018] is that we specialize the network to find the linear causalities hidden in data.

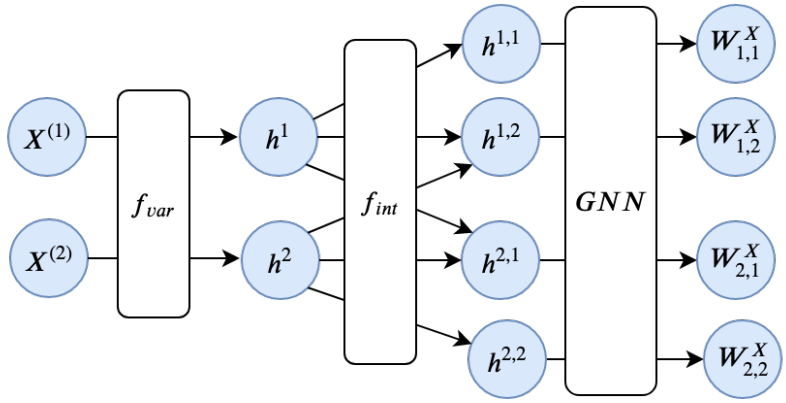


Figure 3.2 – Graphical model of the expanded relational neural network, with $d = 2$, specialize to infer tensor W^X given time series X. *GNN* means graph neural networks.

Remark 24. *Beyond the known interest of the relational inductive bias [Battaglia et al., 2018], the interest of the RelNN compared to a standard neural network for time series data (e.g. recurrent neural networks) is illustrated in Section 3.4.*

We note the RelNN encoder $F_\phi : \mathbb{R}^{d \times T} \to \mathbb{R}^{K \times d \times d}$. The global encoder-decoder problem of Seq2VAR is then:

$$\min_{\phi} \sum_{X \in \mathcal{X}} \left\{ \sum_{t=K+1}^{T} \left\| X_t - \sum_{k=1}^{K} F_\phi(X)_k X_{t-k} \right\|^2 + \lambda \Omega \left(F_\phi(X) \right) \right\}. \tag{3.2}$$

where Ω is a regularization function and $\lambda \in \mathbb{R}_+$ its coefficient of importance. This regularization can be used, in particular, to induce and control sparsity in estimated parameters $\hat{W}^X$.

Seq2VAR is illustrated in Figure 3.3.

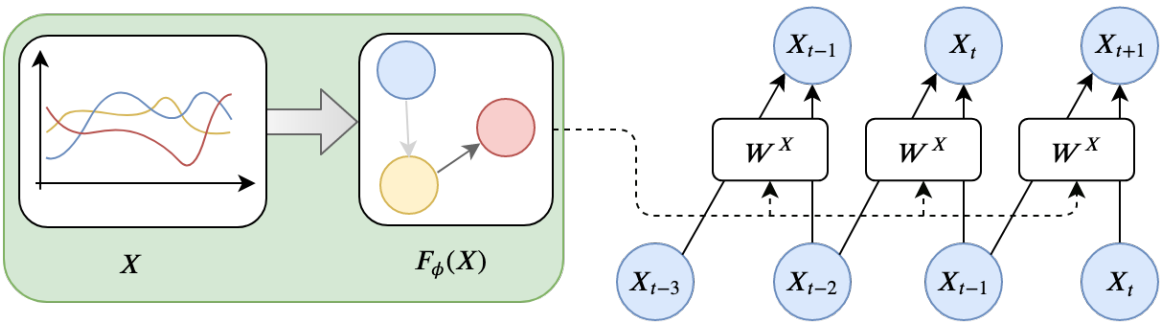


Figure 3.3 – Seq2VAR model infers a VAR parameter from time series data. Here $K = 2$.

Remark 25. *The decoder's parameters ϕ trained on many examples retain some general structural information about the data, hence about the system that generated the data (see Figure 3.4), naturally helping the decoder to generalize to new samples.*

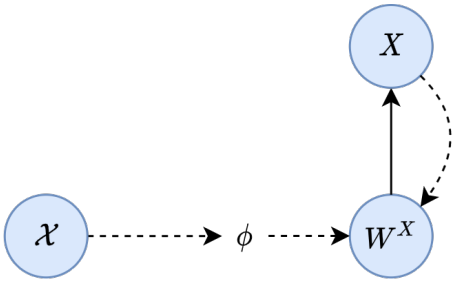


Figure 3.4 – Graphical model of the inference of W^X (broken arrow) and generation of X (plain arrow). The parameters ϕ are inferred from the whole dataset.

Yet, this dataset-level regularization is latent. In Section 3.2, we propose a more explicit dataset-level regularization of the problem to improve representation quality.

3.1.3 Adapting Seq2VAR for constrained VAR parameter inference

3.1.3.1 Sparse Seq2VAR for causality detection

We remind that in VAR(d, K) (3.1), the sparsity is directly linked to the notion of Granger causality (Section 2.3.1): if for $(i, j) \in [\![1, d]\!]^2$, there exists $k \in [\![1, K]\!]$ such that $W^X_{k,i,j} \neq 0$, then $X^{(i)} \overset{\text{GC}}{\Longrightarrow} X^{(j)}$. The non-zero values of W^X define the causalities in X. De facto, sparsity in VAR model plays an important role in the quality of the representation of the sample X by parameters W^X since it is the signature of the causal structure. Moreover, as stated in the introduction, sparse VAR models reduce the statistical ambiguity that exists because of correlation between variables.

In theory, in a regression, true sparsity is achieved with L_0 regularization of the weights. The L_0 norm of a vector is the number of non-zero entries. This norm is not differentiable. It cannot be used as Ω penalty. It is generally replaced by a LASSO penalization, i.e. a L_1 norm.

In VAR model (3.1), it is standard to penalize at the same time all lags parameter W^X. It enables to shut down *all* the causalities from a variable to another. It is called the group-LASSO penalization [Lozano et al., 2009]. In the introduction (Figure 2.10), we saw that a simple LASSO penalization of tensor W^X, i.e. $\Omega(W^X) = \sum_{i,j,k} |W^X_{i,j,k}|$, would possibly converge to the solution $W^X_{1,2,1} = W^X_{1,1,2} = 0$ and $W^X_{2,2,1} = W^X_{2,1,2} = 0$. Hence, it would allow crossed causalities, creating an ambiguous causality structure. A contrario, penalizing all lags at the same time, i.e. $\Omega(W^X) = \sum_{i,j} \|W^X_{i,j}\|_F$ would induce the model to find $W^X_{1,2,1} = W^X_{1,2,2} = 0$ or $W^X_{2,1,1} = W^X_{2,1,2} = 0$, i.e. no cross-causalities.

Remark 26. *Different types of grouped penalization exist, depending on the a priori we have on the sough causality [Jacob et al., 2009, Tank et al., 2018]. We focus on variable group sparsity.*

In our case, we need a specific sparsity for each sample X, since the X have specific parameter W^X. We therefore need the encoder F_ϕ to be enriched so that it can infer not just a VAR parameter but a sparse VAR parameter for each sample X. We propose to use a sample-adaptative binary masks. This type

of masks is commonly used in attention mechanisms [Bahdanau et al., 2014, Xu et al., 2015] or in long-memory recurrent neural network (LSTM) [Hochreiter and Schmidhuber, 1997] to filter the useless memory information.

We have seen in the introduction that for encoder-decoder problems, we can add a probabilistic model to the latent representation (see latent variable models Section 2.2), and estimate the posterior distribution of the latent representation with neural networks. Here, since we are interested in sparsity, we propose to use a Bernoulli distribution as probabilistic model for the edges of the (binary) Granger causality graph, noted $\mathcal{G} = (V, E)$, with V is the set of d nodes (here the d variables of the multivariate time series X) and E the set of edges (the causalities). We note A its binary adjacency, such that $A_{i,j} = 1$ if $(i, j) \in E$, $A_{i,j} = 0$ otherwise. For Seq2VAR, we a priori consider that edges are independently distributed:

$$p(\mathcal{G}) = \prod_{(i,j)\in E} p^{A_{i,j}}(1-p)^{1-A_{i,j}} \tag{3.3}$$

The independence of the edges justifies from the fact that causalities in W^X are assumed independent conditionally to X.

The posterior distribution is estimated with neural networks. We note $P_\phi(X) := p_\phi(\mathcal{G}|X)$ the probabilities of the Bernoulli sampling estimated with neural network P_ϕ.

Remark 27. *We use the notation ϕ for the parameters of P_ϕ since in practice, we only change the output dimensions of F_ϕ to obtain P_ϕ. Hence $F_\phi(X) \in \mathbb{R}^{2K\times d\times d}$ for LASSO, $F_\phi(X) \in \mathbb{R}^{(K+1)\times d\times d}$ for group-LASSO. And $P_\phi(X) = F_\phi(X)_{\geq K}$.*

To obtain a LASSO-like result, we need $P_\phi(X) \in \mathbb{R}^{K\times d\times d}$. To obtain a group-LASSO-like result, we need $P_\phi(X) \in \mathbb{R}^{1\times d\times d}$ (lags of the variables are turned on and off at the same time). Hence, we have the posterior distribution $p(\mathcal{G}|X) := P_\phi(X)$. We note $G_\phi(X)$ the binary gates sampled with probability $P_\phi(X)$.

To obtain truly binary gates (i.e. whose distribution is concentrated at 0 and 1) and not probabilities of gates in $[0, 1]$ as usual in approximate binary decision, we propose to use an adaptation of the Gumbel-softmax trick [Jang et al., 2016, Maddison et al., 2016] proposed in [Li et al., 2018]:

$$G_\phi(X) = \sigma\left(\frac{P_\phi(X) + \log U - \log(1-U)}{\tau}\right), \tag{3.4}$$

where σ is the sigmoid function, τ is a temperature parameter. It controls the relaxation of the binary sampling. When $\tau \to 0$, the sampling tends to a true Bernoulli sampling but loses its differentiability. A contrario, a high temperature implies more continuous relaxation: concentration around 0 and 1 is relaxed, and samples spread in $[0, 1]$, towards uniform sampling as τ grows. U is a random tensor with same size than $P_\phi(X)$, whose entries are independent identically distributed $\mathcal{U}(0, 1)$. σ, log and $+$ are scalar operators.

Now we can control the sparsity of F_ϕ by controlling the gates/edges posterior distribution $P_\phi(X)$, hence without affecting the values of the entries of $F_\phi(X)$:

$$\sum_{X\in\mathcal{X}}\left\{\sum_{t=K+1}^{T}\left\|X_t - \sum_{k=1}^{K}\left(G_\phi(X)\odot F_\phi(X)\right)_k X_{t-k}\right\|^2 + \lambda\Omega\left(P_\phi(X)\right)\right\}, \tag{3.5}$$

where $\odot$ is the Hadamard product. In [Louizos et al., 2017], they propose similar method to prune a neural network, with a slight modification of the relaxed binary sampling.

In variational inference, the penalization is

$$\Omega\left(P_\phi(X)\right) = \sum_{(i,j)\in E} P_\phi(X)_{i,j} \log\left(\frac{P_\phi(X)_{i,j}}{p}\right) + (1 - P_\phi(X)_{i,j}) \log\left(\frac{1 - P_\phi(X)_{i,j}}{1-p}\right) \tag{3.6}$$

i.e. the Kullback Leibler divergence between $P_\phi(X)$ and $\mathcal{B}er(p)$. We found that in certain situations, like in VAE problem, KL penalization sometimes collapse such that all entries of the posterior $P_\phi(X)$ are p, for all X. The inference estimator P_ϕ is stacked in a confortable local minima. To circumvent this problem, penalizing $G_\phi(X)$ instead of $P_\phi(X)$ proves mostly stabler behavior (no collapse). Moreover, we remark that we can approximate the L_0 norm of the estimated W^X as follows:

$$\|G_\phi(X) \odot F_\phi(X)\|_0 \overset{\tau\to 0}{\sim} \|G_\phi(X)\|_1$$

Hence, we can simply use the following penalization:

$$\Omega\left(G_\phi(X)\right) = \left|\frac{1}{C}\|G_\phi(X)\|_1 - p\right| \tag{3.7}$$

where $C = Kd^2$ or $C = d^2$ (depending on the grouping of LASSO penalty) instead of KL (3.6).

In practice, without a strong a priori on p, we may chose $p = 0$ to obtain the sparsest possible graph, or more commonly chose uniform prior $p = 0.5$. The hyper-parameter λ determines the importance of the apriori during the learning of parameter ϕ. If we choose weak a priori cited above, we generally choose a low λ.

We therefore have a representation model for multivariate time series based on the linear dynamics assumption. This assumption enables to explicitly take the existence of causalities between the variables into account, thanks to a *sample-wise* sparsity inference.

Remark 28. *A particular version of sparse Seq2VAR is the binary Seq2VAR (B-Seq2VAR), where the tensor W^X is binary, i.e. estimated only with $G_\phi(\cdot)$. This case is illustrated in Section 3.3.1.2.*

3.1.3.2 Symmetric Seq2VAR

Another constraint that can be easily imposed to Seq2VAR is the *symmetry* of the tensor W^X. We can simply force W^X to be symmetric by imposing $W^X = 1/2(F_\phi(X) + F_\phi(X)^{T(1,2)})$, where $T(1,2)$ is the transposition of dimensions 1 and 2 of the tensor. This setting can be important when the causalities are bidirectional but not synchronized (otherwise it would be a correlation). It is the case where the interactions between variables are physical or mechanical.

In this section, we have proposed Seq2VAR, an encoder-decoder framework that learns to efficiently infer VAR parameters, and in particular under sparsity and symmetry constraints. In the next section, we focus on a particular case and an extension of Seq2VAR where the whole dataset shares a common causal model.

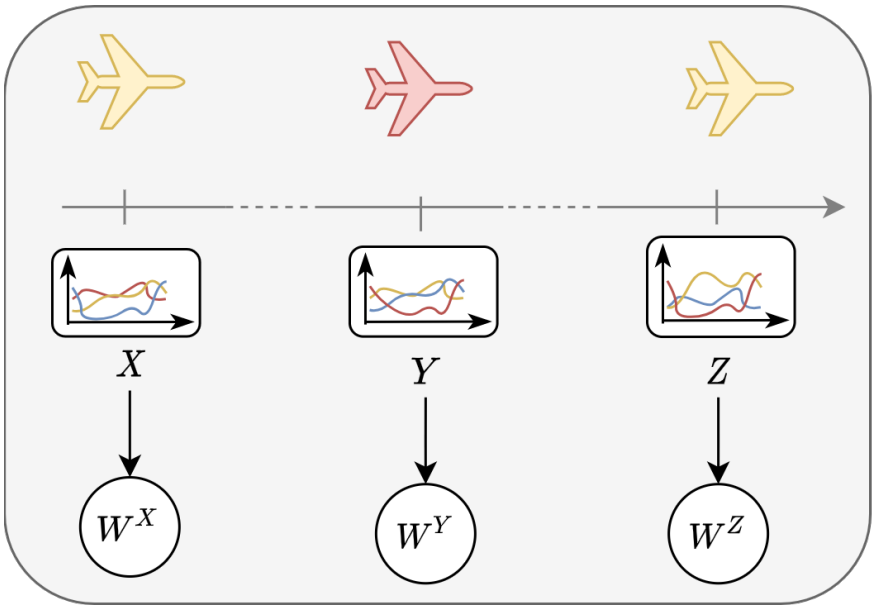


Figure 3.5 – We transform time series X, Y and Z into weights of a unique causal graph $\mathcal{G}$ that represents the causal structure underlying the mechanical system (here a plane). The objective is to compare the weights W^X, W^Y and W^Z to characterize the *relative state* of the system, represented by its color. For example, if the causality assumption is relevant, W^Z should be closer from W^X than from W^Y. We note that a maintenance was effectuated between Y and Z to restore the system's state.

3.2 Seq2Graph: Seq2VAR for system's state monitoring

3.2.1 Context

When observed samples are generated from a unique mechanical system $\mathcal{S}$, it is common to assume a *unique* causality structure $\mathcal{G}$ for the whole dataset $\mathcal{X}$. For example, $\mathcal{G}$ can be the skeleton of an articulated body or represent the causal relationships (statistical or physical) between sensors arranged in an engine, from which we observe samples. An illustration is given in Figure 3.5. In this situation, the causal structure $\mathcal{G}$ is shared between all observed samples. When the assumption of a unique graphical structure for all samples is valid, like in the aforementioned examples, a causality graph $\mathcal{G}$ can be extracted directly from the studied dataset $\mathcal{X}$, using an appropriate statistical method. $\mathcal{G}$ is then an abstract representation of the system $\mathcal{S}$ (e.g. a mechanical system) that generated all the observed samples.

Yet, Granger causality inference only aims at discovering and interpreting causalities between observed variables at the scale of the dataset (i.e. one graph for all samples). In this section, we propose to relegate the inference of the causal graph $\mathcal{G}$ as a preliminary task preceding the representation inference of individual MTS samples. Our assumption is the following: since $\mathcal{G}$ is the graphical model of the system $\mathcal{S}$ that generated all the observed samples (e.g. a mechanical system), $\mathcal{G}$ is also a natural meaningful latent structure on which each data sample can be represented. In particular, each sample X can have its own set of edge's weights W^X on $\mathcal{G}$, hence its own causality-based representation. We can then consistently compare MTS samples by comparing their edge's weights on $\mathcal{G}$. We propose to use the previously presented Seq2VAR model to obtain these graph weights.

3.2.2 Representation on a dataset-level causality graph

We know that in VAR model, $W^X_{\cdot,i,j} = 0$ means the absence of causality from variable $X^{(i)}$ to variable $X^{(j)}$, for a given sample X, [Eichler and Didelez, 2012]. In order to have all samples X represented on the same causality graph $\mathcal{G}$, all W^X should share the same zeros.

In consequence, we propose to use a sparse *random coefficient regression* (RCR) [Muthén et al., 2015] to model our samples (see Appendix B.3) and we assume that each tensor W^X has three underlying components: a dataset-level component $\bar{W} \in \mathbb{R}^{K\times d\times d}$, a sample-wise component $P^X \in \mathbb{R}^{K\times d\times d}$ and dataset-level binary adjacency $A \in \{0,1\}^{d\times d}$, such that:

$$W^X = \bar{A} \odot \left(\bar{W} + P^X\right) \tag{3.8}$$

where $\odot$ is the Hadamard product and $\bar{A}$ the adjacency A extended to match the dimensionality of $\bar{W}$. $\bar{W}$ and A are shared between all samples X. The entries of the sparse tensor $A \odot \bar{W}$ are the edge features of the causal graph $\mathcal{G}$. The entries of the tensor $A \odot W^X$ are the adjustment of the graph to match the properties of sample X.

Remark 29. *The difference between standard RCR [Muthén et al., 2015] and ours is the shared sparsity given by A.*

In practice, we first infer a sparse tensor $\bar{W}$. Then we define a graph adjacency A from $\bar{W}$: $A_{i,j} = \mathbb{1}\{\sum_{k=1}^K |\bar{W}_{k,i,j}| > 0\}$. We then build and train a neural network P_ϕ with parameters ϕ that directly and efficiently outputs the adjustment P^X in the RCR. Hence, the previously introduced neural network F_ϕ is defined as $F_\phi(X) := \bar{W} + P^{\mathcal{G}}_\phi(X)$, where $P^{\mathcal{G}}_\phi(X) = \bar{A} \odot P_\phi(X)$.

The Seq2Graph framework is illustrated in Figure 3.6.

Remark 30. *In term of graph signal processing, $\mathcal{G}$ is a directed graph with adjacency A such that $A^{(ij)} = \mathbb{1}\{\sum_{k=1}^K \|\bar{W}_{k,i,j}\|_2 > 0\}$, the $W_{i,j}$ are the $(K \times l)$-dimensional edge attributes, $X^{(j)}$ is the signal living on node j and the nodes-to-node signal propagation rule is defined by the GVAR (3.9).*

3.2.3 Dataset-level nonlinearities

The same way all the samples may share a unique causal structure $\mathcal{G}$, they may also share a common set of *nonlinearities* to *enrich* and *regularize* the Seq2Graph model. Hence, let $g = \{g_{\theta_j}\}_{j=1}^d$ be a set of shallow neural networks with parameters $\theta = \{\theta_j\}_{j=1}^d$, $g_{\theta_j} : \mathbb{R}^l \to \mathbb{R}$, such that for each sample $X \in \mathcal{X}$ we can find a tensor $W^X \in \mathbb{R}^{K\times d\times d\times l}$ such that $\forall j \in [\![1, d]\!]$:

$$X_t^{(j)} = g_{\theta_j}\left(\sum_{k=1}^K W^X_{k,\cdot,j} X_{t-k}\right) \tag{3.9}$$

with $l \in \mathbb{N}^*$. (3.9) is a neural generalized linear version of a *vector autoregressive* (VAR) model, called *generalized-linear VAR* (GL-VAR) [Tank et al., 2018].

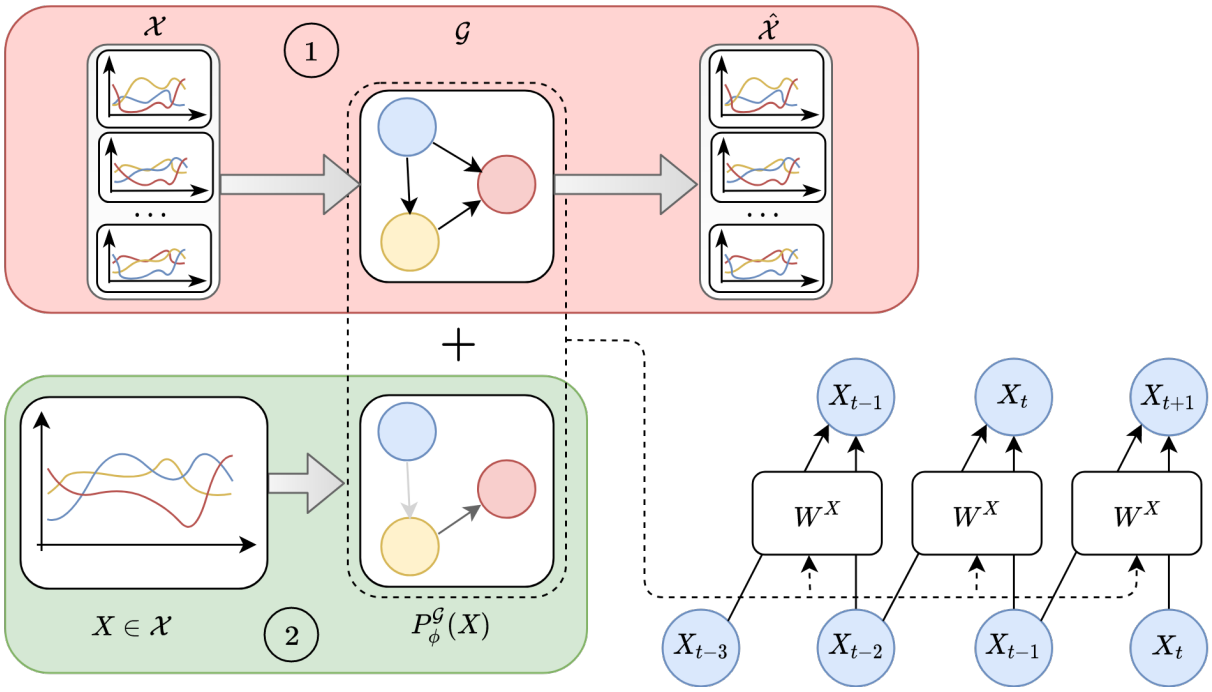


Figure 3.6 – Representation of a sample X with Seq2Graph. ① is the dataset-level causal graph $\mathcal{G}$ inference. $\hat{\mathcal{X}}$ is the set of predictions for the whole dataset $\mathcal{X}$. $\mathcal{G}$ is built such that it explains the mean dynamical behavior of the dataset. ② is the representation inference. $\hat{X}$ is the prediction of the sample X. The adjustments $P_\phi^{\mathcal{G}}(X)$ are built such that they explain the specific dynamical behavior of sample X, along the edge of the mean causal graph.

While in the linear case the full dynamics information is in W^X, here the information about the data is shared between W and g. Yet, the functions g are shared between all data samples. Hence, when representing X with W^X, the weights still contain sample specific information that can be monitored to follow the evolution of the model, as in the linear case.

The advantage of Seq2Graph is twofold. First, sharing the graph means sharing the sparsity. It implies that all samples will be represented in the same low-dimensional space. Second, the RCR structure centers the distribution of the samples representation. It lower the uncertainties of the VAR models fitted in few data (we remind that each sample has its own VAR parameter).

3.2.4 Seq2Graph training

Seq2Graph is a two-step method. This section describes the training of these steps.

3.2.4.1 Dataset-level causal structure inference

The first step of Seq2Graph consist in building the causal graph on which we will represent MTS samples. We search a sparse $\bar{W}$ from RCR model, and in the nonlinear case (3.9) we also learn $\theta = \{\theta_j\}_{j=1}^d$ the set of parameters of the neural link functions. We therefore search a solution to the following mean-squared regression with group-lasso regularization:

$$\min_{\bar{W},\theta} R(\theta,\bar{W}) := \mathbb{E}_{X\sim\mathcal{X}}\left[\sum_{j=1}^{d}\sum_{t=K+1}^{T}\left\|X_t^{(j)} - g_{\theta_j}\left(\sum_{k=1}^{K}\bar{W}_{k,.,j}X_{t-k}\right)\right\|_2^2\right] + \lambda\sum_{i,j=1}^{d}\left\|\bar{W}_{.,i,j}\right\|_F + \gamma\sum_{j=1}^{d}\|\theta_j\| \tag{3.10}$$

where $\|.\|_F$ is the Frobenius norm. The group-lasso penalty for W associated with coefficient λ encourages each $\|\bar{W}_{.,i,j}\|_F$ to be null, meaning that all causal links from $X^{(i)}$ to $X^{(j)}$ would be cut. $\lambda \in \mathbb{R}^+$ controls the speed and intensity of the pruning. Regularization $\|\theta_j\|$ compensates the effect of the group-lasso to avoid the (theoretical) situation where $\bar{W}$ goes to zero while parameters θ tend to infinite sensitivity.

(3.10) is pretrained using a stochastic gradient descent method. Then, in order to obtain a truly sparse weight $\bar{W}$ (with exact zeros), we apply a proximal gradient descent (PGD) method [Parikh et al., 2014]. Principles of proximal optimization and details of the proximal optimization for our problem are given in Appendix B.4.

3.2.4.2 Sample-wise causality adjustment

We assume that sparse $\bar{W}$ and/or non-linearity g_θ have been learned (see Section 3.2.4). We therefore can build the causal structure $\mathcal{G}$ on which we want to represent the samples $X \in \mathcal{X}$. We now train a relational neural network P_ϕ that infers the sample-wise causality adjustment of the RCR model (3.8). We use a similar approach than in Seq2VAR experiments: we note $P_\phi^{\mathcal{G}} = \bar{A} \odot P_\phi$ the adjustment constrained to the inferred causal graph. $\bar{A} \in \mathbb{R}^{K\times d\times d\times l}$ is the adjacency matrix A of $\mathcal{G}$ expanded to fit the output dimensions of tensor $P_\phi^{\mathcal{G}}$, $\odot$ is the Hadammard product. Then the problem to solve is:

$$\min_{\phi} \mathbb{E}_{X\sim\mathcal{X}}\left[\sum_{j=1}^{d}\sum_{t=K+1}^{T}\left\|X_t^{(j)} - g_{\theta_j}\left(\sum_{k=1}^{K}\left(\bar{W} + P_\phi^{\mathcal{G}}(X)\right)_{k,.,j}X_{t-k}\right)\right\|_2^2 + \eta\Omega\left(P_\theta^{\mathcal{G}}(X)\right)\right] \tag{3.11}$$

η is a parameter controlling the intensity of the penalty function Ω. This penalty has been added to enhance the consistency of the samples representation. In fact, we remind that the final objective of the Seq2Graph model is to compare the samples. Gathering the representations around the mean representation $\bar{W}$ insures a consistent behavior.

3.2.4.3 Some implementation details

Sparsity inducing training for the dataset-level causality-graph inference Problem (3.10) is first optimized with stochastic gradient descent. Then, apply PGD [Parikh et al., 2014] as fine-tuning optimization procedure in order to obtain true zeros in $\bar{W}$. If we had a target sparsity, we could stop the PGD when the level is achieved. In our experiment, we do not have (or assume to not have) the true sparsity level. Instead, we propose to monitor the impact of the sparsity on prediction performance, and chose the maximal sparsity that does not degrade the prediction capacity of the model. See ageing detection experiments in Section 3.3.2 for illustration.

Multi-multivariate time series There are cases where the d components of a MTS are multidimensional. For example, if the MTS has d variables situated in a 2D space (see Experiment 3.3.2.1), hence each variable

is represented by a 2D time series. More formally, the problem extends from $\mathcal{X} \subseteq \mathbb{R}^{d \times T}$ to $\mathcal{X} \subseteq \mathbb{R}^{d \times m \times T}$, i.e. $\forall X \in \mathcal{X}\ X_t \in \mathbb{R}^{d \times m}$, with m the dimension of individual time series variables. The approach presented in our paper adapts to this general case by replacing $W^X \in \mathbb{R}^{K \times d \times d}$ by $W^X \in \mathbb{R}^{K \times d \times d \times 1}$. The additional dimension in W^X enables to consider the m time series of each variable as a whole.

3.3 Experiments

We split the experiments in two main subsections. A first subsection shows on two synthetic datasets the ability of Seq2VAR to infer relevant VAR parameters from new data after training, and in particular sparse Seq2VAR and symmetric Seq2VAR presented in Section 3.1.3 are tested out. A second subsection introduces the context in which the dataset-level regularization presented in Section 3.2 is relevant and presents experiments on datasets that contains samples from ageing mechanical systems.

3.3.1 Experiments: causality detection with Seq2VAR

3.3.1.1 Methodology

As a preliminary experimental work, we illustrate our approach on several synthetic datasets, each with several levels of difficulty to illustrate the capacity of Seq2VAR to infer causalities in time series samples. First, in order to assess the *generalization* capacity of Seq2VAR, we use a test set with intrinsic parameters (causality graphs, interaction intensities) that differ from those of the train set in all experiments. Second, *causality discovery* is assessed through the F1-score between inferred and ground truth causality graphs. Third, we can assess the learned representation of samples with a standard downstream task like classification using the inferred tensors W^X. The classification is a 1-nearest-neighbor on the tensors with l^2 norm. The classification is completed in test set representation. Test set is divided in train and test subset for the classification task.

For each experiment, we compare Seq2VAR to NRI [Kipf et al., 2018] (see Section 2.4.5) and VAR [Toda and Phillips, 1994] (see Section 2.3.1), from which we respectively inspire for encoder and decoder. We remind that NRI is a VAE whose latent space contains binary relational graphs. In this setting, the prior (required in VAE settings) of each latent graph is the proportion of ones wanted in the latent graph adjacency. Hence, we always give to NRI model the proportion of ones as prior.

To provide fair comparison benchmarks, the experiments are built such that both VAR and NRI model fit in. We use the implementation of VAR from the *Statsmodel* python library [Seabold and Perktold, 2010].

3.3.1.2 Causality detection in binary linear setting

For our first experiment, we first aim at showing that binary sampling approach is relevant for Seq2VAR model and then at analysing how Seq2VAR behaves in the presence of noise. We place ourselves in a favorable setting for VAR, NRI and Seq2VAR. We generate samples from a stationary 1-order linear autoregressive model, where the linear transition matrix is a permutation matrix. We add several level of additive Gaussian noise to the observations (see Figure 3.7 for an illustration).

We chose $d = 10$, $T = 50$, $K = 1$. We generate 10 permutation matrices for the train set and 10 other

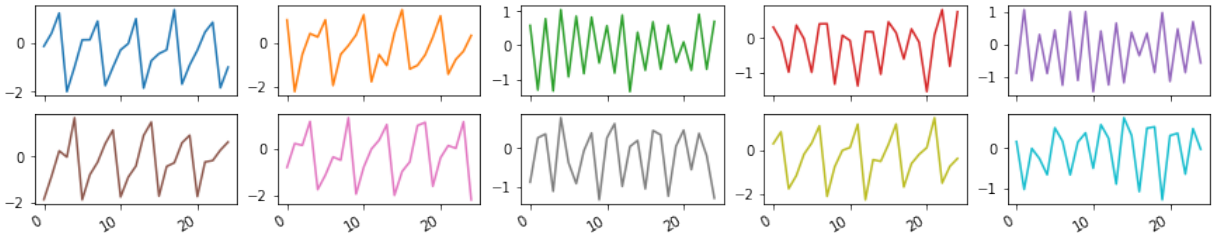


Figure 3.7 – 10-dimensional permutation MTS with observation noise $\mathcal{N}(0, 0.3)$.

permutation matrices for the test set. From each matrix we generate 100 samples with random $\mathcal{N}(0, 1)$ initial conditions. We train two versions of the Seq2VAR: dense (without the binary mask) and pure binary. After training, we show that Seq2VAR has generalized the notion of permutation. We note that the binary version is naturally the only one to find the real permutation matrices. Figure 3.8 shows the outputs of different versions of Seq2VAR encoder at test time, with different level of observation noise.

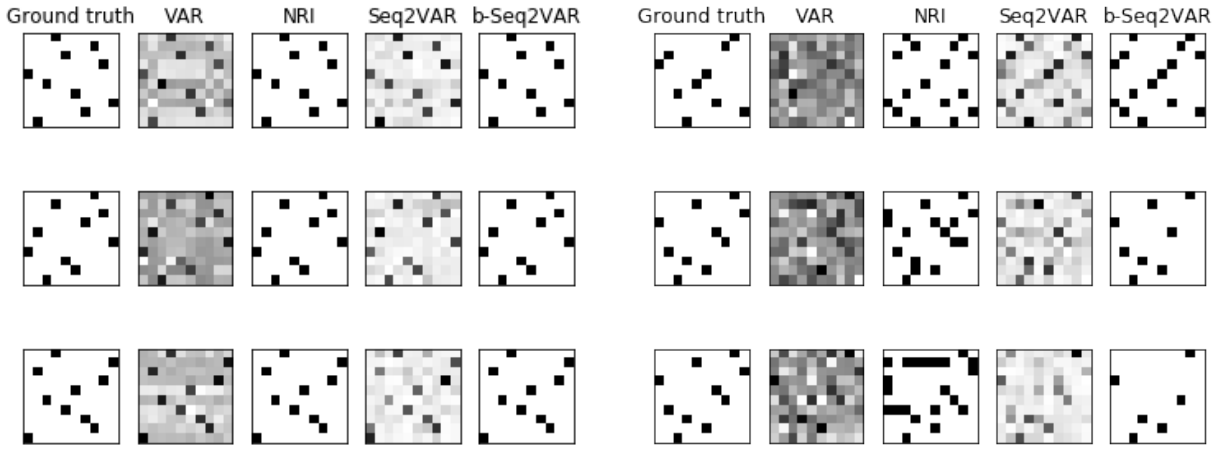


Figure 3.8 – Illustration of inferred transition matrices for different test samples, using VAR, NRI, Seq2VAR and binary Seq2VAR, compared to ground truth, depending on the level of observation noise: $\mathcal{N}(0, 0.1)$ (left), $\mathcal{N}(0, 0.5)$ (right).

We see in Figure 3.8 an illustration of the problems created by an increase of observation noise variance. While signal-to-noise ratio decreases, disentanglement of the permutations from the noise becomes harder. We see that VAR is not robust to noise and that NRI overestimates the density of the binary matrix. On the contrary, Seq2VAR (without regularization) is parsimonious (but not sparse), i.e. it it gathers many entries of $\hat{A}$ around zero (see Figure 3.9).

We explain this behavior by two facts. First the inference mechanism of Seq2VAR is shared by all samples and has integrated the noise by seeing numerous noisy examples. Second, the low expressive decoder cannot integrate noise nor deal with complex mixture of noisy signals. This second point explains why Seq2VAR resists better to noise than NRI that has a powerful decoder that is therefore more sensitive to noise.

We present in Table 3.1 the classification accuracy and causality recovery in the presence of different level of observation noise. Seq2VAR approaches outperform VAR and NRI when noise gets stronger. Binary-Seq2VAR give also better results, as its assumption fits the data better.

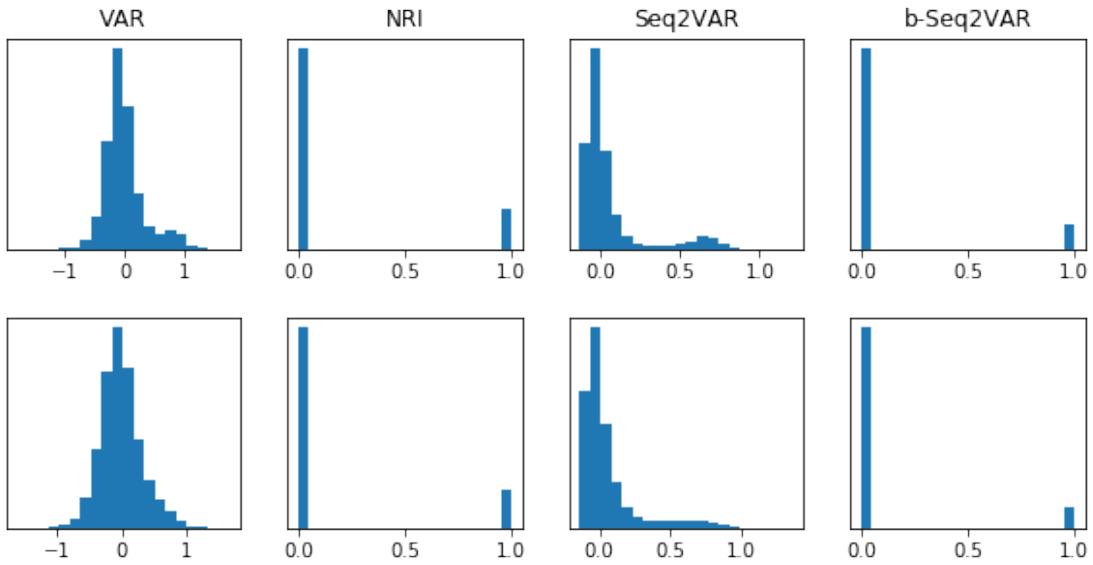


Figure 3.9 – Histogram of the distribution of the entries of inferred transition matrices over a test set, using VAR, NRI, Seq2VAR and binary Seq2VAR, compared to ground truth, depending on the level of observation noise: $\mathcal{N}(0, 0.1)$ (top), $\mathcal{N}(0, 0.5)$ (down).

Dataset	Perm. + $\mathcal{N}(0, 0.1)$		Perm. + $\mathcal{N}(0, 0.3)$		Perm. + $\mathcal{N}(0, 0.5)$	
Tasks	Supervised classification	Causality detection	Supervised classification	Causality detection	Supervised classification	Causality detection
VAR	**100** ± 0.0	72.2 ± 0.2	96.85 ± 3.8	61.8 ± 4.5	96.5 ± 1.6	52.6 ± 3.9
NRI	**100** ± 0.0	95.63 ± 3.1	97.6 ± 3.4	84.3 ± 4.9	97.0 ± 2.1	68.3 ± 3.7
Seq2VAR	**100** ± 0.0	**97.3** ± 0.3	97.0 ± 4.3	92.5 ± 2.1	**97.8** ± 3.9	83.6 ± 2.3
B-Seq2VAR	**100** ± 0.0	97.2 ± 0.1	**100** ± 0.0	**94.6** ± 2.7	97.0 ± 4.3	**90.1** ± 2.9

Table 3.1 – Test classification accuracy (%) and causality discovery (F1-score). The standard deviations correspond to the variation in results between different generated datasets (train and test). B-Seq2VAR stands for binary Seq2VAR.

In Figure 3.10, we see the boxplot representations of the distribution of the L_1 distance between ground truth and inferred causality graph within the test set, for several levels of observation noise. We first see that all methods suffer from noise. When noise is low, both NRI, Seq2VAR and binary-Seq2VAR offers almost perfect graph discover. However, the outliers (red crosses) of NRI spread further than the one of Seq2VAR, which means that NRI fails to find the latent graph not by simply missing some entries but by finding a completely different graph that still fits the decoding requirements (thanks to its powerful decoder).

3.3.1.3 Causality detection in interacting Newtonian system

We now propose to assess the capacity of Seq2VAR to find Granger causality graph hidden in physical data. We use 10-ball-springs system data, consisting of the simultaneous trajectories of 10 identical balls in a 2D space, each ball being connected to others by springs with probability 0.5. The connection network is called interaction graph. The system can be represented as a Granger causality graph [Eichler and Didelez, 2012]

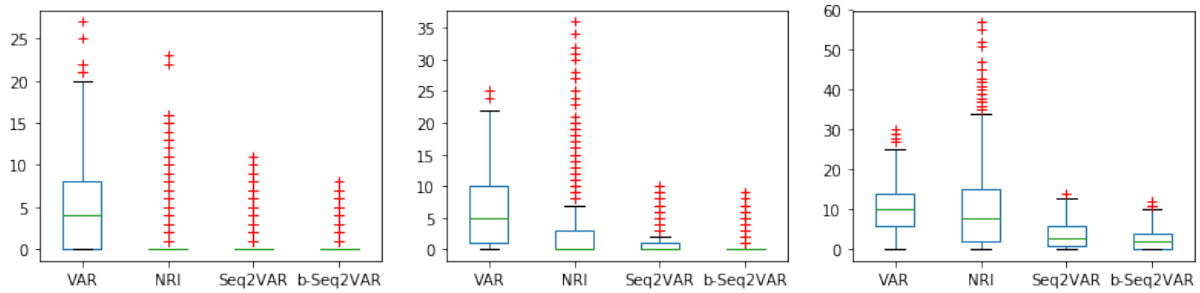


Figure 3.10 – Quartiles of the distribution of the L_1 distances between true and inferred causality graphs with VAR, NRI, Seq2VAR and binary Seq2VAR respectively with observation noise $\mathcal{N}(0, 0.1)$ (left), $\mathcal{N}(0, 0.3)$ (middle) and $\mathcal{N}(0, 0.5)$ (right). b-Seq2VAR stands for binary Seq2VAR.

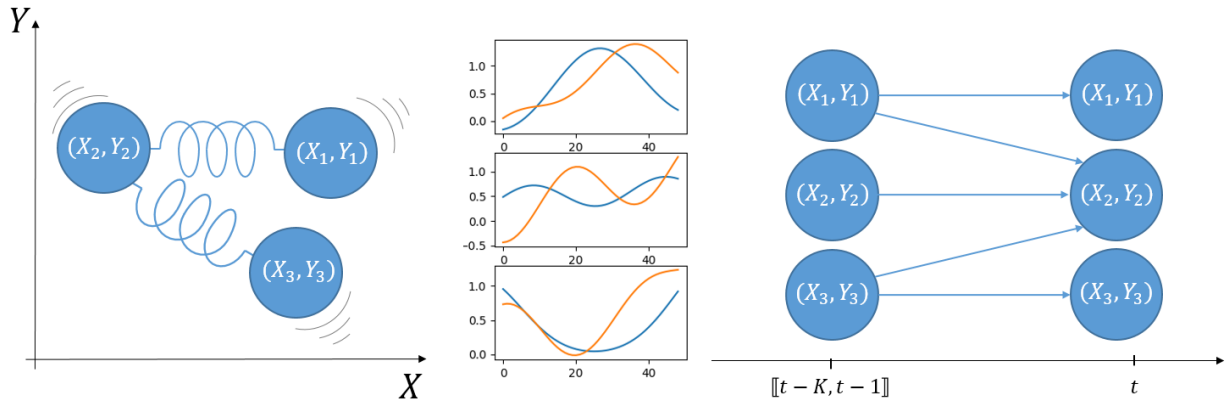


Figure 3.11 – **Left**: example of a 3-balls-springs system. **Middle**: a MTS sample, with 3 components representing the location of the ball in the 2D space. **Right**: the associated Granger causality graph (right). The causality graph represents the first-order dynamical dependencies between balls position at each time t and the positions of their direct neighbors at previous time steps $t-1 \dots t-K$. For all $t = K \dots T$ and $i \in \{1, 2, 3\}$, arrows indicates a dynamical dependency between $(X_t^{(i)}, Y_t^{(i)})$ and $\left\{(X_{t-1}^{(j)}, Y_{t-1}^{(j)}), \dots, (X_{t-K}^{(j)}, Y_{t-K}^{(j)})\right\}_{j \in pa(i)}$, with $pa(i)$ the set of parents of node i in the directed graph.

(see Figure 3.11). Note that this experiment is used in NRI paper [Kipf et al., 2018]. Each sample is 49 timesteps long. More information about the simulation of ball-spring system is given in Appendix C.

For the experiments of this section, we sample 20 different balls-spring binary interaction graphs, 10 for the train set and 10 for the test set. Each of the 20 dynamical systems associated to the 20 graphs is built at random, with balls linked by a spring with probability 0.5. Each graph characterizes a class. We propose two different datasets: one with identical rigidity 1 for all springs (*unweighted* interaction graph) and one with variable rigidity (*weighted* interaction graph). For the later, the rigidity of each spring is uniformly sampled in $[0.75, 1]$. Each binary graph characterizes a class. We use 1000 samples per class. As for permutation MTS (see 3.3.1.2), train and test set have different causality structures, in order to challenge the generalization capacity of Seq2VAR.

We trained respectively a VAR, a Seq2VAR, a symmetric Seq2VAR and a sparse Seq2VAR. We chose $K = 2$ for all the experiments. For the later, we impose a slight regularization on the level of sparsity, i.e.

on the number of null entries in the matrix. In Equation 3.5 we use $\Omega(G_\phi(X)) = \left|\frac{1}{100}\|G_\phi(X)\|_1 - 0.5\right|$ and λ is set at $1e^{-3}$, 0.5 being the sparsity prior. Without this regularization, the sparse Seq2VAR generally converges naturally towards the right proportion of true zeros. The regularization is added for preventing an eventual trivial solution where the mask is a matrix of ones. The only occurrence of this problem happened in the weighted problem.

Table 3.2 gathers the results. We see that Seq2VAR scores better on both quality measures than the usual VAR approach learned on the test set. If NRI gives very good results of causality detection on unweighted springs, its causality discovery performance drops when dealing with weighted springs. On the contrary, Seq2VAR gives good results for both unweighted and weighted rigidity graph.

Dataset	Unweighted springs		Weighted springs	
Tasks	Supervised classification	Causality detection	Supervised classification	Causality detection
VAR	57.0 ± 5.7	55.7 ± 5.3	56.2 ± 6.3	54.0 ± 4.8
NRI	$\mathbf{100} \pm 0.0$	$\mathbf{96.1} \pm 1.8$	$\mathbf{100} \pm 0.0$	$78.5 \pm 6.8*$
Seq2VAR	$\mathbf{100} \pm 0.0$	89.4 ± 2.0	$\mathbf{100} \pm 0.0$	84.5 ± 3.7
Symmetric Seq2VAR	99.9 ± 0.1	91.4 ± 1.3	$\mathbf{100} \pm 0.0$	$\mathbf{90.4} \pm 2.8$
Sparse Seq2VAR	99.7 ± 0.1	88.2 ± 3.7	99.2 ± 0.3	81.4 ± 4.4

Table 3.2 – Test classification accuracy (%) and causality discovery (F1-score). *Hyperparameters different than the one used for unweighted springs case (from the original paper [Kipf et al., 2018]) to obtain better results.

The results also confirms that using an expressive inference network that explicitly model dependencies between variables capitalize on the global information of the dataset: it generalizes well on new data. It is also interesting to notice that variable spring rigidity makes the causality less identifiable in both linear (Seq2VAR) and nonlinear (NRI) decoding process, with respect to our experimental setup. Yet, since the representation W^X from Seq2VAR is continuous coupled with a simple decoder (that is very close to the physic of the system), the results are more robust in Seq2VAR framework.

We find in Figure 3.12 an illustration of the inference capacities of our model.

Remark 31. *For NRI, all other parameters are the one of the original paper for the unweighted springs rigidity for the unweighted springs. For the weighted springs, the parameter prediction_steps is set to 5 instead of 10 and τ is set to 0.1. These parameters gave the best average results. In fact, due to the highly expressive form of its decoder, NRI was able to build good predictor with not the good graph. We played with parameters to get more stable and better results. For the experiments, we also tried to change the skip first parameter that is set to False or True in the original paper [Kipf et al., 2018], depending on the dataset studied. It did not change the results of the experiments.*

3.3.2 Experiments: unsupervised ageing detection on causality graphs

We propose two experiments to illustrate the interest of Seq2Graph in the context of health monitoring (HM) for Safran. Both are based on multivariate time series data generated from mechanical systems. For the first,

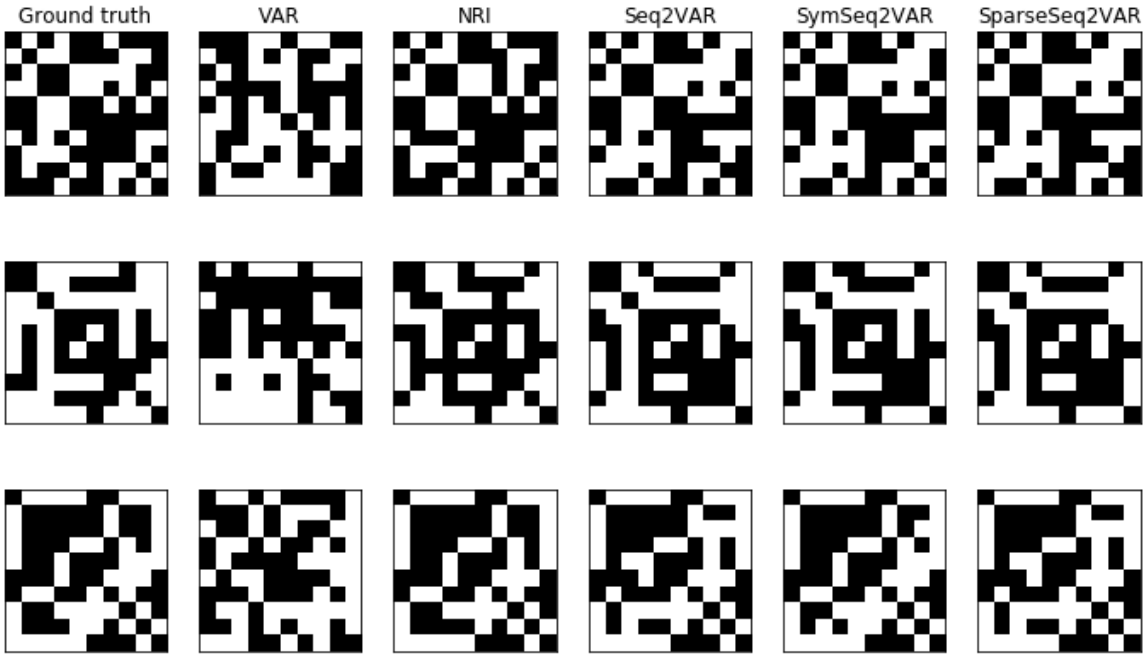


Figure 3.12 – Three examples of inferred transition matrices over a test set, using VAR, NRI, Seq2VAR, Symmetric Seq2VAR, Sparse Seq2VAR and Sparse Symmetric Seq2VAR, compared to ground truth.

we use synthetic data with controlled causal structure to show the interest of the regularization with global causality graph $\mathcal{G}$ with a sparse random coefficient model (3.8). For the second, we use a NASA dataset to assess Seq2Graph on real representation task. In all experiments, we compare Seq2Graph to three time series representation methods: a sequential autoencoder (SAE) [Malhotra et al., 2017], to the unsupervised scalable time series representation (USTR) [Franceschi et al., 2019] and to the sequence-to-VAR (Seq2VAR) presented above.

The hyperparameters are given in Table 3.3.

	λ	γ	η
Experiment 3.3.2.1	10^{-3}	-	10^{-5}
Experiment 3.3.2.2	5×10^{-3}	5×10^{-3}	5×10^{-4}

Table 3.3 – Hyperparameters for our experiments.

3.3.2.1 Ageing interacting Newtonian system

Dataset We simulate samples from a 10-ball-springs system, consisting of the simultaneous trajectories of 10 identical balls of unit mass in a 2-dimensional space, each ball being connected to some others by springs (the rate of connection is 56%). This system has a natural bidirectional causal structure: each ball's trajectory acts as a cause for changes in the trajectory of the neighbor balls, and conversely. Using the previously introduced notations, we have $d = 10$ (10 balls) and $m = 2$ (in a 2-dimensional space, see implementation details). System dynamics follows Newton's law of motion. We assume that the system is

ageing and is regularly restored. All samples share a common graph graphical structure $\mathcal{G}$ whose adjacency is the interaction matrix formed by the springs.

We simulate a synthetic dataset of 15000 samples (trajectories), 5000 for train, 5000 for validation and 5000 for test. Each trajectory is 49 time-steps-long ($T = 49$). For each batch b of 50 samples, a constant ageing factor $\alpha_b \sim \mathcal{U}([0.9, 1])$ is applied to the system: at each sample X whose index is $s \in [\![0, 50]\!]$ (within the batch b of 50 samples), we randomly choose a spring (i, j) and multiply its rigidity by α_b^s, i.e. an exponential ageing coefficient with respect to sample index. Every 50 samples, we *restore* the state of the system and another ageing factor is sampled and applied to the next batch of 50 samples. For some trajectories, $\alpha_b = 1$, i.e. there is no ageing: the initial hidden causality graph has binary adjacency and remains the same along the life of the system. When $\alpha_b < 1$, the initial graph is deteriorating during along the life (observed through 50 samples) of the system, until restoration. An illustration is given in Figure 3.13.

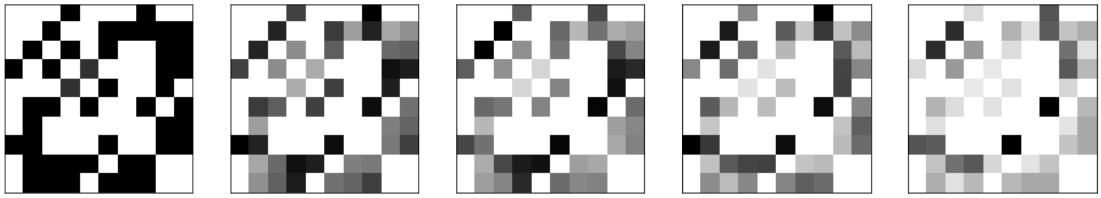


Figure 3.13 – Representation of the springs degradation in the coefficients of rigidity in the weighted adjacency matrix of the interaction graph.

Model For this first experiment, we assume that the model is linear, i.e. that functions g_{θ_j} are identities. The objective is to illustrate the impact of representing data on the same causality graph $\mathcal{G}$. We determine by cross-validation that $K = 2$. The level of sparsity is determined by the quality of the prediction for different levels of sparsity of the causality graph, as shown in Figure 3.14. The prediction is almost invariant until a sparsity of about 56%. We note that we find back the true adjacency.

Metrics and results We assess the quality of the representation inference function $P_\phi^{\mathcal{G}}$. We test if we can represent the ageing of the system with respect to a reference *healthy* sample X^{ref} (first sample of a batch) picked in the validation set. We then build the test ageing curve

$$X \longmapsto \left\| \sum_{k=1}^{K} (P_\phi^{\mathcal{G}}(X^{ref}) - P_\phi^{\mathcal{G}}(X))_k \right\|_2^2 \tag{3.12}$$

for all samples $X \in \mathcal{X}^{test}$. Results are presented in Figure 3.15 and Table 3.4. We note that all curves are min-max re-scaled.

Table 3.4 gives system ageing detection results. The *Ageing score* is the correlation between estimated and real ageing curve.

We see that Seq2Graph outperforms both SAE, USTR and Seq2VAR for unsupervised representation learning, when meaningful information is fully contained in the causality. In particular, SAE and USTR

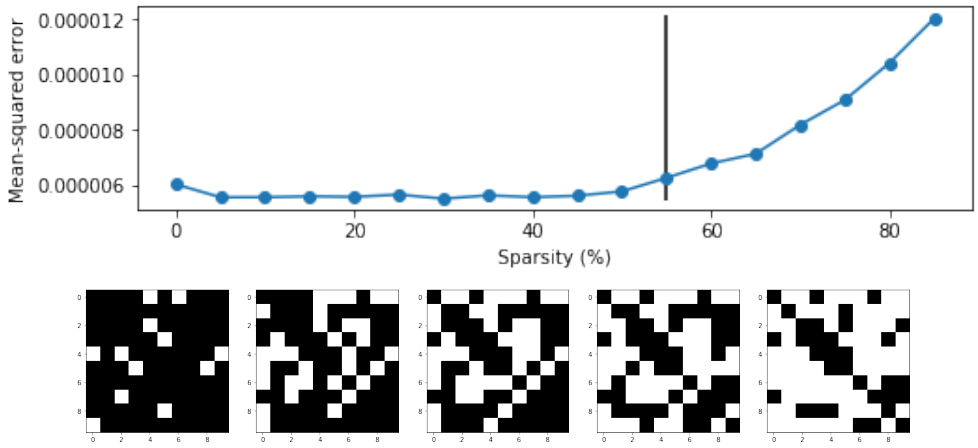


Figure 3.14 – Learning $\bar{W}$ from ball-springs data. **Top**: Prediction MSE. Black line is the true sparsity. **Bottom**: Causal graphs for different sparsity levels. The third figure is the inferred causal graph, which matches the ground truth.

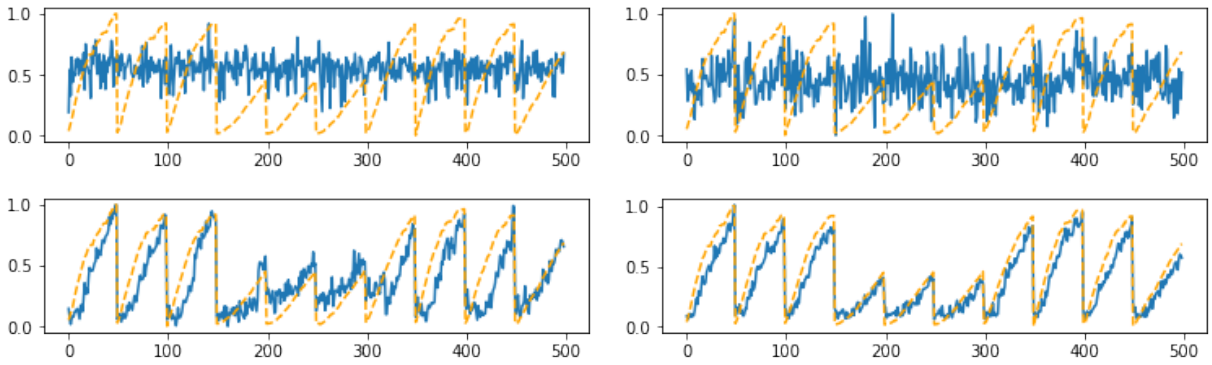


Figure 3.15 – Unsupervised estimation of ageing curves for the 10 first batches of the test set. **Top-left**: USTR [Franceschi et al., 2019], **top-right**: SAE [Malhotra et al., 2017], **bottom-left**: Seq2VAR [Pineau et al., 2019], **bottom-right**: Seq2Graph. Orange curve is the ground truth.

completely miss the consistent ageing information, due to the fact the the inductive bias towards the causality (that fully contains the ageing information) is null. Although consistent, Seq2VAR seems to suffer when the causalities become lower. We can relate it to the observation that the causality detection in Experiment 3.3.1.3 becomes harder when causalities are heterogeneous. In fact, lowering the causality improves the difficulty to capture them, hence prevents to find the trend hidden in causality. Adding a common causal structure $\bar{W}$ in Seq2Graph naturally helps the identification and the consistency of the sample representations.

3.3.2.2 NASA turbofan degradation simulation dataset

Dataset NASA public Commercial Modular Aero-Propulsion System Simulation dataset (CMAPSS) is a tool for simulation of realistic large commercial turbofan engine data [Saxena and Goebel, 2008]. An engine degradation simulation was carried out using CMAPSS, under different conditions and different faults. We use the *FD001* dataset which contains 100 time series recorded at sea level with one fault mode for

Models	MSE	Ageing score*
SAE	2.2×10^{-5}	0.09
USTR	-	0.05
Seq2VAR	2.3×10^{-7}	0.62
Seq2Graph	4.4×10^{-7}	**0.97**

Table 3.4 – Performance of several models plus ours on ageing mass-springs problem. *Higher is better. MSE stands for mean squared error and serves only as a sanity check (for MSE-based methods).

each (degradation of the high-pressure compressor, a fundamental turbofan piece). The time series are the output of the turbine-engine system that takes a fuel flow as input and outputs 21 variables, whose 13 are not constant (we only keep these 13 variables). Time series are 206 time-steps long on average. Each time series is the recording of a turbine engine going to failure. The engine is operating normally at the start of each time series and develops a fault of unknown initial magnitude in its first moments. We only know that the impact of this fault on the system grows in magnitude until system failure.

For the results of the paper, we split the dataset: the first 60 time series are train set, the 15 next are validation set and the last 25 are test set. We extract from these time series sub-trajectories of length 25, with a rolling window with stride 5 to make our dataset. Hence, as for previous experiment, we have several batches of samples. A batch corresponds to the life of the system from start to failure. At the end of each batch, the engine is *restored* and another batch of samples is recorded.

Model Using the previously introduced notations, $d = 13$. All samples share a common (unknown) structure which is the turbine engine mechanics. We assume that this structure can be represented by a sparse causality matrix. For this experiment each link function g_{θ_j} is a MLP with 2 hidden layers of 4 channels. We determine by cross-validation that $K = 2$. The maximal sparsity is determined by the quality of the prediction for different levels of sparsity, as shown in Figure 3.16. The prediction is almost invariant until a sparsity of about 75%.

We solve (3.11) using the previously found causal graph. Contrary to the previous experiment, the system is not *isolated* since the observed variables are the response to an unobserved command (the fuel flow). Yet, the variables can effectively interact with each other and statistical causality still makes sense as a representation assumption.

Metrics We assess the quality of the learned $P_\phi^{\mathcal{G}}$ by testing if we can extract *ageing information* from the representations, as for the previous experiment. Yet, we remind that we do not know the importance of the initial fault. What we know is that the 100 engines go to failure and the degradation of the state is monotonic until restoration. We propose a two-step process to predict the imminence of a failure.

First, we build an ageing indicator assuming that is a relative position compare to a healthy sample. We pick a healthy sample X^{ref} (first sample of a batch) in the validation set and build the ageing curve $\| \sum_{k=1}^{K} (P_\phi^{\mathcal{G}}(X^{ref}) - P_\phi^{\mathcal{G}}(X))_k \|_2^2$ for all $X \in \mathcal{X}^{valid}$. We compute a *failure threshold* τ^{valid} that must indicate when an engine goes to failure. We set τ^{valid} to the maximal threshold that ensures turbine engine failure

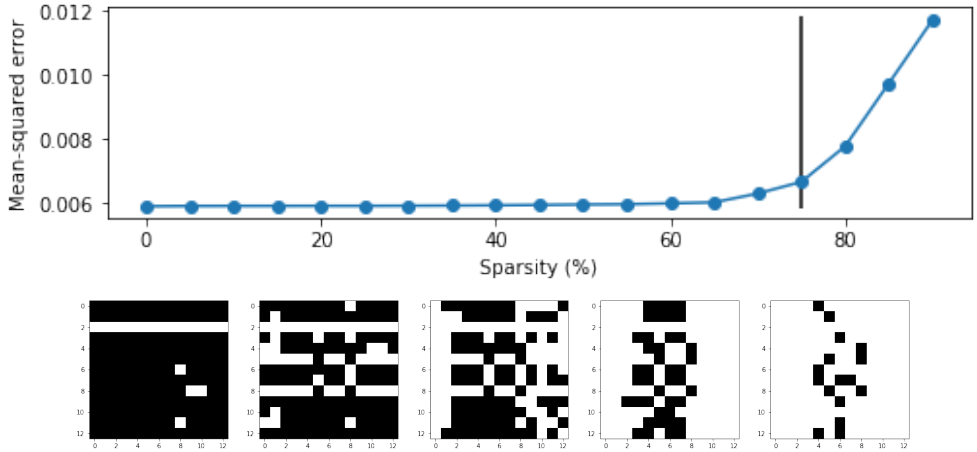


Figure 3.16 – Learning $\bar{W}$ from CMAPSS data. **Top**: MSE of the prediction. Black line is the sparsity of the inferred causal graph. **Bottom**: Causal graphs for different sparsity levels. Fourth image is the inferred causal graph.

detection for all validation batches, that is:

$$\tau^{valid} = \min_{X \in \mathcal{X}^{val,fail}} \left\| \sum_{k=1}^{K} (P_\theta^{\mathcal{G}}(X^{ref}) - P_\theta^{\mathcal{G}}(X))_k \right\|_2^2 \tag{3.13}$$

where $\mathcal{X}^{val,fail}$ is the set of validation samples preceding the engine failure. We note that τ^{valid} has no safety margin, i.e. any threshold above τ^{valid} misses at least one engine failure in the validation set (under monotony assumption underlying the ageing of a mechanical system). It is possible to add a margin by setting $\mathcal{X}^{val,fail}$ as the set of validation samples preceding by $k \in \mathbb{N}$ samples index (in the batch) the engine failure.

Second, we build the test ageing curve (3.12) for all $X \in \mathcal{X}^{test}$. We apply the detection test using τ^{valid} (represented by the horizontal dotted line in Figure 3.17.

Results As a first assessment, we see in Figure 3.17 that the estimated ageing curves built from SAE, Seq2VAR and Seq2Graph are almost monotonic inside each batch (between two vertical orange lines). We recall that monotony is the only ground truth information we have on the ageing of the system. The fact that SAE, Seq2VAR and Seq2Graph unveils monotonic signal means the ageing information is present both in patterns and values (SAE) and in causality (Seq2VAR and Seq2Graph). We do not find consistent representations with USTR. We also observe that the batches do not begin at the same value (dashed horizontal lines in Figure 3.17), whatever the method. It is partly imputed to the fact that the mechanical faults are located at the beginning of each batch and that they vary in intensity. Hence, the inferred first samples of each batch do not have to be equal.

Remark 32. *Not finding a temporally consistent representation of samples does not mean that the USTR is generally useless. At contrary, USTR has proven impressive performance on downstream classification tasks on standard datasets. Yet, here USTR does not capture by itself the physical states of the turbine systems that is hidden within the samples. Additional inductive bias is required.*

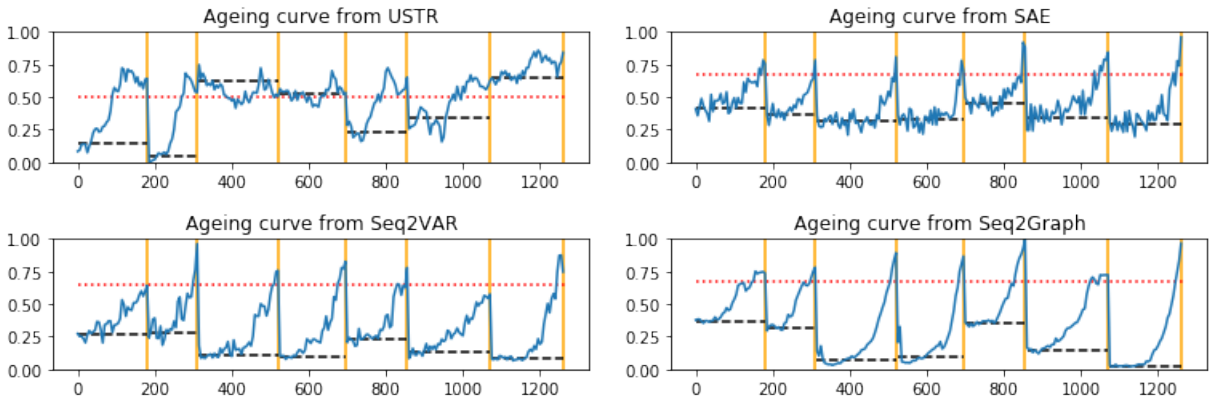


Figure 3.17 – Unsupervised estimation of ageing curve with different models on the 7 first test batches. **Top-left**: USTR [Franceschi et al., 2019], **top-right**: SAE [Malhotra et al., 2017], **bottom-left**: Seq2VAR [Pineau et al., 2019], **bottom-right**: Seq2Graph. Orange picks are engine failures and repair. Long red dotted horizontal line is the threshold τ^{valid}. Black dashed horizontal lines are the estimated initial states of each engine, computed as the mean value of the curve on the 10 first samples of each batch. They are meant to indicate that the first states are more uncertain in term of causality parameters (Seq2VAR and Seq2Graph) than in term of values (SAE).

We observe that the batch's ageing curves do not begin at the same value (dashed horizontal lines in Figure 3.17), whatever the method. It is partly imputed to the fact that the mechanical faults are located at the beginning of each batch and that they vary in intensity. Hence, the inferred first samples of each batch do not have to be equal.

We now compare the ability of the different MTS representations to detect failures. In Figure 3.18, built with extracted signal show in Figure 3.17, gives the proportion of alarm at different time steps before actual failure happens. First, we note that all models detect almost 100% of failures before it happens. Second, we want detection of the coming failures to be reasonably early to avoid false alarms. If curves cross threshold too early, the MTS representation is useless. Figure 3.18 shows that Seq2Graph is the most consistent in early detection with no alarms far from failure, due to the consistency of the extracted monotonic signal. On the contrary, SAE always finds early failures. We note that Seq2Graph also has lower standard deviation, illustrating the interest of the regularizing effect over Seq2VAR.

In Figure 3.19, we give the average precision scores (APS), that summarizes precision-recall curves as the weighted mean of precision achieved at each failure detection threshold. We see that Seq2VAR and Seq2Graph are globally better at finding exact failure in advance.

We finally observe on all figures that the results of Seq2Graph effectively completes Seq2VAR (also causality-based) with a regularizing effect due to the sparse prior on the causality graph representation space and the link functions g_{θ_j} of the GVAR (3.9) that contains global causal information about the engine.

We have built a representation of the samples that both describe the system dynamics and are consistent with the unknown ageing process since the distance from reference is almost everywhere monotonic before failure, without supervision.

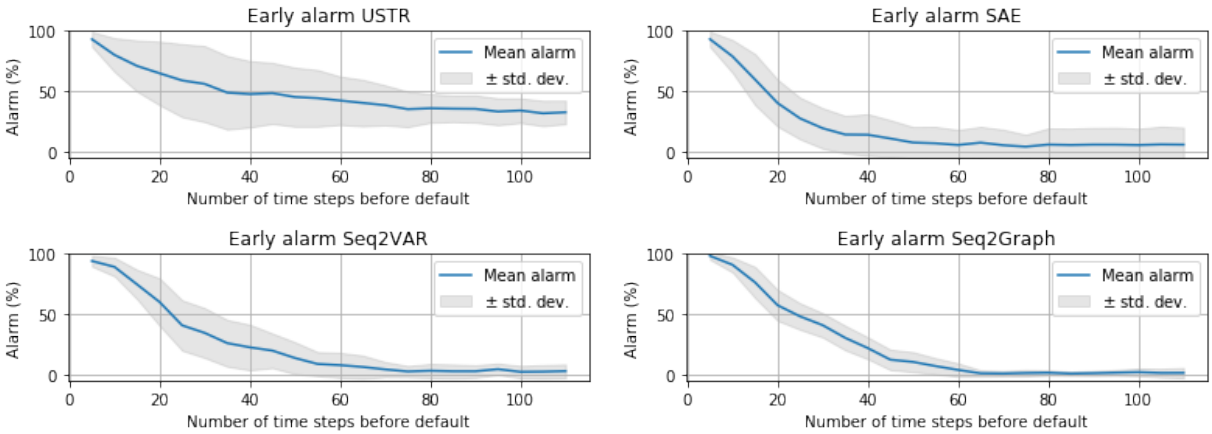


Figure 3.18 – Early alarm on CMAPSS data using MTS representation models USTR, SAE, Seq2VAR (see related work for details) and Seq2Graph. Means and standard deviations are built using all batch's first samples as X^{ref} and several encoders trained with different seeds.

3.4 Discussions

This section proposes discussions around our Seq2VAR approach for MTS representation with VAR matrices and causality graph.

3.4.1 Remarks about the relation between Seq2VAR and NRI

As mentioned above, NRI is a graph relational inference model, set as a VAE with binary latent space. The inferred unweighted graphs are used as a variable selection procedure for a prediction model (nonlinear neural decoder). This configuration specifically applies to physical interacting systems MTS (like ball-springs system). Three major differences appear between Seq2VAR and NRI.

First the form of the decoder. NRI decoding scheme is a nonlinear network that takes as input an embedding of the pairwise variable interactions at each time step and output the incremental change to predict next time step from current time step. On the contrary, we propose to leverage the simplicity of a linear autoregressive decoder that is potentially less expressive but do not require additional parameters.

Second, as a consequence of the form of the decoder, the latent representation is meaningful. We infer both binary and real latent representation to respectively represent existence and intensity of the causal interactions in the data. The real part is implicit in NRI. Experiment 3.3.1.3 shows that NRI does not disentangle existence and intensity of the interactions: when springs are not equally rigid, NRI is perturbed and finds a latent graph that does not correspond to physical reality. Our Seq2VAR, thanks to its continuous part, explicitly disentangles latent causal structure from other information and finds a better causal graph.

Third difference, which is also as a consequence of the decoder: the minimal input information requirement. In fact, the notion of time lag is absent from NRI and lagged information needs to be furnished as input. For example, with the ball-springs systems data, Seq2VAR only needs measures of the location of each ball at each time step while NRI requires both location and velocity. Beyond the minimal input information requirement, the absence of lag in NRI modeling imposes that causality graph remains the same for all lags, like in physical structures.

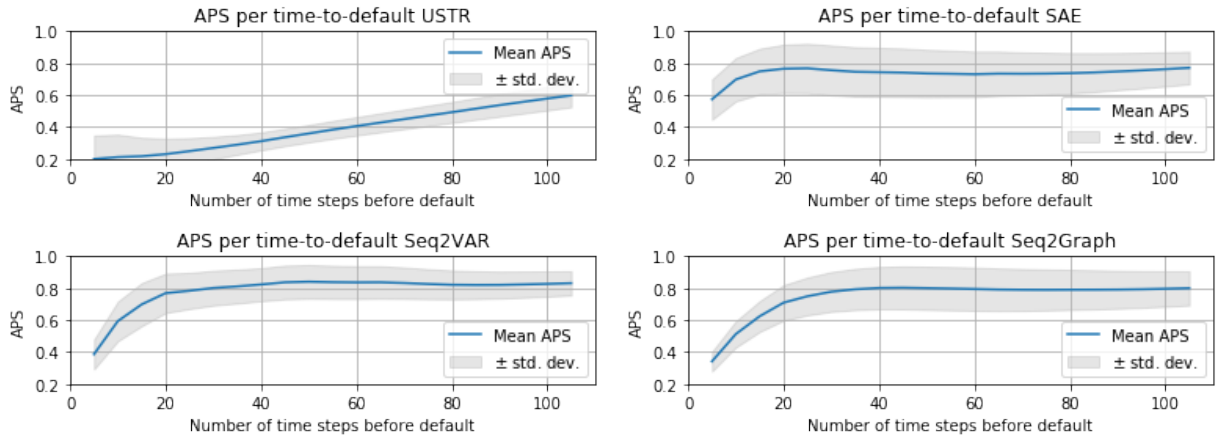


Figure 3.19 – Average Precision Score (APS) of the failure detection using MTS representation models USTR, SAE, Seq2VAR (see related work for details) and Seq2Graph. Means and standard deviations are built using all batch's first samples as X^{ref} and several encoders trained with different seeds.

Experiments	Permutations (batch size=128)		Ball-springs (batch size=64)	
	Number of parameters	CPU time per epoch (s)	Number of parameters	CPU time per epoch (s)
NRI [Kipf et al., 2018]	65031	5.1	72966	38.8
Seq2VAR	47811	1.2	52550	4.7
Symmetric Seq2VAR	-	-	52550	4.7
Binary Seq2VAR	47811	1.4	-	-
Sparse Seq2VAR	-	-	61071	5.7

Table 3.5 – Memory and computing time. Absence of results means that model is not used.

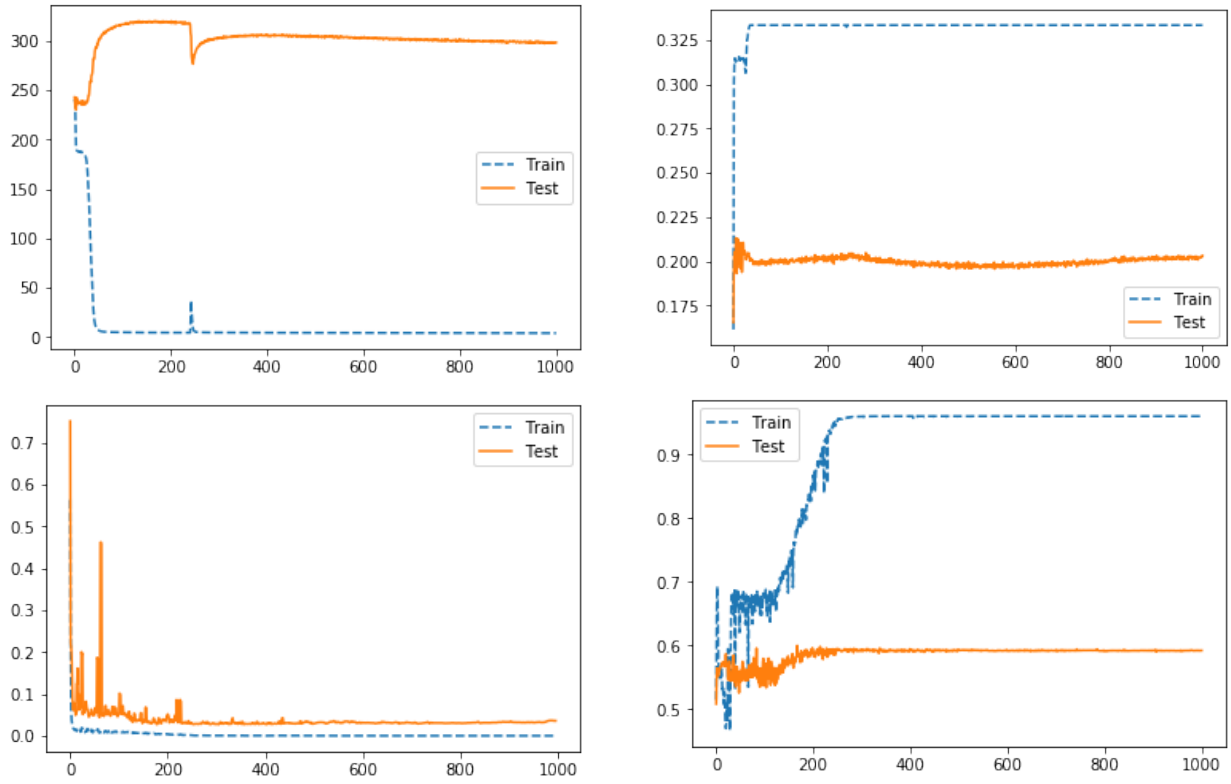
Finally, these differences materialize with the results of the experiments. Note that Seq2VAR, with the simplification compared to NRI, also procure a significant advantage in term of complexity, assessed in Table 3.5.

3.4.2 RelNN vs. RNN as Seq2VAR encoder

The main assumption when modeling time series data is the autoregressive structure of the observed signal. A generic and expressive autoregressive model is the recurrent neural network (RNN). Hence, in practice we could use RNN as representation inference network $F_\phi(X)$, as usual for time series, and train it to output the tensor W^X from samples X.

Our different attempts with RNN encoder were not able to generalize over the notion of causal interactions, no matter the regularization technique used to avoid overfitting (reduction of network memory capacities, increasing of depth [Sak et al., 2014], dropout [Zaremba et al., 2014], batch-norm [Merity et al., 2017], L_1-norm and L_2-norm penalty on weights). We show in Figure 3.20 an example of train and test performances evolution during training, respectively for permutation and ball-springs experiments with GRU encoder instead of RelNN encoder. We can see that a Seq2VAR using RNN as encoder overfits.

We explain this by the fact that RelNN explicitly takes as input all the pairs of univariate time series constituting the MTS and outputs a tensor whose entries represent an embedding of each pair. Therefore, this inductive bias incites the network to model one-to-one interactions. We remind that the causality graphs of the test set are different from the causality graphs of the train set. Hence we ask our inference network to generalize over a discontinuous manifold. RelNN inference networks and its explicit one-to-one interactions modeling learns well to generalize. Conversely, RNNs implicitly learn the notion of interactions during training, with a vector output that needs to be folded into the right shape. The generalization is more difficult.



(a) MSE prediction

(b) Causality F1-score

Figure 3.20 – Train and test performances during training of Seq2VAR with GRU encoder instead of RelNN encoder. Column (a): MSE of the 1-step VAR prediction. Column (b): F1-score of the inferred causality graph. Top row: permutation dataset. Bottom row: unweighted 10-ball-springs dataset. We note that using RNN for permutation dataset fails to find the good causalities even for training dataset. It finds another appropriate (more complex) matrix that effectively minimized the MSE.

3.4.3 Explicit temporal consistency

In Seq2Graph experiments, we did not take into account explicitly the temporal relations between samples (i.e. adding an horizontal an arrow from W^X to W^Y in Figure 3.5), whereas it was sough in the experiments. The point was twofold. First, we wanted to show that the causality inductive bias is enough in certain situations. Second, we do not have always the temporal index of the samples in a dataset. Now, if we have

a dataset $\mathcal{X}$ such that the samples are ordered, then we can add an temporal consistency inductive bias (see Section 2.1). We note i^X the index of the sample X in $\mathcal{X}$. Then we can add:

$$\gamma \mathbb{E}_{(X,Y)\sim\mathcal{X}^2}\left[\mathbb{1}_{\{|i^X - i^Y|=1\}} d\left(P_\phi^{\mathcal{G}}(X), P_\phi^{\mathcal{G}}(Y)\right)\right]$$

with $\gamma > 0$ and $d(.,.)$ any divergence between tensors. The norm can be any norm. Such penalization effectively smooths the sequence of inferred representation for a sufficiently high γ. The condition $\mathbb{1}_{\{|i^X - i^Y|=1\}}$ is a constraint of the proximity of samples. It can be extended to consistency of longer term $n \in \mathbb{N}^*$, for example $\mathbb{1}_{\{|i^X - i^Y|\leq n\}}$ (hard n-long consistency) or $|i^X - i^Y|^{-n}$ (progressively vanishing consistency).

In practice, choosing γ can be difficult. In particular, for a dataset like the spring-balls, for a not too high γ the representation became constant, i.e. one stationary model for the whole dataset, since this local minimum was a stable solution from which it was difficult to get out (the mean-squared error of the prediction was 5.10^{-3} with stationary assumption 5.10^{-4} with sample-wise causality adjustment). It is explained by the fact that the nonstationarity induces by trends in causality is of low magnitude.

3.4.4 More general extension of the framework

We limited the representation learning framework to represent data with Granger causality. A perspective for this chapter would be to extend the inductive bias to larger surrogate statistical model. Based on recent advances on neural architecture search [Elsken et al., 2019] and architecture embedding [Cao et al., 2019], we can imagine directly the following method. We fit a neural network g_θ on the dataset $\mathcal{X}$ to explain the dynamics of observed data, i.e. $\min_\theta \mathbb{E}_{X\in\mathcal{X}}\|X_t - g_\theta(X_{<t})\|$. Hence, we find a generative/invertible embedding function N of parameters θ and train an embedding function F_ϕ such that $\phi, N = \arg\min \mathbb{E}_{X\in\mathcal{X}}\|X_t - g_{N^{-1}(N(\theta)+F_\phi(X))}(X_{<t})\|$. We note that the embedding function F_ϕ is not required to be related to Granger causality anymore.

3.4.5 Seq2VAR and Seq2Graph are meta-learning methods

Meta learning consists in learning at the same time a parametric predictive model (here the one-step post-linear prediction model 3.9) and a parameter updating model F that learns how to update parameters W with delta (W^X) for each new situation X, with few data. In Seq2VAR and Seq2Graph, a general model is learned and a RelNN F_ϕ learns to adjust prediction parameters $(g_\theta, \bar{W})$. A particularity of meta learning is the high generalization capacity of the parameter adaptation model, since it should find the right parameters from new situations with few data. It is exactly what we have shown for Seq2VAR , with test set containing unseen Granger causality structures. A good overview of meta-learning techniques is given in [Weng, 2018].

3.5 Conclusion

In this chapter, we proposed a new representation learning based on the assumption that relevant information can be found in hidden causality underlying multivariate time series data. Our approach is an encoder-decoder that learns to infer the parameter of a VAR model explaining the dynamics of the data. We

proposed several features to enrich the basic Seq2VAR model and demonstrated its concrete interest with an application to mechanical system state monitoring.

Chapter 4

Contrastive learning of hidden temporal trends

Abstract In previous chapter, we have shown that using a model-based inductive bias for representation learning enables to find ageing hidden in mechanical system's data. In this chapter, we use a natural inductive bias to focus on ageing detection without assumption on the model describing the data. We propose a new simple tool to extract trends from time series data, based on *temporal contrastive learning*, i.e. a type of learning procedure that uses temporal information to represent data. We prove that our approach identifies trend hidden in data while avoiding standard restrictive assumptions used for trend identification. We show in several experiments that our model achieves good results in a panel of trend identification tasks. This chapter covers two contributions:

- Pineau, E., Razakarivony, S., and Bonald, T. (2020). French patent (information, included title, are confidential until the end of the publication delay in January 2022).
- Pineau, E., Razakarivony, S., and Bonald, T. (2020). Time series source separation with slow flows. *Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models, at International Conference of Machine Learning (INNF+, ICML 2020).*

4.1 Presentation of the problem

We have seen in the introduction section that there are several possible inductive biases that can be used in a representation framework, depending on the downstream task of interest. For time series data, a standard inductive bias assumes the existence of *structural latent components* that are supposed to be meaningful (see 2.4.2). We remind that the standard decomposition in the literature consists in the combination of the trend τ, the cycle c, the seasonality s and the irregularity ϵ. The structural components τ, c, s, ϵ are assumed to be independent, for *identifiability* reasons (see Section 2.2.5). Signal extraction is therefore a type of time series independent component analysis (ICA, Section 2.2.4) where independent features are interpretable. This decomposition always exists without loss of generality, since residual term ϵ^X can manage any latent variable that is not a trend, a cycle or a season.

In particular, in health monitoring (HM) context, trend signal is a good candidate signature of *ageing*. In this chapter, we focus on the detection of the trend component.

To be more concrete, we propose to see the trend identification through an example. Let's consider a mechanical system $\mathcal{S}$. We assume that we observe d simultaneous temporal signals from $\mathcal{S}$ recorded with sensors that form a d-dimensional multivariate time series (MTS), at different moments t^X of the life of the system $\mathcal{S}$. We note $\mathcal{X} \subseteq \mathbb{R}^{d \times T}$ the set of recorded samples. For each X sampled from $\mathcal{S}$, we know its sampling dates t^X. Sample dates are for example counted in *hours of usage*. We assume that the $\mathcal{S}$ ages, i.e. its state degrades. We denote τ^X the true age (unknown) of the system at sampling time t^X. We assume that τ^X is a definition of the system's *state*. We do not have access to ground truth ages τ^X, since ageing process is not linear of the time and depends on the hidden state of the system, that itself depends on unknown factors like conception, usage or exogenous factors. We only know the sampling times t^X. In general, the state τ^X is given by experts that check the system. The objective of our paper is to extract relevant information about the state τ^X of system $\mathcal{S}$ from observations $\mathcal{X}$, without expert supervision. We use a particular contrastive learning (CL), introduced in Section 2.1.4, as a way to build expressive posterior distribution for latent variable models. We call our approach the *contrastive trend extraction* (CTE).

4.2 Contrastive trend extraction problem setup

Let $(X, Y) \in \mathcal{X}^2$ be two observed time series sampled from $\mathcal{S}$, with sampling time t^X and t^Y. At the time of the sampling, the hidden age of the system was respectively τ^X and τ^Y. The objective is to build an estimator of the age τ^X for each sample X, from data. The only assumption we make for ageing detection is the following:

Hypothesis 1. *The degradation of the system $\mathcal{S}$ is monotone with respect to time between two maintenances, i.e. it does not cure by itself. In particular, for all couples of samples $(X, Y) \in \mathcal{X}^2$, we have $t^Y \leq t^Y \iff \tau^X \leq \tau^Y$.*

We note that $\mathcal{X}$ is a set of samples between two maintenance dates, where each $X \in \mathcal{X}$ is associated to a time index t^X. Hence we assume monotonicity of the hidden states as in Hypothesis 1. We propose a framework to estimate the ages from sampling times.

Let $F_\phi : \mathcal{X} \to \mathbb{R}^{d_e}$ be a neural network (NN) with parameters ϕ that embeds the MTS samples in a d_e-dimensional vector space. Let $g_\beta(.,.) : \mathbb{R}^{d_e} \times \mathbb{R}^{d_e} \to [0, 1]$ be a parametric logistic regressor with parameters β, such that:

$$g_\beta\left(F_\phi(X), F_\phi(Y)\right) = \sigma\left(\beta^\top (F_\phi(X) - F_\phi(Y))\right) \tag{4.1}$$

with $\sigma(x) = (1 + e^{-x})^{-1}$ the sigmoid function. In notations, we separate the embedding from the classifier since in this thesis, we split the embedding from the downstream task. Yet, both are trained end-to-end.

We note $C_{XY} = \mathbb{1}_{\{\tau^X < \tau^Y\}}$ the variable that describes the trend direction of (X, Y). We want to learn the posterior distribution $p(C_{XY}|X, Y)$ with F_ϕ and β, i.e. finding:

$$p(C_{XY} = 1|X, Y) = g_\beta(F_\phi(X), F_\phi(Y)) \tag{4.2}$$

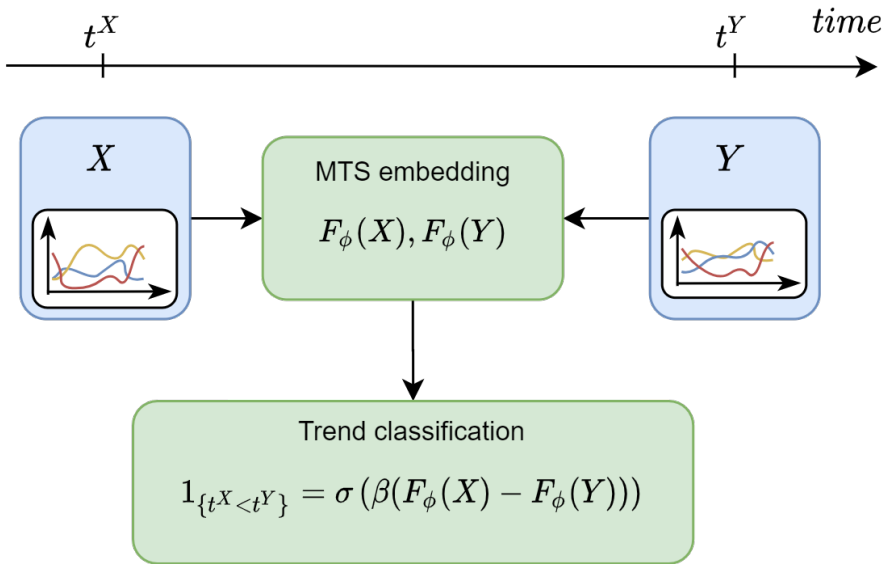


Figure 4.1 – Illustration of a trend estimation between samples.

Like for common binary classification problem, training is done by minimizing the binary cross entropy (BCE) between C_{XY} and the $g_\beta(F_\phi(X), F_\phi(Y))$, for all pairs of samples in the dataset, i.e. by minimizing:

$$R(\beta, \phi; \mathcal{X}) = -\mathbb{E}_{(X,Y)\in\mathcal{X}^2}[C_{XY} \log g_\beta(F_\phi(X), F_\phi(Y))] \tag{4.3}$$

Our CTE method is illustrated in Figure 4.1.
For the rest of the paper, we note $F_{\beta,\phi}(X,Y) := \beta^\top(F_\phi(X) - F_\phi(Y))$.

4.3 Identifiability study

In this section, we study the identifiability (see Definition 3) of the system's trend, and in particular the age τ^X of each sample X, using our method.

We assume in this section that F_ϕ is an universal approximation function (property of the NNs, see Section 2.1.5) and that the amount of data is high enough (theoretically tends to infinity) such that the true classifier can be estimated (equality (4.2)). Under these assumptions, we can analyze the identifiability of the system's age for each sample X. To do so, we need the following definition and results.

Definition 6. ***(Minimal sufficiency)*** *A sufficient statistic T is minimal sufficient if for any sufficient statistic U there exists a function h such that $T = h(U)$. If U is also minimal, then h is one-to-one.*

Proposition 1. *$F_{\beta,\phi}(X,Y)$ is a minimal sufficient statistic for C_{XY}.*

Proof. First we remind that logistic regression learns likelihood ratios, i.e. $F_{\beta,\phi}$ is a log-likelihood difference. In fact, using Bayes rule, we get

$$p(C_{XY} = 1|X,Y) = \frac{p(X,Y|C_{XY}=1)p(C_{XY}=1)}{p(X,Y)}.$$

Moreover, using properties of sigmoid function σ and equality (4.2) we have

$$p(C_{XY}=0|X,Y) = e^{F_{\beta,\phi}(X,Y)}p(C_{XY}=1|X,Y)$$

Finally we obtain,

$$e^{F_{\beta,\phi}(X,Y)} = \frac{p(C_{XY}=1|X,Y)}{p(C_{XY}=0|X,Y)} = \frac{p(X,Y|C_{XY}=1)p(C_{XY}=1)}{p(X,Y|C_{XY}=0)p(C_{XY}=0)}.$$

We note that the prior $p(C_{XY}=1) = p(C_{XY}=0)$ since we randomly choose X and Y simultaneously in $\mathcal{X}$. Hence,

$$e^{F_{\beta,\phi}(X,Y)} = \frac{p(X,Y|C_{XY}=1)}{p(X,Y|C_{XY}=0)}. \tag{4.4}$$

We have a likelihood ratio. Finally we refer to Theorem 2 in [Goh, 2001] stating that the density ratio is a minimal sufficient statistics and Definition 6 to conclude that $F_{\beta,\phi}(X,Y)$ is a minimal sufficient statistic for C_{XY}.

□

Remark 33. *(4.4) is a likelihood ratio. Learning such quantity without explicitly knowing the likelihood is standard. It is called likelihood-free inference [Thomas et al., 2016]. Related to our approach, in [Gutmann et al., 2018] is explained how classification can be used to do likelihood-free inference. More generally, it is the principle of CL. Its main advantage compared to maximum likelihood approaches lives in the fact that contrasting between two models enables to cancel out some computationally untractable terms, like the log-determinant of the Jacobian of the embedding function, for example. Many ML approaches use a version of this trick, like generative adversarial networks (GANs) [Goodfellow et al., 2014] or restricted Botlzman machines (RBMs) [Hinton and Salakhutdinov, 2006]. More details are given in Appendix B.5.*

Lemma 1. *$\exists h$ monotone such that $F_{\beta,\phi}(X,Y) = h(\tau^Y - \tau^X)$.*

Proof. $\tau^Y - \tau^X$ is a minimal sufficient statistic for C_{XY} since $C_{XY} = \mathbb{1}_{t^Y - t^X > 0} = \mathbb{1}_{\tau^Y - \tau^X > 0}$. Hence, since $F_{\beta,\phi}(X,Y)$ is minimal (Proposition 1), Definition 6 says that there exists a one-to-one function h defined in $\mathbb{R}$ such that $F_{\beta,\phi}(X,Y) = h(\tau^Y - \tau^X)$. □

Remark 34. *The monotonicity comes from the one-to-one properties. More intuitively, for samples X, Y, Z of respective age τ^X, τ^Y, τ^Z, we have by assumption that $\tau^Y - \tau^X \leq \tau^Z - \tau^X \Rightarrow F_{\beta,\phi}(X,Y) \leq F_{\beta,\phi}(X,Z)$. Hence h is necessary monotone.*

Theorem 1. *We can identify the true age τ^X, up to a monotone transformation h, in the limit of infinite data.*

Proof. We consider a reference sample X^{ref}, sampled from the newest possible system. Hence, any future sample is sampled from a degraded system. Without loss of generality, we shift the time index and the trend factor such that $t^{X^{ref}} = 0$ and $\tau^{X^{ref}} = 0$. Hence, under the assumption that (4.2) is achievable (infinite data

and universal approximation function F_ϕ), from Lemma 1, we have (β, ϕ) such that $F_{\beta,\phi}(X^{ref}, X) = h(\tau^X)$. Hence, if we denote $C = \beta^\top F_\phi(X^{ref})$, then

$$\beta^\top F_\phi(X) = h(\tau^X) + C$$

□

Prior assumption for h The problem is now to identify the function h. We can assume that h is related to a prior assumption on the distribution of the couple of ages $p(\tau^X, \tau^Y | C_{XY})$. For example:

- Without assumption on the behavior of the relative ageing, we a priori choose constantly ageing system, i.e. $(2C_{XY} - 1)(\tau^Y - \tau^X) \sim \mathcal{U}(0, M)$, where M is a maximum expected life of the ageing system. Then we may choose $h\left(\tau^Y - \tau^X\right) \propto \tau^Y - \tau^X$, hence $F_\phi(X, Y) \propto \tau^Y - \tau^X$
- If we assume a slow ageing for the first part of the system's life, and then continuous acceleration of the ageing after a certain relative time, then a priori $(2C_{XY} - 1)(\tau^Y - \tau^X) \sim Exp(\lambda)$, with $\lambda > 0$ for $p(X, Y | C_{XY})$. The higher the λ, the longer the time with slow ageing, i.e. the higher the density of low values of $(2C_{XY} - 1)(\tau^Y - \tau^X)$. Then we can use for example $h(\tau^Y - \tau^X) \propto \sinh\left(\lambda(\tau^Y - \tau^X)\right)$
- The previous point does not take into account the usage time t^X and t^Y. But the relative age usually depends on the usage, i.e. exponential parameter depends on time $\lambda(t)$. For accelerated ageing, $\lambda(t)$ is time-decreasing. We may choose h depending on temporal information (t^X, t^Y): $h(\tau^Y - \tau^X | t^X, t^Y) \propto \sinh\left(\lambda\left(\min(t^X, t^Y)\right)(\tau^Y - \tau^X)\right)$
- If, at contrary, we assume a fast ageing for the first part of the system's life (running-in period), and then a stabilization of the state of the system, then we can use for example $h(\tau^Y - \tau^X) \propto \tanh\left(\lambda(\tau^Y - \tau^X)\right)$

This list is not exhaustive but shows in which extent a simple assumption about the evolution of the system's state can brings *identifiability up-to monotone transform* towards *identifiability*.

4.4 Relation with other models

In this section, we relate CTE to two large families of hidden variable estimations: the *contrastive learning methods* and the *survival analysis methods*.

4.4.1 Relation with other time contrastive learning methods

We have introduced contrastive learning (CL) in Section 2.1.4. We mentioned that CL is a representation learning framework that can be used as a way to improve the expressiveness of posterior distribution of latent data representation. We also introduced an adaptation of CL for time-indexed data called *time-contrastive learning* (TCL) in Section 2.4.3. In this section, we relate our model to existing advances in time series representation learning with TCL.

First, we mentioned that time series decomposition in structural components is related to blind source separation (BSS) and independent component analysis (ICA, see Sections 2.2.4). Since structural components are assumed to be independent, each independent component is related to at most one structural component. Hence, identifying the independent components, and in particular the trend, means identifying the structural components, up-to variable-wise transformation. It can be done with CL, as proposed in the recent works of Hyvarinen (and derived works), presented in Section 2.4.4.

Second, other contrastive learning methods are related to our work. In [Misra et al., 2016], they embed frames of a video with a convolutional neural network (CNN) F_ϕ. They train F_ϕ by learning to verify if several frames are in the correct temporal order (binary classification supervised by correct/incorrect order label). In [Fernando et al., 2015], they learn to rank video frames to capture temporal consistent representation of object in the video.

It appears that our approach mixes several aspect of time contrastive learning, applied to a particular component identification of time series. Further information about general contrastive learning is given in Appendix B.5.

4.4.2 Relation with survival analysis

A naturally related field of statistics is the *survival analysis* (SA), also called *time-to-event analysis*. The objective of SA it to estimate the lifespan of a system under study, from data. For example, the time-to-death of a patient, the time-to-failure of a mechanical system. Using previously used notations, we have X an observed sample (e.g. a patient) at time t^X, T the failure time (e.g. death), τ^X its age at sampling time t^X, $T - \tau^X$ its *remaining useful life* (RUL). Generally, the lifespan is observed for a subsample of the dataset. If not observed (right-censored), only the time from first to last observation is known. We note δ_X the censor indicator, such that $\delta_X = 1$ is we know the true lifespan, $\delta_X = 0$ otherwise. The objective is to model the conditional *survival function* $S(\tau^x|x) = P(T > \tau^x|X = x)$, hence the probability that the current age is lower than death-time. A standard way to estimate S is to use a particular model for the age, using a *hazard function* h defined as $h(\tau|x) := -\partial_\tau \log(S(\tau|x))$. Several features from survival analysis are used in our study. First, lifespan is relative to a (generally unknown) reference state: we do not have absolute notion of the state τ^X. Second, lifespan is generally not linear with time index. Third, the most used performance metric for survival analysis is similar to our loss and is called the *concordance index* (CI) [Harrell Jr et al., 1996]. The CI measures the fraction of pairs of samples (X, Y) that can be ordered in term of estimated lifespan (from eligible pairs, i.e. pairs for which we have the order information). The CI is then the mean accuracy of our CTE model. Directly learning C_{XY} from data is equivalent to training the model to maximize CI and is the learning procedure of CTE. We can check the performance of CTE on a dataset with a CI, computed as follows:

$$CI(\beta, \phi; \mathcal{X}) = -\mathbb{E}_{(X,Y)\in\mathcal{X}^2}\left[\mathbb{1}_{\left\{C_{XY}=\mathbb{1}_{\left\{g_\beta(F_\phi(X),F_\phi(Y))>0.5\right\}}\right\}}\right] \tag{4.5}$$

The idea of directly training the model to maximize the CI exists in survival analysis literature. In [Steck et al., 2008], they describe the ranking problem similarly to (4.3), and relate it to the standard

proportional hazard model (PHM) $h(\tau|x) = h_0(\tau)\exp(F_\phi(x))$ [Cox, 1972]. They nevertheless commonly restrict it to linear $F_\phi(x) = \phi^T x$, for computational reasons and because first order is sufficient in many cases. In [Katzman et al., 2018], they finally use a neural network for F_ϕ, to create a personalized treatment recommender system. In particular, they show that the subsequent recommendation system, under PHM assumption, is the difference between the embedding of two samples. In [Jing et al., 2019] they learn a lifespan prediction model by pair, which regresses the difference between two samples embedding on the difference between samples target RUL (supervised learning). They call it *ranking loss* since it is the alternative name of metric learning (see Appendix B.5). Finally, in [Kalderstam et al., 2013] they propose to learn the CI with an ensemble of NNs trained with genetic algorithms.

An alternative to PHM is the multi-task logistic regression (MTLR) [Yu et al., 2011]. It consists in building a series of logistic regression models fitted on different time intervals to estimate the probability that the event of interest (e.g. death) happened within each interval. Another alternative to Cox model is the *proportional odds model* (POM) $O(\tau|x) = O_0(\tau)\exp(F_\phi(x))$ [Bennett, 1983], where $O(\tau|x)$ is the odd of individual surviving beyond time τ. We remind that an odd is the ratio $\frac{S(\tau|x)}{1-S(\tau|x)}$, such that $\exp(F_\phi(x)) = \frac{S(\tau|x)}{1-S(\tau|x)}$ for a constant baseline function $O_0(\tau)$. The POM is therefore directly related to CTE ratio (4.4).

Yet, we surprisingly found no reference of CTE-like survival analysis, with NNs trained with standard gradient descent that learns the CI C_{XY} from pairs of samples (X, Y), hence without assumptions on the shape of the survival model and using only a neural logistic regressor. We therefore did the experiments on standard survival analysis datasets as a side-contribution of our CTE approach, detailed in Section 4.6.3.

4.5 Related work on trend detection

The trend is a very general concept underlying data distribution. It is any monotone data's underlying factor, a "general direction and tendency" [Goldsmith, 2012], without additional specification. The trend detection, also, encompasses several purposes: trend magnitude estimation, change detection in trend, trend-based alarm when trend is substantial or when trend value reaches a threshold.

The trend can then be a drift in moments, model parameters, interactions or more generally a monotonic generative factor underlying data distribution. Depending on the a priori, different methods are proposed. In this section, we propose a review of commonly used trend detection methods, that we distributed in several classes. We keep the previously introduced notations.

Monitoring of statistics For each sample $X \in \mathcal{X}$, a set of statistics s^X can be inferred. If samples are sufficiently large, we can estimate variables mean, covariance and autocovariance matrices, any high-order moments or cumulants (or cross-cumulants), parameters of a fitted model (e.g. experiments 3.3.2 of Chapter 3), etc. Hence, the sequence $\{s^x\}_{x\in\mathcal{X}}$ may be monitored to find nonstationarity. If nonstationarity is monotone, it is a trend. More generally, any embedding $F(X)$ may be monitored while $F(X)$ is informative about X. We have seen in this thesis several example of embedding methods for time series data.

Regression of time on observations A standard trend detection method consists in regressing time index t on a response variable X_t. For example, $X_t = \beta_0 + \sum_{p=1}^{P} \beta_p(t)^p$. In general, time is rescaled in $[0, 1]$. The null hypothesis is generally $H_0 : \beta_p = 0\ \forall p$ (the absence of trend). Yet, the polynomial

shape is unpractical and/or restrictive. A more general flexible model use f_t a smooth function of time as trend estimate. f_t can be estimated with smoothing splines, spline regression, kernel smoother [Gray, 2007]. The interest of such methods is that a *trend curve* can be extracted. We note that this method can be used by regressing time not on observation X but on the set of statistics computed from X, i.e. $F(X) = \beta_0 + \sum_{p=1}^{P} \beta_p(t)^p$. Our CTE model does it, with adapted and efficient learning procedure.

Correlation with time index Directly related to regression of time methods, but without model assumption, we may define a correlation coefficient between time and data. In [Thomas, 1996], they use the Kendall's τ^K correlation coefficient. τ^K measures the *ordinal association* between variables, i.e. the correlation between variables' ranking. If one of the variable is the time index, hence τ^K captures the existence of a trend if null hypothesis $H_0 : \tau^K = 0$ is rejected. Yet, it does not quantify the trend nor give trend curves. Same limit for any nonlinear relational coefficient between time and observation (e.g. mutual information).

Remark 35. *If we train an embedding function F_ϕ such that $F_\phi(X)$ has maximum correlation with time index, we find a particular case of CTE where the relation between data and time depends on the correlation measure we use.*

Residual of decomposition into stationary components Another general and standard trend extraction method is the observation of a residual after decomposition of time series in stationary components (e.g. Fourier analysis [Körner, 1989]). In [Huang et al., 1998], they propose to use the *empirical mode decomposition* (EMD). They provide a general framework for decomposing time series into oscillatory sources $\{c_k\}_{k=1}^K$, built from data, whose number of modes is strictly decreasing. By construction, $X = \sum_{k=1}^K c_k + r_K$ such that r_K has only local min/max modes are also global modes. Hence if trend exists, it is in r_k. An efficient multivariate extension can be found in [Lang et al., 2018]. EMD is a *geometric* construction of trend. There exists several application for climate change [Wu et al., 2007], EEG monitoring [Lang et al., 2018] or electrical data [Mhamdi et al., 2010]. Yet, the trend decomposition is restricted additive trend and particular time series shapes. If for example, the nonstationarity is that the frequency of data underlying oscillations increases in time, each c_k may contain the trend and the residual r_k would be useless.

Independent component analysis We mentioned above the relation between trend and identifiable ICA using TCL. In [Blaschke et al., 2007] they use slowness (SFA, Section 2.4.2) to decompose time series into nonlinear independent components, by showing that slowness is a good supplementary assumption for identification. In particular, we note the trend is the slowest nonconstant component underlying any time series data.

There are many possible trend detection methods, that can be coupled. But, what is the advantage of CTE among these many available and simple trend detection methods?

Qualitative comparison between standard methods and CTE With the proposed CTE model, we do not claim outperformance against all above-cited methods in all situations. For each problem, a possibly expert-based or appropriate competitive method that fits the time series properties may be developed and used. Yet, we claim several advantages:

- *Specialization*: our model explicitly seeks and infers trend while majority of existing methods simply observe a posteriori the time series decomposition and then seek a trend
- *Universality*: we have no assumption on the type of trend (see the experiments for an illustration) nor the type of data
- *Generalization capacity*: neural networks have good generalization capacities at the condition that the architecture is chosen wisely. Hence, once trained our CTE can be used to infer new data state
- *Efficient inference* : neural networks are efficient at inference time
- *Simplicity*: many of the above-named methods are difficult to implement and/or require particular knowledge. Our CTE approach is simple to understand and to implement, in particular with the recent development of user-friendly deep learning libraries like Pytorch [Paszke et al., 2017].

4.6 Experiments

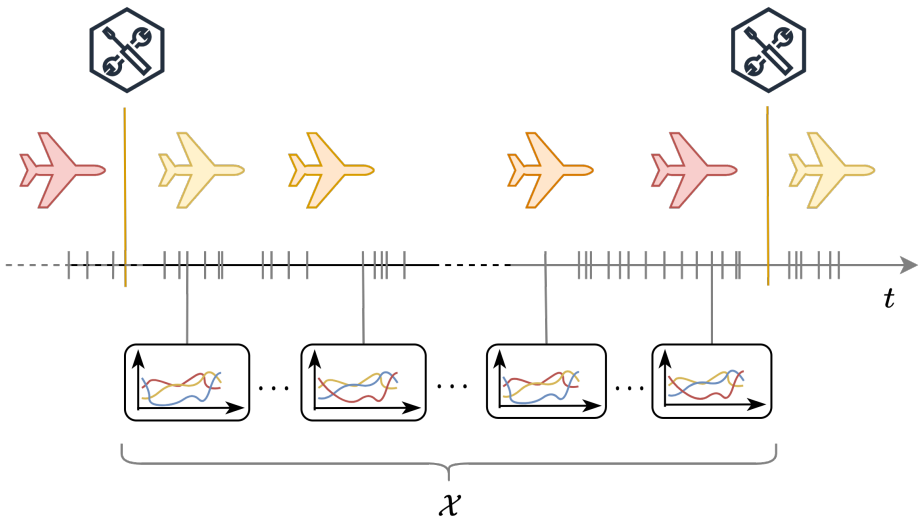


Figure 4.2 – Illustration of the creation of each dataset $\mathcal{X}$ for CTE experiments. Each dataset contains all consecutive samples between two maintenances. Planes illustrate the system $\mathcal{S}$ that generated the MTS samples. Vertical long orange lines are maintenances/restorations. Vertical black short lines are observation.

In this section, we propose experiments to illustrate our approach on several datasets with different types of hidden trend.

We note that in the experiments, we have one or several datasets $\mathcal{X}$ composed with samples X sorted with respect to their sampling date t^X, and we note $\mathbb{X}$ the set of datasets. The presence of several datasets is explained by the fact that we may have several batches of samples separated by a break in the monotonicity of the trend (e.g. a maintenances during the life of a mechanical system). Hence, each dataset is the set of samples between two monotonicity breaks. In this situation, the problem to solve is the sum of (4.3) over $\mathbb{X}$. Hence, no information is directly shared between the datasets, except the parameters (ϕ, β) to learn. Illustration is given in Figure 4.2.

Remark 36. *At training time, inter-dataset comparison may be inappropriate. For example, we assume that dataset $\mathcal{X} \in \mathbb{X}$ has a hidden ageing coefficient $\alpha_{\mathcal{X}}$ and $\mathcal{Y} \in \mathbb{X}$ has a hidden ageing coefficient $\alpha_{\mathcal{Y}}$ such that $\alpha_{\mathcal{X}} < \alpha_{\mathcal{Y}}$. Then for a sample X from $\mathcal{X}$ and a sample Y from $\mathcal{Y}$, we can have $t^X < t^Y$ and $\tau^X > \tau^Y$, which stands against the identifiability assumptions.*

4.6.1 Toy examples

For the toy examples, we compare CTE to the most appropriate trend extraction methods in different situations.

4.6.1.1 Evolving correlation

We generate several samples of multivariate Gaussian samplings. We first generate a random sparse correlation matrix Σ. We generate 20 sets of samples (10 for training, 10 for testing). For each set $\mathcal{X}$ we generate a random sparse positive semi-definite matrix $S^{\mathcal{X}}$ with entries in $[-10^{-3}, 10^{-3}]$ that shares at least the sparsity of Σ. In each set $\mathcal{X}$, we generate 1000 Gaussian samples, such that the sample $X \in \mathcal{X}$ associated with time index $t^X \in \mathbb{N}$ contains 100 random point from $\mathcal{N}(0, \Sigma + t^X S^{\mathcal{X}})$. We compare the CTE model with a maximum-likelihood covariance estimation for each sample (from `covariance` package of `scikit-learn` [Pedregosa et al., 2011]). Inferred ageing curves are drawn in Figure 4.3.

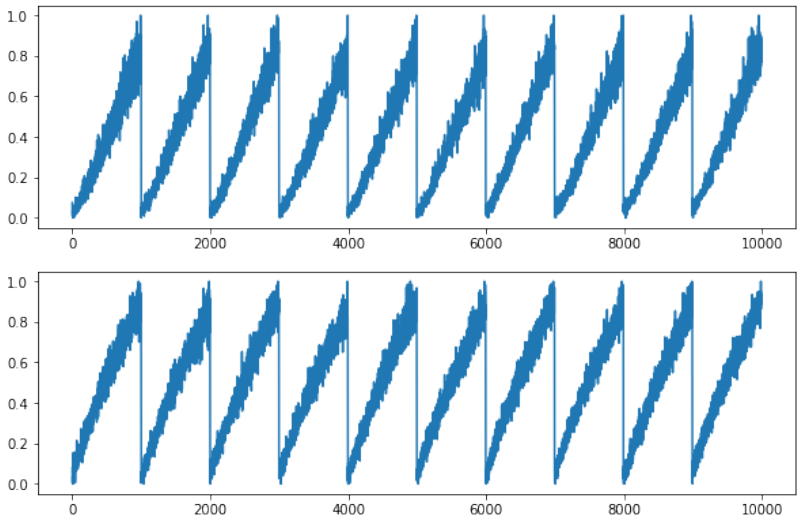


Figure 4.3 – Ageing curves computed on the 10 test sets (each monotonic segment matches a set of ageing correlation). **Top**: using estimated correlation matrix. **Bottom**: using curves $\beta^T F_\phi(X)$.

The variance of the increments of the estimated ageing curves is 0.004 for correlation-based estimation and 0.006 for CTE-based (equal on train set). The correlation between the two curves is 0.98. Hence, without the assumption that the drift was contained in the correlation, CTE achieves the quality of the trend estimation that uses the true nature of the trend.

We note that we can use several types of neural networks, as soon as it is sufficiently expressive to be considered as a universal approximator with respect to the complexity of the data.

4.6.1.2 Structural independent components

We use an experiment presented as an example in a preliminary experimental work [Pineau et al., 2020a]. This work presents a short guideline on how to simply achieve time series BSS using flow-based models (FBMs). This model is called slow flow-based model (S-FBM) and is quickly introduced above.

S-FBM We have seen in Section 2.4.2 that slowness plays an important role in standard time series representation learning. In Section 2.2.4, we have introduced flow-based models (FBM), that use normalizing flows to create powerful latent variable models (LVMs) and in Section 2.4.4 that the identifiability in LVMs solved by maximum likelihood depends on the existence of auxiliary variable U_t on which we condition the representation Z_t of each observation X_t. In particular, we can use $U_t = Z_{t-1}$, finding back the SFA prior (2.17). It is equivalent to add temporal differential operator Δ in the chain of normalizing flows F_ϕ, and maximize the likelihood of $\Delta \circ F_\phi(X)$ under Gaussian assumption (2.17). We call it *slow-FBM*. Finally, using the recent advances in neural BSS (see Section 2.6), we can say that with enough data (asymptotic result) and sufficiently large neural network F_ϕ, slow-FBM has the necessary conditions to recover the true sources up to linear transform A, i.e. we can learn ϕ such that $\exists A : S = AF_\phi(X)$. Matrix A is assumed full-rank and is post-learned with a linear ICA.

This work is a preliminary work and is not sufficiently advanced to have a full chapter. We present it here as a method for time series BSS more than a contribution, as a meaningful comparative model for the current experiment.

We generate a time series composed with one trend, two cycles and one seasonality. These four components are corrupted by additive Gaussian noise $\mathcal{N}(0, 0.2)$. See illustration in Figure 4.4. The four components are assumed to be independent. We mix the components with an invertible flow-based neural network (FBM, see Section 2.2.4).

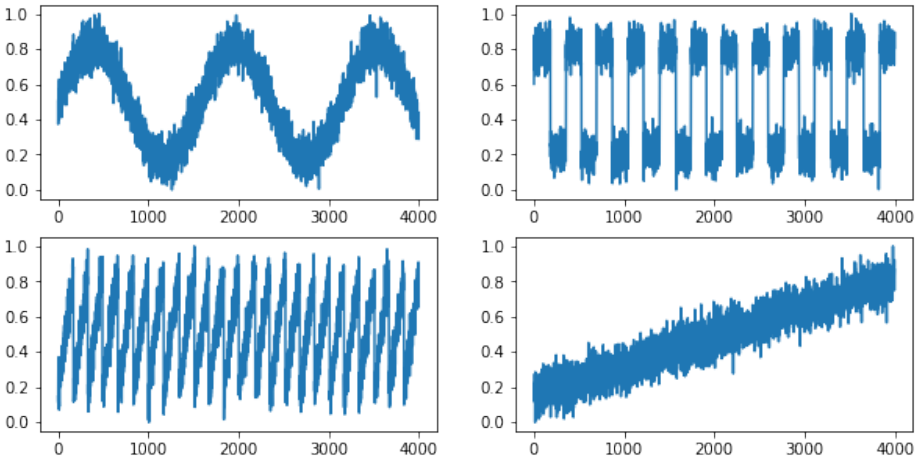


Figure 4.4 – Structural components.

Since we know that one of the sources is effectively a trend and that the sources are independent, nonlinear independent BSS is an optimal approach. We can use the method S-FBM described above. We confront CTE to this method. Results are given in Table 4.2.
Hence, CTE achieves the results of an appropriate estimate of the hidden trend.

S-FBM + ICA	CTE
96.7 ± 3.2	98.9 ± 0.5

Table 4.1 – Absolute correlation between estimated and true trend.

4.6.2 Mechanical systems's health monitoring

In this section, we apply our CTE in the same data than in Chapter 3 to have comparison of CTE with different neural-based inductive biases.

4.6.2.1 Ball-springs dataset

Dataset We use the same dataset than for Seq2Graph experiments. Reminder: we simulate a synthetic dataset of 15000 samples (trajectories), 5000 for train, 5000 for validation and 5000 for test. Each trajectory is 49 time-steps-long ($T = 49$). For each batch b of 50 samples, a constant ageing factor $\alpha_b \sim \mathcal{U}([0.9, 1])$ is applied to the system: at each sample X whose index is $t^X \in [\![0, 50]\!]$ (within the batch b of 50 samples), we randomly choose a spring (i, j) and multiply its rigidity by $\alpha_b^{t^X}$, i.e. an exponential ageing coefficient with respect to sample index. Every 50 samples, we *restore* the state of the system and another ageing factor is sampled and applied to the next batch of 50 samples. For some trajectories, $\alpha_b = 1$, i.e. there is no ageing: the initial hidden causality graph has binary adjacency and remains the same along the life of the system. When $\alpha_b < 1$, the initial graph is deteriorating during along the life (observed through 50 samples) of the system, until restoration.

We choose a relational neural network (RelNN) as embedding function F_ϕ. We jointly train F_ϕ and a linear classifier β by minimizing (4.3). Once trained, we build the curve $\{\beta^\top F_\phi(X)\}_{X \in \mathcal{X}^{test}}$, where $\mathcal{X}^{test}$ is the test set of samples. Estimated ageing curves are given in 4.5.

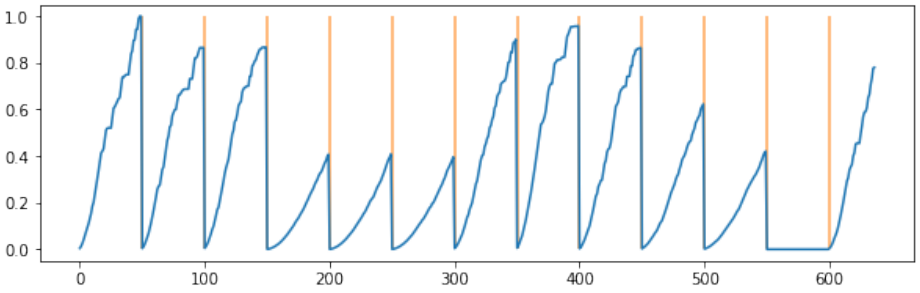
Seq2Graph	CTE
0.97 ± 0.03	0.96 ± 0.01

Table 4.2 – Absolute correlation between estimated and true trend. Seq2Graph is the model presented in Chapter 3.

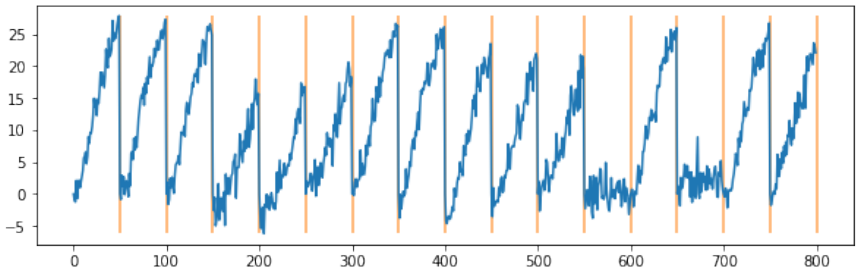
Results The concordance index of the trend detection computed as (4.5) is equal to 96% on the test set. We find back the quality of the results that we obtained with Seq2Graph (relational encoder, VAR decoder and unique causal graph, Section 3.3.2.1). The difference is that here we swapped physical inductive bias enforced by the autoregressive sparse decoder and unique causal graph against the trend inductive bias enforced by the trend contrastive learning. Such a high correlation means that the representation has inferred the right underlying ageing coefficient α.

4.6.2.2 NASA dataset

Dataset We use the same dataset than for the second experiment of Seq2Graph, Section 3.3.2.2. The objective is to extract monotonous signals from each lifetime batches (samples from plant to failure). Since



(a) Ground truth ageing curves



(b) Estimated ageing curves $\{\beta^\top F_\phi(X)\}_{X\in\mathcal{X}}$.

Figure 4.5 – Ageing curves of ageing balls-springs system, from 16 datasets of 50 consecutive samples. The vertical orange lines delimit the datasets.

we do not have access to the true ageing curve, we do the same analysis than in experiments of Chapter 3: building ageing curves and seeing if we can infer interesting information, like for example the failure moment. We compare with the previously presented results.

We choose a recurrent neural network (RNN) as embedding function F_ϕ. We jointly train F_ϕ and a linear classifier β by minimzing (4.3). Once trained, we apply the same experimental protocol than in Section 3.3.2. We build an ageing curve $\{\beta^\top F_\phi(X)\}_{X\in\mathcal{X}^{valid}}$, where $\mathcal{X}^{valid}$ is the validation set of samples. We compute a failure threshold τ^{valid} that must indicate when an engine goes to failure. We set τ^{valid} to the maximal threshold that ensures turbine engine detection for all validation trajectories, that is:

$$\tau^{valid} = \min_{X\in\mathcal{X}^{val,def}} \beta^\top F_\phi(X) \tag{4.6}$$

where $\mathcal{X}^{val,def}$ is the set of validation samples preceding the engine failure. We note that τ^{valid} has no safety margin, i.e. any threshold above τ^{valid} misses at least one engine failure in the validation set. It is possible to add a margin, with $\mathcal{X}^{val,def}$ the set of validation samples preceding with k index the engine failure. Then, we build an ageing curve $\{\beta^\top F_\phi(X)\}_{X\in\mathcal{X}^{test}}$, where $\mathcal{X}^{test}$ is the test set of samples. We apply a failure detection test using τ^{valid}, represented by the horizontal dotted line in Figure 4.6.

Results As a first assessment, we see in Figure 3.17 that the estimated ageing curves are monotonic inside each trajectory (between two vertical orange lines): the concordance index of the trend detection computed as (4.5) is equal to 98.5% on the test set.

Hence the model has well generalized on the trend extraction problem, which is its only *explicit* objective.

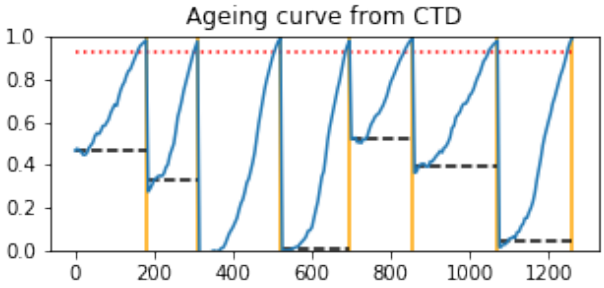


Figure 4.6 – Estimated ageing curves $\{\beta^\top F_\phi(X)\}_{X\in\mathcal{X}}$ on the test set of CMAPSS data. We kept the same samples than Figure 3.17 for comparison. Black dashed horizontal lines are the estimated initial states of each engine, computed as the mean value of the curve on the 10 first samples of each batch.

We note that the monotonicity is perfect, contrary to curves extracted from Seq2Graph (we remind that Seq2Graph do not solve a trend extraction problem but an unsupervised representation problem). We observe that the trajectories do not begin at the same value as well (dashed horizontal lines in Figure 3.17). We impute it to the fact that the mechanical faults are located at the beginning of each trajectory and that they vary in intensity. Hence, the inferred first samples of each trajectory do not have to be equal.

We now look at the ability of CTE-based representation to detect failures, like in previous chapter experiments, from which we also take comparative models. In Figure 4.7 (left), built with extracted signal show in Figure 4.6, gives the proportion of alarm at different time steps before actual failure happens. Second, we want detection of the coming failures to be reasonably early to avoid false alarms. If curves cross threshold too early, the MTS representation is useless. Figure 4.7 (left) shows that CTE is consistent in early detection, with no alarms far from failure, thanks to the consistency of the extracted monotonic signal. In Figure 4.8 (right), we give the average precision scores (APS), that summarizes precision-recall curves as the weighted mean of precision achieved at each failure detection threshold.

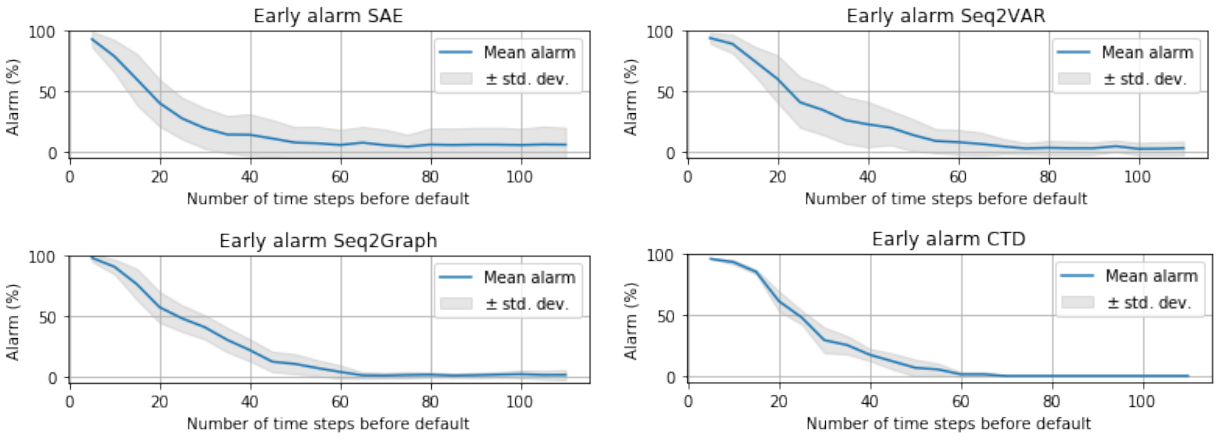


Figure 4.7 – Early alarm on CMAPSS data using MTS representation models SAE, Seq2VAR, Seq2Graph (from Chapter 3) and CTE.

We have built a representation of the CMAPSS samples that explicitly represents the age of the system (up to monotone transformation, see Section 4.3) at the moment of the sampling. It proved to be very

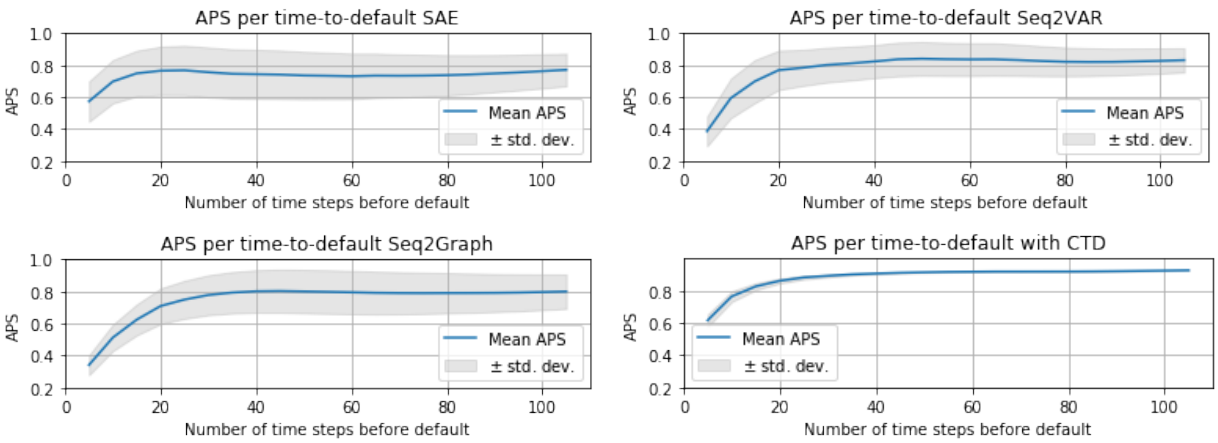


Figure 4.8 – Average Precision Score (APS) of the failure detection using MTS representation models SAE, Seq2VAR, Seq2Graph (from Chapter 3) and CTE.

consistent in common ageing detection problems. Compared to samples representation learned using Granger causality graphs (and concurrent methods) presented in Section 3.3.2.2, the CTE finds better ageing curves from which we can infer failure detection methods, as expected from this new inductive bias.

4.6.3 Side-contribution: survival analysis with CTE

We showed in Section 4.4.2 that our trend extraction method is related to survival analysis methods. In this section, we show that our CTE model gives state-of-the-art survival analysis results on five public survival analysis datasets.

Datasets Customer churn prediction (Churn) consists in estimating the percentage of customers that stop using the products and services of a company. The survival analysis for customer churn helps companies predicting when a customer is likely to quit considering its personal features.

Credit risk (Credit) is the risk carried by a loan company when people borrow money. It corresponds to the likelihood of borrower's credit default with respect to personal features. Survival analysis for credit risk predicts if and when a borrower is likely to fail.

Treatment effects on survival time are fundamental for pharmaceutical laboratory. It is possible to do survival analysis of patients. Two public datasets exist. First, German Breast Cancer Study Group (GBCSG2) contain a subset of variables from the German breast cancer study [Schumacher et al., 1994]. It studies the effects of a treatment with hormones on survival time without cancer recurrence. Second, Mayo Clinic Primary Biliary Cirrhossis (PBC) [Therneau and Grambsch, 2000] studies the effects of the drug D-penicillamine on the survival time of patients.

Finally, more interesting for Safran, the predictive maintenance of mechanical equipment consists in predicting when an equipment will fail in order to prevent the failure by processing a maintenance. We use a public dataset, called Maintenance, whose data is extracted from sensors on manufacturing machines to predict which will fail soon.

We compare our CTE surival analysis with five other standard models: the linear and neural Cox

proportional hazard models (Cox-PHM) [Cox, 1972, Katzman et al., 2018], the extra-tree and random forest survival analysis (RFS) [Ishwaran et al., 2008] and a multi-task logistic regression (MTLR) survival analysis [Yu et al., 2011]. In Cox models, hazard function h has the form $h(\tau|x) = h_0(\tau)\exp(F_\phi(X))$. If F_ϕ is linear, we call it *linear Cox*, if F_ϕ is a neural network, we call it *neural Cox*. In tree-based survival, an estimation of the *cumulative hazard function* ($\int_t h(t|x)dt$) is done with bags of trees. The MTLR models are a built as a series of logistic regression models built on different time intervals so as to estimate the probability that the event of interest happened within each interval. We do not provide additional information here. An exhaustive introduction to these models is provided in the website of the PySurvival library [Fotso et al., 19] that we used to implement the comparative methods.

For the experiments, we used a 10-folds train-test setup. Each dataset is divided into 10 folds used for cross-validation i.e., one fold serves as the testing set while the other ones compose the training set. This 10-folds train-test separation is repeated several times for robustness of the results. Table 4.3 show the mean and standard-deviation of the concordance index (CI, see Section 4.4.2) computed on the test samples.

	Churn	Credit	GBCSG2	PBC	Maintenance
Linear Cox	87.9 ± 1.4	76.0 ± 3.1	66.3 ± 5.5	79.0 ± 8.3	96.5 ± 1.3
Neural Cox	87.8 ± 2.5	76.5 ± 2.6	64.4 ± 4.2	$\mathbf{80.2} \pm 8.4$	99.3 ± 0.6
Extra Tree Survival	85.4 ± 1.9	71.2 ± 4.3	66.8 ± 6.4	78.9 ± 8.5	94.1 ± 1.5
Random Forest Survival	84.0 ± 2.7	72.1 ± 3.2	67.6 ± 6.4	79.0 ± 7.5	93.8 ± 2.1
Multitask	88.6 ± 1.2	72.4 ± 3.7	$\mathbf{67.9} \pm 7.7$	74.8 ± 8.0	93.4 ± 1.9
CTE (ours)	$\mathbf{89.9} \pm 1.1$	$\mathbf{76.9} \pm 2.1$	$\mathbf{67.9} \pm 5.7$	80.1 ± 7.5	$\mathbf{99.6} \pm 0.6$

Table 4.3 – Concordance index of the survival analysis experiments on four datasets, using state-of-the-art models versus our CTE learning method.

Remark 37. *Bold numbers are the best results in term of means; yet taking into account the standard deviations, we cannot claim that CTE is the best survival model, only that it achieves state-of-the-art results without standard survival analysis assumption (in particular assumption on survival model). It is not surprising considering the relation between CTE and survival analysis frameworks, described in Section 4.4.2.*

The main difference between CTE for general trend extraction and CTE for survival analysis lies in the fact that self-supervision is partial. In standard time series, label is easily extracted from two data samples by looking at their time index (see above). Yet, in survival analysis, certain samples are right-censored. It means that, for each sample we observe a time index that could be either the survival time or the censored time (event of interest has not been observed). The censored time will be a lower bound for the survival time. Hence, for two samples X and Y, we cannot robustly compare samples (hence creating a contrasting label) in the following situations:

- $\delta_X = \delta_Y = 0$, X and Y are right-censored
- $\delta_X = 0$, $\delta_Y = 1$ and $t^X < t^Y$
- $\delta_X = 1$, $\delta_Y = 0$ and $t^Y < t^X$

In all other situations, the comparison of time indices t^X and t^Y give robust information about the relative age of the samples X and Y. Yet, since samples come from different individuals (e.g. one sample for one patient in health survival analysis), comparing samples even when time indices are eligible for robust comparison is not always appropriate, as explained in Remark 36. In fact, each individual has proper features, and in particular its own ageing trend. Yet, we have no choice to consider that all individuals may have similar behavior to train the model, since we have only one sample per individual, but it is not specific to our approach: all models suffer from the same limit.

4.7 Discussion on noisy CTE

In real world data, the trend may be noisy. A typical example in monitoring of mechanical system's state exists when maintenance procedures (cleaning or piece changing) are effectuated, creating partial (and usually not indicated in data) restoration of the state. In this situation, with a such powerful embedding function as neural networks, the noise can be learned, preventing the network to capture the relevant information. A short analysis of the impact of the noise if given below.

We note that, by definition, only the trend component contains trend information. Then we have

$$p(C_{XY}|X,Y) = p(C_{XY}|\tau^X,\tau^Y).$$

We now consider that there is noise in the trend, i.e. the existence of outliers such that for example $\mathbb{1}_{\{\tau^Y-\tau^X>0\}} \neq C_{XY}$, or more generally $p(C_{XY}=1|\tau^X,\tau^Y) < p(1-C_{XY}=0|\tau^X,\tau^Y)$ when $\tau^Y > \tau^X$. This noise can be seen as a contamination of the density $p(\tau^X,\tau^Y|C_{XY})$ with another density $\nu(\tau^X,\tau^Y|C_{XY})$:

$$\begin{aligned} p^\nu\left(\tau^X,\tau^Y|C_{XY}\right) &:= (1-\eta)\,p\left(\tau^X,\tau^Y|C_{XY}\right) + \eta\nu\left(\tau^X,\tau^Y|C_{XY}\right) \\ &= p\left(\tau^X,\tau^Y|C_{XY}\right)\left(1+\eta\left[\frac{\nu\left(\tau^X,\tau^Y|C_{XY}\right)}{p\left(\tau^X,\tau^Y|C_{XY}\right)}-1\right]\right) \end{aligned}$$

with $\eta \in [0,1]$ the prevalence of the contamination as named in [Fujisawa and Eguchi, 2008]. Then

$$\begin{aligned} \log p^\nu\left(\tau^X,\tau^Y|C_{XY}\right) &= \log\left(p\left(\tau^X,\tau^Y|C_{XY}\right)\right) + \log\left(\left(1+\eta\left[\frac{\nu\left(\tau^X,\tau^Y|C_{XY}\right)}{p\left(\tau^X,\tau^Y|C_{XY}\right)}-1\right]\right)\right) \\ &= \log\left(p\left(\tau^X,\tau^Y|C_{XY}\right)\right) + \eta\left[\frac{\nu\left(\tau^X,\tau^Y|C_{XY}\right)}{p\left(\tau^X,\tau^Y|C_{XY}\right)}-1\right] + O\left(\eta^2\right) \end{aligned}$$

using the Taylor expansion of the logarithm for small η. Finally

$$\begin{aligned} \log\frac{p^\nu(X,Y|C_{XY}=1)}{p^\nu(X,Y|C_{XY}=0)} &= \log\frac{p(X,Y|C_{XY}=1)}{p(X,Y|C_{XY}=0)} + \eta\left[\frac{\nu\left(\tau^X,\tau^Y|C_{XY}=1\right)}{p\left(\tau^X,\tau^Y|C_{XY}=1\right)}-1\right] \\ &\quad - \eta\left[\frac{\nu\left(\tau^X,\tau^Y|C_{XY}=0\right)}{p\left(\tau^X,\tau^Y|C_{XY}=0\right)}-1\right] + O\left(\eta^2\right) \\ &= h\left(\tau^Y-\tau^X\right) + \eta\left(\frac{\nu\left(\tau^X,\tau^Y|C_{XY}=1\right)}{p\left(\tau^X,\tau^Y|C_{XY}=1\right)} - \frac{\nu\left(\tau^X,\tau^Y|C_{XY}=0\right)}{p\left(\tau^X,\tau^Y|C_{XY}=0\right)}\right) + O\left(\eta^2\right) \end{aligned}$$

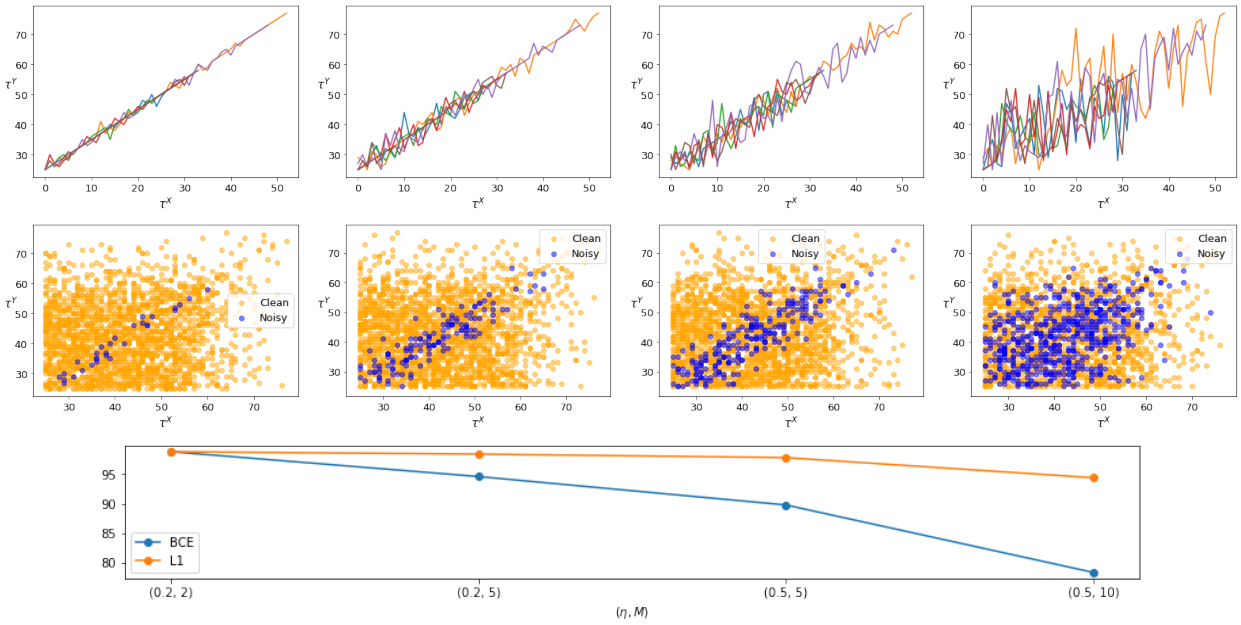


Figure 4.9 – Example of noisy CTE on CMAPSS data, with noise distribution given by (4.7). The experimental design is the same than in Experiment 4.6.2.2. **Top**: Four examples of noisy curves $x \mapsto x + 25$ with couples (η, M) respectively $(0.2, 2), (0.2, 5), (0.5, 5), (0.5, 10)$ (increasing noise), on time indices of 5 datasets (5 curves per subfigure) of CMAPSS data. We see that, even with large local noise, the trend is visually still existing. **Middle**: Distribution of couples (t^X, t^Y) for random couples of time series data $(X, Y) \in \mathcal{X}^2$, for different levels of noise. **Bottom**: Concordance index (4.5), with F_ϕ a gated RNN with two 128-dimensional hidden layers. We use the L1 norm as robust classification loss (result from [Ghosh et al., 2017]).

where $h\left(\tau^Y - \tau^X\right)$ is a monotone function minimal sufficient statistic for C_{XY} (see Section 4.3).

The interesting point here is that we can understand the degradation of the CTE in terms of likelihood ratio between clean density and noise density. Under universal approximation properties of the representation function, we learn parameters β, ϕ such that $F_{\beta,\phi}(X, Y)$ is close to $\log \frac{p^\nu(X,Y|C_{XY}=1)}{p^\nu(X,Y|C_{XY}=0)}$. Yet, in the case where $(\tau^X, \tau^Y)|C_{XY}$ is unlikely under p compared to ν (because of the contamination), the learning can be severely disrupted.

Yet the situation is not desperate since there exist several ways to limit the impact of noisy labels in a classification task. In particular, for trend extraction, the impact of the noise may be different depending on the type of noise.

First, we may have local noise, that consists in having local uncertainty in the ageing monotonicity, while long term trend is still monotone, like in the second toy experiment 4.6.1.2. For example, if two samples X and Y have close sampling times t^X and t^Y, we may have

$$\nu(\tau^X, \tau^Y|C_{XY}) \propto e^{-|\tau^X - \tau^Y|} \mathbb{1}_{|\tau^X - \tau^Y| < M} \tag{4.7}$$

with $M \geq 0$. In this situation, since we compare randomly chosen samples, if noise is localized (low M) the model will mainly see true labels even with a η level of noise, since the temporal scale of trend (and the training) will be larger than the temporal scale of the noise. Hence, the ratio $\frac{\nu(\tau^X,\tau^Y|C_{XY}=1)}{p(\tau^X,\tau^Y|C_{XY}=1)}$ will not

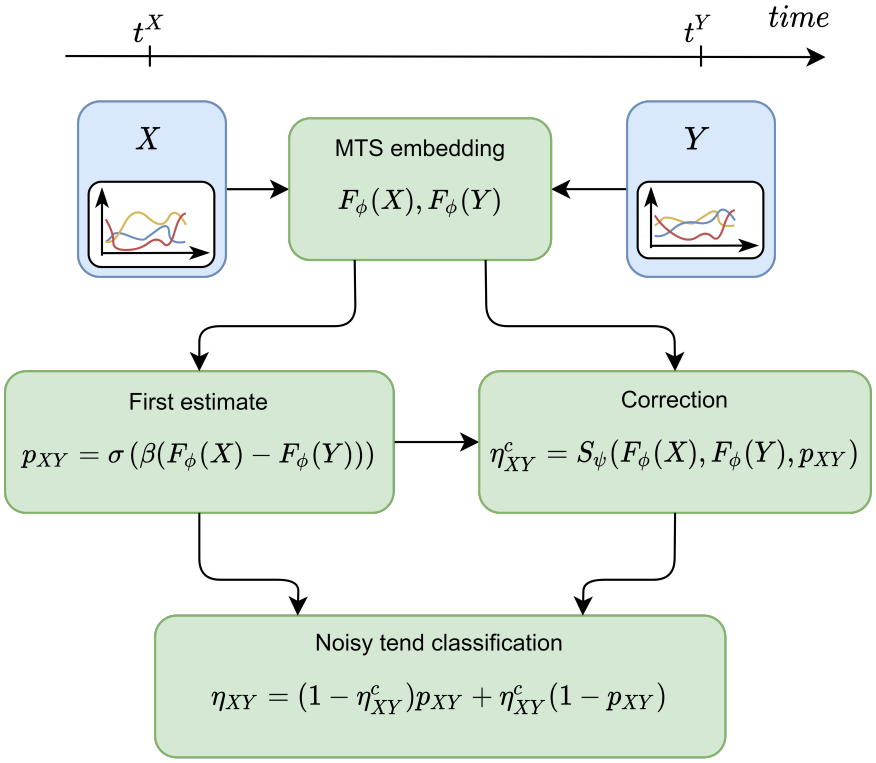


Figure 4.10 – Corrected CTE to learn from noisy labels.

explode since the noisy couples (t^X, t^Y) are situated in the dense part of the support of $p\left(\tau^X, \tau^Y | C_{XY}\right)$ (see scatter plots in Figure 4.9). Hence, with an expressive embedding function F_ϕ (with inductive bias) and an adequate learning procedure (e.g. symmetric loss [Ghosh et al., 2017] or regularization), a representation and generalization capacity may be obtained even in high-capacity regime (e.g. universal approximation embedding function). An illustration is given in Figure 4.9.

Second, the global noise. For example, we do not know that an intervention was done on studied ageing system, creating long-term inversions of the trend. In this situation, the ratio $\frac{\nu\left(\tau^X, \tau^Y | C_{XY}=1\right)}{p\left(\tau^X, \tau^Y | C_{XY}=1\right)}$ explodes, since supports of $\nu\left(\tau^X, \tau^Y | C_{XY}\right)$ and $p\left(\tau^X, \tau^Y | C_{XY}\right)$ have low overlap (see scatter plot in Figure 4.11). Hence, the standard robust classification losses are not sufficient.

In [Fujisawa and Eguchi, 2008] they propose a method to robustly estimate parameters of a model under *heavy contamination*, using a modified cross-entropy function called γ-entropy. The γ-entropy has been adapted in [Sasaki et al., 2019] to the problem of robust ICA with TCL (TCL is detailed in the Section 4.4.1). Another possibility is to add a likelihood estimate (or another meaningful information) of the individual representations $F_\phi(X)$ to the loss (4.3).

Another possibility is to add an additional network that corrects the first classification decision to meet the noisy labels. This methods relies on the fact that, when using appropriate neural network architecture, the clean labels are easier to learn than noisy labels (gradient path is smoother and more monotone in the direction of the truth). Hence, using the same principle than attention mechanism (a sample-dependent variable selection method that helps the learning procedure), we may propose the following learning procedure illustrated in Figure 4.10, where $S_\psi : \mathbb{R}^{d_e} \times \mathbb{R}^{d_e} \times [0,1] \to [0,1]$ a neural network with parameters

ψ learns a switch parameter η^c_{XY} that changes the estimated probability $p_{XY} := g_\beta\left(F_\phi(X), F_\phi(Y)\right)$ into $\eta_{XY} = (1 - \eta^c_{XY})p_{XY} + \eta^c_{XY}(1 - p_{XY})$, with respect to the trend detection objective. Like for attention, switching model is useful if it helps the training. Considering the fact that an appropriate model (not over-parametrized compared to the difficulty of the task) that easily learns from data is better than a model that overfits (overfitting generally comes from inappropriate model), if F_ϕ is appropriate with respect to data and task (here trend detection), switching model will help. In this situation, the estimated probability p_{XY} will contain the more easily learned estimation of label C_{XY}, while η_{XY} will estimate the noisy labels seen by the model during training. Each η^c_{XY} will be the estimation of the probability that C_{XY} is noisy.

Remark 38. *Like for attention mechanism, the success of such approach depends on the fact that the true labels are much more easier to learn than the noisy labels. We can regularize the model F_ϕ to limit the overfitting capacities and force the label switcher η^c_{XY} to help [Zhang et al., 2016].*

Some results are given in Figure 4.11. We note that only the training set is noisy, not the validation and test sets. We see that effectively, our method based on an *unsupervised noise detection* helps the model finding true labels under the hypothesis that true labels are easier to learn than noisy labels. The CI on test set (clean) is 98.7%, i.e. the level obtained with clean training data.

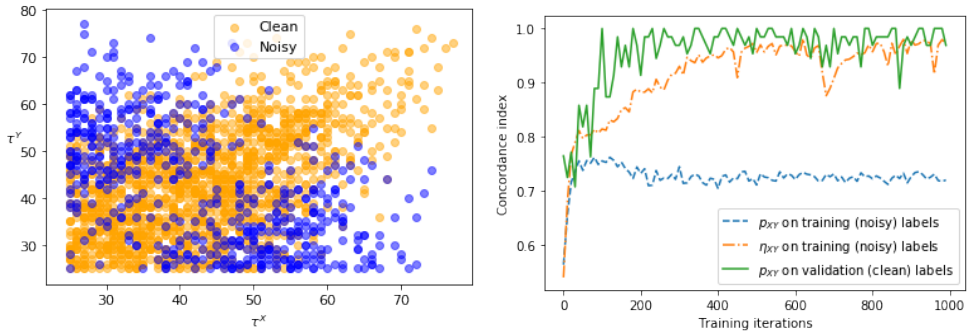


Figure 4.11 – **Left**: Distribution of the noisy labels. **Right**: Results on CMAPSS data when we ignore the maintenance procedures (i.e. we only have one dataset that contains all the 25-long time series samples), using the method proposed in Figure 4.10.

4.8 Conclusion

In this chapter, we proposed a new universal trend extraction methods using contrastive learning procedure. Our model is simple, easy to implement and is supported by theoretical identifiability results. In particular, it theoretically contains all the already existing trend extraction methods thanks to the usage of universal approximation functions coupled to a generic formulation of the trend detection and extraction problem.

Chapter 5

Recurrent graph classification

Abstract The variable size and absence of natural ordering of the nodes are exotic features of graphs that make them difficult to embed. To circumvent variable size problem, we use the following inductive bias: a graph is a sequence of nodes. Under this assumption, it is possible to use recurrent neural networks, a high-capacity neural network for sequential data, to represent graphs. Moreover, under label supervision, *isomorphism invariance* problem (introduced in Section 2.5) is greedy solved by introducing randomness in the node ordering for each graph at each iteration of the learning process. We show that using these two features, we obtain a good graph classifier. Inspired by natural language processing methods, we also study the influence of a node-level self-supervised additional task as regularization. This chapter covers one contribution:

- Pineau, E. and de Lara, N. (2019). Variational recurrent neural networks for graph classification. *Workshop on Representation Learning on Graphs and Manifolds, at International Conference on Learning Representation (RLGM, ICLR 2019).*

5.1 Sequential embedding of sequence of nodes

We consider $G = (V, E, W)$ a graph where V is the set of vertices (also names nodes) and $E \subset V \times V$ is the set of edges. We have seen in Section 2.5 that G is fully identified, modulo any permutation π over its nodes index, by its (possibly weighted) adjacency matrix W^π such that $W^\pi_{ij} \in]0, 1]$ if $(i, j) \in E$ and $W^\pi_{ij} = 0$ otherwise. In classific ation problem, each graph G has a label $y^G \in [\![0, C-1]\!]$. For example, G is a chemical component and $y^G = 0$ if G is carcinogen, $y^G = 1$ otherwise. We note $\mathcal{X}$ the dataset of graphs that contain all the couples (G, y^G).

We want to learn a graph classifier that estimate the label posterior distribution $p(y|G)$. It consists in building and training an embedding function F_ϕ with parameters ϕ and a classifier D_ψ with parameters ψ such that $D_\psi \circ F_\phi(G) = y^G$, with D_ψ a simple (e.g. linear) classifier. We note $p_{\phi,\psi}(y|G)$ the estimated posterior distribution.

Yet, graphs raise two main difficulties, as illustrated in Figure 2.18 of Section 2.5.

First, graphs have variable size, whose usual representations are edge list or adjacency matrix. Hence, it is required that F_ϕ takes variable sized inputs, and outputs fixed-size embedding.

Second, in a classification task, the label of a graph is independent from the indices of its nodes, so the model used for prediction should be *invariant* to node ordering as well. Hence, if a graph $G \in \mathcal{X}$ has n_G nodes $V = \{v_i\}_1^{n_G}$ and if π is a permutation of $[\![1, n_G]\!]$ and if we note G^π the graph with nodes $V^\pi = \{v_{\pi(i)}\}_{i=1}^{n_G}$ (G and G^π are called *isomorphic* graphs, i.e. we can find a bijective reindexing of the nodes to align the graphs), hence we expect to have $F_\phi(G) = F_\phi(G^\pi)$. It is called *isomorphism-invariance*. We remind that a permutation of the index of the nodes is equivalent to a simultaneous permutation of the rows and columns of the adjacency matrix W.

The isomorphism-invariance for graph embedding can be achieved by using certain types of embedding methods that use permutation-invariant functions, like sums or histograms, on isorphism-invariant embedding of the nodes. The difficulty is then to find such node embedding. More recently, an active research on graph neural networks (GNNs, see Appendix A.1.1) proposed neural networks architectures designed to be invariant to permutation of the nodes [Xu et al., 2018], which is required for permutation.

This chapter's contribution was done in parallel to the rise of these expressive GNN architectures. At that time, we proposed to test out a different inductive bias instead of improving GNNs for graph classification. The inductive bias is the following: a graph can be seen as a sequence of nodes. This point of view, that is true, opened new possibilities to proposed a new classification technique based on sequence embedding tools, using recurrent neural networks (RNN, see Appendix A.1.4) noted F_ϕ:

$$h_i = F_\phi(v_i, h_{i-1}) \tag{5.1}$$

and the representation of G is the last memory cell of the RNN $h_{n_G} := F_\phi(G) = F_\phi(v_{n_G}, h_{n_G-1})$ that accumulated all the information of the sequence of nodes.

The interest is two-folds. First, given a node representation, the sequential embedding of the nodes solves the variable-size problem. Second, if we choose a sufficiently expressive RNN F_ϕ and a simple classifier D_ψ, and learn parameters ϕ and ψ such that $D_\psi \circ F_\phi(G) = y^G = l_{G^\pi} = D_\psi \circ F_\phi(G^\pi)$ then we obtain an isomorphism-invariant supervised representation of graph G.

In this chapter, we propose a guideline to build such recurrent graph classifier (RGC). Besides, we experiment the addition of a stochastic node prediction task to see if it helps the model to capture the structure of the graphs, inspiring from sentence embedding in natural language processing (NLP) [Sanh et al., 2018]. The augmented model is denoted *variational recurrent graph classifier* (VRGC).

5.2 Recurrent graph classifier

We propose to use a sequential approach to embed graphs with a variable number of nodes and edges into a vector space of a chosen dimension. This latent representation is then used for classification.

Our model, called *recurrent graph classifier* (RGC), is divided in three main parts: the preprocessing sequential representation of the graph, the classification and a result aggregation phase. Consistently, we assume that F_ϕ is composed by a preprocessing F^0 and two consecutive RNNs, i.e. $F_\phi = F_\phi^2 \circ F_\phi^1 \circ F^0$.

Node ordering and pre-embedding Before being processed by the neural network, the adjacency matrix of a graph is transformed on-the-fly [You et al., 2018] by a preprocessing F^0. First, a node v_R with

index R is selected at random in V and used as *root* for a breadth first search (BFS) over the graph. A BFS consists in, given a root node, exploring all neighbor nodes, depth level by depth level. An illustration of the BFS is given in Figure 5.1.

The rows and columns of the adjacency matrix are then reordered according to the sequence of node indices returned by the BFS. The interest of using BFS as a preprocessing step is the following: without changing the nature of the embedding (still the adjacency matrix), the observed structure contains graph-level information. BFS preprocessing arranges the adjacency in a favorable shape.

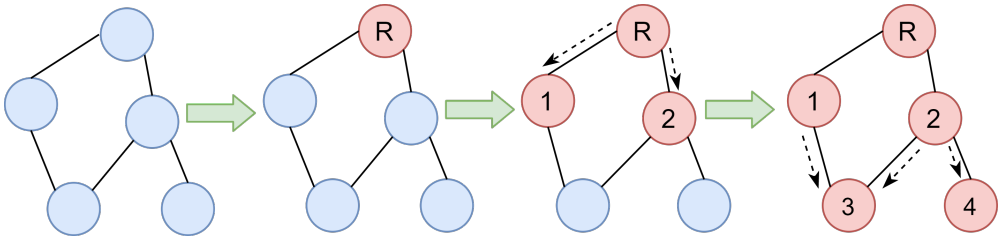


Figure 5.1 – BFS ordering.

Once we have a good adjacency disposition, we still have for each G a list of n_G vectors of size n_G. Hence, we cannot use standard RNN shared between all graphs. We propose the following approach. Each row i (corresponding to the i^{th} node in the BFS ordering) is truncated to keep only the connections of node v_i with the $\min(i, d)$ nodes that precede v_i in the BFS (illustration in Figure 5.2). The d first nodes are zero padded since they have less than d connections in the BFS ordered list. This way, each node becomes d-dimensional such that we now have G represented as a sequence of n_G vectors of dimension d. The full pre-processing to build a sequence from graph G is dependent on root R and we note it $h^0_R := F^0(G, R) \in [0, 1]^{n_G \times d}$ the output of the preprocessing.

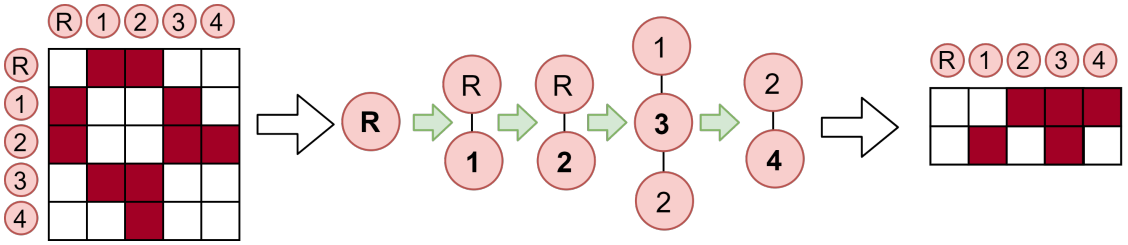


Figure 5.2 – Truncation of the BFS-ordered nodes.

Neural embedding After node ordering and pre-embedding, each graph is processed as a sequence of d-dimensional nodes by a gated recurrent unit (GRU) RNN noted F^1_ϕ. The GRU is a special RNN with extended ability to learn long term dependencies by solving vanishing gradient effect [Cho et al., 2014b]. We note that the choice of GRU over Long Short Term Memory ([Hochreiter and Schmidhuber, 1997]) networks is arbitrary as they have overall equivalent modeling power [Chung et al., 2014].

In order to help the recurrent network training, we add a multi-layer perceptron (MLP, see Appendix A.1.2) between pre-embedding and recurrent embedding: the nodes will be presented to the GRU as continuous vectors instead of binary adjacency vectors. Finally, the GRU sequentially embeds the BFS-ordered

and truncated set of nodes h_R^0 such that $h^1 = F_\phi^1(h_R^0)$ is the embedding of the graphs as a sequence of well preprocessed and recurrently embedded nodes. See top line of Figure 5.3 for illustration.

Classification After the embedding step, we use an additional GRU noted F_ϕ^2 dedicated to classification that takes h^1 as input, i.e. $h = F_\phi^2(h^1)$. The last memory cell h_{n_G} is an embedding of the graph G that feeds a softmax multilayer perceptron (MLP) D_ψ which performs class prediction. Hence, in the end, for each graph we obtain an estimate of the distribution $p(y|G,R)$ with $p_{\phi,\psi}(y|G,R) := D_\psi \circ F_\phi(G,R)$, which is dependent on the root R. To limit the dependency on R, we propose the following procedure for training and testing.

Training The model parameters ϕ and ψ are learned by minimizing the cross-entropy loss between ground-truth and $p_{\phi,\psi}(y|G,R)$ class membership probability vector for all graphs G given all the possible indices of the BFS root R. We call this objective $\mathcal{L}_{classif}$:

$$\mathcal{L}_{classif} = \mathbb{E}_{G\sim\mathcal{X}}\left[\mathbb{E}_{R\sim\mathcal{U}([1,n_G])}\left[\sum_{y^G=1}^{C} \mathbb{1}_{\{y=y^G\}} p_{\phi,\psi}(y|G,R)\right]\right] \tag{5.2}$$

In practice, at each epoch of the training, only one R is sampled for each G, which constitutes weak estimation of the expected value. Since at the end of the training, the network cannot have seen all the BFS orders, we rely on the generalization capacity of neural networks coupled with the strong proposed inductive bias to find relevant information in unseen roots and graphs.

Aggregation of the results at test time The node ordering step from random root introduces randomness in the classification. On the one hand, it helps learn more general graph representations during the training phase, but on the other hand, it might produce different outputs for the same graph during the testing phase, depending on the root of the BFS. In order to counter this side effect, we add the following aggregation step for the testing phase. For each graph we sample K subsets of N roots, noted $R_k = \{R \sim \mathcal{U}(\llbracket 1, n_G \rrbracket)\}_{i=1}^N$ (i.e. $\#R_k = N$). For each R_k, we compute an aggregated posterior

$$p_{\phi,\psi}(y|G,R_k) := \frac{1}{N}\sum_{R\in R_k} p_{\phi,\psi}(y|G,R)$$

This soft vote is repeated for each r_k, resulting in K probability vectors $\left\{p_{\phi,\psi}(y^G|G,R_k)\right\}_{k=1}^K$ for each graph G. The estimated class prediction for graph G is then:

$$y^G = \max_{k\in\llbracket 1,K\rrbracket} p_{\phi,\psi}(y|G,R_k)$$

The whole classification process is illustrated in Figure 5.3.

5.3 Regularization with autoregressive node prediction

For the moment, we have treated the problem of graph classification as a pipeline of two representation mechanisms: BFS and truncation, and recurrent neural networks. We have trained it as a fully supervised

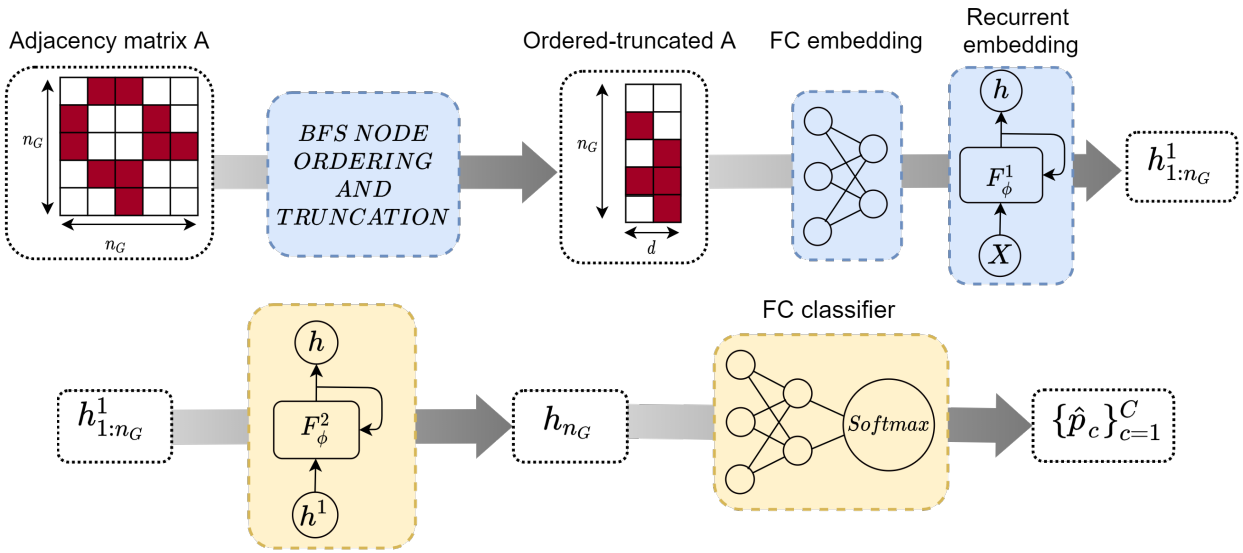


Figure 5.3 – Two-step RGC model. **Top**: node ordering and embedding. **Bottom**: scheme of the classification. $\{\hat{p}_c\}_{c=1}^C$ is the set of estimated membership probabilities, i.e. $\hat{p}_c = p_{\phi,\psi}(y = c|G, R)$.

representation problem. In this section, we propose to study the addition of a standard pretext-task for sequence embedding: the *one-step prediction*.

Principle To force each node embedding h_i to contain information about the graph, we add an *auto-regression* block to our model: at each node, the network makes a prediction for the next node adjacency. To do so, we propose to use a latent variable model $T_\theta = (T_{\theta_1}, T_{\theta_2})$ inspired from variational state-space model (see Section 2.4.2). For each graph G, a MLP T_{θ_1} with parameters θ_1 transforms each embedding h^1_{i-1} of node v_{i-1} into a low-dimensional stochastic Gaussian latent variable $Z_i = T_{\theta_1}(h^1_{i-1})$. A second MLP T_{θ_2} transforms Z_i into to initial representation h^0_{Ri} of node v_i. This block is a type of autoregressive VAE for node representation. We call is called *variational node auto-regression* (VNAR), that models the following distribution:

$$p(G, Z|h) = p(h^0_{R1}, Z_1) \prod_{i=2}^{n_G} p_{\theta_2}(h^0_{Ri}|Z_i) q_{\theta_1}(Z_i|h_{i-1}).$$

In practice, p_{θ_2} and q_{θ_1} are modelled by T_{θ_1} and T_{θ_2} respectively. Since h^0_{Ri} lives in $[0, 1]^d$, we use sigmoid output for T_{θ_2}. Training is done by maximizing the variational lower bound of the log-likelihood of the observation:

$$\mathcal{L}_{VAR} = \mathbb{E}_{G\sim\mathcal{X}}\left[\sum_{i=2}^{n_G} \mathrm{KL}\left(q_{\theta_1}(Z_i|h_{i-1})||p(Z_i)\right) - \sum_{i=2}^{n_G} \mathbb{E}_{q_{\theta_1}(Z_i|h_{i-1})}\left[\log p_{\theta_2}(h^0_{Ri}|Z_i)\right]\right] \tag{5.3}$$

which is a lower bound of the negative marginal log-likelihood $\mathbb{E}_{G\sim\mathcal{X}}[\log p_{\theta_2}(G)]$. We chose the standard Gaussian prior for $p(Z_i)$.

The regularization part is illustrated in Figure 5.4. In the end, the model is trained by minimizing the total loss

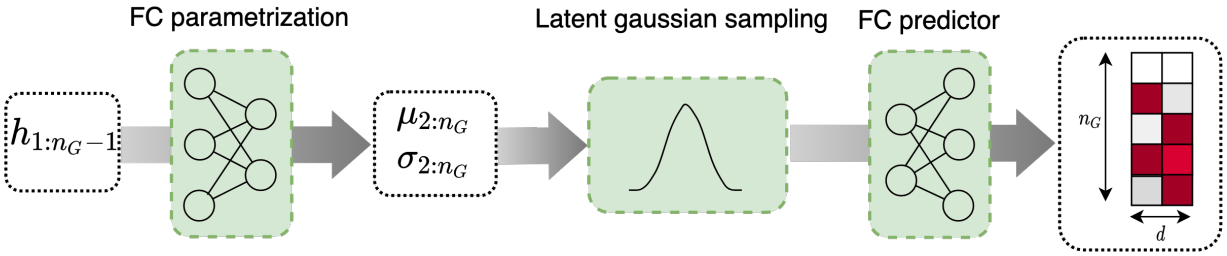


Figure 5.4 – Variational auto-regression of the nodes. Input $h_{1:n_G-1}$ is the output of node embedding part presented above.

$$\mathcal{L} = \mathcal{L}_{classif} + \alpha \mathcal{L}_{VAR}$$

with respect to parameters ϕ, ψ, θ_1 and θ_2.

Intuition Classification of graphs has similarity with classification of sentence. In fact, the embedding of a node is based on the embedding of its relation with other nodes of the graph. In NLP, standard word embedding are based on the relation between a word and its context, i.e. the co-ocurrence of neighboring words [Mikolov et al., 2013]. Moreover, multi-task learning is a common leverage to learn rich word and sentence representation [Sanh et al., 2018] in NLP. In particular, helping structural information extraction with auto-regression has been used for complex sequence classification tasks like sentiment analysis [Latif et al., 2017, Xu et al., 2017]. The sentiment analysis as the particularity to predict ambiguous labels from sequential discrete data, which is close to our graph classification problem. In NLP, the proposed regularization (5.3) is the equivalent to the prediction of the i^{th} word of a sentence, given an aggregated representation of this sentence up to word $i-1$.

5.4 Experiments

Experimental setup for classification of graphs For classification, we use the standard 10-folds train-test setup for model training and evaluation. Each dataset is divided into 10 folds such that the class proportions are preserved in each fold for all datasets. These folds are then used for cross-validation i.e, one fold serves as the testing set while the other ones compose the training set. Results are the mean and standard-deviation computed using the ten results from the ten folds tests.

Hyperparameters The input size d of the recurrent neural network is chosen for each dataset according to the algorithm described in [You et al., 2018], namely 11 for MT, 25 for EZ, 80 for PF and 11 for NCI1. All the other hyperparameters are shared between all datasets. We fix learning rate to 10^{-3} of the Adam stochastic gradient descent [Kingma and Ba, 2014].

Comparison with other models We compare our model [Pineau and de Lara, 2019] to different graph-classification methods picked among those presented in Section 2.5.2. Standard graph classification methods are: Earth Mover's Distance [Nikolentzos et al., 2017] (EMD), Pyramid Match [Nikolentzos et al., 2017]

(PM), Feature-Based [Barnett et al., 2016] (FB), Dynamic-Based Features [Gómez and Delvenne,] (DyF), Graphlet Kernel [Shervashidze et al., 2009] (GK) and family of graph spectral distances [Verma and Zhang, 2017] (FGSD). All of these methods represent graphs without supervision and then use support vector classifier (SVC) over extracted features. Deep learning methods are: Graph Convolutional Network [Kipf and Welling, 2016] (GCN), Deep Graph CNN [Zhang et al., 2018] (DGCNN), Capsule GNN [Xinyi and Chen, 2018] (Caps-GNN), Graph Isomorphism Network [Xu et al., 2018] (GIN) and GraphSAGE [Hamilton et al., 2017]. All deep learning methods are end-to-end graph classifiers (embedding function and classifier are trained concurrently. More details about these models are given in the introduction, Section 2.5.2.

The results are reported below. All values are directly taken from the aforementioned papers as we use a setup similar to their. For algorithms presenting results with and without node features, we reported the results without node features, to obtain comparable results. For those presenting results with several sets of hyper-parameters, we reported the results for the parameters that performed the best on the largest number of datasets. We note that contrary to our approach, GNN-based models use the node features when there are node features. For molecular graphs, we do not report GNN-based results since they use node features. For social networks, there are no node features; hence we can compare with GNNs.

Molecular graphs We use four datasets for the experiments: Mutag (MT), Enzymes (EZ), Proteins Full (PF) and National Cancer Institute (NCI1) [Kersting et al., 2016]. All graphs are chemical components. Nodes are atoms or molecules and edges represent checmical or electrostatic bindings. We note that molecular graphs contain node attributes, that are used by all neural networks based models. Hence for the molecular graphs, we only compare to non-neural methods. Description and statistics of molecular datasets are presented in Table C.1, Appendix C.

	MT	EZ	PF	NCI1
EMD	86.1 ± 0.8	36.8 ± 0.8	-	72.7 ± 0.2
PM	85.6 ± 0.6	28.2 ± 0.4	-	69.7 ± 0.1
FB	84.7 ± 2.0	29.0 ± 1.2	70.0 ± 1.3	62.9 ± 1.0
GK	81.7 ± 2.1	27.1 ± 0.8	71.7 ± 0.6	26.6 ± 1.0
DyF	86.3 ± 1.3	26.6 ± 1.2	73.1 ± 0.4	66.6 ± 0.3
FGSD	**92.1**	-	73.4	79.8
RGC	88.3 ± 7.9	48.7 ± 6.1	72.4 ± 3.1	78.3 ± 2.3
VRGC $\alpha = 0.1$	86.3 ± 8.6	48.4 ± 6.2	**74.8** ± 3.0	**80.7** ± 2.2
VRGC $\alpha = 1$	88.7 ± 8.9	**49.3** ± 6.1	73.8 ± 3.4	79.9 ± 1.9

Table 5.1 – Accuracy (%) of classification with different graph representations, on molecular graphs.

Social network graphs We use four datasets for the experiments: IMDB-Binary (IMBD-B), IMDB-Multi (IMDB-M), REDDIT-Binary (REDDIT-B) and COLLAB. All graphs are social networks. The graphs of these datasets do not contain node attributes. Therefore, we can more appropriately compare RGC and VRGC to GNN-based classification. We also compare with three standard methods for which we have the

figures. Statistics about social networks datasets are presented in Table C.2, Appendix C.

	IMDB-B	IMDB-M	REDDIT-B	COLLAB
GK	65.9 ± 1.0	50.6 ± 0.6	69.6 ± 0.9	77.8 ± 0.2
DyF	72.9 ± 4.1	48.1 ± 3.6	89.5 ± 2.0	80.6 ± 1.6
FGSD	73.6	52.4	86.5	80.0
GCN	74.0 ± 3.4	51.9 ± 3.8	-	79.0 ± 1.8
DGCNN	70.0 ± 0.9	47.8 ± 0.9	76.0 ± 1.7	73.8 ± 0.5
CapsGNN	73.1 ± 4.8	50.3 ± 2.7	-	79.6 ± 0.9
GIN-0	**75.1** ± 5.1	**52.3** ± 2.8	**92.4** ± 2.5	**80.2** ± 1.9
GraphSAGE	72.3 ± 5.3	50.9 ± 2.2	-	-
RGC	70.4 ± 2.8	47.8 ± 2.4	87.5 ±3.1	79.1 ± 1.5
VRGC ($\alpha = 0.1$)	71.4 ± 2.7	49.1 ± 2.6	89.7 ± 2.1	77.5 ± 1.9
VRGC ($\alpha = 1$)	71.2 ± 2.8	48.1 ± 2.6	88.9 ± 2.6	77.4 ± 1.0

Table 5.2 – Classification accuracy (%) of different deep learning based models plus ours over standard social networks datasets. Graphs of these datasets does not have node features.

Analysis of the results The classification results above illustrate the capacity of RGC to capture graph structural information from a greedy supervised learning. In molecular experiments, on which we compare with feature-based and kernel-based methods, we see that RGC and VRGC achieve or outperform state-of-art results. In particular, the advantage of using expressive embedding functions is important for the Enzymes classification that have 6 classes.

In social network experiments, we can compare RGC and VRGC with GNNs since these graphs have no node features. We see that GIN [Xu et al., 2018] outperforms, as expected (see Section 2.5.2). Our approach yet achieves the performance of other GNN architectures.

Remark 39. *We see that the relatively high standard-deviations impeach to have a clear view of the out-performance. We therefore only look at average score to appreciate it.*

Greedy learning of isomorphism invariance Our model is not inherently designed to create graph embedding that are independent from node ordering. Yet, the randomly rooted BFS performed on each graph at each epoch associated with a supervised end-to-end learning of a label invariant to the node-root forces the embedding F_ϕ to greedy learn an invariant representation of the graphs. Indeed, as illustrated in Figure 5.5, the projections corresponding to the same graphs form a heap in the low dimensional representation of the latent space, as expected. The greedy approach enables the network to learn the quasi-isomorphism rule $F_\phi(G) \approx F_\phi(G^\pi)$.

Contribution of the regularization to classification The variational regularization term seems to help the model finding a more meaningful latent representation for classification in almost all experiments.

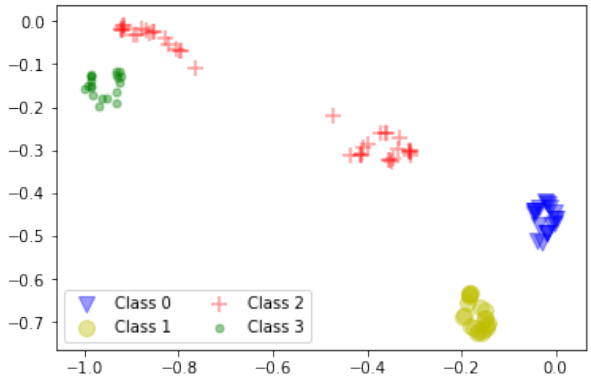


Figure 5.5 – TSNE projection of the latent state preceding classification for five graphs of EZ each initiated with 20 different BFS. Colors and markers represent the respective classes of the graphs.

Nevertheless, even when the average score is better with regularization, we cannot affirm that it generally contributes to a better classification, due to the very close scores and the relatively high standard-deviations.

Remark 40. *We note that the extra cost of training the regularization is marginal with respect to the training of the RNNs.*

5.5 Conclusion

The contribution of this chapter to supervised graph classification is the following: we can greedily learn with label's supervision a graph representation for classification that is almost invariant to isomorphism, from numerous iterations on randomly rooted BFS-ordered graph using powerful neural networks coupled with a strong inductive bias. We have seen that node-level additional task may enrich the classifier to estimate a better label's posterior distribution. Yet, the results remain unclear.

This work was a short break into the neural-network-based graph classification from the perspective of sequential data representation. Our results are encouraging and may need to pick new intuitions in NLP recent models and solutions developed for text and document embedding. Yet, we have the intuition that the salvation of useful and powerful graph representation and classification more likely lives in GNNs inductive bias (presented in Appendix A.1.1) rather than in sequential embedding.

Chapter 6

Laplacian spectrum for graph classification

Abstract In this chapter, we focus on two important inductive biases for unsupervised representation of graphs preceding a classification task: the *consistency* and the *isomorphism-invariance*. While state-of-the-art methods presented in the previous chapter seek such properties with powerful neural-networks trained end-to-end, we propose to look at the classification capacity of a known and simple graph feature that satisfies the aforementioned two key attributes: the Graph Laplacian Spectrum (GLS). To do so, we first derive bounds for the Euclidean distance between two GLS and show how it relates to the *divergence to isomorphism*, a standard computationally expensive graph divergence. We then experiment GLS as graph feature representation through consistency tests and classification tasks, and show that it can be a strong baseline for graph classification. This chapter covers a publication and a preprint:

- de Lara, N. and Pineau, E. (2018). A simple baseline algorithm for graph classification. *Workshop on Relational Representation Learning, at Advances in Neural Information Processing Systems (R2L, NIPS 2018)*.
- Pineau, E. (2019). Using Laplacian Spectrum as Graph Feature Representation. *Preprint*.

6.1 Introduction and intuition

We have seen in previous chapter a fully supervised graph classification where a recurrent neural network learns to extract and arrange graph-level features while a linear classifier learns to discriminate their underlying class.

Here we are interested in the case where an unsupervised graph representation is learned with the objective to feed a classifier as a downstream task. In this situation, we need to define features that have certain properties that match natural classification inductive biases.

First, we need to find a feature that is invariant to non-significant factors underlying data. In particular, as seen in previous chapter, for graph-level tasks we need *isomorphism-invariance* (see definition in Section 2.5). Second, we need to have separate features for graphs that live in the same class. Since we do not

know how a class is characterized in graph domain and since we do not learn explicitly the characterization (unsupervised representation), we use the following inductive bias: graphs in the same class have similar structures, i.e. close adjacency matrices up to permutation of node index and addition of nodes and edges.

Remark 41. *This assumption is relevant since we work on graphs that belong to consistent datasets where graphs have similar properties (e.g. datasets of comparable proteins).*

In this chapter, based on the intuitions given below, we focus on the spectrum (eigenvalues) of a particular matrix associated to graphs: the Laplacian matrix (see Section 2.5). In particular, we show that Graph Laplacian Spectrum (GLS) respects the two aforementioned inductive biases and propose experiments to illustrate its representation power for graph classification.

Why Laplacian sepctrum? The initial intuition of using GLS as graph feature representation for graph classification is based on known properties of GLS. First, the GLS is invariant to isomophism (see a proof in Appendix B.6). Second, the Laplacian eigenvalues give many structural information like the presence of communities and partitions, the regularity, the closed-walks enumeration, the diameter or the density of the graph [Brouwer and Haemers, 2011, Newman, 2013]. GLS is also interpretable in term of graph signal processing [Shuman et al., 2016] or mechanics [Bonald et al., 2018]. Hence, GLS can be used in several contexts to infer information about graphs. In [Wilson and Zhu, 2008], they propose an experimental approach to show that comparing graphs using their GLS (or more generally the spectrum of their adjacency and the spectrum of the normalized Laplacian) is meaningful.

This chapter extends this latter work by showing why GLS is a candidate feature that is relevant for classification, with respect to the aforementioned two inductive biases for graph classification, through the following contributions:

- We analyze the consistency between structural deformation of the graph and its GLS by deriving bounds for the distance between the GLS of two graphs, using a ad-hoc perturbation-based framework
- We validate the consistency, the classification power and reasonableness of truncating the GLS (to solve the variable-size problem) on synthetic and real graphs

The whole graph classification with GLS is illustrated in Figure 6.1.

The rest of the chapter is built as follows. A presentation of the mathematical framework and the analysis of GLS are displayed in Section 6.2. Section 6.3 exposes the experiments. Finally, after the conclusion in Section 6.4, the Section 6.5 gathers the proofs of different results of the chapter.

6.2 Analysis of the relevance of GLS for classification

6.2.1 Presentation of the perturbation-based framework

We consider two undirected and weighted graphs $G_1 = (V_1, E_1, W_1)$ and $G_2 = (V_2, E_2, W_2)$ with respective adjacency matrix W_1 and W_2, degree matrix D_1 and D_2. These matrices are set with respect to an arbitrary indexing of the nodes. We remind that the Laplacian matrix L_i of G_i is defined as $L_i = D_i - W_i$. We aim at using the GLS to build fixed-dimensional representation that encodes structural information to compare

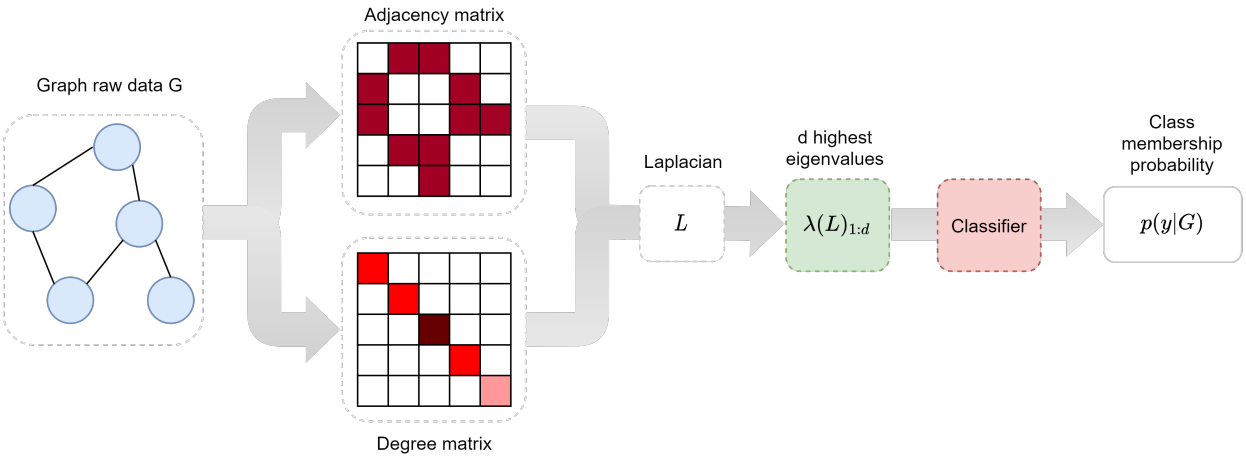


Figure 6.1 – Classification procedure using truncated GLS (t-GLS)

any graphs G_1 and G_2 (i.e. that are not aligned nor equally sized). For the rest of the paper, and without loss of generality we postulate that $|V_1| \leq |V_2|$. The rest of this section introduces the definitions, hypothesis and notations needed for our theoretical analysis of the GLS.

Definition 7. *Let $G = (V, E, W)$ an undirected weighted graph with n nodes, with $W \in \mathcal{M}_{n\times n}$ the $n \times n$ weighted adjacency matrices. We define $P \in \mathcal{M}_{n\times n}$ a symmetric matrix with $P_{ii} = 0$, $P_{ij} \in [-W_{ij}, W_{ij}]$ such that $W_{ij} + P_{ij} \in [0, 1]\, \forall (i, j)$. We define the two following perturbations applied on graph G:*

- *Adding isolated nodes:* $\overline{W} = \begin{bmatrix} W & 0_{n\times m} \\ 0_{m\times n} & 0_{m\times m} \end{bmatrix}$
- *Adding or removing edges:* $W^P = W + P$

We call edge-perturbation the addition or removal of edges, and node-perturbation the addition of nodes. A complete perturbation is done by adding isolated nodes (i.e. without incident edges) and perturbing the augmented graph with edge addition or removal. If graph G is unweighted, i.e. with binary adjacency, then edge perturbations $P_{ij} \in \{-1, 0, 1\}$.

From Definition 7 the importance of the perturbation is contained in P. The intuition is that the denser P, the higher the difference between initial and perturbed graphs. We note L_P the Laplacian of P. We remind that the permutation of node indices is not a relevant perturbation. Hence, we have the following definition:

Definition 8. *We say that G^{P^*} is a perturbed version of G if we have*

$$\begin{cases} W^{P^*} = {\Pi^*}^T(\overline{W} + P^*)\Pi^* \\ s.t.\ \Pi^* = \arg\min_{\Pi\in\mathcal{P}(n)} \|W^{P^*} - \Pi^T\overline{W}\Pi\|_1 \end{cases}$$

i.e. such that P^* is the sparsest possible i.e. does not include permutations.

Notations We note P^* *the sparsest perturbation* as defined in Definition 8. We note $\overline{G}$ the completion of G with isolated nodes. If M is a matrix associated to G, we note $\overline{M}$ the equivalent matrix for $\overline{G}$. We note $\lambda(X)$ the eigenvalue of a square matrix X in ascending order, $\lambda_i(X)$ the i^{th} smallest eigenvalue.

Remark 42. *In this part, the representation function F is the eigenvalue algorithm applied on the graph Laplacian. Hence, F is the composition of $G \mapsto L$ and λ. We note that λ can be any eigenvalue algorithm. Since we work on real symmetric matrices, we can use for example the Jacobi eigenvalue algorithm for F. In this chapter, we do not assume that the graph representation unveils generative factors, only interesting properties.*

From the introduced concepts above, we can propose the following assumption on which we will build our study:

Hypothesis 2. *Without loss of generality, we assume that G_2 is a perturbed version of G_1, i.e. there exist P^* the $|V_2|$-square sparsest perturbation matrix associated with a permutation matrix $\Pi^* \in \mathcal{P}(|V_2|)$ such that $W_2 = {\Pi^*}^T \left(\overline{W_1} + P^*\right) \Pi^*$.*

We have defined a notion of smooth consistency between graphs that has a natural and simple interpretation: any graph G_2 is a perturbed version of graph G_1, and the larger the perturbation the higher the structural dissimilarity between G_1 and G_2. Under classification inductive bias cited above, a smaller perturbation between two graphs increases the probability that they are in the same class.

Yet, finding P^* or its norm for all pairs of graphs of a dataset is not tractable. In the next section, we use the previously presented mathematical framework to show that the GLS can help solving this intractable problem.

6.2.2 Analysis

We place ourselves under the Hypothesis 2 saying that the difference between graphs G_1 and G_2 is characterized by the unknown deformation P^*. A good embedding of these graphs should be close when level of deformation is low, and far otherwise. This level of deformation can be quantified by the structure of P^*, which can be represented by its Laplacian noted L_{P^*}.

We use this idea to propose an analysis of the distance between two GLS. All proofs are detailed in Section 6.5.

6.2.2.1 Consistency under deformation and relation to graph isomorphism

We remind that two graphs G_1 and G_2 are isomorphic if and only if $\exists \Pi \in \mathcal{P}(|V_1|)$ such that $L_2 = \Pi^{-1} L_1 \Pi$ [Merris, 1994], hence when they are structurally equivalent irrespective to the vertex ordering. Several papers has proposed to use a notion of *divergence to graph isomophism* (DGI) to compare graphs [Grohe et al., 2018, Rameshkumar et al., 2013]. The DGI between graphs G_1 and G_2 is generally $\min_\Pi \left\| L_1 - \Pi^{-1} L_2 \Pi \right\|_F$. Considering this definition, the following Lemma links the graph-isomorphism problem and the Laplacian of the hypothetical perturbation P^*:

Lemma 2. *Using the notations from Hypothesis 2, we have $L_2 = {\Pi^*}^T \left(\overline{L_1} + L_{P^*}\right) \Pi^*$, with $L_{P^*} = diag(P^* \mathbf{1}_{|V_1|}) - P^*$ the Laplacian of P^* and $\mathbf{1}_n$ the n-dimensional unit vector. In particular, $\min_\Pi \|L_2 - \Pi^T \overline{L_1} \Pi\|_F = \|L_{P^*}\|_F$.*

We remind that the DGI is known to be NP-hard [Grohe et al., 2018]. The Propositions 2 and 3, and the different comments, relate the distance between GLS to the DGI and show the interest of GLS for whole-graph comparison hence classification.

Proposition 2. *Using Hypothesis 2 and Lemma 2:* $\|\lambda(L_2) - \lambda(\overline{L_1})\|_2 \leq \|L_{P^*}\|_F$.

The above result tells us that the higher the difference between GLS, the larger the hypothetical perturbation P^* i.e. the higher the structural dissimilarity.

The implication of GLS closeness is less clear, since it tackles the notion of *non-isomorphic L-cospectrality*, which is the idea that two graphs can have equal eigenvalues while having different Laplacian matrix [Brouwer and Haemers, 2011]. In fact, there exist families of graphs that are not fully determined by their spectrum, like trees [Schwenk, 1973].

Yet, especially for graph classification, non-isomorphic cospectrality is not necessarily a problem. First, almost all graphs are determined by their spectrum [Brouwer and Haemers, 2011, Haemers, 2016]. Second, when two graphs are non-isomorphic cospectral, it means that they share certain intrinsic properties [Shuman et al., 2016]. For example, we have mentioned that eigenvalues are related to properties of the signal living on a graph (for example the edge weights). Hence, equal GLS means equal properties of the signalμ. We note that the signal can be the weights of the graph. If the class label associated to graphs is related to the signal living on it, which is not rare, the cospectrality is meaningful.

Nevertheless, the above intuition is not fully satisfying. We accordingly propose the Proposition 3 to better understand GLS proximity even when graphs are non-isomorphic cospectral.

Proposition 3. *The closer the GLS, the closer to unitary-similarity the Laplacian matrices.*

We remind that two real n-square matrices A and B are *unitary-similarity* if there exists an orthogonal matrix O such that $B = OAO^T$. Similarity is an *equivalence relation* on the space of square-matrices. Moreover, divergence to unitary-similarity defined by $\min_{O\in\mathcal{O}(|V_2|)}\|\overline{L_1} - OL_2O^T\|_F$ is a relaxed version of the divergence to graph-isomorphism [Grohe et al., 2018], where the permutation matrix space is replaced by a unitary matrix space. Finally from Proposition 2 and 3 we can bound the distance between GLS as follows:

$$\min_{O\in\mathcal{O}(|V_2|)}\|\overline{L_1} - OL_2O^T\|_F \leq \|\lambda(\overline{L_1}) - \lambda(L_2)\|_2 \leq \|L_{P^*}\| \tag{6.1}$$

In this section, we have shown that structural similarity (divergence) between graphs can be reasonably approximated by the similarity (divergence) between their GLS, at least for graph-level problems in which the relevant similarity is related to structural similarity between graphs, like for example graph classification.

6.2.2.2 Some practical aspects

Previous section showed the capacity of the distance between Laplacian spectrum to serve as proxy for graph similarity. We note that the computation of spectrum is backed by efficient and robust approximate eigen-decomposition algorithms enabling to scale on large graphs and datasets [Halko et al., 2011].

Another problem is the variable size of the graphs, since a graph G with n_G nodes has a GLS of size n_G. In practice, a fixed embedding dimension d must be chosen for all graphs in dataset $\mathcal{D}$. According to

previous analysis, the most obvious dimension is $d = \max_{G \in \mathcal{D}} |V|$ and all graphs with less than d nodes may be padded with isolated nodes. We note that padding with isolated nodes is equivalent to adding zeros in the GLS. Nevertheless, in some datasets, some graphs can be significantly larger and the padding can become abusive. We therefore propose for these graph to have $d < \max_{G \in \mathcal{D}} |V|$. We simply truncate the GLS such that we keep only the highest d eigenvalues. This method also enables to save computation time with efficient truncated singular value decomposition [Yuan and Zhang, 2013].

The problem with this method is that we may lose information for graphs with more than d nodes. Yet, in practice, for large graphs, the contribution of the lowest eigenvalues to the distance between GLS as a proxy for graph divergence is negligible. In particular, large graph have many sparse areas, such that many eigenvalues are very low, hence truncating the bottom part of the GLS is usually not a problem. We assess the impact of the truncation in the experimental section.

Remark 43. *We note that we can also propose several ways to limit information loss for truncated GLS (t-GLS), like low-dimensional embedding of the lowest eigenvalues with moments or histograms for example.*

6.3 Experiments

6.3.1 Preliminary experiments

As a first illustration of deformation-based results presented in Section 6.2.2, we sample a random graph from Erdos-Rényi model [Erdős and Rényi, 1959], with parameter $n_G = 80$ and $p = 0.05$ (graph randomly chosen among the set of graphs with a certain number of nodes n_G and a certain edge density p). On that random graph, we do two simple illustrative experiments.

First, the distance between the Laplacian spectrum of a graph and a perturbed version of this graph is related to the number of perturbations. We can find the experimental illustration in Figure 6.2 (similar to those in [Wilson and Zhu, 2008]). We see that the number of perturbations is directly related to the distance between GLS features for edge addition and edge withdrawal.

Remark 44. *A relation between graph sparsity and Laplacian eigenvalues can be seen for example through the Gershgorin circle theorem [Gershgorin, 1931].*

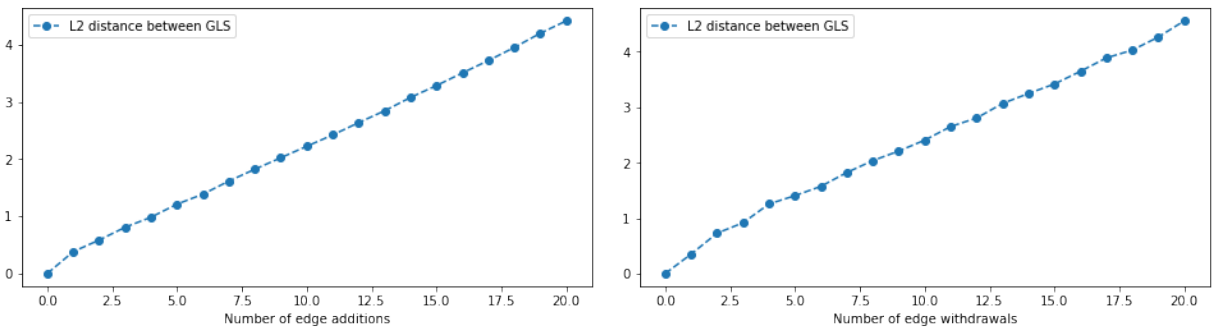


Figure 6.2 – Experimental results to illustrate how GLS behaves under edge addition and withdrawal. **Left**: edge addition. **Right**: edge withdrawal. In this case, studied adjacency and perturbation matrix are binary.

Second, we mentioned that when a graph is significantly bigger than other graphs of a dataset, we can use a truncated GLS (t-GLS). In Figure 6.4, we iteratively add 20 nodes with various random connections to a random Erdos-Rényi graph G. We keep the same dimensionality n_G for all the t-GLS at each iteration. We illustrate the fact that the t-GLS is consistent with node addition and the intensity of the connection of the new nodes. We do the same experiment with a real molecular graph from MUTAG dataset (see Appendix C.3).

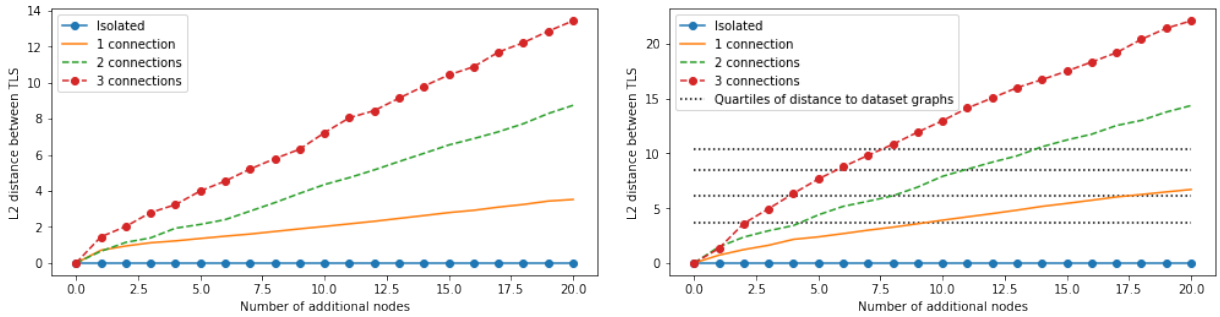


Figure 6.3 – Experimental results illustrate how t-GLS behaves under iterative addition of 20 new nodes with respectively 0, 1, 2 and 3 random connections. **Left**: synthetic a 80-nodes Erdos-Rényi graph. **Right**: a 28-nodes molecular graph from MUTAG dataset. Horizontal dotted lines (right figure) are the quartiles 25, 50, 75 and 100 of the distances between the GLS of the 28-nodes graph and the other 187 graphs of the dataset. The curves are computed as the l^2-norm between each t-GLS and the initial GLS.

6.3.2 Classification of molecular and social network graphs

We evaluate GLS for classification on molecular graphs and social network graphs. We remind the inductive bias: two structurally close graphs belong to the same class. We know that GLS is consistent with structure deformation. We now challenge the assumption and previous results with the following classification experiments.

Experimental setup for classification of graphs Like for experiments of Chapter 5, we use the standard 10-folds train-test setup for model training and evaluation. Each dataset is divided into 10 folds such that the class proportions are preserved in each fold for all datasets. These folds are then used for cross-validation i.e, one fold serves as the testing set while the other ones compose the training set. Results are the mean and standard-deviation computed using the ten results from the ten folds tests.

Hyperparameters We use the support vector classifier (SVC) from scikit-learn [Pedregosa et al., 2011]. We chose Radial Basis Function as kernel, i.e. $\mathcal{K}(\lambda(\overline{L_1}), \lambda(L_2)) = \exp\left(-\gamma \|\lambda(\overline{L_1}) - \lambda(L_2)\|_2^2\right)$ since directly related to Euclidean distance between GLS for which we have bounds (6.1). Hence, our theoretical results remain consistent with our experiments. Hyper parameters C and γ are searched at each fold training among respectively $\{0.5, 1, 5\}$ and $\{0.0001, 0.001, 0.01, 0.1, 0.5, 1, 5\}$ for the molecular datasets, and $\{0.5, 1, 5, 25, 50\}$ and $\{0.0001, 0.001, 0.01, 0.1\}$ for the social network datasets (in practice, using a global pool for all the datasets gives equivalent results, but hyperparameter inference becomes expensive with a too large grid, in

particular when repeated ten times). The nested hyperparameter search works as follows: in each training fold we perform a 5-fold random search cross-validation. We therefore avoid the problem of overfitting related to model selection that appear when using non-nested cross-validation [Cawley and Talbot, 2010].

For the dimension $d \in [\![1, \max_{G \in \mathcal{D}} n_G]\!]$, representing the number of eigenvalues we keep to build the t-GLS, we chose the percentile 95 of the distribution of graph sizes in each dataset. A study of the impact of the truncation is given in paragraph *On the reasonableness of using t-GLS* (Section 6.3.2).

Comparison with other models We compare GLS+SVC with the same models that in previous chapter, presented in Section 2.5.2. We also add the results of our graph classification method presented in Chapter 5.

Molecular graphs We use four datasets for the experiments: Mutag (MT), Enzymes (EZ), Proteins Full (PF) and National Cancer Institute (NCI1) [Kersting et al., 2016]. All graphs are chemical components. Nodes are atoms or molecules and edges represent checmical or electrostatic bindings. We note that molecular graphs contain node attributes, that are used by all neural networks based models. Hence for the molecular graphs, we only compare to non-neural methods. Description and statistics of molecular datasets are presented in Table C.1, Appendix C.

	MT	EZ	PF	NCI1
EMD + SVC	86.1 ± 0.8	36.8 ± 0.8	-	72.7 ± 0.2
PM + SVC	85.6 ± 0.6	28.2 ± 0.4	-	69.7 ± 0.1
FB + SVC	84.7 ± 2.0	29.0 ± 1.2	70.0 ± 1.3	62.9 ± 1.0
DyF + SVC	86.3 ± 1.3	26.6 ± 1.2	73.1 ± 0.4	66.6 ± 0.3
FGSD + SVC	**92.1**	-	73.4	79.8
RGC	88.3 ± 7.9	48.7 ± 6.1	72.4 ± 3.1	78.3 ± 2.3
VRGC $\alpha = 0.1$	86.3 ± 8.6	48.4 ± 6.2	74.8 ± 3.0	**80.7** ± 2.2
VRGC $\alpha = 1$	88.7 ± 8.9	**49.3** ± 6.1	73.8 ± 3.4	79.9 ± 1.9
GLS + SVC	87.9 ± 7.0	40.7 ± 6.3	**75.3** ± 3.5	73.3 ± 2.1

Table 6.1 – Accuracy (%) of classification with different graph representations, on molecular graphs. SVC stands for support vector classifier. Comparative models are divided into two groups: feature + SVC and end-to-end deep learning. *Models using node attributes.

Social network graphs We use five datasets for the experiments: IMDB-Binary (IMBD-B), IMDB-Multi (IMDB-M), REDDIT-Binary (REDDIT-B) and COLLAB. All graphs are social networks. The graphs of these datasets do not contain node attributes. Therefore, we can more appropriately compare GLS + SVC to deep learning based classification. Statistics about social networks datasets are presented in Table C.2, Appendix C.

Analysis of the results The classification results above illustrate the capacity of GLS to capture graph structural information, under the assumption that structurally close graphs belong to the same class. In

	IMDB-B	IMDB-M	REDDIT-B	COLLAB
GK	65.9 ± 1.0	50.6 ± 0.6	69.6 ± 0.9	77.8 ± 0.2
DyF	72.9 ± 4.1	48.1 ± 3.6	89.5 ± 2.0	80.6 ± 1.6
FGSD	73.6	52.4	86.5	80.0
GCN	74.0 ± 3.4	51.9 ± 3.8	-	79.0 ± 1.8
DGCNN	70.0 ± 0.9	47.8 ± 0.9	76.0 ± 1.7	73.8 ± 0.5
CapsGNN	73.1 ± 4.8	50.3 ± 2.7	-	79.6 ± 0.9
GIN-0	**75.1** ± 5.1	**52.3** ± 2.8	**92.4** ± 2.5	**80.2** ± 1.9
GraphSAGE	72.3 ± 5.3	50.9 ± 2.2	-	-
RGC	70.4 ± 2.8	47.8 ± 2.4	87.5 ±3.1	79.1 ± 1.5
VRGC ($\alpha = 0.1$)	71.4 ± 2.7	49.1 ± 2.6	89.7 ± 2.1	77.5 ± 1.9
VRGC ($\alpha = 1$)	71.2 ± 2.8	48.1 ± 2.6	88.9 ± 2.6	77.4 ± 1.0
GLS + SVC	73.2 ± 4.2	48.5 ± 2.5	87.4 ± 3.4	78.5 ± 1.1

Table 6.2 – Classification accuracy (%) of different deep learning based models plus ours over standard social networks datasets. Graphs of these datasets does not have node features. SVC stands for support vector classifier.

molecular experiments, on which we compare with feature-based and kernel-based methods, we see that GLS is a strong baseline that achieves or outperform state-of-art results. Yet, we see that the results achieved by the our model RGC (only NN-based graph classifier that do not use node features) presented in Chapter 5 still outperform thanks to its high expressiveness.

In social network experiments, we can compare GLS+SVC with GNNs since these graphs have no node features. We see that GIN [Xu et al., 2018] outperforms, as expected (see Section 2.5.2). GLS yet achieves the performance of other GNN architectures.

These experiments, we can say that GLS is a simple way to represent graphs in an unsupervised manner, with theoretical background, simplicity of implementation (there exist many efficient implementation of eigenvalues inference) and competitive downstream classification results.

Remark 45. *We see that the relatively high standard-deviations impeach to have a clear view of the outperformance of certain models against each others. We therefore only look at average score to appreciate the results.*

On the reasonableness of using truncated GLS We assess the impact of truncating the GLS. We remind that using t-GLS enables to reduce the computational cost for large graphs and reduce the dimensionality of the graph representation for all graphs (Section 6.2.2.2). Results are presented in Figure 6.4 for molecular datasets.

We see that truncating GLS is not highly impacting classification results. Only ENZYMES multi-class classification, which is a particularly difficult task (see experiments in Section 6.3.2), suffers from truncation.

Figure 6.5 illustrates the reasonableness of using only the highest eigenvalues of the Laplacian spectrum as whole-graph feature representation. We take the original and final graphs of the deformation-consistency

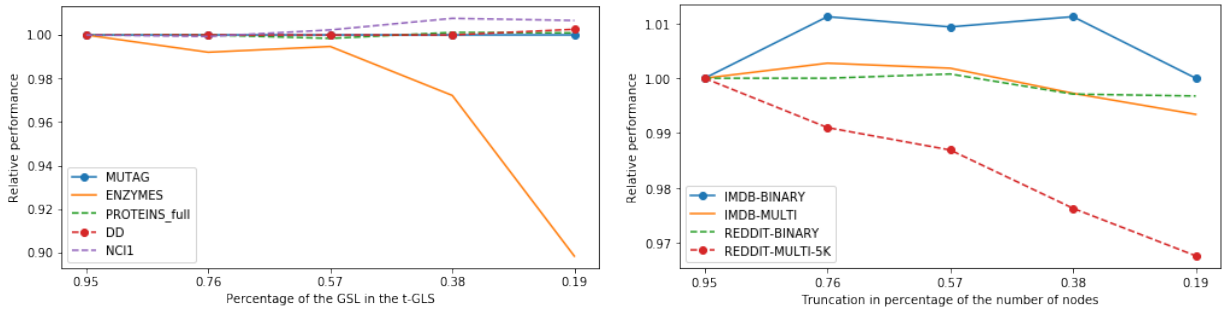


Figure 6.4 – Illustration of the impact of the truncation in term of classification accuracy. **Left**: molecular graphs. **Right**: social networks. We represent the impact relatively to the 95-percentile truncation adopted for classification experiments.

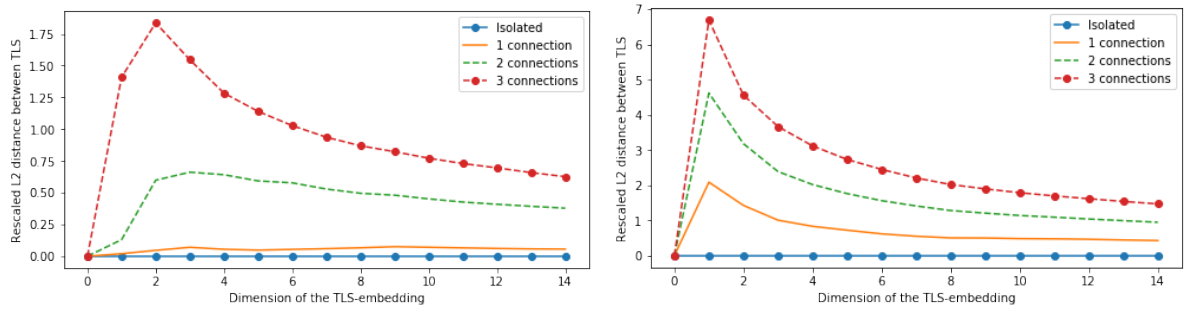


Figure 6.5 – Illustration of the relative importance of the dimensionality of GLS-embedding, after the iterative addition of 20 new nodes with respectively 0, 1, 2 and 3 random connections with graph. **Left**: synthetic a 80-nodes Erdos-Rényi graph. **Right**: a 28-nodes molecular graph from MUTAG dataset. We see that the first largest eigenvalues of the Laplacian are the most important to discriminate a graph and its perturbed version.

test presented in Figure 6.4. We compute the L_2 distance between t-GLS with dimension d and divide it by d, for d varying from 1 to 15. The objective is to confirm that first eigenvalues are relatively more important to discriminate to structurally different graphs, which is the case. We note that for the Erdos-Rényi case with few connected additional nodes, first eigenvalues are not as relatively important as for the other example. In fact, adding nodes with stochastic connections is the construction process of Erdos-Rényi graphs. Hence, discriminating augmented graph from the original one is difficult based only on the structural information.

6.4 Conclusion

In this chapter, we analyzed the graph Laplacian spectrum (GLS) as whole graph representation for graph classification. We showed that comparing two GLS is a good proxy for the divergence between two graphs in terms of structural information. We also showed that truncating the GLS to align graphs with different sizes is acceptable. We coupled these results with the simplicity of implementation, the computational efficiency offered by modern randomized eigenvalues algorithms and the rare occurrence of detrimental L-cospectral non-isomorphic graphs to propose the GLS as a strong baseline for graph classification.

6.5 Proofs

This section gathers the proofs that complete the results below.

6.5.1 Proof of Lemma 2

$$\begin{aligned}
L_2 &= D_2 - W_2 \\
&= \mathrm{diag}(W_2\mathbf{1}) - W_2 \\
&= \mathrm{diag}({\Pi^*}^T\left(\overline{W_1} + P^*\right)\Pi^*\mathbf{1}) - \Pi^*\left(\overline{W_1} + P^*\right)\Pi^* \\
&= \mathrm{diag}({\Pi^*}^T\overline{W_1}\Pi^*\mathbf{1}) + \mathrm{diag}({\Pi^*}^T P^*\Pi^*\mathbf{1}) - {\Pi^*}^T\overline{W_1}\Pi^* - {\Pi^*}^T P^*\Pi^* \\
&= {\Pi^*}^T\overline{D_1}\Pi^* - {\Pi^*}^T\overline{W_1}\Pi^* + {\Pi^*}^T D_{P^*}\Pi^* - {\Pi^*}^T P^*\Pi^* \\
&= {\Pi^*}^T\overline{L_1}\Pi^* + {\Pi^*}^T L_{P^*}\Pi^*
\end{aligned}$$

with $L_{P^*} = \mathrm{diag}(P^*\mathbf{1}) - P^* = D_{P^*} - P^*$ and $\mathbf{1}$ the unit vector.

Therefore,

$$\min_{\Pi}\|L_2 - \Pi^T\overline{L_1}\Pi\|_F = \min_{\Pi}\|\Pi^T L_{P^*}\Pi\|_F = \|L_{P^*}\|_F$$

6.5.2 Proof of Proposition 2

From lemma 2 we have $L_2 = {\Pi^*}^T\overline{L_1}\Pi^* + {\Pi^*}^T L_{P^*}\Pi^*$. Moreover, from Weyl's eigenvalues inequalities and since eigenvalues are isomorphism invariant:

$$\begin{cases}
\lambda_i(L_2) \leq \lambda_i(\overline{L_1}) + \lambda_{|V_2|}(L_{P^*}) \\
\lambda_i(\overline{L_1}) + \lambda_1(L_{P^*}) \leq \lambda_i(L_2)
\end{cases}$$

Hence: $\lambda_1(L_{P^*}) \leq \lambda_i(L_2) - \lambda_i(\overline{L_1}) \leq \lambda_{|V|}(L_{P^*})$.

Now let (λ, x) be any eigen couple of a matrix $M \in \mathcal{M}_{n\times n}$. We can always pick $i \in \{1 \dots n\}$ and build x such that $|x_i| = 1$ and $|x_{j\neq i}| < 1$. Hence:

$$\begin{aligned}
(Mx)_i = \lambda x_i &\iff \sum_{j=1}^{n} m_{ij}x_j = \lambda x_i \\
&\iff \lambda^2 \leq \sum_{j=1}^{n}(m_{ij}x_j)^2 \\
&\implies \lambda^2 \leq \sum_{j=1}^{n}(m_{ij})^2 \\
&\implies \lambda^2 \leq \frac{1}{n}\sum_{i,j=1}^{n} m_{ij}^2
\end{aligned}$$

Using previous results we get:

$$\sum_{i=1}^{|V_2|}\left(\lambda_i(L_2) - \lambda_i(\overline{L_1})\right)^2 \leq |V_2|\frac{1}{|V_2|}\sum_{i,j=1}^{|V_2|} {L_{P^*}}_{ij}^2 = \|L_{P^*}\|_F^2,$$

with $\|X\|_F = \sqrt{\sum_i \sum_j |X_{ij}|^2}$ the Frobenius norm.

6.5.3 Proof of Proposition 3

Denoting $\mathcal{O}(n)$ the n-orthogonal matrices group (orthogonal since real), we want to show that:

$$\min_{O \in \mathcal{O}(|V_2|)} \|\overline{L_1} - O^T L_2 O\|_F \leq \|\lambda(\overline{L_1}) - \lambda(L_2)\|_2$$

We note $L_i = Q_i \Lambda_i {Q_i}^T$ the eigendecomposition of the Laplacian L_i with $\Lambda_i = \text{diag}(\lambda(L_i))$. Since $\overline{Q_1}$ is unitary and using property of Frobenius norm, we have, $\forall O \in \mathcal{O}(|V_2|)$:

$$\|\overline{L_1} - O^T L_2 O\|_F = \|\overline{\Lambda_1} - \overline{Q_1}^T O^T Q_2 \Lambda_2 {Q_2}^T O \overline{Q_1}\|_F,$$

We know that $\overline{Q_1}$ and Q_2 are orthogonal since they are respectively eigenvector matrices of symmetric matrix $\overline{L_1}$ and L_2. We therefore have:

$$(Q_2^T O \overline{Q_1})^T (Q_2^T O \overline{Q_1}) = \overline{Q_1}^T O^T Q_2 Q_2^T O \overline{Q_1} = I_{|V_1|}$$

Moreover $\forall \tilde{\Pi} \in \mathcal{P}(|V_2|) \subset \mathcal{O}(|V_2|)$, if $O = Q_2 \tilde{\Pi} \overline{Q_1}^T$ then $O \in \mathcal{O}(|V_2|)$.

Hence,

$$\begin{aligned}
\min_{O \in \mathcal{O}(|V_2|)} \|\overline{L_1} - O^T L_2 O\|_F &= \min_{O \in \mathcal{O}(|V_2|)} \|\overline{\Lambda_1} - \overline{Q_1}^T O^T Q_2 \Lambda_2 {Q_2}^T O \overline{Q_1}\|_F \\
&\leq \min_{\tilde{\Pi} \in \mathcal{P}(|V_2|)} \|\overline{\Lambda_1} - \overline{Q_1}^T (Q_2 \tilde{\Pi} \overline{Q_1}^T)^T Q_2 \Lambda_2 {Q_2}^T (Q_2 \tilde{\Pi} \overline{Q_1}^T) \overline{Q_1}\|_F \\
&= \min_{\tilde{\Pi} \in \mathcal{P}(|V_2|)} \|\overline{\Lambda_1} - \tilde{\Pi}^T \Lambda_2 \tilde{\Pi}\|_F \\
&= \min_{\tilde{\Pi} \in \mathcal{P}(|V_2|)} \|\overline{\Lambda_1} - \tilde{\Pi} \Lambda_2 \tilde{\Pi}^T\|_F \\
&= \min_{\tilde{\Pi} \in \mathcal{P}(|V_2|)} \|\overline{\Lambda_1} - \tilde{\Pi} \Lambda_2 \tilde{\Pi}^T\|_F \\
&= \min_{\sigma \in \mathcal{S}(|V_2|)} \|\lambda(\overline{L_1}) - \lambda(L_2)_{\sigma(1:|V_2|)}\|_2 \\
&\leq \|\lambda(\overline{L_1}) - \lambda(L_2)\|_2
\end{aligned}$$

with $\mathcal{S}(n)$ the permutation group of $\{1 \dots n\}$.

Chapter 7

Conclusion

In this thesis, we have explored several inductive biases and methods for the representation of multivariate time series (MTS) and graphs. The goal was twofold. First, contribute to the representation of MTS and graphs problem. Second, propose ideas and tools to help data analysts of Safran to build health monitoring (HM) solutions for turbine engines.

Chapter 3 proposed a consistent model-based framework to represent MTS data, in particular MTS samples recorded on mechanical systems. The principle was the following: we substitute the true unknown physical model describing the data dynamics by a simple statistical model whose first parameter controls the causality between observed variables. We trained a neural inference function to infer these parameters, using an encoder-decoder framework. We claimed and showed that the inferred MTS representations were relevant for monitoring the state of mechanical systems (a Newtonian system and a turbine engine).

Chapter 4 proposed a simple yet powerful and universal model called *contrastive trend extraction* (CTE) to extract the trend signal from data with generic neural networks (NNs), without assumption on the shape of the model underlying the data. Trend extraction is interesting under the assumption that a mechanical system ages monotonously. Hence, searching monotonous trends in data is equivalent to searching ageing signal. We proposed both experimental and theoretical results to explain in which extent this approach is relevant. We also showed that CTE is related to another interesting topic for Safran: the *survival analysis*. In particular, besides the exhibition of the relation, we used our CTE model on survival analysis datasets and found out that it achieves state-of-the-art results. We finally proposed an analysis of the impact of the noise on the trend detection quality.

Both chapters illustrated the interest of using adapted inductive biases to learn relevant MTS representation, in particular for ageing signal extraction for health monitoring.

A natural extension of these works, in which we developped methods to search particular signals underlying data under generative assumption (e.g. ageing), uses recent literature about blind source separation (BSS). This literature proposes new identifiability results for data decomposition into generative factors that can be applied to time series data. The idea is the following: with appropriate representation tools and inductive biases, we can extract the *true* factors that generated the data, in which we might find the ageing signal but also other interesting information about the studied mechanical system. The preliminary work of such perspective has been studied and presented as a contribution in [Pineau et al., 2020a], not sufficiently

advanced for a full chapter in this thesis report but described in the experiment part of Chapter 4.

Several additional perspectives exist that are both interesting for ML community and Safran. First, we may have specific information about the monitored mechanical system, for example physical model, causal rules, etc. How can we integrate these information within a statistical data representation model to obtain more meaningful, interpretable or robust monitoring indicators? Second, for certain mechanical systems we have access to simulators; for example, some stages of turbine engines can be simulated. How can we use the simulator or the simulations to enhance the representation model? Third, it is common in real world problems to have few data. Can we propose specific MTS few-shot learning processes to learn robust representations of data?

These open questions are of major interest in many domains where research in ML may bring solutions.

We also used our knowledge about sequence embedding to a very studied and experimented topic: the classification of graphs. In fact, we experimented in Chapter 5 an inductive bias for the classification of graphs with NNs that is different than the standard end-to-end classification model with graph neural networks (GNNs). The main assumption is that graphs are sets of nodes, that can be sequentially embedded with respect to a given node indexing. Using a standard node-ordering called breadth first search, we turned each graph into a sequence of nodes. Therefore, we leveraged the high capacity of recurrent neural networks (RNNs) to achieve state-of-the-art results. We also tested out the addition of a node-level task to help the graph-embedding model to generalize better by forcing it to keep local information during the graph-embedding phase. Despite the justified interest of such recurrent graph embedding for classification, the more recent GNN architectures outperform our approach.

Finally, in Chapter 6, we showed with a simple graph-feature which satisfies graph classification inductive biases, that the usage of the high capacity of GNNs for graph classification is not always fully justified, in particular on the commonly studied datasets.

We have seen in this thesis that the recent advances in graph classification are numerous, yet limited in their experiments to simple datasets (small graphs and binary classification). Hence, a first perspective would be to extend the benchmarks to large datasets of large graphs. A second perspective is for sequential graph classification. Since sequential graph embedding is inspired from NLP, we could use the recent advances (e.g. transformers) to find more expressive representation of graphs, using recent advances in text/document embedding.

This thesis was a journey in machine learning world, with focus on the inductive biases required to learn relevant data representation. In particular, we leveraged the high capacity and flexibility of neural networks coupled with adapted biases to build original embedding frameworks for MTS data, in particular in order to bring knowledge and solutions for health monitoring problems, and for graph classification.

Appendices

Appendix A

Background: neural networks

This appendix gives details on neural networks (NNs) that can be useful for the self-contained understanding of the thesis. We decompose the appendix in two sections. A first section presents the main neural networks architectures. A second section gives details on the learning procedure of the neural networks weights.

A.1 Different neural networks for different inductive biases

Neural networks are parametric functions composed as a stack of linear operations and nonlinear activation functions (sigmoid, rectified linear unit, tanh, etc.). They have been recently widely used in machine learning because of their high representation and generalization capacities. The purpose of this section is not to explain why neural networks are powerful but to introduce standard architectures.

The important knowledge to have is the following: NNs have shapes that depends on the a priori we have on data structure. This a priori can be though as a graphical structure on which data lives. For this reason, we first introduce graph neural network, since graphs are the most general data structure (contains all the others).

Global remark on the notations. For simplicity we share the same notations for all the presented architectures. Yet, in practice, depending of the scale data is regarded, the notations change.

A.1.1 Graph neural network

Graphs are data structures that model a set of objects (nodes) and their pairwise interactions (edges between nodes). Hence, data leaving on a graph domain can not be managed with tools developed for Euclidean (regular) data structure.

Presentation The main purpose of graph neural networks (GNNs) [Sperduti and Starita, 1997] is to represent objects living in a graph domain, by taking into account the neighboring information. We consider a graph $G = (E, V, W, X)$ with E the edge list, V the vertex set, W the adjacency and X a set of five d-dimensional objects with features $X = \{X_i\}_{i=1}^{5} \in \mathbb{R}^{5\times d}$, illustrated in Figure A.1.
The representation of each object X_i should take into account its conditional terms, i.e. other objects that interact with it. We note $A \in \mathbb{R}^{5\times 5}$ the adjacency of the graph G such that $A_{ij} \neq 0$ if edge $X_i \to X_j$ exists.

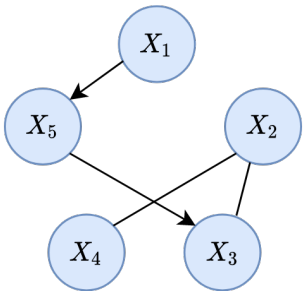


Figure A.1 – Five objects in graph domain. Arrows are conditional dependencies. Lines are joint distributions (double arrow).

GNNs are based on the recursive *message passing/neighborhood aggregation* embedding. At iteration k of the aggregation process, a representation Z_u^k of node $u \in V$ is defined as

$$Z_u^k = AGGREGATE^{(k)}\left(\left\{h_v^{(k-1)} : v \in \mathcal{N}(u)\right\}\right)$$

where $\mathcal{N}(u)$ is the set of neighbors for node u and $h_v^{(k)} = COMBINE^{(k)}\left(h_v^{(k-1)}, Z_v^k\right)$ $\forall v \in \mathcal{N}(u)$. AGGREGATE and COMBINE have several definitions. The most famous compose the graph convolutional neural network (GCN) proposed in [Kipf and Welling, 2016]. The multilayer GCN with latent dimensions $\{d_l\}_{l=1}^L$ consists in the following embedding: $Z^l = \sigma\left(\bar{A}Z^{l-1}W^{l-1}\right)$, with $W^l \in \mathbb{R}^{d_l \times d_{l-1}}$, $h^0 = X$, $d_0 = d$, $\bar{A} = \tilde{D}^{-\frac{1}{2}}A\tilde{D}^{-\frac{1}{2}}$ with $\tilde{A} = A + I$, $\tilde{D}_{ij} = \sum_k \tilde{A}_{ik}\delta_{ij}$ and σ any activation function (e.g. sigmoid). The product by $\bar{A}$ enables to only propagate node information contained in X to neighboring nodes: in the first layer only direct neighboring information is passed, in the second layer the neighbor's neighbors information is passed, etc. At the end of the training, Z^L contains a d_l-dimensional embedding of the nodes of G that can be use for downstream tasks (possibly learned end-to-end). We note that all objects $X_i \in X$ share the weights $W^{(l)}$.

Many particular GCNs have been proposed (with several AGGREGATE and/or COMBINE functions) to both outperform the expressiveness of basic GCN and solve precise problems on graph domains. In [Wu et al., 2020] is proposed a survey of on GNNs.

If the GCN is sufficiently deep to reach equilibrium (i.e. $Z^l = Z^{l-1}$) and if all layers share the same weights (i.e. $W^l = W^0$ for all layers l), we find the convergence and principle of recurrent graph neural network (RGNN) [Sperduti and Starita, 1997] which is the pioneer work of GNNs.

Remark 46. *From nodes embedding, it is possible to create an embedding $Z^{i\to j}$ of edges $i \to j$ by embedding couples $[Z^i, Z^j]$. Conversely, an embedding Z^j of the node j can be extracted from the edge embeddings $\{Z^{i\to j}\}_{i:A_{ij}\neq 0}$, for example $Z^j = \sum_i Z^{i\to j}$.*

Whole-graph embedding In Chapters 5 and 6 we are interested in the classification of whole graphs. Yet, GNNs presented above are made to embed nodes of the graph by taking into account the graph structure they live on. For whole-graph embedding, a READOUT function is required: $Z_G = READOUT\left(\left\{Z_v^K : v \in V\right\}\right)$. For example, READOUT function can be a mean or a max pooling, or a most sophisticated one (see Section 2.5.2 for examples).

A.1.2 Multi-layer perceptron

When we have no information nor a priori on the data graphical structure relating the observed objects, we may consider a fully disconnected graph as illustrated in Figure A.2.

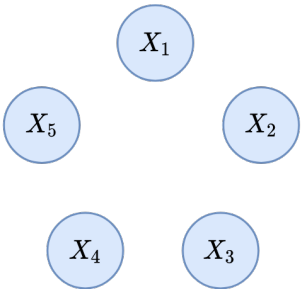


Figure A.2 – Absence of graphical a priori.

It is equivalent to have $A = 0$, i.e. l^{th} layer values of the graph has values $Z_i^l = \sigma\left(Z_i^{l-1}W^{l-1} + b\right)$ for objects i, with $Z^0 = X_i$. It is the multilayer perceptron (MLP). Yet, it is still possible to find hidden interactions within data by using Remark 46. Since all objects are disconnected, the linear operations Z^lW^l can be parallelized for faster computation.

A.1.3 Convolutional neural network

A common prior on graphical structure of image data is the regular grid. Each pixel (generally 3-dimensional since RGB color decomposition) is a node (the objects are here the pixels). Core pixels have a minimum of eight neighbors, border pixels have a minimum of five neighbors and corner pixels have a minimum of three neighbors, as illustrated in Figure A.3. Edges between neighboring pixels can be weighted by a function of the pixels (nodes) values.

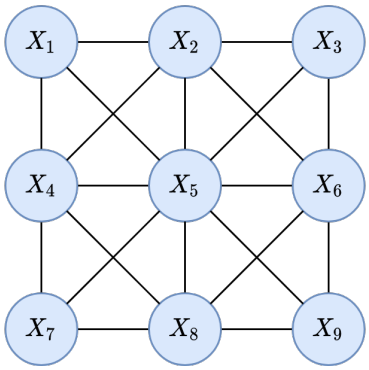


Figure A.3 – Graphical a priori for image (each object is a pixel, this is a very little image).

In general, since interactions between nodes are very local and structured (A is block-diagonal) and since generally images are high-dimensional (with thousands of objects), the image is treated as a fully-disconnected set of connected pixels. The local connection between pixel is implicitly taken into account by using localized convolutional filters. Hence, the parallelization property of MLP can also be used.

A.1.4 Recurrent neural network

The main a priori on sequential data structure is the auto-regression, introduced in Section 2.3, Figure 2.7. Using the notations above, it would require A to be upper triangular and dense in its upper part. When t grows, the number of edges to X_t grows also and computational/memory problems arise (we note that the problem exists for large graphs). Hence, a simplification of the a priori graph, thanks to hidden variables (see Section 2.3). The set of objects $\{X_i\}_{i=1}^T$ are assumed to be disconnected like in MLP. Nevertheless, to keep the autoregressive aspect, all layers of the neural networks are assumed to have a Markov chain structure, as illustrated in Figure A.4.

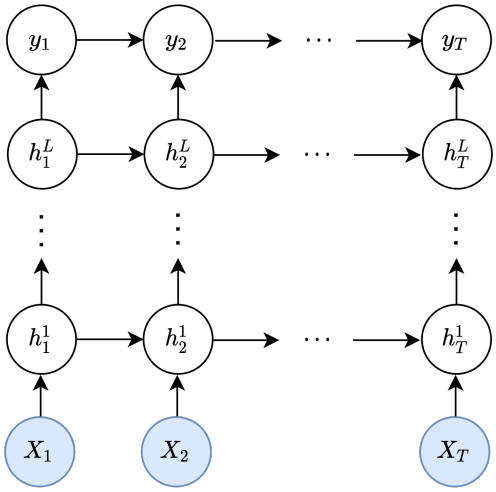


Figure A.4 – Graphical representation of a recurrent neural network. White balls are unobserved (latent).

It is the *recurrent neural networks* (RNN). A *memory cell* h accumulates the information such that h_t contains all (relevant) information from past and present time steps $X_{1:t}$. Using notations above plus additional weights $\{U^l\}_{l=1}^L$ for the memory cell, we have $h_t^l = \sigma(h_t^{l-1}W^{l-1} + h_{t-1}^l U^l + b_Z)$ and $y_t = \sigma(h_t^L U_y + b_y)$.

Remark 47. *RNN can be trained and used as a sequential generative model, if y_t serves as prediction of X_{t+1} that feeds the.*

In practice, it is known that RNN fails at modeling long-term dependencies in its memory cell Z. Information *vanishes*. Long-short term memory cells for recurrent neural networks (LSTM) has been proposed in [Hochreiter and Schmidhuber, 1997] to improve the long-term dependency modeling in RNN. LSTM leans on the usage of binary gates that chose, with respect to memory and observed data, if the past information should go through, without being diluted. In [Li et al., 2018], they observe that in practice, these binary gates are not really binary, and in many cases are half-open. They improve the LSTM by sampling the gates as (relaxed) binomial variables, hence truly binary. An equivalent (with little less parameters) is the gated recurrent unit (GRU) [Cho et al., 2014a, Chung et al., 2014].

Remark 48. *GRU has been used in [Li et al., 2015] to model a particular type of GCN (Section A.1.1) called recurrent GNN (RNN). Using gated neighbor aggregation facilitates the convergence towards an equilibrium of the message passing process.*

A.1.5 Relational neural networks

A relational neural network consists in finding an embedding of the relation between objects. We consider five objects $X = \{X_i\}_{i=1}^n, X_i \in \mathcal{X}$. First, a function $f_{var} : \mathbb{R}^{\mathcal{X}} \to \mathbb{R}^{d_v}$ embeds the d variables X_i of a sample X in a latent space of dimension d_v. Second, a function $f_{int} : \mathbb{R}^{2\times d_v} \to \mathbb{R}^{d_i}$ takes as input each couple $[f_{var}(X_i), f_{var}(X_j)]$ and embeds it into second latent space of dimension d_i. We note $h^i = f_{var}(X_i)$ and $h^{i,j} = f_{int}([h^i, h^j])$.

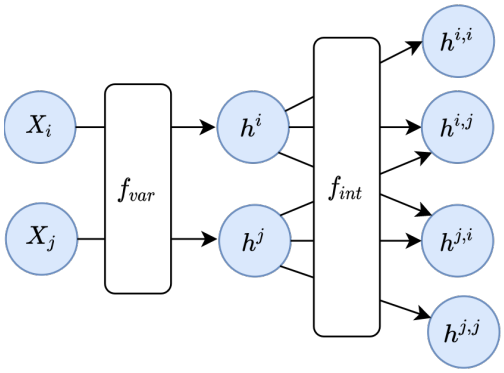


Figure A.5 – Graphical model of a relational neural network, embedding the relation between two objects (i, j).

In [Kipf et al., 2018], the RelNN is expanded with a GNN that takes as input the fully-connected graph with n^2 nodes, whose edge features are the $h^{i,j}$. This GNN layer enables the model to explicitly take into account the fact that the interactions between objects (i, j) are also dependent on the respective interactions of i and j with the other objects $\{1, \dots, n\} \setminus \{i, j\}$. We use the expanded version of RelNN in Chapters 3.

Remark 49. *All the neural networks implementations have been made using PyTorch library [Paszke et al., 2019].*

Appendix B

Background: algebra, statistics and optimization

This section gives the minimal background about algebra, statistics and optimization to fully understand the thesis.

B.1 Note on variational inference

The *variational inference* (VI) is a method used in Bayesian statistics to approximate densities of probability through optimization techniques. In particular, it is often used for the approximation of marginal likelihood of observed data or posterior distributions of unobserved parameters conditionally to observed data. VI is generally faster and easier to scale to high dimensional data and large datasets than Markov chain Monte Carlo (MCMC) estimation [Hastings, 1970]. In this section, we give minimal needed information about VI to understand to content of the thesis. Exhaustive information can be found in [Blei et al., 2017].

Consider the problem of estimating posterior $p(Z|X)$ in latent variable models (2.5). It is generally untractable. The VI consists in replacing $p(Z|X)$ by a *variational distribution* q selected from a *variational family of distibution* $\mathcal{Q}$, such that:

$$q = \arg\min_{q \in \mathcal{Q}} KL\left(q(Z|X)||p(Z|X)\right)$$

where KL is the *Kullback-Leibler divergence* (KLD). Yet, the quantity still depends on $p(Z|X)$. We change the point of view. As usual, the problem can be related to the maximization of data log-likelihood $\log p(X)$. We remark that

$$
\begin{aligned}
\log p(X) &= \log \int p(X,Z)dZ \\
&= \log \int \frac{q(Z|X)}{q(Z|X)} p(X,Z)dZ \\
&= \log \mathbb{E}_{q(Z|X)} \frac{p(X,Z)}{q(Z|X)} \\
&\geq \mathbb{E}_{q(Z|X)} \log \frac{p(X,Z)}{q(Z|X)} \\
&= \mathbb{E}_{q(Z|X)} \log \frac{p(X|Z)p(Z)}{q(Z|X)} \\
&= \mathbb{E}_{q(Z|X)} \log p(X|Z) + KL\left(q(Z|X)||p(Z)\right)
\end{aligned}
$$

Hence, finding the best variational approximiation q must maximize the lower bound of the log-likelihood, called *evidence lower bound* (ELBO). The first term of the last line is the *reconstruction term*, related to the estimated likelihood. The second term is a regularization of to match prior and posterior distributions.

Generally, the family $\mathcal{Q}$ is much simpler than the true posterior distribution, e.g. the exponential family, generally Gaussian. Otherwise, the ELBO would not be tractable. The maximization with respect to q then becomes a maximization with respect to the parameters of the variational family.

Remark 50. *The ELBO is the loss of variational autoencoders (VAE), the most known family of neural latent variable models, introduced in Section 2.2.4.*

B.2 Change of variable formula

If we have $z \sim p(z)$ a d-dimensional variable and $f : \mathbb{R}^d \to \mathbb{R}^d$ an invertible function such that $x = f(z)$, then the density:

$$
x \sim p\left(f^{-1}(x)\right) \left| \det \frac{\partial f^{-1}(y)}{\partial y}\bigg|_{y=x} \right| = p\left(f^{-1}(x)\right) \left| \det \frac{\partial f(y)}{\partial y}\bigg|_{y=x} \right|^{-1}
$$

the second equality coming from two properties. First, the Jacobian of the inverse is the inverse of the Jacobian. Second, the determinant of the inverse is the inverse of the determinant.

B.3 Note on random coefficient regression

In standard regression model, the parameters (e.g. slope β and intercept b of a linear model) are fixed after fitting a dataset. The assumption is that data is *stationary*. In *random coefficient models* (RCR), the parameters are allowed to vary according to a distribution [Muthén et al., 2015]. For example, if the variable to predict is the baccalauréat score Y from in-class evaluation scores X, the regression parameters may depend on characteristics C_p of each pupil p, like the school, the socio-professional status, etc. We would therefore have the additional assumption $(\beta_p, b_p) \sim p(\beta, b|C_p)$ such that the predicted baccalauréat score Y_p for pupil p is $\beta_p X + c_p$.

B.4 Note on proximal algorithms

B.4.1 Principles of proximal gradient descent

A standard minimization problem in machine learning consists of an objective function and a regularization function, in the following generic shape:

$$\min_{\theta} f(\theta) + \lambda g(\theta) \tag{B.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is generally a smooth (differentiable) function and $g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is non-smooth, like LASSO penalty for example. We assume that f and g are convex.

We call *proximal operator* of convex function f the function:

$$\text{prox}_f(\tilde{\theta}) := \arg\min_{\theta} f(\theta) + \frac{1}{2}\|\tilde{\theta} - \theta\|_2^2 \tag{B.2}$$

The proximal operator has the following property

$$\theta = \text{prox}_f(\tilde{\theta}) \Leftrightarrow \tilde{\theta} - \theta \in \partial f(\theta) \tag{B.3}$$

where $\partial f(\theta)$ is the subderivative of f at θ, defined by:

$$\partial f(\theta) = \{v | f(z) \geq f(\theta) + \langle v, z - \theta \rangle, \forall z \in \text{dom} f\} \tag{B.4}$$

We note that if f is differentiable, then $\nabla f(\theta) \in \partial f(\theta)$. It implies two important features for our problem. First, proximal operation can be seen as a gradient step, with a minimum of f being achieved at a fixed point of $\text{prox}_{\lambda f}(\theta)$. If f is differentiable, $\text{prox}_{\lambda f}(\theta) \approx \theta - \lambda \nabla f(\theta)$. Second, a fixed point of prox_f is a minimizer of f. More details can be found in [Parikh et al., 2014]. We generically call *proximal gradient method* (PGM) the learning procedure based on proximal gradient.

In problem (B.1), the proximal operator is required for g. We note λ_r the gradient descent step. The splitting of the problem in differentiable and non-differentiable functions is treated with two steps [Polson et al., 2015]. Fist, the parameter θ is updated in the direction of $\nabla f(\theta)$, that exists since f is differentiable. Second, the updated $\theta - \lambda_r \nabla f(\theta)$ is mapped towards a minimum of function g using $\text{prox}_{\lambda g}$. The parameter λ, as a stepsize, controls the importance of the mapping towards the minimum of g. It is summaried in Algorithm 1.

Algorithme 1 : Proximal gradient step for a problem (B.1)

Input : Initial parameter θ

Output : Updated parameter θ

$\theta \leftarrow \theta - \lambda_r \nabla f(\theta)$

$\theta \leftarrow \text{prox}_{\lambda g}(\theta)$

We note that this algorithm is still meaningful for nonconvex smooth functions f [Polson et al., 2015], like neural networks [Ma et al., 2019]. Some pretraining might be useful to avoid convergence anomalies, by placing θ in a favorable location before processing proximal gradient steps. We use this proximal algorithm in our experiments of Chapter 3.

We note that efficient proximal algorithm is still an active field of research [Yao et al., 2016]. For this thesis, we do not go further than the basis PGM.

B.4.2 Application of proximal gradient to sparse VAR problem

This section details the proximal gradient descent applied on the sparse VAR problem of Chapter 3. We do not remind the notations in this Appendix.

In order to obtain truly sparse weights $\bar{W}$ (with exact zeros), we apply a proximal gradient method [Parikh et al., 2014]. Principles of proximal optimization of given in Appendix B.4. If we note λ_r the learning rate and $G(W) := \sum_{i,j=1}^{d} \|W_{.,i,j}\|_F$ the group-LASSO penalty, we use a *block soft thresholding* proximal operator:

$$\text{prox}_{\lambda G}(W_{.,i,j}) = \left(1 - \frac{\lambda}{\|W_{.,i,j}\|_F}\right)_+ W_{.,i,j} \tag{B.5}$$

Referring to the proximal Algorithm 1 (Appendix B.4), we learn θ and $\bar{W}$ with the procedure described in Algorithm 2.

Algorithme 2 : Proximal gradient descent for sparse GVAR training

Input : θ, $\bar{W}$

λ_r, λ **Output :** Updated parameter θ, $\bar{W}$

while *not converged* **do**

  for $j \leftarrow 1$ **to** d **do**

    $\theta_j \leftarrow \theta_j - \lambda_r \nabla_\theta R(\theta_j, \bar{W})$

  for $i \leftarrow 1$ **to** d **do**

    for $j \leftarrow 1$ **to** d **do**

      $\bar{W}_{.,i,j} \leftarrow \bar{W}_{.,i,j} - \lambda_r \nabla_{\bar{W}_{.,i,j}} R(\theta, \bar{W})$

      $\bar{W}_{.,i,j} \leftarrow \text{prox}_{\lambda_r \lambda G}(\bar{W}_{.,i,j})$

B.5 Note on contrastive divergence and contrastive learning

The *contrastive learning* (CL) is a set of methods to transform unsupervised learning into supervised learning. We split the CL into two branches: the *contrastive divergence learning* and the *contrastive metric learning*.

Contrastive divergence learning A first version of CL emerged in *modern* ML to treat untractable probabilistic modeling that maximum-likelihood estimation (MLE) could not. It is based on *contrastive divergence* (CD), an approximate maximum-likelihood estimation (MLE) [Hinton, 2002] to learn the parameters θ of *unnormalized* parametric statistical models $f_\theta(X)$ (i.e. that does not integrate to one)

[Carreira-Perpinan and Hinton, 2005]. To apply MLE, we need to normalize f_θ with the *partition function* $\mathcal{Z}(\theta) = \int f_\theta(x)dx$ such that $p_\theta(x) = f_\theta(x)/\mathcal{Z}(\theta)$. Then MLE consists in $\arg\min_\theta \mathbb{E}_\mathcal{X}[-\ln p_\theta(x)]$, where $\mathbb{E}_\mathcal{X}$ is the empirical expected value over the observed data $\mathcal{X}$. The main problem with this estimation is that $\mathcal{Z}(\theta)$ is often untractable, and so its gradient. In [Hinton, 2002], they simply remark that, given p^n_θ an approximation of p_θ with n steps of MCMC (we remind that $p^n_\theta \xrightarrow{n\to\infty} p_\theta$), then $\partial_\theta \mathbb{E}_\mathcal{X}[\ln p_\theta(x)] \approx \mathbb{E}_{X\sim\mathcal{X}}[\partial_\theta f_\theta(X)] - \mathbb{E}_{X\sim p^n_\theta}[\partial_\theta f_\theta(X)]$, i.e. partition function disappears from the estimation problem (we note that we can sample from p^n_θ without computing the partition function since MCMC methods involve density ratios $p_\theta(x')/p_\theta(x) = f_\theta(x')/f_\theta(x)$). Finding the best θ then becomes finding the model that do not contrast between MCMC-generated data and observed data, here with respect to the unnormalized density. This is the essence of contrastive divergence learning: finding the set of parameters such that contrast between generated and observed data is null. It is the principle of many likelihood-free inference models.The most known usage of this contrastive estimation of the density gradient is to train restricted Boltzman machine (RBM) [Hinton and Salakhutdinov, 2006, Carreira-Perpinan and Hinton, 2005], an unsupervised representation learning framework. In [Gutmann and Hyvärinen, 2010], they show that the parameter inference of any unnormalized model f_θ, hence not solvable with maximum-likelihood methods like mentioned above, can be approximated by learning to contrast between observed data $\mathcal{X}$ and generated data (noise) $\mathcal{N}$ with distribution p_n. Observed data have labels 1, noise data have labels 0. They train the classifier $h_\theta(x) = \sigma(\ln f_\theta(x) + c - \ln p_n(x))$ with respect to θ, c using these labels. Here, c is a substitute to $\mathcal{Z}(\theta)$. Hence, by minimizing $\mathbb{E}_{(X,\epsilon)\sim\mathcal{X}\times\mathcal{N}}[-\ln h_\theta(X) - \ln h_\theta(1-\epsilon)]$, they show that the estimated θ is the maximum-likelihood estimator, and c the associated (untractable) log-partition function. It is called *noise contrastive estimation* (NCE). We can find NCE in recent advances in neural generative modeling of high-dimensional data [Gao et al., 2019]. A particular case of NCE is the *negative sampling estimation* [Dyer, 2014], whose most famous example is the *word2vec* [Mikolov et al., 2013]. More recently, generative adversarial networks (GANs) [Goodfellow et al., 2014] rely on the same principle. A neural network G_θ parametrized by θ implicitly represents density of the data. Fake data is generated from G_θ. A discriminator D_ϕ with parameters ϕ learns to contrast between generated and observed data. When there is no parameters ϕ such that the discriminator can find contrast between generated and true data, then θ is the most likely data density parameter. Finally, the CD approach is a very general and powerful framework for learning the parameters of complex models.

Metric learning A second version of CL is the *metric learning* or *ranking learning*. It is the CL that has been introduced in Section 2.1.4. It consists in learning representation of data by learning to distance dissimilar samples and reconcile similar samples, in a latent representation space. At the end, with a sufficiently expressive data representation procedure, CL ranks the samples in term of relative similarity in the latent space. This CL is obviously related to the latter paper presented in previous paragraph. This time, noise sampling is replaced by dissimilar sampling. If $X \sim \mathcal{X}$, X^- a dissimilar (negative) sample, X^+ a similar (positive) sample, then the objective is to minimize $h_\theta(X, X^+) = d(f_\theta(X), f_\theta(X^+))$ and maximize $h_\theta(X, X^-) = d(f_\theta(X), f_\theta(X^-))$, with d any divergence. In [Saunshi et al., 2019] they theoretically explain CL by introducing the notion of *latent classes*, such that similar samples are implicitly in the same latent class, and conversely. These latent classes overlaps, such that similarity is a continuous notion in observation space. This paper gives good view of what is possible and impossible with CL.

The main problem is then to define the a priori similarity label between sampels. Yet, there are several domains where it is available. When samples have auxiliary attributes (e.g. labels), the distance between attributes is a simple information about semantic similarity in observation space [Hadsell et al., 2006]. Hence, CL naturally proved to be very efficient for classification downstream task [Chen et al., 2020] when classes are chosen as similarity indicator. In natural language processing (NLP), the similarity between words is generally their number of co-occurence [Mikolov et al., 2013]. This idea extends to all sequential data. Similarity between time series is the co-occurence of patterns [Lin and Li, 2009], or the proximity in term of time-index.

B.6 Proof of GLS isomorphism-invariance

We propose a simple proof of the known GLS isomorphism-invariance used as an important inductive bias for graph-level tasks, like classification, in Chapter 6.

Let $G = (V, E, W)$ be an undirected and weighted graph, L its Laplacian matrix and $\Pi \in \mathcal{P}(|V|)$ be a permutation matrix. A permutation of node indexing implies a permutation of both rows and columns of the Laplacian matrix.

The spectrum of L is the set of roots of $P(\lambda) = det\left(L - \lambda I_{|V|}\right)$. We want the spectrum of $\Pi^+ L \Pi$:

$$\begin{aligned} P^{\Pi}(\lambda) &= det\left(\Pi^T L \Pi - \lambda I_{|V|}\right) \\ &= det\left(\Pi^T L \Pi - \lambda \Pi^+ \Pi\right) \\ &= det\left(\Pi^T \left(L - \lambda I_{|V|}\right) \Pi\right) \\ &= det\left(\Pi^T\right) det\left(L - \lambda I_{|V|}\right) det\left(\Pi\right) \end{aligned}$$

Yet we know that the determinant of a matrix is invariant to transpose and that the determinant of a permutation matrix is equal to its signature:

$$det\left(\Pi\right) = det\left(\Pi^T\right) = (-1)^{|V| - \sum_{i=1}^{|V|} \Pi_{ii}}$$

Hence:

$$\begin{aligned} P^{\Pi}(\lambda) &= (-1)^{2\left(|V| - \sum_{i=1}^{|V|} \Pi_{ii}\right)} det\left(L - \lambda I_{|V|}\right) \\ &= det\left(L - \lambda I_{|V|}\right) \\ &= P(\lambda) \end{aligned}$$

Appendix C

Datasets

C.1 Ball-springs datasets

The *balls-springs* is a simulated data used in several chapters of our thesis. It consists in the trajectory in a 2-dimensional space of N balls randomly linked with springs, with probability 0.5. The springs follow the Newton's law of motion and are very practical for several reasons. First, they contain an underlying graph structure that can be easily controlled for simulation. Second, they contain several independent controllable factors, that are also independent of the graph adjacency, like initial velocity, initial position, rigidity of the springs. like many simple dataset used for interpretable/disentangled representation learning have equivalent properties. It is a very challenging and complete time series toy dataset for unsupervised representation learning tasks. We used the implementation given along the paper [Kipf et al., 2018].

In particular, we use an ageing version of the ball-springs system for experiments in Chapters 3 and 4, that are illustrated in Figure C.1.

Remark 51. *Newton's law tells that the position of a mass is proportional to its acceleration (i.e. its second time derivatives) that can be estimated with Euler method. The acceleration is itself proportional to the force applied to the mass, i.e. to the acceleration of the others masses. It is the origin of Granger causality in mass-spring data. Nevertheless, we note that our time series are subsampled, the estimation of the derivative is not perfect, and so the causality detection [Gong et al., 2015].*

C.2 NASA C-MAPSS datasets

NASA public Commercial Modular Aero-Propulsion System Simulation dataset (C-MAPSS) is a tool for simulation of realistic large commercial turbofan engine data [Saxena and Goebel, 2008]. The common repository proposes four datasets, that consist of sets of multivariate time series. Each time series is from a different engine. Each engine is considered as new, like coming out of the plant. Hence, each engine starts with different degrees of initial wear and manufacturing variation which is unknown to the user. This wear and variation is considered normal, i.e., it is not considered a fault condition. There are three operational settings that have a substantial effect on engine performance. These settings are also included in the data. The data is contaminated with sensor noise.

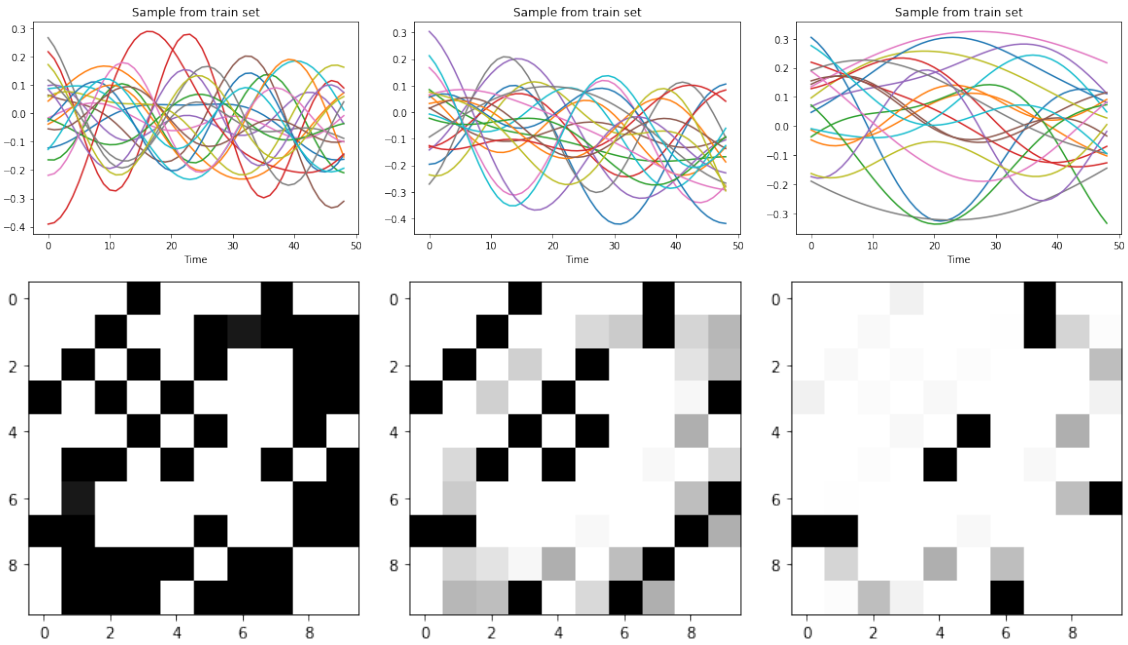


Figure C.1 – Examples of three samples from the same balls-springs system, after respectively 0, 20 and 40 steps of ageing with ageing coefficient $\alpha_b = 0.9$.

The engine is operating normally at the start of each time series, and develops a fault at some point during the series. In the training set, the fault grows in magnitude until system failure. Each time series from start to failure is called *trajectory* in the thesis.

C.3 Graph datasets

In Chapters 5 and 6, we use five molecular datasets and five social network datasets for the experiments [Kersting et al., 2016]. Tables C.1 and C.2 gives statistics of the differents datasets. All used datasets can be found at the following address: https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets.

Molecular graphs datasets are Mutag (MT), Enzymes (EZ), Proteins Full (PF), Dobson and Doig (DD) and National Cancer Institute (NCI1). In MT, the graphs are either mutagenic and not mutagenic. EZ graphs are tertiary structures of proteins from the 6 Enzyme Commission top level classes. In DD, compounds are secondary structures of proteins that are enzyme or not. PF is a subset of DD without the largest graphs. In NCI1, graphs are anti-cancer or not. The graphs of these datasets have node labels that can be leverages by graph neural networks.

Social networks datasets are IMDB-Binary (IMBD-B), IMDB-Multi (IMDB-M), REDDIT-Binary (REDDIT-B), REDDIT-5K-Multi (REDDIT-M) and COLLAB. REDDIT-B and REDDIT-M contain graphs representing discussion thread, with edges between users (nodes) when one responded to the other's comment. Classes are the subreddit topics from which thread have originated. IMDB-B and IMDB-M contain networks of actors that appeared together within the same movie. IMDB-B contains two classes for *action* or *romance* genres and IMDB-M three classes for *comedy*, *romance* and *sci-fi*. COLLAB graphs represent scientific collaborations, with edge between two researchers meaning that they co-authored a paper. Labels of the

	MT	EZ	PF	DD	NCI1
# graphs	188	600	1113	1178	4110
# classes	2	6	2	2	2
bias (%)	66.5	16.7	59.6	58.7	50.0
min./max. $\|V\|$	10/28	2/125	4/620	30/5736	3/106
avg. $\|V\|$	18	33	39	284	30
avg. $\|E\|$	39	124	146	1431	65
Node attributes	✓	✓	✓	✓	✓

Table C.1 – Molecular datasets statistics. Bias indicates the proportion of the dominant class.

graphs correspond to subfields of Physics. The graphs of these datasets have no node attributes and therefore enable fair comparison with deep learning methods.

	IMDB-B	IMDB-M	REDDIT-B	REDDIT-M	COLLAB
# graphs	1000	1500	2000	4999	5000
# classes	2	3	2	5	3
bias (%)	50.0	33.3	50.0	20.0	52.0
min./max. $\|V\|$	12/136	7/89	3/3760	22/3606	32/492
avg. $\|V\|$	20	13	426	501	75
avg. $\|E\|$	97	66	496	590	2458
Node attributes	✗	✗	✗	✗	✗

Table C.2 – Social network datasets statistics. Bias indicates the proportion of the dominant class.

References

[Ackert, 2015] Ackert, S. (2015). Engine maintenance management. *Managing technical aspects of leased assets, Madrid*, pages 1–31.

[Akintayo and Sarkar, 2015] Akintayo, A. and Sarkar, S. (2015). A symbolic dynamic filtering approach to unsupervised hierarchical feature extraction from time-series data. In *2015 American Control Conference (ACC)*, pages 5824–5829. IEEE.

[Alemi et al., 2018] Alemi, A., Poole, B., Fischer, I., Dillon, J., Saurous, R. A., and Murphy, K. (2018). Fixing a broken elbo. In *International Conference on Machine Learning*, pages 159–168.

[Ancona et al., 2004] Ancona, N., Marinazzo, D., and Stramaglia, S. (2004). Radial basis function approach to nonlinear granger causality of time series. *Physical Review E*, 70(5):056221.

[Antonucci et al., 2015] Antonucci, A., De Rosa, R., Giusti, A., and Cuzzolin, F. (2015). Robust classification of multivariate time series by imprecise hidden markov models. *International Journal of Approximate Reasoning*, 56:249–263.

[Arora et al., 2019] Arora, S., Khandeparkar, H., Khodak, M., Plevrakis, O., and Saunshi, N. (2019). A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229*.

[Asadi-Aghbolaghi et al., 2017] Asadi-Aghbolaghi, M., Clapés, A., Bellantonio, M., Escalante, H. J., Ponce-López, V., Baró, X., Guyon, I., Kasaei, S., and Escalera, S. (2017). Deep learning for action and gesture recognition in image sequences: A survey. In *Gesture Recognition*, pages 539–578. Springer.

[Atwood and Towsley, 2016] Atwood, J. and Towsley, D. (2016). Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1993–2001.

[Azizian and Lelarge, 2020] Azizian, W. and Lelarge, M. (2020). Characterizing the expressive power of invariant and equivariant graph neural networks. *arXiv preprint arXiv:2006.15646*.

[Bach and Jordan, 2002] Bach, F. R. and Jordan, M. I. (2002). Kernel independent component analysis. *Journal of machine learning research*, 3(Jul):1–48.

[Badiane et al., 2018] Badiane, M., O'Reilly, M., and Cunningham, P. (2018). Kernel methods for time series classification and regression. In *AICS*, pages 54–65.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Baldi and Hornik, 1989] Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58.

[Banville et al., 2019] Banville, H., Moffat, G., Albuquerque, I., Engemann, D.-A., Hyvärinen, A., and Gramfort, A. (2019). Self-supervised representation learning from electroencephalography signals. In *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE.

[Bar-Hillel et al., 2003] Bar-Hillel, A., Hertz, T., Shental, N., and Weinshall, D. (2003). Learning distance functions using equivalence relations. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 11–18.

[Barnett et al., 2016] Barnett, I., Malik, N., Kuijjer, M. L., Mucha, P. J., and Onnela, J.-P. (2016). Feature-based classification of networks. *arXiv preprint arXiv:1610.05868*.

[Battaglia et al., 2018] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

[Bell and Sejnowski, 1995] Bell, A. J. and Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159.

[Bengio et al., 2013] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

[Bennett, 1983] Bennett, S. (1983). Analysis of survival data by the proportional odds model. *Statistics in medicine*, 2(2):273–277.

[Blaschke et al., 2007] Blaschke, T., Zito, T., and Wiskott, L. (2007). Independent slow feature analysis and nonlinear blind source separation. *Neural computation*, 19(4):994–1021.

[Blei et al., 2017] Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

[Bonald et al., 2018] Bonald, T., Hollocou, A., and Lelarge, M. (2018). Weighted spectral embedding of graphs. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 494–501. IEEE.

[Borgwardt and Kriegel, 2005] Borgwardt, K. M. and Kriegel, H.-P. (2005). Shortest-path kernels on graphs. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE.

[Bostrom and Bagnall, 2017] Bostrom, A. and Bagnall, A. (2017). A shapelet transform for multivariate time series classification. *arXiv preprint arXiv:1712.06428*.

[Bowman et al., 2016] Bowman, S., Vilnis, L., Vinyals, O., Dai, A., Jozefowicz, R., and Bengio, S. (2016). Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21.

[Brouwer and Haemers, 2011] Brouwer, A. E. and Haemers, W. H. (2011). *Spectra of graphs*. Springer Science & Business Media.

[Bruna et al., 2013] Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

[Cai et al., 2018] Cai, H., Zheng, V. W., and Chang, K. (2018). A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Transactions on Knowledge and Data Engineering*.

[Cao et al., 2019] Cao, S., Wang, X., and Kitani, K. M. (2019). Learnable embedding space for efficient neural architecture compression. *arXiv preprint arXiv:1902.00383*.

[Cardoso, 1999] Cardoso, J.-F. (1999). High-order contrasts for independent component analysis. *Neural computation*, 11(1):157–192.

[Carreira-Perpinan and Hinton, 2005] Carreira-Perpinan, M. A. and Hinton, G. E. (2005). On contrastive divergence learning. In *Aistats*, volume 10, pages 33–40. Citeseer.

[Casals et al., 2018] Casals, J., Garcia-Hiernaux, A., Jerez, M., Sotoca, S., and Trindade, A. A. (2018). *State-space methods for time series analysis: theory, applications and software*. Chapman and Hall/CRC.

[Cawley and Talbot, 2010] Cawley, G. C. and Talbot, N. L. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11(Jul):2079–2107.

[Chakrabarti et al., 2002] Chakrabarti, K., Keogh, E., Mehrotra, S., and Pazzani, M. (2002). Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Transactions on Database Systems (TODS)*, 27(2):188–228.

[Chan and Fu, 1999] Chan, K.-P. and Fu, A. W.-C. (1999). Efficient time series matching by wavelets. In *Proceedings 15th International Conference on Data Engineering (Cat. No. 99CB36337)*, pages 126–133. IEEE.

[Chechik et al., 2010] Chechik, G., Sharma, V., Shalit, U., and Bengio, S. (2010). Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(Mar):1109–1135.

[Chen, 2006] Chen, A. (2006). Fast kernel density independent component analysis. In *International Conference on Independent Component Analysis and Signal Separation*, pages 24–31. Springer.

[Chen et al., 2020] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*.

[Cho et al., 2014a] Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

[Cho et al., 2014b] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Chouakria-Douzal, 2003] Chouakria-Douzal, A. (2003). Compression technique preserving correlations of a multivariate temporal sequence. In *International symposium on intelligent data analysis*, pages 566–577. Springer.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[Cohen et al., 2007] Cohen, P., Adams, N., and Heeringa, B. (2007). Voting experts: An unsupervised algorithm for segmenting sequences. *Intelligent Data Analysis*, 11(6):607–625.

[Cox, 1972] Cox, D. R. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202.

[Csáji et al., 2001] Csáji, B. C. et al. (2001). Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24(48):7.

[Davis et al., 2016] Davis, R. A., Zang, P., and Zheng, T. (2016). Sparse vector autoregressive modeling. *Journal of Computational and Graphical Statistics*, 25(4):1077–1096.

[de Lara and Pineau, 2018] de Lara, N. and Pineau, E. (2018). A simple baseline algorithm for graph classification. *Relational Representation Learning Workshops (NIPS 2018)*.

[Defferrard et al., 2016] Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852.

[Diks and Panchenko, 2006] Diks, C. and Panchenko, V. (2006). A new statistic and practical guidelines for nonparametric granger causality testing. *Journal of Economic Dynamics and Control*, 30(9-10):1647–1669.

[Dinh et al., 2014] Dinh, L., Krueger, D., and Bengio, Y. (2014). Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.

[Dinh et al., 2016] Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*.

[Duvenaud et al., 2015] Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232.

[Dyer, 2014] Dyer, C. (2014). Notes on noise contrastive estimation and negative sampling. *arXiv preprint arXiv:1410.8251*.

[Eichler, 2001] Eichler, M. (2001). Granger causality graphs for multivariate time series.

[Eichler and Didelez, 2012] Eichler, M. and Didelez, V. (2012). Causal reasoning in graphical time series models. *arXiv preprint arXiv:1206.5246*.

[Elsken et al., 2019] Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural architecture search: A survey. *Journal of Machine Learning Research*, 20:1–21.

[Erdős and Rényi, 1959] Erdős, P. and Rényi, A. (1959). On random graphs i. *Publ. Math. Debrecen*, 6:290–297.

[Esling and Agon, 2012] Esling, P. and Agon, C. (2012). Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):12.

[Esmael et al., 2012] Esmael, B., Arnaout, A., Fruhwirth, R. K., and Thonhauser, G. (2012). Multivariate time series classification by combining trend-based and value-based approximations. In *International Conference on Computational Science and Its Applications*, pages 392–403. Springer.

[Fabius and van Amersfoort, 2014] Fabius, O. and van Amersfoort, J. R. (2014). Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*.

[Fabrigar and Wegener, 2011] Fabrigar, L. R. and Wegener, D. T. (2011). *Exploratory factor analysis*. Oxford University Press.

[Faloutsos et al., 1994] Faloutsos, C., Ranganathan, M., and Manolopoulos, Y. (1994). Fast subsequence matching in time-series databases. *Acm Sigmod Record*, 23(2):419–429.

[Fernando et al., 2015] Fernando, B., Gavves, E., Oramas, J. M., Ghodrati, A., and Tuytelaars, T. (2015). Modeling video evolution for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5378–5387.

[Fortuin et al., 2018] Fortuin, V., Hüser, M., Locatello, F., Strathmann, H., and Rätsch, G. (2018). Somvae: Interpretable discrete representation learning on time series.

[Fotso et al., 19] Fotso, S. et al. (2019–). PySurvival: Open source package for survival analysis modeling.

[Fraccaro, 2018] Fraccaro, M. (2018). Deep latent variable models for sequential data.

[Franceschi et al., 2019] Franceschi, J.-Y., Dieuleveut, A., and Jaggi, M. (2019). Unsupervised scalable representation learning for multivariate time series. *arXiv preprint arXiv:1901.10738*.

[Fujisawa and Eguchi, 2008] Fujisawa, H. and Eguchi, S. (2008). Robust parameter estimation with a small bias against heavy contamination. *Journal of Multivariate Analysis*, 99(9):2053–2081.

[Gao et al., 2019] Gao, R., Nijkamp, E., Kingma, D. P., Xu, Z., Dai, A. M., and Wu, Y. N. (2019). Flow contrastive estimation of energy-based models. *arXiv preprint arXiv:1912.00589*.

[Ge and Smyth, 2000] Ge, X. and Smyth, P. (2000). Deformable markov model templates for time-series pattern matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 81–90.

[Gershgorin, 1931] Gershgorin, S. A. (1931). Uber die abgrenzung der eigenwerte einer matrix. (6):749–754.

[Geyer, 1992] Geyer, C. J. (1992). Practical markov chain monte carlo. *Statistical science*, pages 473–483.

[Ghahramani and Hinton, 1996] Ghahramani, Z. and Hinton, G. E. (1996). Parameter estimation for linear dynamical systems. Technical report, Technical Report CRG-TR-96-2, University of Totronto, Dept. of Computer Science.

[Ghosh et al., 2017] Ghosh, A., Kumar, H., and Sastry, P. (2017). Robust loss functions under label noise for deep neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[Ghosh et al., 2019] Ghosh, P., Sajjadi, M. S., Vergari, A., Black, M., and Schölkopf, B. (2019). From variational to deterministic autoencoders. *arXiv preprint arXiv:1903.12436*.

[Gidaris et al., 2019] Gidaris, S., Bursuc, A., Komodakis, N., Pérez, P., and Cord, M. (2019). Boosting few-shot visual learning with self-supervision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8059–8068.

[Goh, 2001] Goh, C. (2001). Econ 2 0a: Sufficiency, minimal sufficiency and the exponential family of distributions.

[Goldsmith, 2012] Goldsmith, F. B. (2012). *Monitoring for conservation and ecology*, volume 3. Springer Science & Business Media.

[Gómez and Delvenne,] Gómez, L. G. and Delvenne, J.-C. Dynamics based features for graph classification. In *Benelearn 2017: Proceedings of the Twenty-Sixth Benelux Conference on Machine Learning, Technische Universiteit Eindhoven, 9-10 June 2017*, page 131.

[Gong et al., 2015] Gong, M., Zhang, K., Schoelkopf, B., Tao, D., and Geiger, P. (2015). Discovering temporal causal relations from subsampled data. In *International Conference on Machine Learning*, pages 1898–1906.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

[Granger, 1969] Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, pages 424–438.

[Grathwohl et al., 2019] Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., Norouzi, M., and Swersky, K. (2019). Your classifier is secretly an energy based model and you should treat it like one. *International Conference on Learning Representation*.

[Grauman and Darrell, 2007] Grauman, K. and Darrell, T. (2007). The pyramid match kernel: Efficient learning with sets of features. *Journal of Machine Learning Research*, 8(Apr):725–760.

[Gray, 2007] Gray, K. L. (2007). Comparison of trend detection methods.

[Gretton et al., 2005] Gretton, A., Herbrich, R., Smola, A., Bousquet, O., and Schölkopf, B. (2005). Kernel methods for measuring independence. *Journal of Machine Learning Research*, 6(Dec):2075–2129.

[Grohe et al., 2018] Grohe, M., Rattan, G., and Woeginger, G. J. (2018). Graph similarity and approximate isomorphism. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[Gutmann and Hyvärinen, 2010] Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

[Gutmann et al., 2018] Gutmann, M. U., Dutta, R., Kaski, S., and Corander, J. (2018). Likelihood-free inference via classification. *Statistics and Computing*, 28(2):411–425.

[Hadsell et al., 2006] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

[Haemers, 2016] Haemers, W. H. (2016). Are almost all graphs determined by their spectrum. *Not. S. Afr. Math. Soc*, 47:42–45.

[Halko et al., 2011] Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

[Hamilton et al., 2017] Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034.

[Harrell Jr et al., 1996] Harrell Jr, F. E., Lee, K. L., and Mark, D. B. (1996). Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in medicine*, 15(4):361–387.

[Hastings, 1970] Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications.

[Haufe et al., 2010] Haufe, S., Müller, K.-R., Nolte, G., and Krämer, N. (2010). Sparse causal discovery in multivariate time series. In *Causality: Objectives and Assessment*, pages 97–106.

[Hinton, 2002] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

[Hinton and Salakhutdinov, 2006] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Hollocou et al., 2018] Hollocou, A., Bonald, T., and Lelarge, M. (2018). Multiple local community detection. *ACM SIGMETRICS Performance Evaluation Review*, 45(3):76–83.

[Hu and Liang, 2014] Hu, M. and Liang, H. (2014). A copula approach to assessing granger causality. *NeuroImage*, 100:125–134.

[Huang et al., 1998] Huang, N. E., Shen, Z., Long, S. R., Wu, M. C., Shih, H. H., Zheng, Q., Yen, N.-C., Tung, C. C., and Liu, H. H. (1998). The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London. Series A: mathematical, physical and engineering sciences*, 454(1971):903–995.

[Hyvärinen, 1997] Hyvärinen, A. (1997). *Independent component analysis by minimization of mutual information*. Helsinki University of Technology Helsinki.

[Hyvärinen et al., 2001] Hyvärinen, A., Karhunen, J., and Oja, E. (2001). Methods using time structure. *Independent Component Analysis, John Wiley & Sons, Inc*, pages 341–354.

[Hyvarinen and Morioka, 2016] Hyvarinen, A. and Morioka, H. (2016). Unsupervised feature extraction by time-contrastive learning and nonlinear ica. In *Advances in Neural Information Processing Systems*, pages 3765–3773.

[Hyvarinen and Morioka, 2017] Hyvarinen, A. and Morioka, H. (2017). Nonlinear ica of temporally dependent stationary sources. Proceedings of Machine Learning Research.

[Hyvärinen and Oja, 2000] Hyvärinen, A. and Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430.

[Hyvärinen and Pajunen, 1999] Hyvärinen, A. and Pajunen, P. (1999). Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3):429–439.

[Hyvarinen et al., 2018] Hyvarinen, A., Sasaki, H., and Turner, R. E. (2018). Nonlinear ica using auxiliary variables and generalized contrastive learning. *arXiv preprint arXiv:1805.08651*.

[Ienco and Interdonato, 2020] Ienco, D. and Interdonato, R. (2020). Deep multivariate time series embedding clustering via attentive-gated autoencoder. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 318–329. Springer.

[Ishwaran et al., 2008] Ishwaran, H., Kogalur, U. B., Blackstone, E. H., Lauer, M. S., et al. (2008). Random survival forests. *The annals of applied statistics*, 2(3):841–860.

[Jacob et al., 2009] Jacob, L., Obozinski, G., and Vert, J.-P. (2009). Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*, pages 433–440.

[Jang et al., 2016] Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

[Jing et al., 2019] Jing, B., Zhang, T., Wang, Z., Jin, Y., Liu, K., Qiu, W., Ke, L., Sun, Y., He, C., Hou, D., et al. (2019). A deep survival analysis method based on ranking. *Artificial intelligence in medicine*, 98:1–9.

[Jutten and Karhunen, 2003] Jutten, C. and Karhunen, J. (2003). Advances in nonlinear blind source separation. In *Proc. of the 4th Int. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003)*, pages 245–256.

[Kalderstam et al., 2013] Kalderstam, J., Edén, P., Bendahl, P.-O., Strand, C., Fernö, M., and Ohlsson, M. (2013). Training artificial neural networks directly on the concordance index for censored data using genetic algorithms. *Artificial intelligence in medicine*, 58(2):125–132.

[Kalman, 1960] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems.

[Katzman et al., 2018] Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., and Kluger, Y. (2018). Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC medical research methodology*, 18(1):24.

[Keogh et al., 2001] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. (2001). Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286.

[Keogh et al., 2005] Keogh, E., Lin, J., and Fu, A. (2005). Hot sax: Efficiently finding the most unusual time series subsequence. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. Ieee.

[Kersting et al., 2016] Kersting, K., Kriege, N. M., Morris, C., Mutzel, P., and Neumann, M. (2016). Benchmark data sets for graph kernels. http://graphkernels.cs.tu-dortmund.de.

[Khemakhem et al., 2019] Khemakhem, I., Kingma, D. P., and Hyvärinen, A. (2019). Variational autoencoders and nonlinear ica: A unifying framework. *arXiv preprint arXiv:1907.04809*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kingma et al., 2014] Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589.

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[Kipf et al., 2018] Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. (2018). Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*.

[Kipf and Welling, 2016] Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[Kohonen, 1982] Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69.

[Kondor and Borgwardt, 2008] Kondor, R. and Borgwardt, K. M. (2008). The skew spectrum of graphs. In *Proceedings of the 25th international conference on Machine learning*, pages 496–503. ACM.

[Kondor et al., 2009] Kondor, R., Shervashidze, N., and Borgwardt, K. M. (2009). The graphlet spectrum. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 529–536. ACM.

[Körner, 1989] Körner, T. W. (1989). *Fourier analysis*. Cambridge university press.

[Krishnan et al., 2017] Krishnan, R. G., Shalit, U., and Sontag, D. (2017). Structured inference networks for nonlinear state space models. In *Thirty-first aaai conference on artificial intelligence*.

[Kuehne et al., 2018] Kuehne, H., Richard, A., and Gall, J. (2018). A hybrid rnn-hmm approach for weakly supervised temporal action segmentation. *IEEE transactions on pattern analysis and machine intelligence*.

[Lang et al., 2018] Lang, X., Zheng, Q., Zhang, Z., Lu, S., Xie, L., Horch, A., and Su, H. (2018). Fast multivariate empirical mode decomposition. *IEEE Access*, 6:65521–65538.

[Latif et al., 2017] Latif, S., Rana, R., Qadir, J., and Epps, J. (2017). Variational autoencoders for learning latent representations of speech emotion. *arXiv preprint arXiv:1712.08708*.

[LeCun et al., 1989] LeCun, Y. et al. (1989). Generalization and network design strategies. In *Connectionism in perspective*, volume 19. Citeseer.

[Lei et al., 2017] Lei, Q., Yi, J., Vaculin, R., Wu, L., and Dhillon, I. S. (2017). Similarity preserving representation learning for time series analysis. *arXiv preprint arXiv:1702.03584*.

[Lelarge, 2018] Lelarge, M. (2018). Community detection with the triplet loss.

[Li et al., 2015] Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. (2015). Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

[Li et al., 2018] Li, Z., He, D., Tian, F., Chen, W., Qin, T., Wang, L., and Liu, T.-Y. (2018). Towards binary-valued gates for robust lstm training. *arXiv preprint arXiv:1806.02988*.

[Lin et al., 2007] Lin, J., Keogh, E., Wei, L., and Lonardi, S. (2007). Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144.

[Lin and Li, 2009] Lin, J. and Li, Y. (2009). Finding structural similarity in time series data using bag-of-patterns representation. In *International conference on scientific and statistical database management*, pages 461–477. Springer.

[Linsker, 1989] Linsker, R. (1989). An application of the principle of maximum information preservation to linear systems. In *Advances in neural information processing systems*, pages 186–194.

[Louizos et al., 2017] Louizos, C., Welling, M., and Kingma, D. P. (2017). Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*.

[Lozano et al., 2009] Lozano, A. C., Abe, N., Liu, Y., and Rosset, S. (2009). Grouped graphical granger modeling for gene expression regulatory networks discovery. *Bioinformatics*, 25(12):i110–i118.

[Lu et al., 2017] Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. (2017). The expressive power of neural networks: A view from the width. In *Advances in neural information processing systems*, pages 6231–6239.

[Luzhnica et al., 2019] Luzhnica, E., Day, B., and Lio, P. (2019). Clique pooling for graph classification. *arXiv preprint arXiv:1904.00374*.

[Ma et al., 2019] Ma, R., Miao, J., Niu, L., and Zhang, P. (2019). Transformed l_1 regularization for learning sparse deep neural networks. *arXiv preprint arXiv:1901.01021*.

[MacKay, 1996] MacKay, D. J. (1996). Maximum likelihood and covariant algorithms for independent component analysis. Technical report, Citeseer.

[Maddison et al., 2016] Maddison, C. J., Mnih, A., and Teh, Y. W. (2016). The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

[Makhzani et al., 2015] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

[Malhotra et al., 2017] Malhotra, P., TV, V., Vig, L., Agarwal, P., and Shroff, G. (2017). Timenet: Pretrained deep recurrent neural network for time series classification. *arXiv preprint arXiv:1706.08838*.

[Maron et al., 2019] Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. (2019). Provably powerful graph networks. In *Advances in Neural Information Processing Systems*, pages 2156–2167.

[Maron et al., 2018] Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. (2018). Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*.

[McGovern et al., 2011] McGovern, A., Rosendahl, D. H., Brown, R. A., and Droegemeier, K. K. (2011). Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction. *Data Mining and Knowledge Discovery*, 22(1-2):232–258.

[Merity et al., 2017] Merity, S., Keskar, N. S., and Socher, R. (2017). Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*.

[Merris, 1994] Merris, R. (1994). Laplacian matrices of graphs: a survey. *Linear algebra and its applications*, 197:143–176.

[Mhamdi et al., 2010] Mhamdi, F., Poggi, J., and Jaidane, M. (2010). Empirical mode decomposition for trend extraction: application to electrical data. In *19th International Conference on Computational Statistics*.

[Mikolov et al., 2013] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[Misra et al., 2016] Misra, I., Zitnick, C. L., and Hebert, M. (2016). Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pages 527–544. Springer.

[Monti et al., 2017] Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124.

[Muthén et al., 2015] Muthén, B., Muthén, L., and Asparouhov, T. (2015). Random coefficient regression.

[Nanni et al., 2017] Nanni, L., Ghidoni, S., and Brahnam, S. (2017). Handcrafted vs. non-handcrafted features for computer vision classification. *Pattern Recognition*, 71:158–172.

[Newman, 2013] Newman, M. E. (2013). Spectral methods for community detection and graph partitioning. *Physical Review E*, 88(4):042822.

[Neyshabur et al., 2017] Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. (2017). Exploring generalization in deep learning. In *Advances in neural information processing systems*, pages 5947–5956.

[Ng et al., 2011] Ng, A. et al. (2011). Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19.

[Ng and Jordan, 2002] Ng, A. Y. and Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848.

[Niepert et al., 2016] Niepert, M., Ahmed, M., and Kutzkov, K. (2016). Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023.

[Nikolentzos et al., 2017] Nikolentzos, G., Meladianos, P., and Vazirgiannis, M. (2017). Matching node embeddings for graph similarity. In *AAAI*, pages 2429–2435.

[Ordonez et al., 2011] Ordonez, P., Armstrong, T., Oates, T., and Fackler, J. (2011). Using modified multivariate bag-of-words models to classify physiological data. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 534–539. IEEE.

[Papamakarios et al., 2019] Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2019). Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*.

[Parikh et al., 2014] Parikh, N., Boyd, S., et al. (2014). Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239.

[Paszke et al., 2017] Paszke, A., Gross, S., Chintala, S., and Chanan, G. (2017). Pytorch.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

[Pearl et al., 2009] Pearl, J. et al. (2009). Causal inference in statistics: An overview. *Statistics surveys*, 3:96–146.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

[Peters et al., 2017] Peters, J., Janzing, D., and Schölkopf, B. (2017). *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press, Cambridge, MA, USA.

[Pineau, 2019] Pineau, E. (2019). Using laplacian spectrum as graph feature representation. *arXiv preprint arXiv:1912.00735*.

[Pineau and de Lara, 2019] Pineau, E. and de Lara, N. (2019). Variational recurrent neural networks for graph classification. In *Representation Learning on Graphs and Manifolds Workshop*.

[Pineau and Lelarge, 2018] Pineau, E. and Lelarge, M. (2018). Infocatvae: Representation learning with categorical variational autoencoders. *arXiv preprint arXiv:1806.08240*.

[Pineau et al., 2019] Pineau, E., Razakarivony, S., and Bonald, T. (2019). Seq2var: multivariate time series representation with relational neural networks and linear autoregressive model. In *International Workshop on Advanced Analysis and Learning on Temporal Data*, pages 126–140. Springer.

[Pineau et al., 2020a] Pineau, E., Razakarivony, S., and Bonald, T. (2020a). Time series source separation with slow flows. *Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models Workshop (ICML 2020)*.

[Pineau et al., 2020b] Pineau, E., Razakarivony, S., and Bonald, T. (2020b). Unsupervised ageing detection of mechanical systems on a causality graph. In *2020 19th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE.

[Plaut, 2018] Plaut, E. (2018). From principal subspaces to principal components with linear autoencoders. *arXiv preprint arXiv:1804.10253*.

[Polson et al., 2015] Polson, N. G., Scott, J. G., Willard, B. T., et al. (2015). Proximal algorithms in statistics and machine learning. *Statistical Science*, 30(4):559–581.

[Prado et al., 2006] Prado, R., Molina, F., and Huerta, G. (2006). Multivariate time series modeling and classification via hierarchical var mixtures. *Computational statistics & data analysis*, 51(3):1445–1462.

[Rabiner, 1989] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[Raiffa and Schlaifer, 1961] Raiffa, H. and Schlaifer, R. (1961). Applied statistical decision theory.

[Rameshkumar et al., 2013] Rameshkumar, A., Palanikumar, R., and Deepa, S. (2013). Laplacian matrix in algebraic graph theory. *Journal Impact Factor*, pages 0–489.

[Rao et al., 2009] Rao, C., Ray, A., Sarkar, S., and Yasar, M. (2009). Review and comparative evaluation of symbolic dynamic filtering for detection of anomaly patterns. *Signal, Image and Video Processing*, 3(2):101–114.

[Rezende and Mohamed, 2015] Rezende, D. J. and Mohamed, S. (2015). Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*.

[Rifai et al., 2011] Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 833–840. Omnipress.

[Robert and Casella, 2013] Robert, C. and Casella, G. (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.

[Romeu et al., 2015] Romeu, P., Zamora-Martínez, F., Botella-Rocamora, P., and Pardo, J. (2015). Stacked denoising auto-encoders for short-term time series forecasting. In *Artificial Neural Networks*, pages 463–486. Springer.

[Rubner et al., 2000] Rubner, Y., Tomasi, C., and Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[Sabour et al., 2017] Sabour, S., Frosst, N., and Hinton, G. E. (2017). Dynamic routing between capsules. In *Advances in neural information processing systems*, pages 3856–3866.

[Sak et al., 2014] Sak, H., Senior, A., and Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*.

[Sanh et al., 2018] Sanh, V., Wolf, T., and Ruder, S. (2018). A hierarchical multi-task approach for learning embeddings from semantic tasks. *arXiv preprint arXiv:1811.06031*.

[Santoro et al., 2017] Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. (2017). A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976.

[Sasaki et al., 2019] Sasaki, H., Takenouchi, T., Monti, R., and Hyvärinen, A. (2019). Robust contrastive learning and nonlinear ica in the presence of outliers. *arXiv preprint arXiv:1911.00265*.

[Saunshi et al., 2019] Saunshi, N., Plevrakis, O., Arora, S., Khodak, M., and Khandeparkar, H. (2019). A theoretical analysis of contrastive unsupervised representation learning. In *International Conference on Machine Learning*, pages 5628–5637.

[Saxena and Goebel, 2008] Saxena, A. and Goebel, K. (2008). Turbofan engine degradation simulation data set. *NASA Ames Prognostics Data Repository*.

[Scarselli et al., 2009] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

[Schumacher et al., 1994] Schumacher, M., Bastert, G., Bojar, H., Huebner, K., Olschewski, M., Sauerbrei, W., Schmoor, C., Beyerle, C., Neumann, R., and Rauschecker, H. (1994). Randomized 2 x 2 trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients. german breast cancer study group. *Journal of Clinical Oncology*, 12(10):2086–2093.

[Schwenk, 1973] Schwenk, A. (1973). Almost all trees are cospectral. *Harary, F., Ed. New Directions in the Theory of Graphs*, pages 275–307.

[Seabold and Perktold, 2010] Seabold, S. and Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. In *Proceedings of the 9th Python in Science Conference*, volume 57, page 61. Scipy.

[Sermanet et al., 2018] Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., Levine, S., and Brain, G. (2018). Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE.

[Shawe-Taylor et al., 2004] Shawe-Taylor, J., Cristianini, N., et al. (2004). *Kernel methods for pattern analysis*. Cambridge university press.

[Shen et al., 2007] Shen, H., Jegelka, S., and Gretton, A. (2007). Fast kernel ica using an approximate newton method. In *Artificial Intelligence and Statistics*, pages 476–483.

[Shervashidze and Borgwardt, 2009] Shervashidze, N. and Borgwardt, K. (2009). Fast subtree kernels on graphs. In *Advances in neural information processing systems*, pages 1660–1668.

[Shervashidze et al., 2011] Shervashidze, N., Schweitzer, P., Leeuwen, E. J. v., Mehlhorn, K., and Borgwardt, K. M. (2011). Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561.

[Shervashidze et al., 2009] Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., and Borgwardt, K. (2009). Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, pages 488–495.

[Shi et al., 2019] Shi, L., Zhang, Y., Cheng, J., and Lu, H. (2019). Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12026–12035.

[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 processing magazine*, 30(3):83–98.

[Shuman et al., 2016] Shuman, D. I., Ricaud, B., and Vandergheynst, P. (2016). Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis*, 40(2):260–291.

[Smith et al., 1962] Smith, G. L., Schmidt, S. F., and McGee, L. A. (1962). *Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle*. National Aeronautics and Space Administration.

[Solo, 2008] Solo, V. (2008). On causality and mutual information. In *2008 47th IEEE Conference on Decision and Control*, pages 4939–4944. IEEE.

[Sorrenson et al., 2020] Sorrenson, P., Rother, C., and Köthe, U. (2020). Disentanglement by nonlinear ica with general incompressible-flow networks (gin). *International Conference on Learning Representation*.

[Sperduti and Starita, 1997] Sperduti, A. and Starita, A. (1997). Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735.

[Sprekeler et al., 2014] Sprekeler, H., Zito, T., and Wiskott, L. (2014). An extension of slow feature analysis for nonlinear blind source separation. *The Journal of Machine Learning Research*, 15(1):921–947.

[Steck et al., 2008] Steck, H., Krishnapuram, B., Dehing-Oberije, C., Lambin, P., and Raykar, V. C. (2008). On ranking in survival analysis: Bounds on the concordance index. In *Advances in neural information processing systems*, pages 1209–1216.

[Stögbauer et al., 2004] Stögbauer, H., Kraskov, A., Astakhov, S. A., and Grassberger, P. (2004). Least-dependent-component analysis based on mutual information. *Physical Review E*, 70(6):066123.

[Taleb and Jutten, 1999] Taleb, A. and Jutten, C. (1999). Source separation in post-nonlinear mixtures. *IEEE Transactions on signal Processing*, 47(10):2807–2820.

[Tank et al., 2018] Tank, A., Covert, I., Foti, N., Shojaie, A., and Fox, E. (2018). Neural granger causality for nonlinear time series. *arXiv preprint arXiv:1802.05842*.

[Therneau and Grambsch, 2000] Therneau, T. M. and Grambsch, P. M. (2000). The cox model. In *Modeling survival data: extending the Cox model*, pages 39–77. Springer.

[Thomas, 1996] Thomas, L. (1996). Monitoring long-term population change: why are there so many analysis methods? *Ecology*, 77(1):49–58.

[Thomas et al., 2016] Thomas, O., Dutta, R., Corander, J., Kaski, S., and Gutmann, M. U. (2016). Likelihood-free inference by ratio estimation. *arXiv preprint arXiv:1611.10242*.

[Tipping and Bishop, 1999] Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622.

[Tishby et al., 2000] Tishby, N., Pereira, F. C., and Bialek, W. (2000). The information bottleneck method. *arXiv preprint physics/0004057*.

[Toda and Phillips, 1994] Toda, H. Y. and Phillips, P. C. (1994). Vector autoregression and causality: a theoretical overview and simulation study. *Econometric reviews*, 13(2):259–285.

[Tomczak and Welling, 2018] Tomczak, J. M. and Welling, M. (2018). Vae with a vampprior. In *21st International Conference on Artificial Intelligence and Statistics, AISTATS 2018*.

[Tong et al., 1991] Tong, L., Liu, R.-W., Soon, V. C., and Huang, Y.-F. (1991). Indeterminacy and identifiability of blind identification. *IEEE Transactions on circuits and systems*, 38(5):499–509.

[Tschannen et al., 2018] Tschannen, M., Bachem, O., and Lucic, M. (2018). Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*.

[Turner and Sahani, 2007] Turner, R. and Sahani, M. (2007). A maximum-likelihood interpretation for slow feature analysis. *Neural computation*, 19(4):1022–1038.

[Uthayakumar et al., 2018] Uthayakumar, J., Vengattaraman, T., and Dhavachelvan, P. (2018). A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. *Journal of King Saud University-Computer and Information Sciences*.

[Vahdat and Kautz, 2020] Vahdat, A. and Kautz, J. (2020). Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*.

[van den Oord et al., 2017] van den Oord, A., Vinyals, O., et al. (2017). Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315.

[Van Der Malsburg, 1986] Van Der Malsburg, C. (1986). Frank rosenblatt: principles of neurodynamics: perceptrons and the theory of brain mechanisms. In *Brain theory*, pages 245–248. Springer.

[Verma and Zhang, 2017] Verma, S. and Zhang, Z.-L. (2017). Hunt for the unique, stable, sparse and fast feature learning on graphs. In *Advances in Neural Information Processing Systems*, pages 88–98.

[Vincent et al., 2010] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408.

[Vishwanathan et al., 2010] Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. (2010). Graph kernels. *Journal of Machine Learning Research*, 11(Apr):1201–1242.

[Wang et al., 2019] Wang, H., Pang, G., Shen, C., and Ma, C. (2019). Unsupervised representation learning by predicting random distances. *arXiv preprint arXiv:1912.12186*.

[Wang and Gupta, 2015] Wang, X. and Gupta, A. (2015). Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802.

[Wei et al., 2017] Wei, X., Xu, L., Cao, B., and Yu, P. S. (2017). Cross view link prediction by learning noise-resilient representation consensus. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1611–1619.

[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 Informatsia*, 2(9):12–16.

[Weng, 2018] Weng, L. (2018). Meta-learning: Learning to learn fast. *lilianweng.github.io/lil-log*.

[Weng, 2019] Weng, L. (2019). Self-supervised representation learning. *lilianweng.github.io/lil-log*.

[Williams and Seeger, 2001] Williams, C. K. and Seeger, M. (2001). Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688.

[Wilson and Zhu, 2008] Wilson, R. C. and Zhu, P. (2008). A study of graph spectra for comparing graphs and trees. *Pattern Recognition*, 41(9):2833–2841.

[Wiskott and Sejnowski, 2002] Wiskott, L. and Sejnowski, T. J. (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770.

[Wu et al., 2007] Wu, Z., Huang, N. E., Long, S. R., and Peng, C.-K. (2007). On the trend, detrending, and variability of nonlinear and nonstationary time series. *Proceedings of the National Academy of Sciences*, 104(38):14889–14894.

[Wu et al., 2020] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

[Xinyi and Chen, 2018] Xinyi, Z. and Chen, L. (2018). Capsule graph neural network. *International Conference on Learning Representations*.

[Xiong and Zhou, 2013] Xiong, J. and Zhou, T. (2013). A kalman-filter based approach to identification of time-varying gene regulatory networks. *PloS one*, 8(10).

[Xiong and Yeung, 2002] Xiong, Y. and Yeung, D.-Y. (2002). Mixtures of arma models for model-based time series clustering. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 717–720. IEEE.

[Xu and Tian, 2015] Xu, D. and Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193.

[Xu et al., 2015] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

[Xu et al., 2018] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018). How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

[Xu et al., 2017] Xu, W., Sun, H., Deng, C., and Tan, Y. (2017). Variational autoencoder for semi-supervised text classification. In *AAAI*, pages 3358–3364.

[Yao et al., 2016] Yao, Q., Kwok, J. T., Gao, F., Chen, W., and Liu, T.-Y. (2016). Efficient inexact proximal gradient algorithm for nonconvex problems. *arXiv preprint arXiv:1612.09069*.

[Yazdi et al., 2018] Yazdi, S. V., Douzal-Chouakria, A., Gallinari, P., and Moussallam, M. (2018). Time warp invariant dictionary learning for time series clustering: application to music data stream analysis. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 356–372. Springer.

[Ye and Keogh, 2009] Ye, L. and Keogh, E. (2009). Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 947–956.

[Ying et al., 2018] Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pages 4800–4810.

[You et al., 2018] You, J., Ying, R., Ren, X., Hamilton, W., and Leskovec, J. (2018). Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*, pages 5694–5703.

[Yu et al., 2011] Yu, C.-N., Greiner, R., Lin, H.-C., and Baracos, V. (2011). Learning patient-specific cancer survival distributions as a sequence of dependent regressors. In *Advances in Neural Information Processing Systems*, pages 1845–1853.

[Yuan and Zhang, 2013] Yuan, X.-T. and Zhang, T. (2013). Truncated power method for sparse eigenvalue problems. *Journal of Machine Learning Research*, 14(Apr):899–925.

[Zaremba et al., 2014] Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

[Zhang et al., 2016] Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.

[Zhang et al., 2018] Zhang, M., Cui, Z., Neumann, M., and Chen, Y. (2018). An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[Zhao et al., 2017] Zhao, S., Song, J., and Ermon, S. (2017). Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*.

[Zhou et al., 2017] Zhou, C., Liu, Y., Liu, X., Liu, Z., and Gao, J. (2017). Scalable graph embedding for asymmetric proximity. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[Zhu et al., 2007] Zhu, Y., Wu, D., and Li, S. (2007). A piecewise linear representation method of time series based on feature points. In *International Conference on Knowledge-based and Intelligent Information and Engineering Systems*, pages 1066–1072. Springer.

ECOLE DOCTORALE

Titre : Contributions à l'apprentissage de représentation de séries temporelles multivariées et de graphes

Mots clés : séries temporelles multivariées; graphes; réseaux de neurones

Résumé :
Dans cette thèse, nous nous sommes intéressés à l'apprentissage de représentation de séries temporelles multivariées (STM) et de graphes. STM et graphes sont des objets complexes qui ont des caractéristiques les rendant difficilement traitables par des algorithmes standards de machine learning. Par exemple, ils peuvent avoir des tailles variables et ont des alignements non-triviaux, qui empêchent l'utilisation de métriques standards. Il est alors nécessaire de trouver pour les échantillons observés (STM ou graphes) une représentation alternative qui les rend comparables. Les contributions de ma thèse sont un ensemble d'analyses, d'approches pratiques et de résultats théoriques présentant des apprentissages de représentation de STM et de graphes.
Deux méthodes de représentation de STM sont dédiées au suivi d'état caché de systèmes mécaniques. La première propose une représentation "model-based" appelée Sequence-to-graph (Seq2Graph). Seq2Graph se base sur l'hypothèse que les données observées ont été générées par un modèle causal simple, dont l'espace des paramètres sert d'espace de représentation. La seconde méthode propose une approche générique de détection de tendances dans des séries temporelles, appelée Contrastive Trend Estimation (CTE). Une preuve d'identifiabilité, une extension à des problèmes d'analyse de survie et une discussion sur le cas des tendances bruitées sont proposées pour mieux comprendre les capacités et limites du modèle.
Deux méthodes de représentation de graphes pour la classification sont aussi proposées. Une première propose de voir les graphes comme des séquences de nœuds et donc de les traiter avec un outil standard de représentation de séquences : un réseau de neurones récurrents. Une seconde méthode propose une analyse théorique et pratique du spectre du Laplacien pour la classification de graphes.

Title : Contributions to representation learning of multivariate time series and graphs

Keywords : multivariate time series; graphs; neural networks

Abstract :
In this thesis, we are interested in learning representations of multivariate time series (MTS) and graphs. MTS and graphs are particular objects that do not directly match standard requirements of ML algorithms. They can have variable size and non-trivial alignment, such that comparing two MTS or two graphs with standard metrics is generally not relevant. Hence, particular representations are required for their analysis using ML approaches. The contributions of this thesis consist of practical and theoretical results presenting new MTS and graphs representation learning frameworks.
Two MTS representation learning frameworks are dedicated to the ageing detection of mechanical systems. First, we propose a model-based MTS representation learning framework called Sequence-to-graph (Seq2Graph). Seq2Graph assumes that the data we observe has been generated by a model whose graphical representation is a causality graph. It then represents, using an appropriate neural network, the sample on this graph. From this representation, when it is appropriate, we can find interesting information about the state of the studied mechanical system. Second, we propose a generic trend detection method called Contrastive Trend Estimation (CTE). CTE learns to classify pairs of samples with respect to the monotony of the trend between them. We show that using this method, under few assumptions, we identify the true state underlying the studied mechanical system, up-to monotone scalar transform. We also show how to extend our approach to survival analysis problems.
Two graph representation learning frameworks are dedicated to the classification of graphs. First, we propose to see graphs as sequences of nodes and create a framework based on recurrent neural networks to represent and classify them. Second, we analyze a simple baseline feature for graph classification: the Laplacian spectrum. We show that this feature matches minimal requirements to classify graphs when all the meaningful information is contained in the structure of the graphs.

Institut Polytechnique de Paris
91120 Palaiseau, France