Representation learning for relational data

Ludovic dos Santos

▶ **To cite this version:**

Ludovic dos Santos. Representation learning for relational data. Machine Learning [cs.LG]. Université Pierre et Marie Curie - Paris VI, 2017. English. NNT : 2017PA066480 . tel-01803188

HAL Id: tel-01803188
https://theses.hal.science/tel-01803188

Submitted on 30 May 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ PIERRE ET MARIE CURIE
École Doctorale Informatique Télécommunications et Électronique (Paris)
Laboratoire d'Informatique de Paris 6

DOCTORAL THESIS

Representation Learning for Relational Data

Author:
Ludovic DOS SANTOS

Supervisor:
Patrick GALLINARI
Co-Supervisor:
Benjamin PIWOWARSKI

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Computer Science

Jury Members:

Mr. Thierry ARTIÈRES	Reporter
Mr. Younes BENNANI	Examiner
Mr. Patrick GALLINARI	Supervisor
Mr. Rémi GILLERON	Reporter
Mrs. Marie-Jeanne LESOT	Examiner
Mr. Benjamin PIWOWARSKI	Co-Supervisor

Wednesday 13th December, 2017

Abstract

The increasing use of social and sensor networks generates a large quantity of data that can be represented as complex graphs. There are many tasks from information analysis, to prediction and retrieval one can imagine on those data where relation between graph nodes should be informative.

In this thesis, we proposed different models for three different tasks:

- Graph node classification
- Relational time series forecasting
- Collaborative filtering

All the proposed models use the representation learning framework in its deterministic or Gaussian variant.

First, we proposed two algorithms for the heterogeneous graph labeling task, one using deterministic representations and the other one Gaussian representations. Contrary to other state of the art models, our solution is able to learn edge weights when learning simultaneously the representations and the classifiers.

Second, we proposed an algorithm for relational time series forecasting where the observations are not only correlated inside each series, but also across the different series. We use Gaussian representations in this contribution. This was an opportunity to see in which way using Gaussian representations instead of deterministic ones was profitable.

At last, we apply the Gaussian representation learning approach to the collaborative filtering task. This is a preliminary work to see if the properties of Gaussian representations found on the two previous tasks were also verified for the ranking one. The goal of this work was to then generalize the approach to more relational data and not only bipartite graphs between users and items.

Contents

List of Figures **1**

List of Tables **3**

1 Introduction **5**
1.1 Context . . . 5
1.1.1 Tasks . . . 6
1.1.2 Types of Graphs . . . 6
1.1.3 Outline . . . 7
1.2 Tasks Studied during the thesis . . . 8
1.2.1 The Classification Task . . . 8
1.2.2 The Forecasting Task . . . 8
1.2.3 The Ranking Task . . . 9
1.3 Learning Representations . . . 9
1.4 Contributions . . . 10
1.4.1 Learning Deterministic Representations for Heterogeneous Graph Node Classification (Chapter 4) . . . 11
1.4.2 Learning Gaussian Representations . . . 11
Learning Gaussian Representations for Heterogeneous Graph Node Classification (Chapter 5) . . . 11
Learning Gaussian Representations for Relational Time Series Forecasting (Chapter 6) . . . 12
Learning Gaussian Representations for Ranking (Chapter 7) . . . 12
1.5 Thesis Organization . . . 12

2 Learning Representations for Relational Data **15**
2.1 Introduction . . . 15
2.2 Learning Representations in Graphs . . . 17
2.2.1 Inductive and Transductive Algorithms . . . 17
Induction . . . 17
Transduction . . . 18
2.2.2 Supervised, Unsupervised and Semi-Supervised Learning . . . 18
Unsupervised Learning . . . 18
Supervised Learning . . . 19
Semi-Supervised Learning . . . 19
Our Contributions . . . 19
2.3 Learning Deterministic Representations . . . 19
2.3.1 Unsupervised Models . . . 20
Learning from Context . . . 20

2.3.2 Supervised and Semi-Supervised Models . . . 22
Learning Nodes and Relationship Representations in Knowledge Bases 22
From Unlabeled to Labeled Data in Classification . . . 24
2.3.3 Other Applications of Representation Learning . . . 25
2.4 Capturing uncertainties in representations . . . 26
Bayesian Approaches . . . 26
Direct Approaches . . . 27
2.5 Conclusion . . . 29

I Learning Deterministic Representations and Application to Classification 33

3 State of the Art: Graph Node Classification 37
3.1 Introduction . . . 37
3.2 Graph Node Classification . . . 38
3.2.1 Simple Transductive Model for Graph Node Classification . . . 39
3.2.2 Other Graph Node Classification Works . . . 39
3.3 Conclusion . . . 41

4 Learning Deterministic Representations for Classification 43
4.1 Introduction . . . 43
4.2 Learning Representation for Graph Node Classification . . . 45
4.2.1 Model . . . 45
Loss Function . . . 45
Classifier . . . 46
Transductive graph model . . . 46
4.2.2 Prior Parameters and Learning Algorithms . . . 47
Prior Parameters . . . 47
Learned Relation Specific Parameters . . . 47
4.2.3 Algorithm . . . 50
4.2.4 Experiments . . . 50
Datasets . . . 50
Construction of the LastFM Datasets . . . 53
Evidence For Heterogeneous Node Reciprocal Influence . . . 53
Comparison With Other Models . . . 58
Evaluation Measures And Protocol . . . 59
Results . . . 61
Importance Of The Relations' Weights . . . 65
Label correlation on the LastFM2 Dataset . . . 67
4.3 Conclusion . . . 68

II Learning Gaussian Representations and Applications 71

5 Learning Gaussian Representations for Classification 75
5.1 Introduction . . . 75
5.2 Graph Node Classification with Gaussian Embeddings . . . 76

5.2.1 Model . . . 76
Loss Function . . . 77
Classifier . . . 77
Graph Embedding . . . 79
Prior Parameters and Learned Relation Specific Parameters . . . 79
Algorithm . . . 80
5.2.2 Experiments . . . 80
Datasets . . . 80
Results . . . 81
Qualitative Discussion . . . 83
5.3 Conclusion . . . 84

6 Learning Gaussian Representations for Relational Time Series Forecasting 87
6.1 Introduction . . . 87
6.1.1 Relational Time Series . . . 87
6.1.2 Contribution . . . 88
6.2 Related Work . . . 89
6.3 Relational Time Series Forecasting with Gaussian Embeddings . . . 91
6.3.1 Model . . . 91
Notations and Task Formal Description . . . 91
6.3.2 Informal description . . . 91
Model Definition . . . 92
Impact of minimizing the KL-divergence on predicted values . . . 95
Inference and Time Complexity . . . 96
Variants . . . 96
6.3.3 Experiments . . . 97
Datasets . . . 97
Baselines . . . 97
Experimental protocol . . . 98
Results . . . 98
6.4 Conclusion . . . 101

7 Learning Gaussian Representations for Collaborative Filtering 103
7.1 Introduction . . . 103
7.1.1 Recommender Systems and Uncertainty . . . 103
7.1.2 Contribution . . . 105
7.2 Learning Gaussian Embeddings for Collaborative Filtering . . . 105
7.2.1 Model . . . 105
The Gaussian Embeddings Ranking Model . . . 106
Loss function . . . 108
Ordering items . . . 109
7.2.2 Experiments . . . 109
Analysis . . . 111
7.3 Conclusion . . . 117

III Conclusion **119**

8 Conclusion et perspectives **121**
8.1 Conclusion . . . 121
8.1.1 Contributions . . . 121
Graph Node Classification . . . 121
Relational Time Series Forecasting . . . 122
Collaborative Filtering . . . 122
8.1.2 Learning Hyperparameters . . . 122
8.1.3 From Deterministic to Gaussian Representations . . . 122
8.2 Perspectives . . . 123
8.2.1 Classification Task . . . 123
8.2.2 Forecasting Task . . . 123
8.2.3 Ranking Task . . . 124
8.2.4 Learning Gaussian Embeddings for Knowledge Bases . . . 124

Bibliography **125**

List of Figures

1.1 Example of a heterogeneous multi-label graph with authors, comments, photos, videos, etc. link together by different types of relations such as friendship, colleagues, video similarity, etc. 7
1.2 A tree-like structure that shows the dependencies between the different chapters of the thesis manuscript. 13

2.1 Drawing of representation learning for relational data where nodes are projected from the initial graph (left) to the common latent space (right) where classifiers are learned. 17
2.2 Example of a low $D_{\mathrm{KL}}(Z_j||Z_i)$. 28
2.3 Learned Gaussian representations with diagonal variances, from (Vilnis et al., 2015). The first letter of each word indicating the position of its mean. . . 29
2.4 Learned two dimensional Gaussian representations of nodes on the Cora dataset from (Bojchevski et al., 2017). Color indicates the class label. 30

3.1 Representation of the LastFM network with users, tracks, albums and artists. The different relations are represented with different types of lines. 38

4.1 Plots illustrating the inter-dependencies between labels of two node types for the LastFM2 dataset. Each plot shows $P(Y_{t_1}|X_{t_2})$ for a specific variable couple (Y_{t_1}, X_{t_2}). The values X_{t_2} and Y_{t_1} have been reordered for clarity (see text). The x axis corresponding to variable X_{t_2} is on the bottom left of each plot, and the y axis corresponding to the Y_{t_1} is on the bottom right. 55
4.2 Plots illustrating the inter-dependencies between labels on the DBLP corpus. A gray square means that there is no relation between the corresponding node types 56
4.3 Plots illustrating the inter-dependencies between labels on the FlickR corpus 56
4.4 Plots illustrating the inter-dependencies between labels on the LastFM2 corpus 57
4.5 Plot of the average of the cumulative sum of conditional probabilities $P(Y_{t_1}|X_{t_2})$ for all relation types on the LastFM2 dataset. The $P(Y_{t_1}|X_{t_2})s$ are in decreasing order on the x axis. $x = 40\%$ means that 40% of the conditional probability values have been considered and the corresponding cumulative value is plotted on the y axis. 58
4.6 Plots of the average of the cumulative sum of conditional probabilities $P(Y_{t_1}|X_{t_2})$ (with cumulative percentages on the x-axis) for all relation types on all datasets. 59
4.7 Evolution of w_r values over training steps for all relations r involving users for LastFM2. At convergence $w_{\text{user}\to\text{user}} = 0.22$, $w_{\text{user}\to\text{track}} = 0.28$, $w_{\text{user}\to\text{album}} = 0.25$, $w_{\text{user}\to\text{artist}} = 0.25$. 67

4.8 LastFM2 corpus: Graph of the cosine between the 30 classifiers having the same string-wise label (e.g. "pop") for artists and tracks. For example, line 15 corresponds to *pop_artist* and column 15 to *pop_track*. The lighter the square, the higher the scalar product. 68
4.9 LastFM2 corpus: Graph of the cosine between the classifiers having the same string-wise label (e.g. "pop") for two different node types. 69

5.1 Qualitative results for the model HCGE(Δ_{EV},S) on the LastFM dataset with 50% of the dataset used for train. In Figure 5.1b, we computed Gaussian kernel density to show high density regions in the plot. 85

6.1 Representation of relational time series as a a graph, where nodes correspond to series, and edges correspond to a prior relation between two series. In this relational setting the task considered in this chapter is forecasting. 88
6.2 The Gaussian embedding $Z_i^{(t)}$, the dynamic function h and the decoding function f are learned. We compute $Z_i^{(t+1)}$ and $Z_i^{(t+2)}$ using h and $Z_i^{(t)}$. Then we compute $X_i^{(t+1)}$ and $X_i^{(t+2)}$ from $Z_i^{(t+1)}$ and $Z_i^{(t+2)}$ respectively using f. . 92
6.3 Comparison of RMSE at T+1 on the four datasets between baselines and our proposed model (RDG) on the prediction task. $\text{RDG}_{k,l}$ corresponds to the variant with losses (Δ_{De_k},Δ_{Dy_l}). 99
6.4 Comparison of RMSE from T+1 to T+5 on GL-T between baselines and our proposed model (RDG) on the prediction task. 99
6.5 Forecasts on GFT (two different time series of the dataset) with the $\text{RDG}_{2,2}$ model showing its range of confidence: $E(f(Z^{(t)})) \pm \text{var}(f(Z^{(t)}))$. Prediction at $25 + n$ corresponds to $f(h^n(Z^{(25)}))$. 100

7.1 User (top) and item (bottom) learned first component representation plots ($\sigma_{\bullet 1}$ vs $\mu_{\bullet 1}$) for the training set with 50 items per user from the MovieLens dataset and a representation dimension of 50. 112
7.2 Subset of user learned components representation plots ($\sigma_{\bullet k}$ vs $\mu_{\bullet k}$ with different k) for the training set with 50 items per user from the MovieLens dataset and a representation dimension of 50. 113
7.3 Subset of item learned components representation plots ($\sigma_{\bullet k}$ vs $\mu_{\bullet k}$ with different k) for the training set with 50 items per user from the MovieLens dataset and a representation dimension of 50. 114
7.4 Violin (density) plots for the variance of users ($\sigma_{\bullet k}$) for different latent space dimensions on the MovieLens dataset. Three training set sizes settings (truncation levels) were used: 10 (top), 20 (middle) and 50 (bottom) 115
7.5 Violin (density) plots for the variance of items ($\sigma_{\bullet k}$) for different latent space dimensions on the MovieLens dataset. Three training set sizes settings (truncation levels) were used: 10 (top), 20 (middle) and 50 (bottom) 116

List of Tables

4.1 Model notations . . . 45
4.2 Statistics for the datasets . . . 52
4.3 Conditional entropies $H(Y|X)$ for the lastFM2 corpus, X and Y being neighbors in the LastFM graph. Lower values mean higher dependency between the two variables . . . 54
4.4 Conditional entropies $H(Y|X)$ for the DBLP corpus . . . 55
4.5 Conditional entropies $H(Y|X)$ for the FlickR corpus. . . . 56
4.6 Hyperparameters of the model . . . 60
4.7 P@1 results of the model LaHNet and baselines on DBLP . . . 62
4.8 P@k results of the model LaHNet and baselines on FlickR . . . 62
4.9 P@k results of the model LaHNet and baselines on LastFM1 . . . 63
4.10 P@k results of the model LaHNet and baselines on LastFM2 . . . 63
4.11 P@k results of the model LaHNet and baselines on IMDB . . . 64
4.12 Comparison of the evolution of P@k performance between LaHNet and the best baseline when increasing the training + validation size (TVS). . . . 64
4.13 Effect of learning the relation-specific weights w_r for FlickR. Performance is expressed as micro and macro precisions as indicated, performance for individual relations is micro precision - see text for further explanation. . . . 65
4.14 Effect of learning the relation-specific weights w_r for LastFM1. Performance is expressed as micro and macro precisions as indicated, performance for individual relations is micro precision - see text for further explanation. . . . 66
4.15 Effect of learning the relation-specific weights w_r for LastFM2. Performance is expressed as micro and macro precisions as indicated, performance for individual relations is micro precision - see text for further explanation. . . . 66

5.1 P@1 results of the model HCGE and baselines on DBLP . . . 81
5.2 P@k results of the model HCGE and baselines on FlickR . . . 82
5.3 P@k results of the model HCGE and baselines on LastFM . . . 83

6.1 RMSE at $T + 1$ on the four datasets. . . . 100

7.1 Datasets statistics by truncation level (10, 20 or 50 ratings by user in the training set). . . . 109
7.2 Collaborative Ranking results. nDCG values ($\times 100$) at different truncation levels are shown within the main columns, which are split based on the amount of training ratings. . . . 111

Chapter 1

Introduction

Contents

1.1 Context 5
1.1.1 Tasks 6
1.1.2 Types of Graphs 6
1.1.3 Outline 7
1.2 Tasks Studied during the thesis 8
1.2.1 The Classification Task 8
1.2.2 The Forecasting Task 8
1.2.3 The Ranking Task 9
1.3 Learning Representations 9
1.4 Contributions 10
1.4.1 Learning Deterministic Representations for Heterogeneous Graph Node Classification (Chapter 4) 11
1.4.2 Learning Gaussian Representations 11
1.5 Thesis Organization 12

1.1 Context

Since the early 2000s there has been a great development and use of on-line services, such as On-line Social Networks (Facebook, LinkedIn, ...), E-Commerce (Amazon, eBay, ...), Video or Music On Demand (Netflix, YouTube, LastFM, ...), and Open Data websites (city traffic, meteorology, parking use, ...). The use of those services by a large number of users generates an impressive amount of data such as ratings, likes and traffic data. This data informs us about many aspects of human behaviors, such as users habits, tastes and friendships.

A lot of user-related collected data can be represented as graphs with various types of entities (such as users, photos, items, streets) interacting with each other. Since these interactions may be used as a complementary source of information, these entities should not be considered alone. They are often linked together through various kinds of relations such as friendship, ratings, geographical proximity. In the following, we will talk of a graph or relational data to refer to this type of linked data.

Such graphs are an important source of knowledge and can be relevant to help inferring missing information. This kind of data is generally very large, from some thousand

connected nodes to billions. That is why it is crucial to use machine learning techniques to solve the various possible tasks relevant for this data.

1.1.1 Tasks

Below are some concrete examples of generic tasks relevant for graph and relational data:

- **Node graph classification**: given a new track on LastFM[1], infer the category (e.g. music style) of a track based on who has uploaded it and who has listened to it;
- **Graph classification**: given a graph representing a protein, predict its category, and thus its properties, based on how molecules are arranged;
- **Link prediction**: given a partial graph, infer its missing edges;
- **Information diffusion**: in a social network like Twitter, predict how information spreads through the nodes of the graph;
- **Clustering**: in a social network, detect user communities based on their interests (sport, science, politics, ...);
- **Regression**: knowing the traffic in some streets of the city, estimate the traffic in a specific non-observed street (or in the future);
- **Learning to rank**: rank movies for a Netflix user, based on past ratings and meta information;
- **Anomaly detection**: given a network of different sensors in an industrial process production, predict when the risk of a product being damaged is high;

1.1.2 Types of Graphs

In this thesis, we focus on node specific tasks, and distinguish two kinds of graphs on which inference and learning can be conducted:

- Attributed graphs are graphs where each node is associated with attributes such as raw images, product indicators or any other features.
- Non attributed graphs are graphs where no features are associated to nodes. Only edges between nodes are given.

In a graph, nodes can have different types, this corresponds to different classes of entities in the real world (e.g. user, song, etc.). There exists different types of graphs based on the different types of nodes and relations between them: homogeneous or heterogeneous, mono or multi-relational.

- Homogeneous graphs correspond to graphs where there is only one type of node and one type of relation between them.
- Heterogeneous graphs correspond to graphs where there exists different types of nodes and different types of relations between different types of nodes.

[1] LastFM is a social network for music.

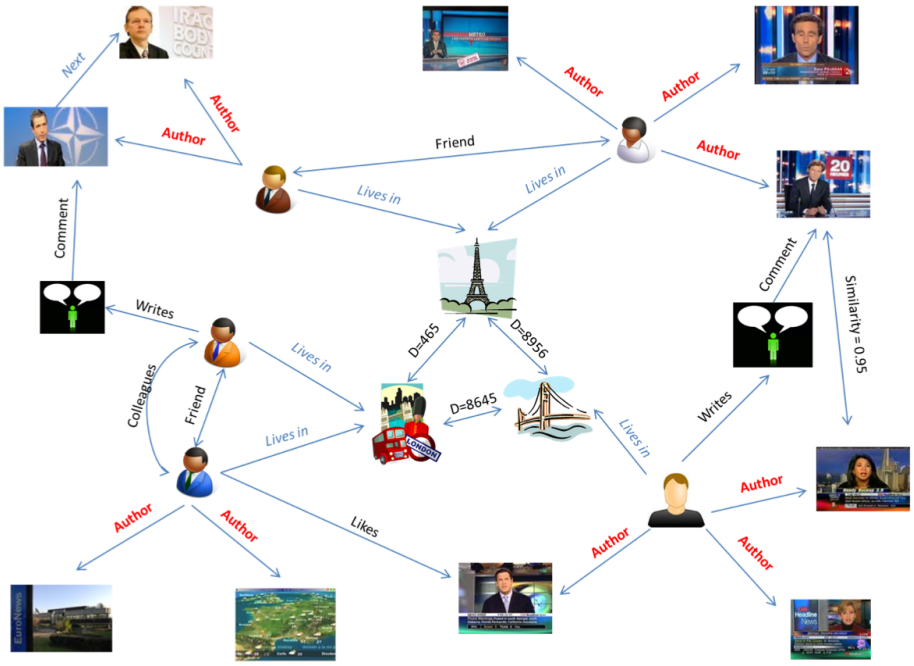


FIGURE 1.1: Example of a heterogeneous multi-label graph with authors, comments, photos, videos, etc. link together by different types of relations such as friendship, colleagues, video similarity, etc.

- Mono-relational graphs correspond to graphs where there is at most one relation between two given nodes.
- Multi-relational graphs correspond to graphs where two nodes can be linked by more than one relation.

We did not specifically focus on the latter type of graphs in our works, even if our models are able to handle multi-relational graphs, none of our datasets is multi-relational.

Figure 1.1 gives an example of a heterogeneous graph with multiple labels associated to the nodes.

In this thesis we focus on non attributed graphs, and on how to learn latent representations based only on the graph structure.

Machine learning techniques have mainly been developed under the assumption that entities are independent identically distributed (i.i.d.), which is clearly not our case since entities depend on each others. Those techniques cannot take into account inter-dependences and there is a real need to develop specific techniques for graph data. That is the main purpose of this thesis.

1.1.3 Outline

The remaining of this chapter is organized as follows: in Section 1.2 we describe the three different tasks studied during the thesis; in Section 1.3 we give an informal introduction to representation learning which is the framework used in all of the models we developed

during the thesis; in Section 1.4 we outline our four contributions. Lastly, Section 1.5 gives the dependencies between the chapters of the thesis.

1.2 Tasks Studied during the thesis

Among the different possible tasks, we focused on three of them, namely classification, forecasting and ranking. In this section, we define them succinctly. Those tasks are detailed in the corresponding chapters. A large part of the thesis work has focused on the classification task (see Section 1.2.1), firstly using deterministic representations, and then Gaussian ones (see Section 1.3). Given the positive results with Gaussian representations, we then have applied Gaussian representations learning on other tasks, namely forecasting (see Section 1.2.2) and ranking (see Section 1.2.3).

1.2.1 The Classification Task

The task of classification consists in identifying the category to which a new instance belongs, based on a training set of observed instances. *Spam filtering* is a typical example of classification, where we want to distinguish junk emails (advertisers, hackers, ...) from others.

In this thesis, we tackle more particularly the graph node multi-label classification task. The difference with classical classification is that the instances are structured through a relational graph, and that we thus use a transductive semi-supervised framework.

For example, in the case of LastFM (one of the datasets used in our contributions) different types of nodes (users, tracks, albums and artists) interact through seven different type of relations such as *friendship*, *most listened tracks*, *authorship*, etc. For example, a track can have the three labels *pop*, *rock* and *live music*, but labels can be missing for some nodes (users, tracks, albums or artists). The task consists in inferring the labels of those nodes using the known labels and the relations that exist between the different types of nodes.

In the context of the thesis, our aim is to leverage representation learning techniques to tackle the graph node multi-label classification task for heterogeneous data, i.e. where both nodes and edges can be of different types. This task is known as heterogeneous graph nodes classification. Note that different sets of labels can be associated to each type of node.

1.2.2 The Forecasting Task

The forecasting task consists in predicting future or missing values, based on the past, of multi-variate time series. A practical example is inventory optimization: knowing how much a company has sold items for the last weeks, we want to predict how much items the company will sell for the coming weeks to produce the right amount.

In this thesis, we view time series as a special type of graph with temporal relations, allowing us to extend the methodology developed for classification. Besides the temporal structure, the proposed approach exploits relationships between the different time series. For instance, a product might be a consumable (e.g. ink) of another (e.g. printer); the relationship can thus have an influence on the predictions. Another example is the traffic conditions of the 50 busiest roads of the city of Lyon (France). Data is aggregated on 20 minutes windows spanning 15 days. The relations between series are based on the geographical

proximity of roads since having traffic at time T in a location might have an impact at time $T + 1$ on neighboring roads. The task here consists in, knowing the traffic from time T_0 to time T, forecasting the traffic at time $T + n$.

In this thesis, to model the joint relational and temporal dynamics of such series we learn Gaussian distributions in a latent space. A decoder then maps the latent representations to the observations. This allows us to model the uncertainty inherent to observations and to predict unobserved values together with a confidence in the prediction.

1.2.3 The Ranking Task

Finally, the ranking task, also known as *learning to rank*, consists in ordering a set of instances according to a specific request. There is a wide variety of applications and thus a variety of possible studied data such as words, documents, items, etc. Document retrieval or recommendation are well known examples of the ranking task. It consists in ordering a collection of documents or items given a query.

In this thesis, we address the ranking task employed for recommendation (collaborative filtering). Here, we want to order a list of items for a specific user based on previous rated items from all users.

Collaborative filtering can be cast into a problem involving relational data. For example, in a music recommendation task, one can imagine having user ratings for the tracks of the LastFM dataset. Leveraging the graph structure so as to learn representations for ranking items in the context of recommendations is a challenging task.

In this thesis, we made a first step in this direction. We learn Gaussian representations in the context of collaborative filtering. The model is based on an approach where, instead of modeling the probability distribution of the likelihood, we directly model the users and items as Gaussian representations. This allows us to compute and maximize the likelihood distribution based on our Gaussian modeling of users and items representations. This Gaussian representation allows us to care for the inherent uncertainty due to lack of information, or conflicting information.

1.3 Learning Representations

This thesis explores representation learning techniques for solving the different tasks presented in Section 1.2. It consists in learning for each entity or node of the graph (for example an image, a word, or a user) a representation in a vector space $\mathbb{R}^d$.

Learning such representations allows us to work with more compact representations than the initial one, to discover new relations or similarities, to simplify the inference, to compare different kind of data in a common latent space or, more importantly, to work with an homogeneous representation space where entities of different types can be represented.

For example, for classical (non graph) classification tasks such as image classification, instead of working with vectors of pixels that exhibit a lot of noise and irrelevant information, representation learning techniques aim at transforming this vector of pixel into a smaller vector where only semantic information is encoded. The same for natural language processing where working with vectors instead of words is far more easier. For example, to find some synonyms one can imagine learning representations such as words with the same meaning are close together in the representation space.

The representation learning framework has recently been applied to different kind of domains, with some strong impacts and breakthrough results during the 2010's in different areas (Bengio et al., 2013):

- Speech recognition (Dahl et al., 2010; Deng et al., 2010; Dahl et al., 2012; Hinton et al., 2012; Mohamed et al., 2012; Seide et al., 2011);
- Signal processing (Boulanger-Lewandowski et al., 2011; Hamel et al., 2011);
- Object recognition (Ciregan et al., 2012; Rifai et al., 2011);
- Natural language processing (Collobert et al., 2011; Weston et al., 2010; Srivastava et al., 2012; Socher et al., 2011; Glorot et al., 2011; Bordes et al., 2012; Mikolov et al., 2013);
- Transfer learning (Bengio, 2012; Chen et al., 2012a; Krizhevsky et al., 2012; Dauphin et al., 2012).

In the case of relational data, we suppose that each relationship corresponds to a geometric constraint between the learned representation of the two linked nodes. While many geometric constraints could be used, we exploited solely the most common one in this thesis, which is based on the hypothesis that *similar entities should be close together in the representation space*.

Representing the nodes simply as points, however, has a crucial limitation: we do not obtain information about the uncertainty of that representation. Uncertainty modeling can be useful when dealing with missing or conflicting information.

Uncertainty is inherent when describing a node in a complex graph: imagine a user for which the different sources of information are conflicting, e.g. pointing to different communities. We would like to encode this uncertainty into its embedding. A solution to this problem is to represent nodes as Gaussian distributions: each node becomes a full distribution (and not a single point only) to capture uncertainty about their representations. Thus, instead of learning deterministic representations we can learn densities with the same kind of constraints but adapted to random variables. Here we want *similar entities to have similar distributions*.

We now detail the main contributions of this thesis.

1.4 Contributions

In this thesis we apply the representation learning framework to three different tasks. For the graph node classification task, we learn a deterministic representation for each node of the graph as well as a classifier (Chapter 4). We extended this model to use Gaussian representations in order to deal with uncertainty and conflicting information (Chapter 5).

Since we got promising results using the new Gaussian representations learning framework, we explored its use for other tasks to see what kind of representations the models could learn and what kind of qualitative properties such representations could have. In particular, we applied the Gaussian representation framework for forecasting (Chapter 6) and ranking (Chapter 7).

1.4.1 Learning Deterministic Representations for Heterogeneous Graph Node Classification (Chapter 4)

In this contribution, we use the representation learning framework for the tasks of node classification in heterogeneous networks. Nodes may be of different types, each type with its own set of labels, and relations between nodes may also be of different types. A typical example is provided by social networks where the node types are e.g. users, content, groups, etc. and relations may be *friendship*, *like*, *authorship*, etc. Learning and performing inference on such heterogeneous networks is a recent task requiring new models and algorithms. Note that at the time of this work, this task was largely unexplored. We propose a transductive approach to classification that learns to project the different types of nodes into a common latent space. This embedding is learned so as to reflect different characteristics of the problem such as the correlation between labels of nodes with different types and the graph topology.

These interdependencies between nodes of different types play an important role in several classification problems, and none of the current methods was able to handle them properly. In this contribution, the proposed model also learns, jointly with the node representations and classifiers, relation specific weights to handle these interdependencies. The application focus is on social graphs but the algorithm is general, and could be used for other domains as well. The model is evaluated on five representative datasets of different instances of social data, and works well compared to competing methods. Experimental results show that the proposed model was able to learn correlations between labels of different types of nodes, notably improving the class label inference.

1.4.2 Learning Gaussian Representations

Learning Gaussian Representations for Heterogeneous Graph Node Classification (Chapter 5)

In this contribution, we have explored the use of uncertainty for learning to represent nodes in the task of heterogeneous graph node classification. This extends the previous contribution, by using Gaussian representations rather than deterministic ones (Chapter 4). The proposed model associates with each node a Gaussian distribution over the representation space, parameterized by its mean and covariance matrix. We have examined four variants of this model, by using either spherical and diagonal covariance matrices, and by using two different loss functions for classification. The proposed model is evaluated on three representative datasets and improves the class label inference compared with the deterministic model proposed in our first contribution.

Based on the experimental results obtained on datasets representative of different situations, our main findings are that (i) integrating uncertainty in representations improved classification (ii) according to our initial intuition, the effect of using uncertainty has generally more impact when the amount of training data is lower and (iii) according to our expectation, highly central nodes have less variance associated to their representation than non central nodes.

Learning Gaussian Representations for Relational Time Series Forecasting (Chapter 6)

The good results obtained with the Gaussian representations learning framework on the heterogeneous graph node classification motivated us to explore its use for another task, namely relational forecasting, predicting the future of relational time series.

In this contribution, we proposed a new dynamical state space model for modeling the evolution of such series. The joint relational and temporal dynamics of the series are modeled as Gaussian distributions. A decoder maps the latent representations to the observations. The two components (dynamic model and decoder) are jointly trained. Using Gaussian representations allows us to model the uncertainty inherent to observations and to predict unobserved values together with a confidence in the prediction.

The proposed model has shown competitive performance on four different datasets compared to state-of-the-art models. Moreover, it allows to model the uncertainty of predictions, providing for example confidence intervals for each prediction.

Learning Gaussian Representations for Ranking (Chapter 7)

In this last contribution, we leveraged Gaussian representations for collaborative filtering in a learning to rank framework. As for relational forecasting and classification, we use Gaussian representations to allow the modeling of uncertainty of the learned representations. This can be useful when a user has a small number of rated items (cold start), or when there are conflicting informations about the behavior of a user or the ratings of an item. We have shown that our model performs well on three representative collections (Yahoo, Yelp and MovieLens). We analyze the learned representations and show that the variance distribution is correlated with the space dimension and the training set size.

1.5 Thesis Organization

The thesis addresses three different tasks (classification, forecasting and ranking) on different kind of data (social networks, relational time series and ratings). To allow for a reading of the different chapter independently, we chose to introduce the field of representation learning in Chapter 2. We then introduce the related works for each task in the chapter where we present the corresponding contribution:

- Chapter 4 for node classification using deterministic representations,
- Chapter 5 for node classification using Gaussian representations,
- Chapter 6 for relational forecasting,
- Chapter 7 for ranking.

Figure 1.2 represents the dependencies between the different chapters of the thesis and summarizes the outline of the thesis.

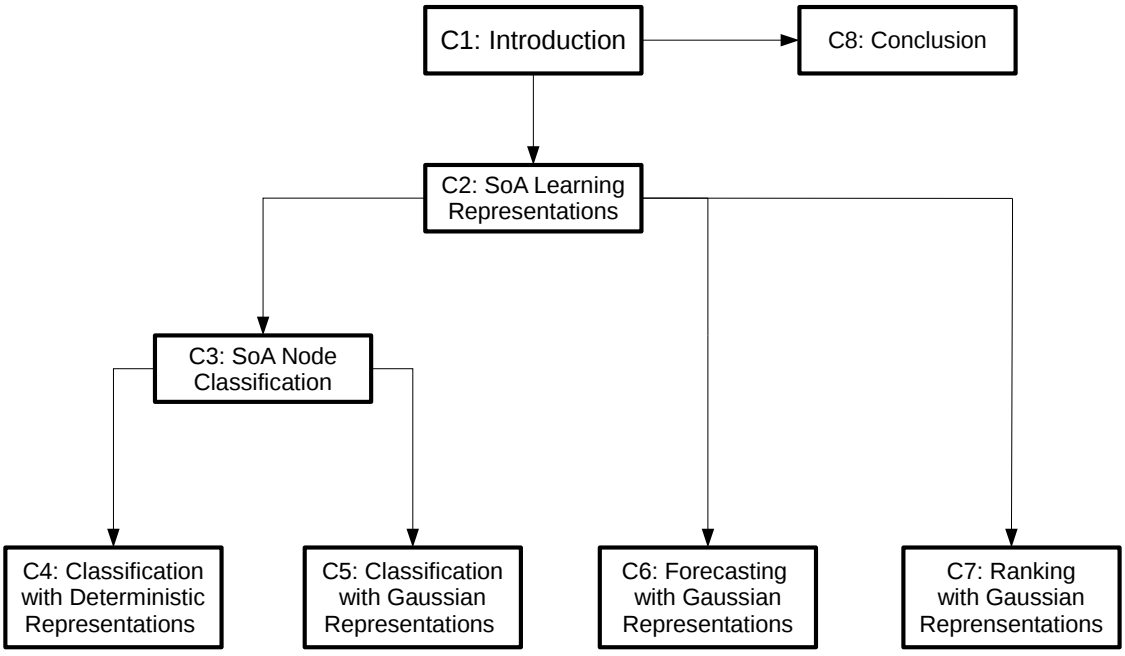


FIGURE 1.2: A tree-like structure that shows the dependencies between the different chapters of the thesis manuscript.

Chapter 2

Learning Representations for Relational Data

Abstract

In this chapter, we present the field of representation learning in the context of relational data. We first give a definition of representation learning. We then present different types of approaches studied during the thesis: inductive, transductive, unsupervised, supervised and semi-supervised. We then give a state of the art of the representation learning framework for learning deterministic and stochastic representations.

Contents

2.1 Introduction 15
2.2 Learning Representations in Graphs 17
2.2.1 Inductive and Transductive Algorithms 17
2.2.2 Supervised, Unsupervised and Semi-Supervised Learning 18
2.3 Learning Deterministic Representations 19
2.3.1 Unsupervised Models 20
2.3.2 Supervised and Semi-Supervised Models 22
2.3.3 Other Applications of Representation Learning 25
2.4 Capturing uncertainties in representations 26
2.5 Conclusion 29

2.1 Introduction

In machine learning, the question of how to represent data is central since the behavior of the model depends on it. Good data representations should emphasize explanatory factors. This is the reason why a lot of works have focused on learning robust representations. We can distinguish two kind of techniques to obtain data representations. The first one is human defined features (or handmade features), which consists in using expert knowledge to define them explicitly. The second one is representation learning which consists in learning the representation of each instance or node in the graph in a vector space from raw data.

One early success of representation learning is image processing. Before representation learning (deep learning approaches) was developed, image processing was mostly based on visual features that were intended to be informative and non-redundant. For example, during several years, image processing models used low level handmade features based on

color, texture, shape, ... Feature descriptors such as SIFT (Lowe, 1999), HOG (Freeman et al., 1995) or SURF (Bay et al., 2008) were introduced. These models and algorithms take an image as input and they output locations of significant points of the image. These descriptors were mainly based on local characteristics computed directly from the image pixels. Models using these descriptors as input for the image classification task (or other image processing tasks) were state of the art references in terms of results.

Deep learning, and more generally representation learning models (Bengio et al., 2013), aim at associating with any object described by raw features (e.g. pixel RGB values or simply one-hot encoding like for text) a latent representation in a (vectorial) space such as $\mathbb{R}^d$ (for $d \in \mathbb{N}$). Geometric relationships between entities encode existing relations or similarities of raw data. Learning representations is a way to discover underlying (unknown) explanatory factors from input data, and allows us to:

- Define compact representations without loss of information;
- Smooth raw data by having similar entities close in the latent space;
- Find new relations or similarities in between data using the latent representations through a natural clustering;
- Simplify the inference since the factors of latent representations are related through simple dependencies;
- Compare different kind of data in a common latent space (such as images and text or nodes of different types).

This last point is key for us, since in this thesis we deal with heterogeneous relational data. Our goal is to learn a representation of an object that encompasses the information from the whole graph. Furthermore, representing heterogeneous data in the same latent space gives us the possibility to learn correlations between different kinds of nodes. That is why, during the last years, representation learning has been largely developed for relational data and has reached extremely promising results on various tasks by automatically building relevant representations.

Representation learning methods aim at projecting instances in a representation space (e.g. $\mathbb{R}^d$). These projections are learned so that distances or similarities between instances in the representation space are consistent with existing relations within the graph.

An illustration of representation learning for relational graphs is given in Figure 2.1. Each of the nodes of the graph is projected onto $\mathbb{R}^d$, and is thus associated with a learned vector representation. In this example, we consider two type of nodes and distinguish them in the figure representing the latent space: the first type is represented as circles and the second as stars. This can be used in classification, as shown in the figure where two classifiers are learned, one for the stars and the other one for the circles. In this example, circles and stars correspond to two different types of nodes. Each type has to be classified among two classes, green/yellow for circles and red/blue for stars, the two types of nodes are linked so that their representations influence each other.

The outline of this chapter is as follows: firstly we discuss the differences between inductive and transductive algorithms (Section 2.2.1), and between supervised, unsupervised and semi-supervised learning (Section 2.2.2). Then we introduce different models by distinguishing on whether they learn deterministic (Section 2.3) or Gaussian (Section 2.4) representations for relational data.

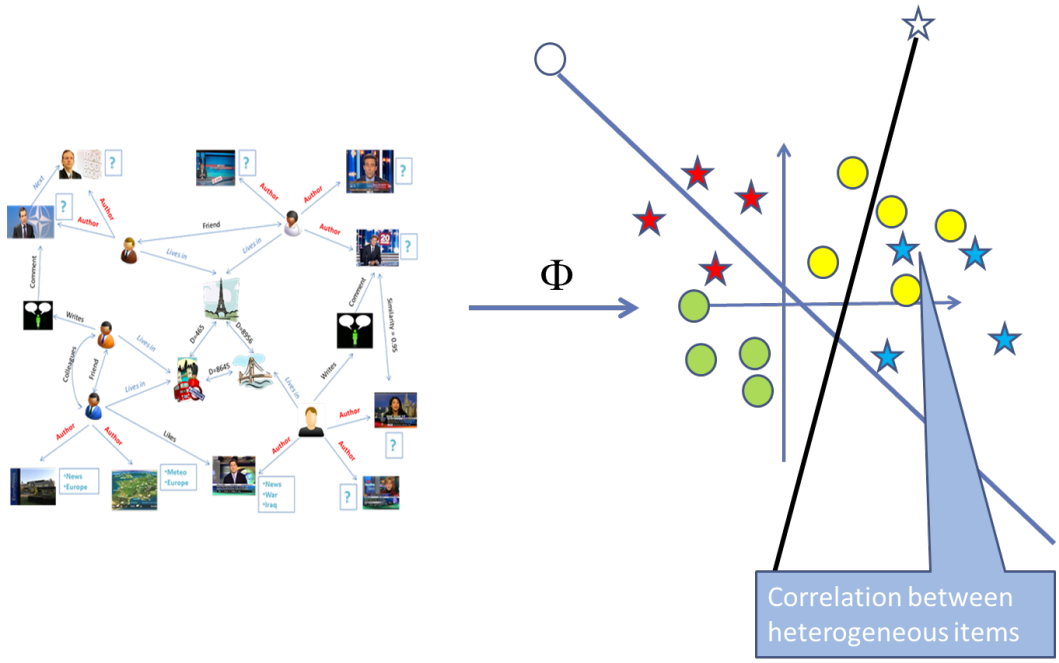


FIGURE 2.1: Drawing of representation learning for relational data where nodes are projected from the initial graph (left) to the common latent space (right) where classifiers are learned.

2.2 Learning Representations in Graphs

In our contributions, the proposed models use transductive (semi-)supervised approaches. Those notions can be source of confusion, so let us firstly clarify some definitions. Since this thesis focuses on relational data, we give an explanation of what it means to work under such frameworks when dealing with graphs.

2.2.1 Inductive and Transductive Algorithms

In this section, we compare two different *philosophical* concepts related to inference in machine learning: induction and transduction. The works presented in this thesis use the transductive approach, but it is important to differentiate the two approaches since, based on the data, both of them or only one of them could be employed.

We take a simple example for the classification task where we have instances $(x_i)_{i\in[1..n]}$ divided in training set, and testing set, as well as their respective labels. Without loss of generality we assume the task of binary classification.

Induction

Inductive algorithms aim at learning a general rule f from observed examples, which is then applied to unseen examples. It learns a predictive model that only depends upon the features of a sample to classify.

More precisely, for classification an inductive approach aims at learning a classifier f that maps the instance space to the label space, based only on the training instances. In

practice, we need input features for all x_i. Generally input features are represented as vectors in $\mathbb{R}^{d'}$. Thus, in our example, the classifier f maps $x_i \in \mathbb{R}^{d'}$ into $\{-1; 1\}$. During the inductive training phase only the labeled training set is used. Inference is then performed by applying the learned classifier f directly on the test set or to any new datum.

The fact that input features are mandatory is important since not all datasets provide readily available input features. As we will see in this thesis, sometimes only the graph is given as input, with no additional feature (e.g. only a weight matrix indicating a relation or a similarity between graph nodes is provided). So, in the case where no input feature is available, the inductive approach is not applicable.

Transduction

Transductive approaches are introduced in (Gammerman et al., 1998). The main difference between inductive and transductive approaches is that the transductive one does not aim at learning a general predictive model but directly classifies test data based on training data: it is an inference from particular to particular. Said otherwise, the output of the transductive algorithm is a vector of labels instead of a classifier f. The goal is to transfer the information from labeled examples to unlabeled ones. The strength of transduction lies in that it is solving a simpler task than induction. Thus during the training phase both the training set (with its respective labels) and the test set (without its respective labels) are used. Since test data is used during the training phase, if we want to label a new test point x_{n+1} the transductive algorithm needs in principle to be run again whereas its inductive counterpart can label it by applying directly the learned classifier f.

Transductive learning assumes that we can however provide a relation between examples (e.g. a way to compare them). This is the case in graphs where relations between nodes give us a prior knowledge on which nodes should be close to each other even if there is no input feature. For example, concerning the classification task, it is very common to assume that *neighbors in the graph are more likely to have similar labels*.

Moreover, speaking about relational data, since most often the main goal is not to return a classifier, the lack of node input features is not crippling because the input graph can be used as the only source of information.

2.2.2 Supervised, Unsupervised and Semi-Supervised Learning

In this section we compare three different *technical* ideas in machine learning: unsupervised, supervised and semi-supervised learning. The works presented in this thesis use semi-supervised techniques.

Unsupervised Learning

Unsupervised learning techniques consist in learning hidden structures (e.g. data densities) from unlabeled data only, i.e. there are no labels y_i provided. All these techniques rely on some pre-defined relation between the samples. A very common example in unsupervised learning is the task of clustering, and more specifically the k-means algorithm (MacQueen, 1967). It aims at finding k points (the center of the cluster) in the vectorial space that minimize the sum of the distances of all instances to the nearest cluster center.

In unsupervised learning, there is no direct evaluation of the algorithm, although the output of the algorithm (clusters, representations, ...) can be used for other tasks.

Supervised Learning

Supervised learning techniques aim at learning a predictor for a task dealing with labels, ratings, values, etc. In our case, it corresponds to learning a classifier f. During the training phase, the goal of the model is, given $(x_i)_{i\in[1..m]}$ and $(y_i)_{i\in[1..m]}$, to try to match $(x_i, f(x_i))$ with (x_i, y_i) as well as possible. This learned classifier can then be used for mapping new examples. All the difficulty is to have a classifier that generalizes well from labeled to unlabeled data.

Semi-Supervised Learning

Semi-supervised learning is a class of supervised learning methods that makes use of both labeled and unlabeled data. The main assumption of semi-supervised learning is that even non labeled data should add relevant information to the model. Actually, unlabeled data give interesting information about the prior distribution of the data $\mathbb{P}(x)$.

In our case it aims at learning a classifier f, when not all the instances of the training set are labeled. Unlabeled training instance are used together with labeled one to represent local relational constraints on the problem by forcing linked nodes to have close representations.

Transductive methods are always semi-supervised since they use information contained within the test points. The difference is more on the reason why they have been proposed. The main difference is that semi-supervised learning does not need specific test instances, whereas transductive algorithms need the test set that will be used in the end to evaluate its performance (Chapelle et al., 2009).

Our Contributions

Concerning our contributions for the classification task (Chapters 4 and 5), using both labeled and unlabeled nodes make our proposed models semi-supervised ones.

Concerning our contributions, for the forecasting task (Chapter 6), target are directly provided since we forecast at time $T+n$ learning from the past known data. For the ranking task (Chapter 7), we also learn to rank from triplets (user, item, ratings). Thus those two contributions belong to the supervised learning framework[1].

2.3 Learning Deterministic Representations

In recent years, there has been a surge of interest in learning deterministic (vector) representations for different machine learning tasks, such as collaborative filtering (Koren et al., 2009), image retrieval (Weston et al., 2011), relation extraction (Riedel et al., 2013), word semantics and language modeling (Bengio et al., 2003; Mnih et al., 2009; Mikolov et al., 2013).

[1]Note that prediction could be considered as unsupervised learning since there is no need for human annotation. However technically the data are also used as targets and this framework include supervised training methods.

In this section we present a state of the art concerning the models learning deterministic representations, focusing on relational data which is the main topic of this thesis.

In the following, we distinguish unsupervised models and supervised/semi-supervised ones since they rely on different sets of techniques. In our experiments, unsupervised models performed worse than supervised one, yet, a lot of (semi-)supervised models are inspired from unsupervised ones.

Recent years have seen an increasing enthusiasm for using neural networks to learn supervised representations. For example, (Zhou et al., 2015; Zhang et al., 2014; Che et al., 2015; Wu et al., 2015; Niepert et al., 2016) propose neural network based models to learn deep representations from raw input data. But neural networks models need rich informative raw input features to learn good representations. In our case, no node specific rich input data are available. Note that, we do not cover the models that learn a representation of the whole graph since we focused on the nodes of the graph.

2.3.1 Unsupervised Models

We first study unsupervised representation learning methods. In the context of graphs, this corresponds to a graph, where nodes may (or not) be associated with raw data like e.g. images or text. These learned representations can then be used for different supervised tasks such as link prediction or classification.

Learning from Context

Learning representations from context often comes to learning entity representations in the context of surrounding words in a text or surrounding objects in an image. Firstly introduced to learn word embeddings (Mikolov et al., 2013), this methodology was extended to other types of high-dimensional data like images. (Mikolov et al., 2013) learn two latent vectors for each word (entity): a representation and a context vector. These two vectors are used to compute the conditional probability of observing a word given its context (another word for instance). The representations are learned by maximizing this conditional likelihood over a corpus.

Most *learning from context* models can be framed within the class of methods called the *exponential family embeddings* (Rudolph et al., 2016). Such a family defines a context function, a conditional probability function (generally modeled from the exponential family) and an embedding structure to form the objective function:

- The context function defines, given an entity instance (e.g. an occurrence of a word in a text), which are the related entities (e.g. surrounding words).
- The conditional probability function is a function of both the representation of the entity and the corresponding context vectors. It is in charge of modeling how representations (mathematically) interact with their corresponding context.
- The embedding structure determines how the model shares the vector representations across the data, i.e. the possible relations between the context vector for a given entity and the representation of the same entity (e.g. same vector for the context and the target representation).

The models proposed in (Mikolov et al., 2013), namely skip-gram and C-BOW, could be set within this framework. We introduce the skip-gram model since it has been adapted to learn representations for relational data.

(Mikolov et al., 2013) proposed an unsupervised model to learn word representations in a vector space, with the goal of using them in natural language processing tasks, by grouping semantically similar words in the same region of the vector space. This is now the basic representation for many neural network models manipulating text.

The goal of the skip-gram model is to learn word representations from which we can infer the context, i.e. surrounding words. Thus words that appear in similar contexts have similar representations. More formally, given a phrase composed of the series of words x_1, x_2, x_3, ..., x_T, the training objective of the model consists in maximizing the log-probability:

$$\sum_{(i,j)} \log p(x_j|x_i) \tag{2.1}$$

where x_t is a specific word and x_c is a word that appears in the context of x_t. The choice of x_c depends on the choice of the context function. For example, we can take a window of a given size centered on x_t and consider that all the nodes x_i in this window are part of the context.

Various conditional probability functions may be used. The basic formulation consists in using a softmax function and learning two representations, z_i and z'_i respectively the "target" and "context" vector representation for x_i:

$$p(x_j|x_i) = \frac{\exp(z'_j \cdot z_i)}{\sum_{k=1}^{W} \exp(z'_k \cdot z_i)} \tag{2.2}$$

where W is the total number of words in the vocabulary.

An interesting property of the learned representations for this model is that some basic algebraic operations on word representation were found to be meaningful and understandable from a natural language point of view (Mikolov et al., 2013). For example, a simple operation such as z("France") + z("capital") is close to z("Paris"). Even complex operations can be done, for example, the closest word representation z("Lisbon") − z("Portugal") + z("France") is z("Paris"). However up to now, there is no real exploitation of these properties.

These ideas have been extended to graphs. (Tang et al., 2015; Perozzi et al., 2014; Cao et al., 2015) learn deterministic representations for relational data with graph nodes instead of words and a list of neighboring nodes instead of window.

The difference between these contributions lies in the way each model constructs the list of neighboring nodes, i.e. the context function. There are various ways to define a sequence of nodes based on a graph structure. (Perozzi et al., 2014; Cao et al., 2015) both use a random walk based approach to generate sequences of nodes. Concerning (Tang et al., 2015), the main difference with (Mikolov et al., 2013) is that the size of the context window is variable depending on the considered node. To define a sequence of nodes, they randomly sort the neighbors of the considered node. Furthermore, in the case of weighted graphs, they assume that the joint probability of two nodes depends on the corresponding

weight. Thus, they weight the conditional probability loss of two nodes accordingly to the edge weight. The objective function maximized by their model is then:

$$\sum_{(i,j)\in\mathcal{E}} w_{ij} \log p(x_j|x_i) \tag{2.3}$$

where $\mathcal{E}$ is the set of edges of the graph and w_{ij} the weight of the (directed) edge (i, j). The conditional probability of neighboring nodes is, as in (Mikolov et al., 2013), modeled as a function of the inner product of their representations.

Similarly, (Grover et al., 2016) proposed an unsupervised learning model that maximizes the likelihood of the network neighborhood (context) knowing a node (target). In practice, this neighborhood is defined through a random walk which can, for example, take into account homophily.

(Tang et al., 2015) have shown on various tasks and datasets that their model was the best competitor over different unsupervised models, including (Perozzi et al., 2014). It is thus used as our unsupervised baseline for the classification tasks.

2.3.2 Supervised and Semi-Supervised Models

Learning Nodes and Relationship Representations in Knowledge Bases

In the context of relational data some graphs distinguish different types of nodes and relations (Chapter 1). For example, in knowledge bases (KB), different types of entities are linked together with different types of relationships. A concrete example of such knowledge base is the Google Knowledge Graph that had, in 2016, more than 70 billions facts and relations about people, places, things.

When multiple types of relations are present, we need more complex models than the simple unsupervised models presented above.

Here we describe a class of supervised models that learn directly the representations without input features. This class of models uses geometric constraints, between two node representations, that depend on the relation type.

Most of these works are concerned with modeling knowledge bases. Information is then provided under the form of triplets (subject (s), relation (r), object(o)). The goal is to learn representations for items (s and o) and relations (r) so as to maximize some loss function (e.g. triplet likelihood). This topic has motivated a series of works (Bordes et al., 2013; Nickel et al., 2011; Bordes et al., 2011; Jenatton et al., 2012; Socher et al., 2013; Socher et al., 2013; Bordes et al., 2014). The main difference between all these approaches lies in how they model the relations between the (s, r, o) elements, and how they calculate the score associated to a triplet (s, r, o). In these models, representations interact with each other through different types of transformations. In the following we describe some representative models of this class. We differentiate them based on which transformation (of the representations) they use: translation, linear or bilinear.

Translation We first present the model introduced in (Bordes et al., 2013), namely TransE, which is a simple but representative example of how these models work.

For each node x_s and for each type of relation r, the model learns latent representations, respectively z_s and z_r, both in $\mathcal{R}^d$. They model relations as a translation in the latent space.

Thus, if there is a directed relation of type r between x_s and x_o (denoted $x_s \xrightarrow{r} x_o$) the vector $z_s + z_r$ should be close to z_o in the latent space.

The constraint on the latent space could be written as: if x_s and x_o are linked through the relation r, $z_s + z_r \approx z_o$, and $z_s + z_r$ should be far away from z_o otherwise. The idea of this model was somehow inspired by the interesting property of representations (coincidentally) found in (Mikolov et al., 2013), where, in the example we give in Section 2.3.1, the *relation* "capital of" was represented by the model as a translation in the embedding space.

To learn such representations, (Bordes et al., 2013) minimize a margin-based ranking loss over the training set:

$$L_{\text{TransE}} = \sum_{(s,o,r)\in E^r} \sum_{(k,l,r)\in E^r_{\text{neg}}} \max(0, 1 + ||z_s + z_r - z_o||^2 - ||z_k + z_r - z_l||^2) \tag{2.4}$$

where E^r represents the set of edges for the relation type r and E^r_{neg} corresponds to the set of edges in E^r where the origin node or the destination node of the relation was changed by a random node of the graph. Here, the score of an edge $x_s \xrightarrow{r} x_o$ is modeled as $\mathcal{S}(s, r, o) = ||z_s + z_r - z_o||^2$, but other models have been proposed using different constraints.

Linear Transformation (Nickel et al., 2011) represent the graph as a 3D tensor where each slice represents a given relationship. They then use tensor (as a matter of fact, matrix) factorization to predict new relations. With $\mathcal{X}$ being a tensor such as $\mathcal{X}_{sro} = 1$ if the relation (s, r, o) exists and zero otherwise. They factorize each slice $\mathcal{X}_r$ as $\mathcal{X}_r \approx AR_rA^\top$, where A is an $n \times m$ matrix and R_r is an $m \times m$ matrix modeling the interactions between the elements according to the relation r.

(Bordes et al., 2011) represent each relationship r with two matrices one for outgoing edges R_{r_1} and the other for incoming ones R_{r_2}. The score of the triplet used in this model is:

$$\mathcal{S}(s, r, o) = ||R_{r_1} z_s - R_{r_2} z_o|| \tag{2.5}$$

(Jenatton et al., 2012) proposed to represent nodes as vectors (z) and relations as matrices (R). They learn the representations of nodes and relations using probabilistic models. They then compute the probability of a triplet through a dot product.

$$\mathcal{S}(s, r, o) = \sigma(< z_s, R_r z_o >) \tag{2.6}$$

At last, (Bordes et al., 2014) introduced two models where nodes and relations are represented in the same latent space but where relations are modeled as tensors instead of translations. In both models, nodes (z_s, z_o) and relations (z_r) are represented in $\mathbb{R}^d$. The first variant they propose for calculating the score of a triplet uses a linear transformation. Giving the representations (z_s, z_o and z_r), the score of the triplet is:

$$\mathcal{S}_{linear}(s, r, o) = -(W_1 z_r + W_2 z_s + b_s)^\top (W_3 z_r + W_4 z_o + b_o) \tag{2.7}$$

where $W_\bullet$ are matrices.

Bilinear Transformation In the second model introduced in (Bordes et al., 2014) nodes (z_s, z_o) and relations (z_r) are also represented in $\mathbb{R}^d$, but they use for calculating the score of a triplet a bilinear transformation. Giving the representations (z_s, z_o and z_r), the score of the triplet is:

$$\mathcal{S}_{bilinear}(s,r,o) = -((W_1 \times z_r)z_s + b_s)^\top ((W_2 \times z_r)E_o + b_o) \tag{2.8}$$

where $W_\bullet$ are tensors of dimension 3 and $\times$ denotes the n-mode vector-tensor product along the 3rd mode (see (Kolda et al., 2009) for more details).

In (Socher et al., 2013), they also use a bilinear model where they compute the score of a triplet (s,r,o) by:

$$\mathcal{S}(s,r,o) = u_r^\top f\left(z_s^\top W_r z_o + V_r \begin{bmatrix} z_s \\ z_o \end{bmatrix} + b_r\right) \tag{2.9}$$

where W_r is a tensor of dimension k, b_r and u_r are in $\mathbb{R}^d$ all representing the relation r.

In practice there is no best model among the ones presented here, since the performances of these models depends on the KB and the task we evaluate the models on.

In this thesis, we proposed models that learn nodes and differentiate relationships based on their type.

All the previous presented learned representations can be used as inputs to learn, for example, a classifier. As seen previously, another way to learn classifier or predictors is to use directly the labels when learning the representations. This is what we discuss in the next section focusing on transductive approaches.

From Unlabeled to Labeled Data in Classification

For graphs, supervised learning models learn representations and classifiers at the same time. Compared with the unsupervised works presented in Section 2.3.1 or the supervised ones on KB presented in Section 2.3.2, the classification task has some specificities. For example, even if labels can be seen as a new type of node (with each nodes linked to its corresponding labels), the node-label relation is very particular and handling them separately in the model performs better. That is why the node-label relation needs to be treated differently in the loss function.

Furthermore, learning a classifier at the same time as the node representation brings information to the learned representation, and makes a huge difference on the learned vector latent space. In this section, we review such graph transductive models.

(Jacob et al., 2014) introduced a supervised model for classification in the context of heterogeneous graphs. The idea developed in this work is to learn, along with classifiers, a latent representation for each node, whatever their type is, into the same latent space.

To do so, they minimize the following objective function[2]:

$$\sum_{i} \max(0, 1 - y_i z_c \cdot z_i) + \sum_{(i,j)\in E} ||z_i - z_j||^2 \tag{2.10}$$

where $y_i \in -1, +1$ is the label of node x_i, z_i and z_j are respectively the representations of x_i and x_j, and z_c is the vector representing the linear classifier. Concerning unsupervised models or models with no labeled data, as said previously, one way to consider labels during the training phase is to see them as nodes and introduce a new relationship "is labeled" denoted r_ℓ.

Let us take the example of the TransE model (Bordes et al., 2013), introduced in the previous section (Equation 2.4), where they test their model on the link prediction task. Transforming it into a supervised model for classification with "labeled" relation r_l and a representation for the classifier z_c, without negative sampling and supposing only null translations, its loss function becomes[3]:

$$\begin{aligned} L_{\text{TransE}} &= \sum_{(i,y_i,r_\ell)\in E^{r_\ell}} \max(0, 1 + y_i||z_i - z_c||^2) + \sum_{(i,j,r)\in E^r, r\neq r_\ell} ||z_i - z_j||^2 \\ &= \sum_{(i,y_i,r_\ell)\in E^{r_\ell}} \max(0, \alpha - y_i z_i \cdot z_c) + \sum_{(i,j,r)\in E^r, r\neq r_\ell} ||z_i - z_j||^2 \end{aligned} \tag{2.11}$$

where α is a constant. This loss is quite close to the supervised model introduced in (Jacob et al., 2014).

In the supervised version of (Bordes et al., 2013) for classification, the modeling omits the major terms and parameters needed to perform well on the classification task such as a parametrized classifier f. As seen in (Jacob et al., 2014), processing differently the "labeled" relationship improves a lot the performances.

Our first contribution (Chapter 4) is based on (Jacob et al., 2014).

2.3.3 Other Applications of Representation Learning

There exists several supervised applications of representation learning for relational data in domains such as information diffusion (Bourigault et al., 2014), recommendation (Chen et al., 2012b), social network analysis (Vu et al., 2015; Zheng et al., 2015).

For example, (Bourigault et al., 2014) propose a model to predict how information spreads into a network. It learns latent representations of users, and models the information diffusion process in the latent space inspired by the heat equation. These representations are then used to predict, given a source of information, which users will be *infected* by some content and in which order.

In (Chen et al., 2012b), they learn a latent representation of each song. They model the probability of a play-list as the product of the conditional probabilities of the successive songs. They learn their representations by maximizing the likelihood of these Markov chains. The conditional probabilities are calculated with a soft-max function depending on the distance between the two representations.

[2]Without loss of generality we consider here the case of binary classification.

[3]We have normalized vectors to simplify the equation.

2.4 Capturing uncertainties in representations

The objective of the approaches exposed in Section 2.3 is to map input instances (such as images, relations, words or nodes in a graph) to vectors representations in a low-dimensional space. The goal is that the geometry of this low-dimensional latent space be smooth with respect to some measure of similarity in the target domain. That is, objects with similar properties (e.g. class) should be mapped to nearby points in the embedded space.

While this approach is highly successful, representing instances as vectors in the latent space carries some important limitations:

- Vector representations do not naturally express uncertainty about the learned representations;
- We cannot model inclusion, or entailment by comparing vector representations (usually done by inner products or Euclidean distance which are symmetric).

In this section we describe different approaches and models able to modeling representations uncertainty through the learning of (latent) densities. There is a long line of previous work in mapping entities to probability distributions, but few represent entities directly as density distributions in a latent function space. Density representations allow to map entities not only to vectors but to regions in space, modeling uncertainty, inclusion, and entailment, as well as providing a rich geometry of the latent space. Using this new framework might solve the previous issues listed above.

In the following, we give a state of the art of models using densities to represent entities. We firstly give a short view of Bayesian approaches (variational Bayes inference) which can be seen as learning density representations. We then focus on Gaussian embeddings, in which both means and variances are directly learned from specific loss function over the densities.

Bayesian Approaches

Concerning the Bayesian framework, a way to learn latent distributions is to find the posterior marginal distributions over all individual latent variables. We suppose that each entity i is modeled by a random variable Z_i in $\mathbb{R}^d$. The goal is to infer $P(\{Z_i\}_i|X)$, where X are the observations.

In the context of relational data, this type of approach has been used in task such as classification (Airoldi et al., 2005; Xiang et al., 2013), data collaborative completion (Kim et al., 2014) or collaborative filtering (Stern et al., 2009). (Airoldi et al., 2005) and (Xiang et al., 2013) address collective classification by inferring the posterior distribution of the class degrees of membership for each node. (Kim et al., 2014) propose a graphical model for binomial mixture for collaborative completion than handle non-random missing data. For collaborative filtering (task studied in Chapter 7), (Stern et al., 2009) proposed an inference model and approximate the posterior distribution by a product of Gaussian distributions. Given a set of rating tuples (user, item, rating), they learn the approximate posterior distributions. The optimization is accomplished using variational message passing (Winn et al., 2005). Thus, they achieve to learn posteriors mean and covariance of each latent representations, which they use to approximate the distribution as a normal. They then use these learned representations to predict the value of an item for a user.

In the most interesting models, algorithms performing exact inference are computationally intractable for all but the simplest models. Variational Bayesian methods (in the context of Bayesian networks) used in variational inference (Jordan et al., 1999) is one way to bypass this limitation.

A first approach is using Monte Carlo Markov Chain (MCMC) sampling from the posterior distribution and then train a (simpler and tractable) model on those samples. However, MCMC methods are often low to converge and in many cases variational methods are preferred since they do not require any sampling, and so are usually faster.

Variational inference aims at approximating the (intractable) posterior distribution, denoted P, by a (simpler and tractable, e.g. making independence assumptions between variables) variational one, denoted Q, i.e. $P(Z|X) \approx Q(Z|X)$, where Z is the set of latent variables and X the observations. We omit the notational dependence of Q on X for clarity.

The principle is then to minimize a measure of similarity between Q and P. One principled way to do so is by considering the KL-divergence[4]:

$$D_{KL}(Q||P) = \sum_{Z} Q(Z|X) \log \frac{Q(Z|X)}{P(Z|X)} dx$$

But the expression of the KL-divergence still involves the intractable posterior distribution $P(Z|X)$, however we can remark that:

$$\log(P(X)) = D_{KL}(Q||P) - \underbrace{\sum_{Z} Q(Z|X) \log \left(\frac{Q(Z|X)}{P(Z,X)} \right)}_{\mathcal{L}(Q)} \tag{2.12}$$

using the fact that $P(Z|X) = \frac{P(Z,X)}{P(X)}$.

Since $P(X)$ is constant w.r.t. Q, minimizing the KL-divergence is equivalent to maximizing $\mathcal{L}(Q)$. Different variational families have been studied to complete this optimization. We focus here on the mean-field variational family (Blei et al., 2017) that makes the simplifying assumption of mutual independence between random variables Z_i conditioned on X. We factorize $Q(Z|X)$ as $Q(Z|X) = \prod_i Q_i(Z_i|X)$, allowing the computation to become tractable. As in Expectation Maximization (EM) algorithms, variational inference algorithms find locally optimal parameters through an alternating scheme, based on mutually dependent equations that cannot be solved analytically.

Direct Approaches

Recently, instead of using Bayesian approaches, several works have proposed to learn these densities directly. The main idea is to derive a cost function which is directly based on the densities, and to train discriminatively instead of estimating posteriors.

All the approaches use Gaussian embeddings since they are well known and simple distributions. They can represent uncertainty, and KL-divergence between Gaussian distributions is straightforward to calculate. The KL-divergence is naturally asymmetric, and has a simple geometric interpretation as an inclusion between families of ellipses.

[4]The KL divergence is an information-theoretic measure of proximity between two densities.

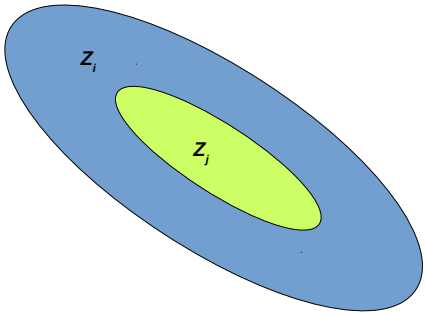


FIGURE 2.2: Example of a low $D_{\text{KL}}(Z_j||Z_i)$.

We begin by describing this new framework, used in three of our contributions, namely Gaussian representation learning. Introduced in (Vilnis et al., 2015), it consists in learning representations in the space of Gaussian distributions. We denote $Z \sim \mathcal{N}(\mu, \Sigma)$ a Gaussian representation characterized by its mean μ and its covariance matrix Σ.

The work presented in (Vilnis et al., 2015) has the same goal as (Mikolov et al., 2013), i.e. learning unsupervised representations of words using their context. To learn Gaussian representations (Vilnis et al., 2015) proposed a ranking-based loss, following (Bordes et al., 2013). They chose a max-margin ranking objective , which pushes the energy of positive pairs bellow negative ones by a margin:

$$L(i, p, n) = \max(0, 1 - E(Z_i, Z_p) + E(Z_i, Z_n)) \tag{2.13}$$

where $E(Z_i, Z_j)$ is the energy associated to the Gaussian representations of words i and j, Z_p is a Gaussian representation of a positive context for word i and Z_n a Gaussian representation of a negative one.

They proposed two functional forms for the energy function E, one symmetric and one asymmetric. The symmetric form did not perform as well, and is less interesting since it does not express the fact that the context of a word should *contain* the word representation. To circumvent this limitation, they proposed to use the Kullback-Leibler divergence, denoted KL-divergence or D_{KL}, which is naturally asymmetric:

$$\begin{aligned} -E(Z_i, Z_j) \overset{def}{=} D_{\text{KL}}(Z_j||Z_i) &= \int_{x \in \mathbb{R}} \mathcal{N}(x; \mu_j, \Sigma_j) \log \frac{\mathcal{N}(x; \mu_j, \Sigma_j)}{\mathcal{N}(x; \mu_i, \Sigma_i)} dx \\ &= \frac{1}{2} \left(\text{tr}(\Sigma_i^{-1}\Sigma_j) + (\mu_i - \mu_j)^T \Sigma_i^{-1} (\mu_i - \mu_j) - d - \log \frac{\det(\Sigma_j)}{\det(\Sigma_i)} \right) \end{aligned} \tag{2.14}$$

KL-divergence is a natural energy function for modeling entailment between concepts. A low KL-divergence $D_{\text{KL}}(Z_j||Z_i)$ indicates that we can encode Z_j easily as Z_i, implying that Z_j entails Z_i. This can be more intuitively visualized and interpreted as a soft form of inclusion between the Gaussian distributions of the two Gaussian representations – if there is a low KL-divergence, then most of the mass of Z_j lies inside Z_i as shown in Figure 2.2. Figure 2.3 shows what kind of Gaussian representations are learned by the model proposed by (Vilnis et al., 2015).

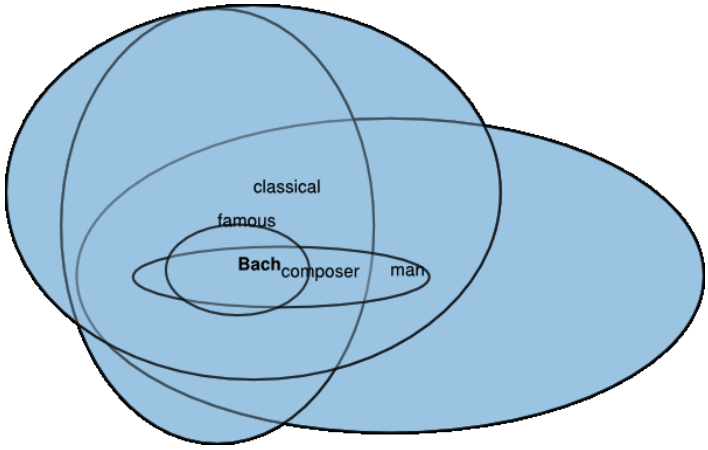


FIGURE 2.3: Learned Gaussian representations with diagonal variances, from (Vilnis et al., 2015). The first letter of each word indicating the position of its mean.

Besides its interesting properties, gradients with respect to the means and covariance matrices for this energy function can be computed in closed form:

$$\begin{aligned}
\frac{\partial E(Z_i, Z_j)}{\partial \mu_i} &= -\frac{\partial E(Z_i, Z_j)}{\partial \mu_j} = -\Delta'_{ij} \\
\frac{\partial E(Z_i, Z_j)}{\partial \Sigma_i} &= \frac{1}{2}(\Sigma_i^{-1}\Sigma_j\Sigma_i^{-1} + \Delta'_{ij}\Delta'^{\top}_{ij} - \Sigma_i^{-1}) \\
\frac{\partial E(Z_i, Z_j)}{\partial \Sigma_j} &= \frac{1}{2}(\Sigma_j^{-1} - \Sigma_i^{-1})
\end{aligned} \tag{2.15}$$

where $\Delta'_{ij} = \Sigma_i^{-1}(\mu_i - \mu_j)$. This allows (Vilnis et al., 2015) to learn the Gaussian representations by gradient descent.

(Vilnis et al., 2015) has inspired several recent work using the same framework on different tasks and datasets such as (He et al., 2015) for knowledge graphs datasets, (Mukherjee et al., 2016) for image classification and (Bojchevski et al., 2017) for attributed graphs with an inductive unsupervised framework. This work has also inspired three of our contributions on graph node classification (Chapter 5), forecasting relational time series (Chapter 6) and collaborative filtering (Chapter 7).

Figure 2.4 shows a visualization of Gaussian representations learned by the unsupervised model presented in (Bojchevski et al., 2017) for the Cora dataset[5]. Colors indicates the class label (not used during training).

2.5 Conclusion

In this chapter we have presented a state of the art concerning representation learning in the context of relational data.

[5]The Cora dataset consists of scientific publications classified into one of seven classes together with a citation network.

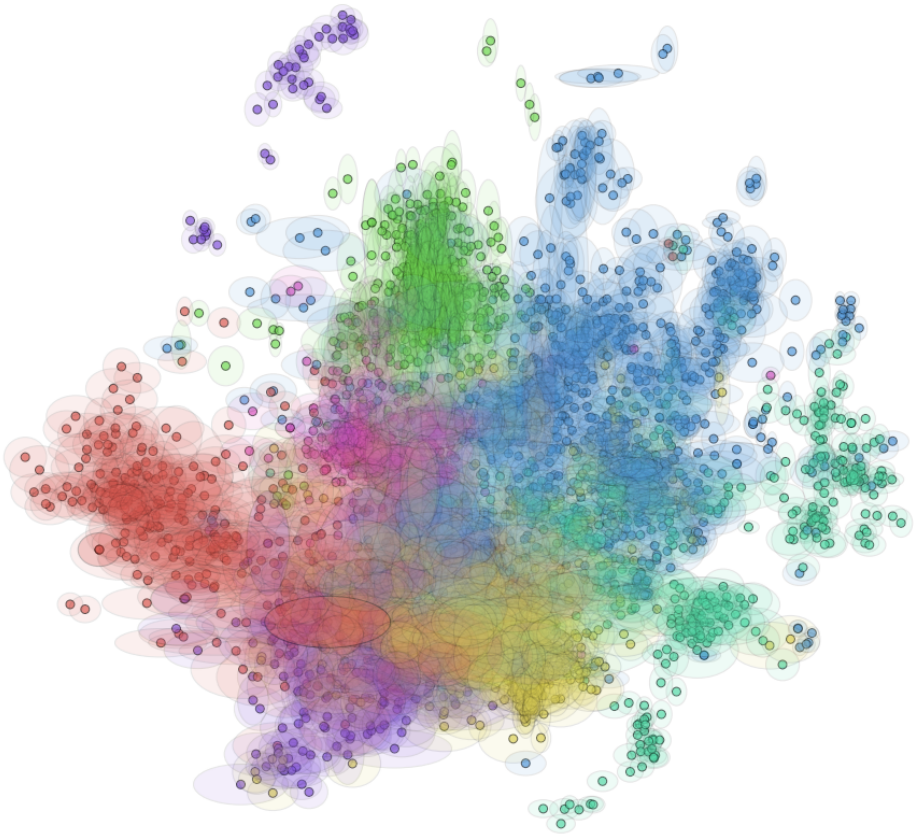


FIGURE 2.4: Learned two dimensional Gaussian representations of nodes on the Cora dataset from (Bojchevski et al., 2017). Color indicates the class label.

We have first presented general notions and concepts pertaining to supervision and transductive learning in order to position the different works presented in this chapter.

We have then described approaches that learn a deterministic representation for relational data, by differentiating unsupervised and (semi-)supervised models. We emphasized the proximity between some supervised and unsupervised models for different tasks. In this thesis, we apply this approach for the classification task (Chapter 4).

We then focused on works that learn distributions over latent variables, instead of deterministic representations, to capture uncertainty in representations. We have presented two ways to learn uncertain representations. The first is Bayesian methods that approximate the posterior distribution using variational techniques. The second is a more direct approach since it directly deals with Gaussian representations in the latent space, and use a discriminative approach to learn Gaussian representations. With this second approach we loose the probabilistic modeling, but gain in computation complexity and performances. We followed this path in this thesis, on three different tasks, namely classification (Chapter 5), forecasting (Chapter 6) and ranking (Chapter 7).

Part I

Learning Deterministic Representations and Application to Classification

In this first part, we tackle classification of nodes in a graph using deterministic vector representations learning.

We first present a state of the art concerning graph node classification for homogeneous and heterogeneous graphs (Chapter 3). In particular, we start by presenting the basis of transductive classification in graph models in the case of homogeneous graphs.

We then present our first contribution (Chapter 4) tackling the heterogeneous graph node classification task, using a deterministic representation learning framework.

The goal of this contribution is to propose one of the very first general algorithm for node classification in heterogeneous graphs. While homogeneous graph classification has been well explored, heterogeneous graph classification was, and still is, a topic largely unexplored.

Chapter 3

State of the Art: Graph Node Classification

Abstract

In this chapter we present a state of the art concerning graph node classification. We give an overview of the different techniques used in homogeneous and heterogeneous graph node classification to show the evolution of such techniques. We present the baseline model of transductive homogeneous graph node classification which inspired representation learning ones. We then describe state of the art models, including the ones used as baselines in our contributions. A lot of those works are specific to homogeneous graphs, whereas our contributions deal with heterogeneous ones.

Contents

3.1 Introduction 37
3.2 Graph Node Classification 38
3.2.1 Simple Transductive Model for Graph Node Classification 39
3.2.2 Other Graph Node Classification Works 39
3.3 Conclusion 41

3.1 Introduction

Social networks (Facebook, Google+, ...) and social media (Twitter, LastFM, ...) are collecting a large amount of data. There exists several types of graphs from different social networks or knowledge bases. Some of them are homogeneous, i.e. composed of nodes of the same type (e.g. CoRA, ...), others are heterogeneous, i.e. composed of different types of nodes (e.g. LibraryThing, ...). Moreover those graphs can be either mono-labeled (one label per node) or multi-labeled (more than one label per node). At last, graphs can have only one type of relation/edge or different types of relations (see Section 1.1.2).

The problem of graph node classification emerged from several application domains like web data mining or biology. Initially, work has mainly focused on homogeneous graphs where nodes and relations are homogeneous – only one type of node and relation. The analysis of more complex data like those coming from social sources or knowledge bases has motivated a growing interest in more complex graphs where nodes may be of different types, share different and sometimes multiple relations. Heterogeneous graph classification is a recent trend and is still an open problem.

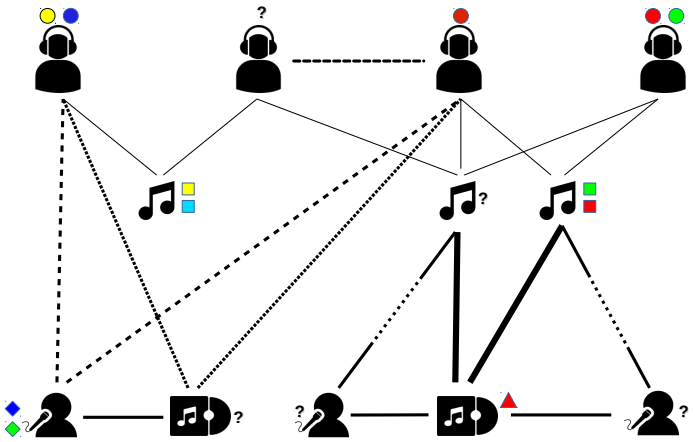


FIGURE 3.1: Representation of the LastFM network with users, tracks, albums and artists. The different relations are represented with different types of lines.

In this part, we focus more particularly on graph node classification for heterogeneous graphs where different types of nodes (users, authors, films, ...) interact through different types of relations (friendship, like, authorship, ...). For example, the LastFM social network used in our experiments (Figure 3.1) links users, tracks, artists and albums via seven different types of relations such as *friendship*, *most listened tracks*, *authorship*, etc.

We review below the main directions for graph nodes classification on both homogeneous or heterogeneous graphs.

3.2 Graph Node Classification

The growing importance of relational data has motivated several works on homogeneous graph nodes classification and ranking (Pimplikar et al., 2014; Namata et al., 2016; Denoyer et al., 2010; Duan et al., 2012; Getoor et al., 2005; Wang et al., 2013; Monner et al., 2013; Gu et al., 2013; Sacca et al., 2013; Belmouhcine et al., 2015). For example, web page classification may be formulated as a homogeneous graph node classification task where nodes are web pages, edges between web pages are hyperlinks and node labels are the web page topics. The problem has been formulated using different frameworks leading to three main families of methods (Bengio et al., 2006; Bhagat et al., 2011; Sen et al., 2008) (i) iterative classification algorithms (ICA) (ii) random walk based methods and (iii) semi-supervised and transductive graph regularized models.

We first present a simple baseline model of transductive homogeneous graph node classification that serves as a basis to other models before presenting the state of the art.

3.2.1 Simple Transductive Model for Graph Node Classification

We describe in this section a transductive model (and its inductive counterpart) for graph node classification outside the representation learning framework. This model, even if it does not learns representation, has a similar approach as our first two contributions. Transductive learning in graph models often uses a loss function with the following form (Zhou et al., 2003; Zhou et al., 2005):

$$(\hat{y}_1, \hat{y}_2, ..., \hat{y}_m) = \underset{\tilde{y}_1,\tilde{y}_2,...,\tilde{y}_m}{\texttt{argmin}} \sum_{i=1}^{m} \Delta(\tilde{y}_i, y_i) + \lambda \sum_{(i,j)\in E} w_{ij}||\tilde{y}_i - \tilde{y}_j||^2 \tag{3.1}$$

where Δ corresponds to the cost of predicting scores $\tilde{y}_i \in \mathbb{R}^m$ instead of true categories y_i for labeled nodes (m possible labels) and $||\tilde{y}_i - \tilde{y}_j||^2$ is a regularization term that encourages connected nodes to have the same score. λ is an hyper-parameter that corresponds to the smoothness of the solution, i.e. how close the distribution over labels should be for two connected nodes, one wants to obtain. Many variations of this formulation have been proposed with different prediction and regularization losses.

As described in Chapters 4 and 5, we use a similar loss function in our contributions but instead of learning scores we learn latent representations.

When input features are available, we can use an inductive version of the previous model by learning a parametric function ϕ_θ. The loss function become:

$$\hat{\theta} = \underset{\theta}{\texttt{argmin}} \sum_{i=1}^{m} \Delta(\phi_\theta(x_i), y_i) + \lambda \sum_{(i,j)\in\mathcal{E}} w_{ij}||\phi_\theta(x_i) - \phi_\theta(x_j)||^2 \tag{3.2}$$

where θ are the learned parameters of the function ϕ_θ.

As explained in Section 2.2.1, the inductive form of the model learns a classification function over the entire space of input features, whereas the transductive form focuses on unlabeled nodes only. Thus, if we want to label an unseen node we need to run the algorithm again considering the transductive model, whereas we can use the learn parametric function in the case of the inductive one.

3.2.2 Other Graph Node Classification Works

Iterative Classification Algorithms Iterative classification algorithms (ICA) extend the classical inductive classification schemes to relational data. They consist in iteratively building a local classifier at each node using as inputs both node characteristics and statistics on the node neighbors current labels. For example (Neville et al., 2000) and (Lu et al., 2003) respectively use Bayesian classifiers and logistic regression in such a scheme where statistics on the neighbors are iteratively updated. During inference, local node classification is then repeatedly performed until convergence. Several variations and extensions of these ideas have been proposed. (Peters et al., 2010) develop an extension of (Jensen et al., 2004) for multi-labeled classification problems on multi-graphs. When iteratively learning the classifier, using only existing edges can be somehow restrictive, i.e. in some cases (e.g. sparsely labeled networks), adding more information than edges and labels can improve the results. To do so, Gallagher et al. proposed an iterative model (Gallagher et al., 2008) that adds

new links to the existing graph using random walks. Another important information that can be used in addition to existing edges is the label distribution: iterative models were introduced for (sparsely) labeled networks, which forces the distribution of the predicted labels to match the distribution of the observed data with a maximum entropy constraint (Pfeiffer III et al., 2015) or a regularization (Mann et al., 2010) based on the KL-divergence between the expected and the empirical label distribution (McDowell et al., 2012). At last, one common way to improve the input graph in iterative classification is to weight the existing links. For example, in (Wang et al., 2013), Wang et al. proposed a weighting scheme based on social context features by capturing the node's intrinsic likelihood of belonging to each class. The weights then serve as a prior for each class when aggregating the neighbors' class labels in the collective inference procedure.

Random Walks Random walks have been used for graph nodes classification: labels are propagated from labeled to unlabeled nodes using the graph structure – this is thus one instance of the transductive setting. For homogeneous graphs, Zhu et al. proposed *Homogeneous Label Propagation (HLP)* (Zhu et al., 2002) which iterates until convergence the walk $Y \leftarrow T \cdot Y$ where Y is the matrix of predicted labels and T characterizes the transition matrix of the graph. This branch of research has motivated a large number of contributions. Recent contributions (Devooght et al., 2014; Zhou et al., 2014; Nandanwar et al., 2016) have focused on how the random walk could take into consideration more specific information, such as graph communities (Devooght et al., 2014), label distribution, etc. For instance, (Zhou et al., 2014) recently proposed a model for heterogeneous graphs which relies on manually defined *activity graphs* designed by an expert for the classification task at hand. The transition matrix used in a random walk then incorporates this knowledge. An interesting model for heterogeneous classification is *Graffiti* (Angelova et al., 2012). It is based on two intertwined random walks. The first one operates between nodes of the same type which are directly connected in the graph while the other one is defined between nodes of the same type which are connected through a node from another type. Although the model only considers 2 hop jumps in the graphs for incorporating heterogeneous information, it is very competitive and according to our experiments with different models represents the state of the art in the domain. It is one of the baselines used in our experiments.

Transductive semi-supervised models The transductive semi-supervised models learn an embedding for each node (either in the training or testing set) to predict the class labels using learned classifiers. Our model is part of this framework. (Yang et al., 2016) have recently proposed a model that learns two different representations for a node, one from some features associated to the node, and another one from the graph alone, in an unsupervised transductive way, based on (Perozzi et al., 2014). The latter learns node representations using only the context, like the SkipGram (Mikolov et al., 2013) model, but instead of words the context is here composed of the neighboring nodes. This can be applied to label prediction and the former uses the label when learning the representations.

The most active research direction for graph nodes classification in the machine learning community is probably semi-supervised learning based on transductive regularized models (Zhou et al., 2005; Belkin et al., 2006; Zhou et al., 2003; Pimplikar et al., 2014). These models optimize an objective function that encourages connected nodes to have the same labels. The loss usually incorporates a classification term that exploits available label information and a propagation term that allows labels to be propagated from labeled to unlabeled nodes in the graph under the hypothesis that neighbors should have similar labels. Contrary to iterative classification algorithms, a latent representation is learned for each node and no input vector is mandatory. As for the random walk models, these methods are intrinsically designed for homogeneous graphs and do not naturally extend to heterogeneous ones. (Ye et al., 2015) tackled the homogeneous graph node classification task when multi-relational graphs are available (i.e. when there exists different types of relations). They assume that relations may have varying levels of informativeness and they propose a weighting scheme to filter out irrelevant types of relations. To do so, they minimize their objective function with the different graphs associated with their level of informativeness (evolving through the learning process) to predict the labels of unlabeled nodes. Models for heterogeneous graphs have been proposed in the case where the label set is the same for all node types (Ji et al., 2010; Hwang et al., 2010). In that case, simply ignoring the node type allows to use frameworks developed for the homogeneous case (Hwang et al., 2010). (Ji et al., 2010) have nevertheless made a step towards heterogeneous networks by introducing weights on relations between node types and thus differentiating in a way the different types of nodes.

3.3 Conclusion

In this chapter, we presented a state of the art concerning the classification task in the context of relational data. We introduced a variety of models tackling the graph node classification task. We have described models for both homogeneous and heterogeneous graphs, and we have shown how important it was to have specific models for heterogeneous graphs. We have presented a simple generic model in its transductive and inductive form, which could serve as a basis for many transductive classification algorithms. We highlighted how few models were developed specifically for heterogeneous graphs. Chapters 4 and 5 will emphasize the necessity of taking into account the heterogeneity of the graph and will show its impact in term of results.

Chapter 4

Learning Deterministic Representations for Classification

Abstract

In this chapter we present our first contribution. We apply the representation learning framework to the task of node classification in heterogeneous networks. Here the nodes may be of different types, each type with its own set of labels, and relations between nodes may also be of different types. A typical example is provided by social networks where the node types are e.g. users, content, groups, etc. and relations may be *friendship*, *like*, *authorship*, etc. Learning and performing inference on such heterogeneous networks is a recent task requiring new models and algorithms. We propose a model, **Labeling Heterogeneous Network (LaHNet)**, a transductive approach to classification that learns to project the different types of nodes into a common latent space. This embedding is learned so as to reflect different characteristics of the problem such as the correlation between labels of nodes with different types and the graph topology. The application focus is on social graphs but the algorithm is general and could be used for other domains as well. The model is evaluated on five datasets representative of different instances of social data.

Contents

4.1 Introduction 43
4.2 Learning Representation for Graph Node Classification 45
4.2.1 Model 45
4.2.2 Prior Parameters and Learning Algorithms 47
4.2.3 Algorithm 50
4.2.4 Experiments 50
4.3 Conclusion 68

4.1 Introduction

We consider here the generic task of multilabel node classification in the general case of heterogeneous graphs with multiple node and relation types. Graph node classification has been investigated in machine learning since the early 2000. Most work until now has focused on homogeneous graphs where nodes are of the same type, share the same label set, and relations reflect a unique type of dependency.

The main assumption in most homogeneous graph classification approaches is that *two connected nodes tend to have the same labels*, so that graph labeling can be performed using some form of label propagation from labeled nodes to their unlabeled neighbors (Bhagat et al., 2011; Abernethy et al., 2008; Zhou et al., 2003). Work in this domain can be grouped in two main families : transduction/diffusion models and induction (feature-based) models. Transduction models use all the nodes (labeled and unlabeled) to predict the labels of the entire graph, in contrast to induction models, where classifiers are trained only on labeled nodes.

Extending the ideas used for homogeneous graphs to heterogeneous ones is not trivial. For example in transductive approaches, the label propagation paradigm becomes irrelevant when neighbors are of different types and hence have different sets of labels. Different attempts have tried to reformulate the problem so as to use already known homogeneous methods. The simplest idea developed by several authors is to project the heterogeneous graph onto a family of homogeneous ones (one node type, one relation type) and then perform classification independently for each projection. This basic approach suffers from several drawbacks i) defining semantically meaningful projections is problem specific, ii) the number of projections could rapidly become extremely large, iii) as shown in our experiments, correlations between different types of nodes may play an important role in the labeling task, and are lost after the projection. Another simple alternative consists in using restrictive assumptions so that homogeneous label propagation can be used. (Ji et al., 2010; Hwang et al., 2010) for example consider heterogeneous graphs where the label set is the same for all node types. Only some works have tackled the general heterogeneous node labeling problem with new algorithms, such as (Angelova et al., 2012) who defined new forms of random walks on heterogeneous graphs specifically adapted to this problem.

While all the above methods have been designed for operating directly on the discrete graph structure, we follow here a different path by using a representation learning approach (Bengio et al., 2013). We propose to map the classification problem onto a continuous space of latent node representations. This representation space is common to all node types thus allowing to handle heterogeneity. Node embedding in this space reflects both the classification objective for each type of node and its associated label set, and the heterogeneous relational structure of the graph. Classification is then performed directly on the latent space. This idea has been introduced in a preliminary work (Jacob et al., 2014). The current contribution extends this work in several ways: we introduce prior parameters based on the graph characteristics, we enrich the model with hyperparameters trained to weight the relative importance of the different relation types and propose an alternate optimization scheme for learning the weights, and we perform extensive experimental evaluations on social datasets representative of different situations .

Summarizing, our contributions are the following:

- We address the heterogeneous graph multi-label classification problem as a representation learning problem in a continuous latent space. We consider a transductive approach to graph node classification.
- We propose new algorithms for learning this representation space able to handle heterogeneity for both nodes and relations.
- We develop an experimental analysis of the algorithm behavior.

- We perform an extensive experimentation with five datasets, representative of different experimental situations and a comparison with baselines, including state of the art models.
- We study how much our hypotheses hold on the different datasets.

4.2 Learning Representation for Graph Node Classification

4.2.1 Model

Our model, **Labeling Heterogeneous Network (LaHNet)**, has been designed for transductive node classification in heterogeneous graphs. The objective is to learn node representations so that each type of node can be correctly classified while exploiting the correlations between the labels of different node types. These interdependencies between nodes of different types play an important role in several classification problems and none of the current methods is able to handle them. The variables to be learned are: a latent representation $z_i \in \mathbb{R}^d$ for each node x_i, relation weights w_r for each relation type r and classifiers parameters θ^t, where t indicates the node type.

Notation	Meaning
Z	Dimension of the latent space
z_i	Latent representation (in $\mathbb{R}^Z$) of node x_i
w_r	Weight of the relation type r
ψ_i	Prior parameter reflecting the relative importance of node x_i
ϕ_{ij}	Prior parameter reflecting the relative importance of relation between i and j
$\mathcal{L}_c$	Set of labeled nodes used for learning the classifiers f_{θ^t} and the labeled node representations z
$\mathcal{L}_w$	Set of labeled nodes used to learn the relation weights w_r

TABLE 4.1: Model notations

Loss Function

The loss function takes the general form of a transductive regularized loss (Ji et al., 2010; Zhou et al., 2003), with a classification term L_C and a regularization term L_G :

$$L(z,\theta) = \sum_{i\in\mathcal{L}_c} \psi_i \Delta_C(f_{\theta^{t_i}}(z_i), y_i) + \lambda \sum_{i\in\mathcal{N}} \sum_{j\in N_i} w_{r_{ij}} \phi_{ij} \Delta_G(z_i, z_j) = L_C + \lambda L_G \qquad (4.1)$$

where Δ_C is a classification loss and Δ_G is a graph regularization loss; $f_{\theta^t}(.)$ is a parametric classifier for nodes of type t – there is one such classifier for each node type; ϕ_{ij}, ψ_i and $w_{r_{ij}} \geqslant 0$ are real parameters that will be discussed later (Section 4.2.2); $\lambda \in \mathbb{R}$ is a regularization weight. Other notations for the model description and parameters are summarized in Table 4.1.

In a classical transductive homogeneous graph formulation (Ji et al., 2010; Zhou et al., 2003; Abernethy et al., 2008; Zhou et al., 2003; Zhou et al., 2005), the z_i are labels and the

regularization term operates as a diffusion equation for propagating labels from labeled nodes to their unlabeled neighbors. Here, the z_is have a different role. They are latent node representations in $\mathbb{R}^d$. The second term in equation 4.1 thus forces neighbors to have close latent representations z whatever their type is. This allows one to exploit correlations between labels of nodes with different types and to handle the classification problem in heterogeneous graphs.

The terms L_C and L_G are optimized simultaneously w.r.t. parameters z_i for each node x_i and w.r.t. θ^t for each node type t.

Let us now detail the components of the loss in equation 4.1.

Classifier

The mapping onto the latent space is learned so that the labels for each type of node can be predicted from the latent representations. For that, we use a type specific classifier denoted by f_{θ^t} for node type t. This classification function takes as input a node representation z_i and outputs a vector of classification scores $f_{\theta^{t_i}}(z_i)$ for the classes associated with the node type.

The parameters θ^t of the classifier are learned by minimizing the first term in equation (4.1) over the labeled dataset $\mathcal{L}_c$. In this first term, $f_{\theta^{t_i}}(z_i)$ and y_i are respectively the classification score and target vector associated to x_i. $\Delta_C(f_{\theta^{t_i}}(z_i), y_i)$ is the loss of predicting labels $f_{\theta^{t_i}}(z_i)$ for the target y_i, and ψ_i represents a prior node relative importance (see Section 4.2.2).

In the experiments, we used a hinge-loss function for Δ_C:

$$\Delta_C(f_{\theta^t}(z), y) = \sum_{k=1}^{\#\mathcal{Y}^t} \max(0; 1 - y^k f^k_{\theta^t}(z)) \tag{4.2}$$

where y^k is equal to 1 if x belongs to category k and -1 otherwise, and $f^k_{\theta^t}(z)$ is the predicted score of category k, i.e. the k-th component of the vector $f_{\theta^t}(z)$.

Transductive graph model

The second term in equation 4.1 exploits the proximity of the nodes in the graph. We suppose that nodes linked in the graph will tend to have similar representations. Nodes of the same type which are close one to the other in the graph, but separated by nodes from other types, will also have similar representations and will tend to have similar labels.

We used a L2 term $\Delta_G(z_i, z_j) = ||z_i - z_j||^2$ for this loss:

$$L_G = \sum_{i \in \mathcal{N}} \sum_{j \in N_i} w_{r_{ij}} \phi_{ij} ||z_i - z_j||^2 \tag{4.3}$$

where N_i denotes the graph neighbors of node i, $w_{r_{ij}} \in \mathbb{R}^+$ is a learned weight for the relation of type r_{ij} (Section 4.2.2), and $\phi_{ij} \in \mathbb{R}$ represents a prior on the relative importance of the relation between x_i and x_j (Section 4.2.2).

4.2.2 Prior Parameters and Learning Algorithms

The loss function (Equation 4.1) makes use of three graph related parameters: ϕ_{ij}, ψ_i and w_r. Introducing these three families of parameters allowed us to significantly improve the model performance. ϕ_{ij} and ψ_i are priors taking into account local graph characteristics and $w_r s$ are learned relation weights reflecting the relative importance of the different relation types for the classification task. The three families of parameters are described below.

Prior Parameters

Let us suppose the ϕ_{ij}, the ψ_i and the w_r parameters be set to 1 in loss function (Equation 4.1). When optimizing this loss, all nodes will have an equal importance (term L_C) and similarly, all the edges will have the same importance (term L_G). This is not desirable for the classification task. Central nodes, with a large number of neighbors, have more influence than others on their neighbors. Wrong decision at these nodes should be over penalized. This is the role of the coefficient ψ_i which is associated with node i. Besides node priors, some relation types between nodes might be important for the classification objective, but if they are under-represented in the graph, they will be ignored. It is then important to weight them up. This is the role of the prior coefficient ϕ_{ij}. Both weights exploit local graph information and could be considered as priors based on local graph characteristics, and we experimented with different values for the two parameters. For ψ_i, we found that the following formula behaved well: $\psi_i = \frac{1}{R}\sum_{r=1}^{R}\frac{N_i^r}{E^r}$, with N_i^r the number of neighbors for relation r and E^r the total number of relations of type r in the graph. It is a measure of centrality of node i. More specifically, it measures the proportion of outgoing edges macro-averaged over the type of relation. This aims at favoring type specific central nodes. For ϕ_{ij}, we found that the formulation proposed in (Angelova et al., 2012) was the most efficient, i.e. $\phi_{ij} = \frac{1}{RN_j^{r_{ij}}E^{r_{ij}}}$, with $N_j^{r_{ij}}$ the number of neighbors of j considering only the relation type r_{ij}. It gives a similar importance to all relation types (the $E^{r_{ij}}$ coefficient), weighted locally according to the destination node j (the $N_j^{r_{ij}}$ coefficient).

Learned Relation Specific Parameters

The graph regularization coefficients w_r are relationship-specific. They reflect the importance of relation r for the inference task. For example, if inference consists in classifying an author research domain, then the authorship relation between the author and his published papers is probably more important than his affiliation relation. For the model, this means that authors representations should be close to their papers representations, whereas no such constraint operates on the affiliation representation. The w_r are hyperparameters which could be learned by grid search and cross-validation. Since there might be several relation types for heterogeneous graph this is not a relevant option here. We used instead the framework of continuous optimization of hyperparameters (Bengio, 2000; Luketina et al., 2016).

This framework has been developed in (Bengio, 2000) for learning regularization hyperparameters. Given a regularized loss such as Equation 4.1, hyperparameters are learned simultaneously to the model parameters by optimizing the unregularized loss on a distinct training set $\mathcal{L}_w$. Contrarily to grid search which also selects regularization hyperparameters using an unregularized loss on a validation set, by testing different preset hyperparameters

values, here parameters and hyperparameters are learned simultaneously and their dynamics are intertwined. There is no formal proof that such procedures converge to an optimal choice of the hyperparameters, but they offer an approximate solution which performs well in many cases, see (Luketina et al., 2016) for a discussion. There has been several instances of this general framework, and we derive below our own version for the specific problem handled here. Our inference problem is classification, so that we use as a loss function for hyperparameter training the classification objective denoted L_W, defined on $\mathcal{L}_w$, a set of labeled nodes distinct from the labeled set $\mathcal{L}_c$ used in Equation 4.1 ($\mathcal{L}_w \cap \mathcal{L}_c = \emptyset$).

$$L_W = \frac{1}{\#\mathcal{L}_w} \sum_{i \in \mathcal{L}_w} \Delta_C(f_{\theta^{t_i}(w)}(z_i(w)), y_i) \tag{4.4}$$

The proposed learning scheme makes use of an alternating optimization algorithm for learning the parameters $\{z_i\}$, $\{\theta^{t_i}\}$, on one side, and the $\{w_{r_{ij}}\}$, on the other side.

The training algorithm iteratively alternates between (step 1) the optimization of Equation 4.1 w.r.t. parameters θ and z, the ws being fixed, and (step 2) optimization of Equation 4.4 w.r.t. parameters w. At each iteration of this process, the value of θ and z after step 1, are dependent of the current ws. This dependency is emphasized in Equation 4.4 by the notations $z(w)$ and $\theta^t(w)$. Note that the optimization for loss Equation 4.1 is performed on all the nodes (set $\mathcal{L}_c$ for the supervised term and all the labeled and unlabeled nodes for the regularization term) while the optimization for Equation 4.4 is performed on the labeled set $\mathcal{L}_w$ only. In order to optimize loss Equation 4.4, let us derive a closed form dependency relation on w for $\frac{\partial L_W}{\partial w}$.

Using the chain rule, the derivative of $L_W(z(w), \theta(w), \mathcal{L}_w)$ w.r.t. w is:

$$\frac{\partial L_W}{\partial w} = \frac{\partial L_W}{\partial \theta} \frac{\partial \theta}{\partial w} + \frac{\partial L_W}{\partial z} \frac{\partial z}{\partial w} \tag{4.5}$$

In order to make this computation tractable, we will make two assumptions.

- Step 1 has reached a minimum of L in Equation 4.1. Based on this hypothesis, by zeroing $\nabla_z L$ on $\mathcal{L}_w$, we can express $z \in \mathcal{L}_w$ as an explicit function of w:

$$z_i^\star = \frac{\sum_{r=1}^R w_r \sum\limits_{j : i \xrightarrow{r} j} z_j \phi_{ij}}{\sum_{r=1}^R w_r \sum\limits_{j : i \xrightarrow{r} j} \phi_{ij}} \tag{4.6}$$

  where $j : i \xrightarrow{r} j$ corresponds to the set of r-neighbors of i.

- The second assumption is technical. We suppose that $\frac{\partial \theta}{\partial w}$ is negligible compared to $\frac{\partial z}{\partial w}$, i.e. $\left|\frac{\partial \theta}{\partial w}\right| \ll \left|\frac{\partial z}{\partial w}\right|$, and that $\frac{\partial L_W}{\partial \theta}$ and $\frac{\partial L_W}{\partial z}$ have roughly the same order of magnitude. This is a reasonable assumption since θ is only indirectly impacted by w, when z directly depends on the values w (this was confirmed experimentally).

  Under this assumption, one can approximate the partial derivative (4.5) as:

$$\frac{\partial L_W}{\partial w} \approx \frac{\partial L_W}{\partial z} \frac{\partial z}{\partial w} \tag{4.7}$$

Under these hypothesis, by plugging Equation 4.6 into Equation 4.4 and for $r \in \mathcal{R}$ the derivative of the unregularized loss over w_r can then be written as:

$$\frac{\partial L_W}{\partial w_r} \approx \sum_{i \in \mathcal{L}_w} \sum_{k=1}^{\#\mathcal{Y}^{t_i}} \underbrace{\psi_i y_{ik} \frac{\sum_{r' \neq r} w_{r'} A_i^{r',r} \left(\theta^{t_i,k} \cdot S_i^{r'} - \theta^{t_i,k} \cdot S_i^{r} \right)}{\left(\sum_{r'=1}^{R} w_{r'} A_i^{r',r} \right)^2}}_{D_{i,r}^k} \tag{4.8}$$

with

$$\begin{aligned} S_i^r &= \frac{\sum_{j : i \xrightarrow{r} j} z_j \phi_{ij}}{\sum_{j : i \xrightarrow{r} j} \phi_{ij}} \\ A_i^{r',r} &= \frac{\sum_{j : i \xrightarrow{r'} j} \phi_{ij}}{\sum_{j : i \xrightarrow{r} j} \phi_{ij}} \end{aligned}$$

$\frac{\partial L_W}{\partial w_r}$ measures the variation of the empirical risk (Equation 4.4) w.r.t w_r changes. In order to develop some intuition on Equation 4.8, let us suppose that all the ϕ and ψ coefficients are equal to 1 (they are all positive anyway). We will consider the inner term in Equation (4.8), i.e. term $D_{i,r}^k$, and more precisely its sign. Without loss of generality, let us consider the case of a target class $y_i^k = 1$. The sign of $D_{i,r}^k$ is then determined by $\sum_{r' \neq r} w_{r'} A_i^{r',r} \left(\theta^{t_i,k} \cdot S_i^{r'} - \theta^{t_i,k} \cdot S_i^{r} \right)$. Since $\left(w_{r'} A_i^{r',r} \right)$ is positive the sign depends on $\theta^{t_i,k} \cdot (S_i^{r'} - S_i^r)$. In this expression, S_i^r is the average of the z_j vectors for all j r-neighbors (neighbors according to relation r) of node i and $\left(\theta^{t_i,k} \cdot S_i^r \right)$ is the mean score for class k of the r-neighbors of node i. Similarly, $\theta^{t_i,k} \cdot S_i^{r'}$ is the mean score for class k of the r'-neighbors of node x_i. The difference $\theta^{t_i,k} \cdot (S_i^{r'} - S_i^r)$ will be negative, leading to an increase of w_r, if the mean score of the r-neighbors is higher than the mean score of the r'-neighbors. The same reasoning holds for $y_i^k = -1$. Said otherwise, the weight of relation r will be increased if the center of mass of the r-neighbors is better located (w.r.t. the classification task) than the center of mass of the other neighbors, i.e. if it reinforces the correct classification, and will be decreased otherwise.

$z_i^\star$ in Equation (4.6) is invariant to a global scaling of w_r for $r \in \mathcal{R}$ leading to an infinity of possible solutions. In order to avoid numerical problems, we normalize the w_r to force a unique solution. We used the following normalization:

$$\text{for any node type } t, \sum_{r : t \to \bullet} w_r = 1 \tag{4.9}$$

where $r : t \to \bullet$ indicates all the relation types r from a node of type t to any other type of node $\bullet$.

4.2.3 Algorithm

Algorithm 1 describes the alternating optimization scheme for learning the θs, zs and ws.

Step 1: learning θ and z corresponds to lines 1 to 18. One samples a pair of connected nodes i and j and then makes a gradient update of the parameters for loss Equation 4.1. A node in $\mathcal{L}_c$ appears in the two terms of Equation 4.1. The update w.r.t the first term is indicated lines 8-9 for node i and 13-14 for node j and consists in successively modifying the parameters of the classification function θ and of the latent representations z_i and z_j so as to minimize the classification loss. For all nodes, be they in $\mathcal{L}_c$ or not, the model updates the parameters w.r.t the graph loss (second term of Equation 4.1) – lines 16-17. These different steps are repeated up to a fixed number of iterations.

Step 2: Using as training set $\mathcal{L}_w$, one then learns the ws by Gradient Descent on L_W. We compute the derivatives w.r.t. the relations by summing over all the nodes in $\mathcal{L}_w$ – line 26 – according to Equation 4.8. Then we update w_r – line 32. Finally, we normalize w_r – line 34 – (Equation 4.9). This alternating scheme is iterated for a fixed number of iterations.

For Step 1, we have been using a stochastic gradient. For Step 2, because of normalization (Equation 4.9), using a batch gradient, as described in Algorithm 1, was more convenient.

Note that in Algorithm 1 ϕ_i and ψ_{ij} do not appear explicitly. They are implicitly computed through the sampling procedure: we firstly sample uniformly the type of edge (line 4), and then choose uniformly an edge given that type (line 5). This sampling corresponds to optimizing Equation 4.3 with hyperparameters $\phi_i = \frac{1}{R}\sum_{r=1}^{R}\frac{N_i^r}{E^r}$ for the classification term and $\psi_{ij} = \frac{1}{RN_j^{r_{ij}}E^r}$ for the graph regularization one.

In Algorithm 1, ϵ is the gradient step, and λ is the trade-off between the classification and smoothness terms. Both parameters were set by grid search.

4.2.4 Experiments

Datasets

Experiments have been performed on five different datasets respectively extracted from DBLP, FlickR, LastFM and IMDB. For all but the first dataset (DBLP), each node can have multiple labels. The datasets are described below. Statistics for the datasets are provided in Table 4.2.

The **DBLP** dataset [1] (Sun et al., 2009) is a bibliographic network composed of authors and papers. We consider here two different sets of labels: authors are labeled with their research domain (4 domains) while papers are labeled with the conference name they were published in (20 labels). Authors and papers are connected through an *authorship* relation. The network is then composed of two types of nodes and is bipartite, with one relation type. Classification is monolabel on papers and authors.

The **FlickR** corpus is composed of photos and users. The photo labels correspond to different possible tags while the user labels correspond to their subscribed groups. The classification problem is multi-label: images and users may belong to more than one category. Photos are related to users through an *authorship* relation, while users are related to other users through a *following* relation. We have kept the image tags that appear in at least

[1] The dataset is available at http://web.cs.ucla.edu/ yzsun/data

Algorithm 1: Alternating Optimization Algorithm

Input: z node representations, w weights, ϵ gradient step, λ trade-off, $\mathcal{L}_c$,$\mathcal{L}_w$;
Output: Learned z and w;

while *not converged* **do**
  for *A fixed number of iterations* **do**
    ▷ Learn the representation z_i and the classifier parameters θ;
    Choose r in $\mathcal{R}$ at random;
    Pick randomly an edge $i \xrightarrow{r} j$;
    if $i \in \mathcal{L}_c$;
    then
      $\theta \leftarrow \theta - \epsilon \nabla_\theta \Delta(f_\theta^{t_i}(z_i), y_i)$;
      $z_i \leftarrow z_i - \epsilon \nabla_{z_i} \Delta(f_\theta^{t_i}(z_i), y_i)$;
    end
    if $j \in \mathcal{L}_c$;
    then
      $\theta \leftarrow \theta - \epsilon \nabla_\theta \Delta(f_\theta^{t_j}(z_j), y_j)$;
      $z_j \leftarrow z_j - \epsilon \nabla_{z_j} \Delta(f_\theta^{t_j}(z_j), y_j)$;
    end
    $z_i \leftarrow z_i - \epsilon \frac{w_{ij}}{N_j^{r_{ij}}} \lambda \nabla_{z_i} ||z_i - z_j||^2$;
    $z_j \leftarrow z_j - \epsilon \frac{w_{ji}}{N_i^{r_{ji}}} \lambda \nabla_{z_j} ||z_i - z_j||^2$;
  end
  ▷ Learn the w_r;
  Let grad_w $= [0, ..., 0]$;
  for $i \in \mathcal{L}_w$ **do**
    for k *in* $\mathcal{Y}^{t_i}$ **do**
      if $1 - y_i^k(\theta^{t_i,k}|z_i^\star) > 0$;
      then
        for r *in* $\mathcal{R}$ **do**
          grad_w[r] +=$D_{i,r}^k$;
        end
      end
    end
  end
  for r *in* $\mathcal{R}$ **do**
    $w_r \leftarrow w_r - \epsilon$grad_w[r];
  end
  Normalize w w.r.t. relations;
end

500 images, and user categories that appear also at least 500 times in the dataset resulting in 21 possible labels for photos and 42 for authors.

DBLP	Nodes	**Type**	**Nb. Nodes**	**Nb. Labeled Nodes**	**Nb. Labels**
		Paper	14,376	14,376	20
		Author	14,475	4,057	4
	Edges	**Type**	**Nb. Edges**		
		Author↔Paper	41,794		
FlickR	Nodes	Photo	46,926	8,766	21
		User	4,760	3,476	42
	Edges	User↔User	175,779		
		User↔Photo	46,926		
LastFM1	Nodes	User	1,013	321	59
		Track	35,181	24,562	28
		Album	32,118	15,966	47
		Artist	17,138	11,564	47
	Edges	User↔User	1,109		
		User↔Album	47,541		
		User↔Artist	47,812		
		User↔Track	47,807		
		Track↔Album	29,647		
		Track↔Artist	35,181		
		Album↔Artist	32,118		
LastFM2	Nodes	User	20,004	5,000	48
		Track	340,356	177,297	32
		Album	247,681	59,226	52
		Artist	103,951	60,388	59
	Edges	User↔User	23,573		
		User↔Album	951,274		
		User↔Artist	960,227		
		User↔Track	970,381		
		Track↔Album	162,364		
		Track↔Artist	340,356		
		Album↔Artist	247,681		
IMdB	Nodes	Film	2,724,246	1,050,323	28
		Actor	3,146,496	0	0
		Director	362,470	0	0
	Edges	Film↔Actor	14,308,748		
		Film↔Actress	8,526,214		
		Film↔Director	2,055,234		

TABLE 4.2: Statistics for the datasets

We used two different **LastFM** datasets denoted **LastFM1** and **LastFM2**, collected independently using the LastFM API[2]. Both datasets are social networks composed of users, tracks, albums and artists. The task is multi-label classification and all the node types have their own set of labels. Users are labeled with the type of music they like (59/48 labels respectively for **LastFM1** and **LastFM2**), tracks with the kind of music they belong to (28/32 labels), albums with their type (47/52 labels) and artists with the kind of music they play the most (47/59 labels). Users are linked to users (*friendship*), tracks (*favorite tracks*), albums (*favorite albums*) and artists (*favorite artists*). Tracks are linked to albums (*belong to*) and artists (*singer*). Finally, albums are linked to artists (*sing in*). Note that both a track and an album may be linked to several artists. This dataset contains tracks labeled by their genres (*rock*, *indie*, ...), users labeled by the type of music they like (*female vocalists*, *ambient*, ...), albums labeled by type (*various artists*, *live*, ...) and artists labeled by the role they hold (*singer*, *songwriter*, ...). The $LastFM$ graph is schematically represented in Figure 3.1. Some labels may overlap between different types of nodes but we suppose that in this case, they are distinct, e.g. *pop* is not the same for an artist or a track.

Finally, the **IMDB** dataset[3] is a movie description dataset composed of actors, actresses, films and directors. Only films are labeled with 28 possible labels corresponding to their type (Documentary, Drama, Comedy, ...). Films are related to actors and actresses using a *casting* relationship and to directors using a *film maker* relationship.

Construction of the LastFM Datasets

The LastFM datasets were extracted for this work, so we describe succinctly the procedure using the methods of the LastFM API. We selected randomly a set of 10 users, and collected user friends using the `User.getFriends` method. We stopped when reaching a given limit (1,000 users for LastFM1 and 20,000 users for LastFM2). Then, for each user of our graph we fetched his top tracks, albums and artists (methods `User.getTopAlbums` `User.getTopArtists` and `User.getTopTracks`). At last, we labeled tracks, albums and artists according to the top tags LastFM users have given to them using the methods `Track.getTopTags`, `Artist.getTopTags` and `Album.getTopTags`. For the users we chose the top tags they added with the method `User.getTopTags`.

Evidence For Heterogeneous Node Reciprocal Influence

Throughout this work, we make the assumption that correlations exist between the labels of different types of nodes, and that modeling this influence is useful for the heterogeneous classification task. We exhibit below some statistics and characteristics of the datasets to support this hypothesis. Let us consider the conditional probability $P(Y_{t_1}|X_{t_2})$ of label Y_{t_1} for a node of type t_1 given the label X_{t_2} of a neighbor of type t_2, for all the possible values of Y_{t_1} and X_{t_2}.

If there is a dependency between the two variables, this means that the conditional probability $P(Y_{t_1}|X_{t_2})$ will be high for some values of the couple (X_{t_2}, Y_{t_1}) and low for the other values, otherwise if the distribution is flat, this means that X_{t_2} does not bring any information on Y_{t_1}. In Figure 4.1 we have plotted some typical conditional distributions between

[2]To access the API go to http://www.lastfm.fr/api. The detailed procedure on how we collected these datasets is provided in Section 4.2.4.

[3]The dataset is available at http://www.imdb.com/interfaces

Y \ X	user	track	album	artist
user	$3.2 \cdot 10^{-3}$	$8.7 \cdot 10^{-5}$	$2.4 \cdot 10^{-4}$	$1.8 \cdot 10^{-4}$
track	$1.1 \cdot 10^{-3}$		$3.0 \cdot 10^{-4}$	$2.1 \cdot 10^{-4}$
album	$1.2 \cdot 10^{-3}$	$1.0 \cdot 10^{-4}$		$2.4 \cdot 10^{-4}$
artist	$1.2 \cdot 10^{-3}$	$4.4 \cdot 10^{-5}$	$1.4 \cdot 10^{-4}$	

TABLE 4.3: Conditional entropies $H(Y|X)$ for the lastFM2 corpus, X and Y being neighbors in the LastFM graph. Lower values mean higher dependency between the two variables

couples of variables. In order to make the graphics more readable, we have ordered labels X_{t_2} by increasing conditional entropy [4] $H(Y|X = k)$ along the x axis, for all the possible values X. The conditional entropy quantifies the amount of information needed to describe the outcome of Y given the value of X, so that the most informative X values are near the x axis origin (the most informative value $X_{t_2} = k$ will be situated at the origin $x = 0$). For each value k of X, i.e. for each x coordinate, we further order the Y values according to decreasing values of $P(Y|X)$ for all the possible values of Y, i.e. for each X value, the stronger dependencies are near the origin of the y axis (the red part of the plots).

Figure 4.1 shows 4 pairwise dependencies as measured by $P(Y_{t_1}|X_{t_2})$ on the LastFM2 dataset. All of them show clear peaks along the 0-y ordinate, and the peak values decrease when x increases. For all these examples, there is a clear dependency between the two variables. If we take a closer look, we can see (first row of the Figure) that $P(Y_{user}|X_{track})$ is more peaked than $P(Y_{user}|X_{user})$, meaning that the music a user listens to is more correlated to the user labels than to his friends labels. The second row of the figure shows a similar phenomenon: $P(Y_{artist}|X_{user}) < P(Y_{artist}|X_{track})$, meaning that the labels for an artist are more correlated to his musical production than to the labels of his followers. Additional figures exhibiting similar behaviors for other relations and corpora are available for DBLP (Figure 4.2), FlickR (Figure 4.3), and LastFM2 (Figure 4.4).

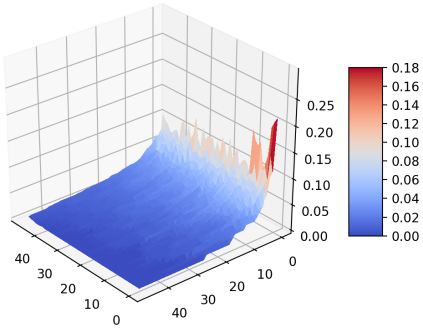
We also show under each figure the conditional entropy $H(Y|X)$ for each couple of variables. The most peaky distributions have an entropy which is two order of magnitude lower than the flatter distributions. Table 4.3 summarizes the conditional entropies computed for the different relations on the LastFM2 dataset, i.e.:

$$H(n_{t_1}|n_{t_2}) = \sum_{x \in \text{Label set}(n_{t_1}), y \in \text{Label set}(n_{t_2})} p(x,y) \log \frac{p(x)}{p(x,y)} \tag{4.10}$$

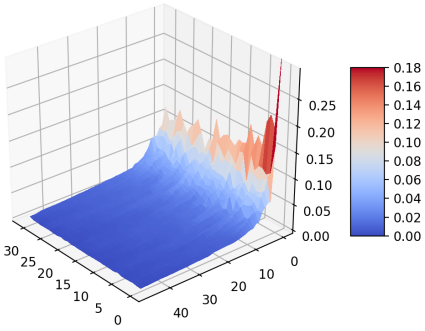
where n_{t_i} denotes the set of nodes of type t_i.

From those values, we could conclude that we should learn more from the tracks a user listen to than from the music his friends listen to. This matches the intuition of the semantics of such relations. We show in section 4.2.4 that this property is properly captured when learning the relation specific parameters w_r. Additional tables exhibiting similar behaviors for other relations and corpora are available for DBLP (Table 4.4) and FlickR (Table 4.5).

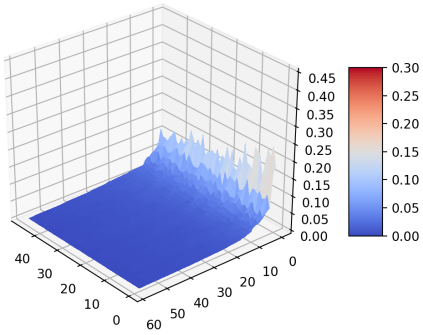
[4] $H(Y|X) = \sum_{X \in \mathcal{X}, Y \in \mathcal{Y}} p(X,Y) \log \frac{p(X)}{p(X,Y)}$



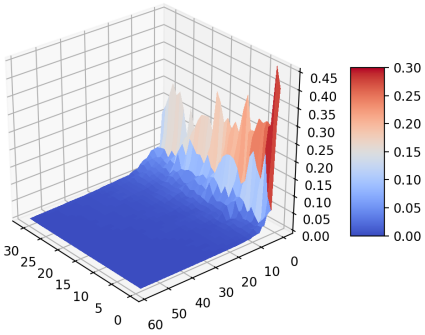
(A) $P(Y_{user}|X_{user})$ Conditional Entropy 3.2e-3



(B) $P(Y_{user}|X_{track})$ Conditional Entropy 8.7e-5



(C) $P(Y_{artist}|X_{user})$ Conditional Entropy 1.2e-3

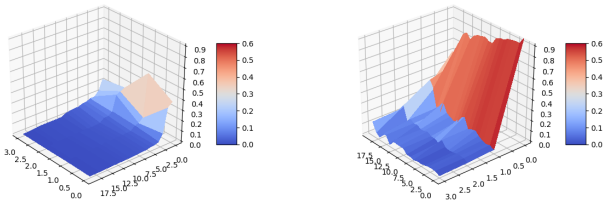


(D) $P(Y_{artist}|X_{track})$ Conditional Entropy 4.4e-5

FIGURE 4.1: Plots illustrating the inter-dependencies between labels of two node types for the LastFM2 dataset. Each plot shows $P(Y_{t_1}|X_{t_2})$ for a specific variable couple (Y_{t_1}, X_{t_2}). The values X_{t_2} and Y_{t_1} have been reordered for clarity (see text). The x axis corresponding to variable X_{t_2} is on the bottom left of each plot, and the y axis corresponding to the Y_{t_1} is on the bottom right.

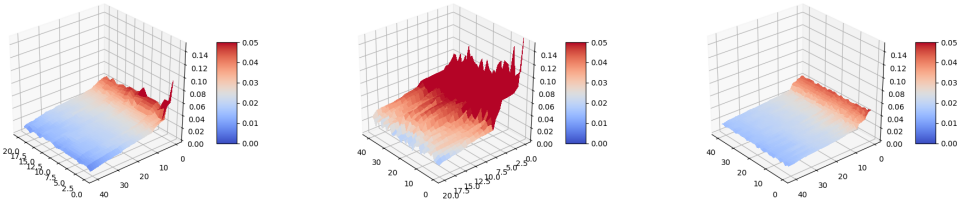
Y \ X	author	paper
author		$7.1 \cdot 10^{-4}$
paper	$2.5 \cdot 10^{-3}$	

TABLE 4.4: Conditional entropies $H(Y|X)$ for the DBLP corpus



(A) $P(Y_{paper}|X_{author})$ (B) $P(Y_{author}|X_{paper})$

FIGURE 4.2: Plots illustrating the inter-dependencies between labels on the DBLP corpus. A gray square means that there is no relation between the corresponding node types



(A) $P(Y_{user}|X_{photo})$ (B) $P(Y_{photo}|X_{user})$ (C) $P(Y_{user}|X_{user})$

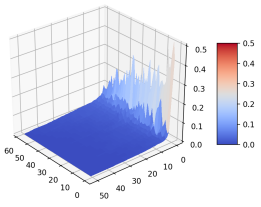
FIGURE 4.3: Plots illustrating the inter-dependencies between labels on the FlickR corpus

Further insights into this dependency are provided by Figure 4.5. It shows the cumulative values of $P(Y_{t_1}|X_{t_2})$ for different relations (Y_{t_1}, X_{t_2}) w.r.t. the percentage of the number of couples considered on the LastFM2 dataset. We start with the highest $P(Y_{t_1}|X_{t_2})$ for any relation, the value of which is plotted at $x = 0\%$ and we accumulate the successive values $P(Y_{t_1}|X_{t_2})$ taken in decreasing order.

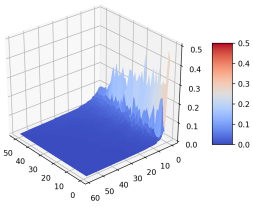
Figure 4.5 shows first that all the relations between the pairs of variables exhibit some dependency: at $x = 20\%$, one already has a cumulated value of the $P(Y_{t_1}|X_{t_2})$ between 58% and 87% regardless of the relation type. Second, some relations are clearly more informative than others, e.g. for $x = 10\%$ the relation *user* $\rightarrow$ *user* has cumulated 38% of conditional

Y \ X	user	photo
user	$4.7 \cdot 10^{-4}$	$8.9 \cdot 10^{-4}$
photo	$8.1 \cdot 10^{-4}$	

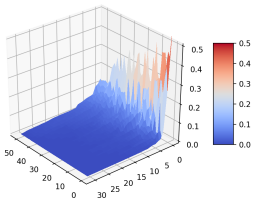
TABLE 4.5: Conditional entropies $H(Y|X)$ for the FlickR corpus.



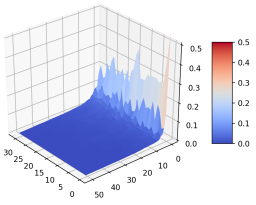
(A) $P(Y_{album}|X_{artist})$



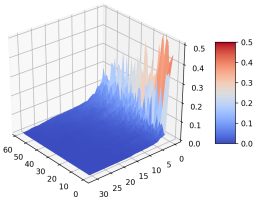
(B) $P(Y_{artist}|X_{album})$



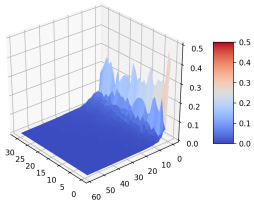
(C) $P(Y_{track}|X_{album})$



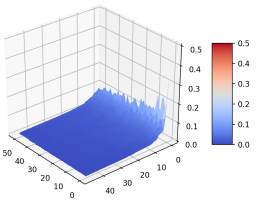
(D) $P(Y_{album}|X_{track})$



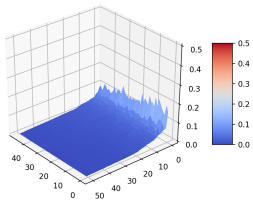
(E) $P(Y_{track}|X_{artist})$



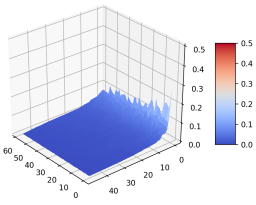
(F) $P(Y_{artist}|X_{track})$



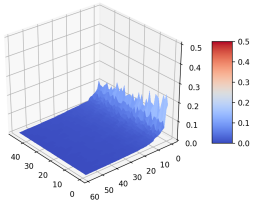
(G) $P(Y_{user}|X_{album})$



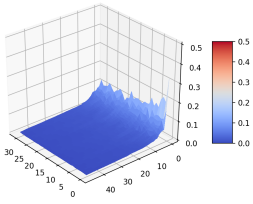
(H) $P(Y_{album}|X_{user})$



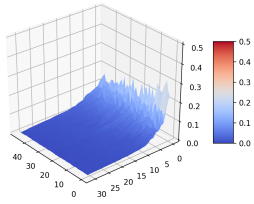
(I) $P(Y_{user}|X_{artist})$



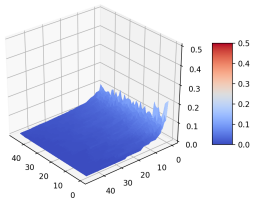
(J) $P(Y_{artist}|X_{user})$



(K) $P(Y_{user}|X_{track})$



(L) $P(Y_{track}|X_{user})$



(M) $P(Y_{user}|X_{user})$

FIGURE 4.4: Plots illustrating the inter-dependencies between labels on the LastFM2 corpus

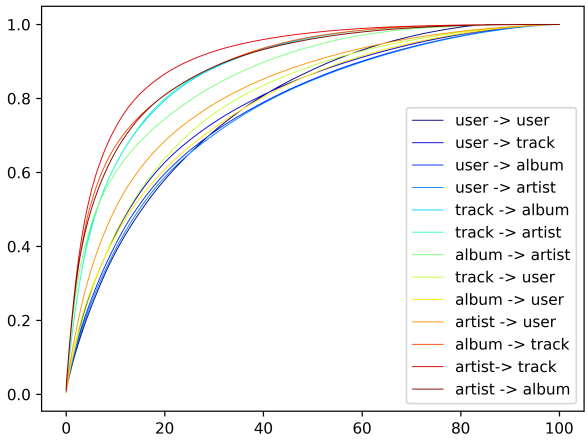


FIGURE 4.5: Plot of the average of the cumulative sum of conditional probabilities $P(Y_{t_1}|X_{t_2})$ for all relation types on the LastFM2 dataset. The $P(Y_{t_1}|X_{t_2})s$ are in decreasing order on the x axis. $x = 40\%$ means that 40% of the conditional probability values have been considered and the corresponding cumulative value is plotted on the y axis.

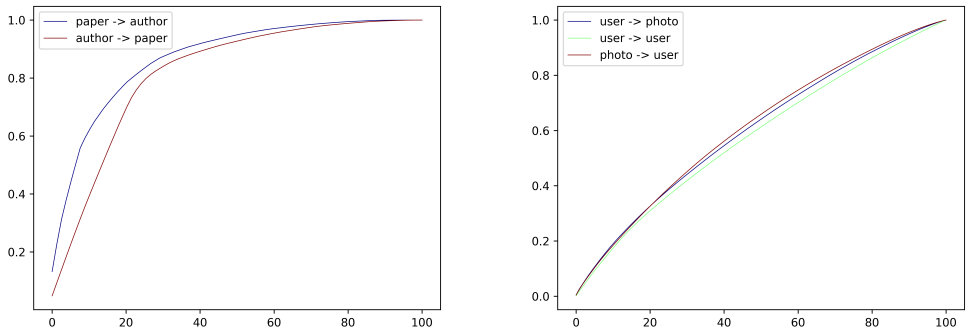
probability whereas the relation *artist* $\rightarrow$ *track* has cumulated 72%. Figures exhibiting similar behaviors for DBLP and FlickR are shown in Figure 4.6.

These statistics clearly show that there exists strong interdependencies between labels of different node types, which could be exploited by an appropriate model, and this confirms our base hypothesis.

Comparison With Other Models

We compare our approach with three baselines described below. The first one, LINE (Tang et al., 2015) is representative of unsupervised learning methods for graph embeddings. Based on (Mikolov et al., 2013), LINE learns distributed representations by maximizing the probability of a node knowing its neighbors. We then performed a logistic regression with the learned representations as inputs. The second one, Homogeneous Label Propagation (HLP) (Zhu et al., 2002), provides a comparison with a representative model of label propagation, typical of semi-supervised transductive learning on homogeneous graphs. There are two main ways to construct an homogeneous graph from an heterogeneous one, each considers a separate classification problem for any node type:

- For each node type, use the whole graph for propagating the labels of this given type: only the nodes from this type will be considered for classification, and all the nodes are considered in the propagation loss. Then repeat the propagation for all types of nodes.
- Construct a projection of the heterogeneous graph for each type of nodes, thus building an homogeneous graph, and then propagate the labels on each homogeneous graphs.



(A) DBLP dataset (B) FlickR dataset

FIGURE 4.6: Plots of the average of the cumulative sum of conditional probabilities $P(Y_{t_1}|X_{t_2})$ (with cumulative percentages on the x-axis) for all relation types on all datasets.

The second option (projection) requires defining projections of the heterogeneous graph onto homogeneous ones. There are many possible choices for that and the results heavily depend on this choice. We hence rather considered the first option. The third baseline, Graffiti, is a state of the art model for the task of classification in heterogeneous graphs (Angelova et al., 2012). It is a modified random walk that captures the influence among nodes of the same type by considering their common neighbors. For example, given a node, it can jump to another node of the same type by either following a direct link or using a 2-hops jump via a common neighbor of another type. Graffiti has been shown to be superior to the following four models on different datasets (FlickR and LibraryThing): a naive Bayesian classifier, TopicalPR (Nie et al., 2006), an hybrid classifier (Myaeng et al., 2000), and an iterative relaxation labeling classifier (Angelova et al., 2006). Although it allows to develop random walks on heterogeneous graphs, it does not take advantage of the correlations between labels as our model does.

Evaluation Measures And Protocol

Evaluation measures We have considered two different evaluation measures: **Precision at 1 (P@1)** measures the percentage of nodes for which the category with the highest predicted score is among the observed labels for this node. For monolabel classification, this should be the target label, while for multilabel classification, this could be any of the target labels. **Precision at k (P@k)** is the proportion of correct labels in the set of k labels with the highest predicted scores. For the mono-label dataset DBLP, we only make use of **Precision at 1 (P@1)**. For the multi-labels dataset, P@k will denote an average over all the node types, with k set to the number of categories a node belongs to. We optimized the different models with regard to micro-average, but we report both micro-average and macro-average precision P@•.

Hyperparameter	Meaning
N	Dimension of the latent vector space: $[50; 200]$
$\texttt{GS}_z$	Step size of the gradient descent when updating the vector representations: $[10^{-4}; 10^{-3}]$
$\texttt{ITER}_z$	Number of iterations of the algorithm: $[100; 1000]$
V	Variance for the initialization of z representations: $[10^{-4}; 10^{-3}]$
λ	Trade-off between classification and smoothness: $[10; 100]$
$\texttt{GS}_w$	Step size of the gradient descent when updating the w_r: $[10^{-1}; 1]$
$\texttt{ITER}_w$	Number of iterations when to stop learning of w_r: $[10; 100]$

TABLE 4.6: Hyperparameters of the model

Hyperparameters **LINE** has four hyperparameters: the mini-batch size, the learning rate, the dimensionality of the unsupervised learned representation and the number of negative samples. **HLP** has one hyperparameter: the number of times the model is iterated. **Graffiti** has four hyperparameters: three characterize the random walk and one the number of iterations. Finally, our model has six hyperparameters described in Table 4.6. The hyperparameters for all the models were optimized by grid search on a validation set.

Protocol Since we are in a transductive setting, all the nodes are used for training – but not all the labels. We partition the labeled nodes into three datasets: train, validation and test. Model parameters are trained as described in Algorithm 1[5]. Model selection is performed on the validation set using a **Micro P@k** measure, which corresponds to the mean of P@k over all nodes. Performance is then evaluated on the test set.

Experiments were performed with different training + validation set sizes: 10%, 30%, 50% of the total size of the labeled dataset. For a training + validation size of 10% of the labeled nodes we used a 50-50 partition (training-validation) of the labeled nodes and a 80-20 partition for the 30% and 50% training + validation sizes. The training nodes are selected at random.

Experiments are performed with 5 random splits. The hyper-parameters are selected for each split using the validation set. We then average 10 runs over each split to account for the random initialization of parameters.

For example, in the LastFM2 dataset, for the user node type, a training + validation set of 10% means that 500 users out of 5,000 labeled users have been considered, 250 of which are used during the parameters training phase and the remaining 250 are used for the validation. Specific to our model, we split the training set of 250 users into two separated training set. The first one (denoted $\mathcal{L}_c$) is used to learn the representations and the classifiers and corresponds to 90% of this training set. The second one (denoted $\mathcal{L}_w$) is used to learn the hyperparameters w_r and corresponds to 10% of this training set.

Note that for this dataset (LastFM2), only 5,000 users out of 20,004 are labeled. Unlabeled nodes only appear in the graph regularization term of equation (4.1).

[5]We split the training set into two distinct training sets for optimizing respectively $\mathcal{L}_c$ and $\mathcal{L}_w$

Results

In this section we compare our model to the baselines on the five datasets described in Table 4.2.4. For each dataset, we report the averaged P@k measures (micro and macro averages on the node types) and the P@k for each node type. The best performance on the test set is in bold. Results are presented for the different datasets in Tables 4.7, 4.8, 4.9, 4.10 and 4.11.

The main conclusions are :

- Supervised models have better performance than the representative unsupervised LINE, which proceeds in two steps: unsupervised representation learning followed by classifier learning.
- On all datasets **HLP** is below **Graffiti** and **LaHNet** for most settings. This clearly shows that modeling the heterogeneity of the graph is essential.
- **LaHNet** outperforms the baselines on four out of five datasets.

Let us first analyze the performance of the different models. Based on the micro-average P@k measure, our model outperforms all the other models on all the data sets except IMDB. LaHNet outperforms (or is close to for IMDB) Graffiti and HLP with on average over all the settings, +2.4 points for DBLP (see Table 4.7), +3.0 for FlickR (see Table 4.8), +3.6 for LastFM1 (see Table 4.9), +1.9 for LastFM2 (see Table 4.10) and -0.8 for IMDB (see Table 4.11) compared to the best competitor. The behavior of the models is slightly different according to the datasets. On DBLP, LaHNet is superior to all the other models for all the settings ($10\%, 30\%, 50\%$ of labeled data used for training). For FlickR LaHNet is better at ($10\%, 30\%$) and all the supervised models are similar for 50%. On the LastFM datasets, LaHNet is better at 30% and 50% while all the supervised models are similar at 10%. On IMDB, Graffiti and LaHNet have similar performance with a slight advantage to Graffiti. The IMDB data set is specific since only movies have labels, hence LaHNet has no advantage over homogeneous models. Graffiti shows a slight advantage over HLP since it explicitly models the 2 hop relations occurring in this dataset.

Table 4.12 shows the performance evolution at different training and validation set sizes (TVS).

We can distinguish two groups: the three datasets DBLP, LastFM1 and LastFM2 and the two datasets FlickR and IMDB. For the first group, the performance increase for LaHNet is higher than for Graffiti even if LaHNet already outperforms Graffiti. This shows that in this case, for low training set size, LaHNet takes greater benefits of additional training data compared to Graffiti. The reverse holds for the second group: for IMDB the increases are comparable and LaHNet remains slightly below Graffiti, for FlickR, LaHNet outperforms HLP for all training set sizes, but the gap between HLP and our model decreases when the training data increases.

While the models are optimized for micro-average P@k, macro-average P@k gives a complementary indication (see Tables 4.7, 4.8, 4.9, 4.10 and 4.11). Macro precision is below micro precision for all the models, but globally the models rank similarly for both measures. Macro precision shows that under-represented types, e.g. users on the LastFM datasets, have low performance. Note that LaHNet is below Graffiti and HLP for these node types, probably because the number of parameters per class is higher for LaHNet.

Train + Val. size	Model	Train **Micro**	**Val** **Micro**	Test **Micro**	Test Macro	Test Author	Test Paper
10%	LINE	25.1	18.9	19.5	23.0	29.1	16.8
	HLP	100	24.7	24.1	27.2	32.6	21.8
	Graffiti	100	32.4	30.9	38.1	50.8	25.3
	LaHNet	99.8	33.8	**32.1**	**40.0**	**53.9**	**26.0**
30%	LINE	24.0	21.5	21.9	24.8	30.1	19.5
	HLP	100	35.8	36.0	41.9	52.4	31.4
	Graffiti	100	39.6	38.5	46.6	61.1	**32.1**
	LaHNet	99.7	43.0	**41.2**	**52.9**	**73.8**	31.9
50%	LINE	24.2	21.1	22.3	25.0	29.8	20.2
	HLP	100	39.7	39.4	46.5	59.3	33.7
	Graffiti	100	41.5	41.2	49.4	64.1	**34.8**
	LaHNet	99.9	45.5	**44.4**	**56.8**	**79.2**	34.5

TABLE 4.7: P@1 results of the model LaHNet and baselines on DBLP

Train + Val. size	Model	Train **Micro**	**Val** **Micro**	Test **Micro**	Test Macro	Test User	Test Photo
10%	LINE	24.4	19.4	20.7	23.2	29.1	17.3
	HLP	100	26.0	26.3	27.8	31.3	24.3
	Graffiti	100	24.3	24.5	27.0	**32.7**	21.2
	LaHNet	99.3	31.8	**31.9**	**31.9**	31.8	**32.0**
30%	LINE	23.0	21.6	21.5	24.2	30.6	17.9
	HLP	100	47.6	47.7	43.7	34.5	53.0
	Graffiti	100	47.5	47.0	43.7	**36.1**	51.3
	LaHNet	100	50.1	**49.0**	**44.3**	33.3	**55.3**
50%	LINE	23.2	21.8	21.8	24.6	31.0	18.2
	HLP	100	54.2	54.1	48.6	35.8	61.4
	Graffiti	100	54.4	54.0	**48.8**	**36.9**	60.8
	LaHNet	99.9	55.8	**54.3**	48.2	33.9	**62.4**

TABLE 4.8: P@k results of the model LaHNet and baselines on FlickR

Train + Val. size	Model	Train **Micro**	**Val** **Micro**	Test **Micro**	Macro	User	Track	Album	Artist
10%	LINE	20.8	20.6	20.4	15.9	5.6	26.0	14.5	17.4
	HLP	98.7	38.1	38.4	30.0	9.9	47.8	27.2	35.1
	Graffiti	100	40.1	**40.0**	**31.4**	**10.6**	49.0	28.1	**38.1**
	LaHNet	100	39.5	39.0	29.2	9.5	**50.7**	**30.8**	25.9
30%	LINE	20.5	20.9	20.5	17.0	10.1	25.9	14.4	17.5
	HLP	98.9	50.2	49.7	40.0	**17.2**	60.5	37.7	44.8
	Graffiti	100	50.8	50.3	40.4	**17.2**	61.7	36.2	46.5
	LaHNet	100	56.6	**55.6**	**43.6**	15.3	**67.1**	**44.7**	**47.3**
50%	LINE	20.5	20.5	20.5	17.0	10.3	26.0	14.4	17.5
	HLP	98.8	51.9	52.1	42.3	**19.4**	63.1	40.2	46.4
	Graffiti	100	53.2	53.5	43.2	19.1	65.4	39.5	48.7
	LaHNet	100	59.2	**59.3**	**46.9**	16.8	**70.4**	**47.8**	**52.7**

TABLE 4.9: P@k results of the model LaHNet and baselines on LastFM1

Train + Val. size	Model	Train **Micro**	**Val** **Micro**	Test **Micro**	Macro	User	Track	Album	Artist
10%	LINE	17.5	17.4	17.5	15.3	13.8	20.1	12.2	15.1
	HLP	98.2	43.5	43.4	35.6	**26.6**	51.7	29.7	34.6
	Graffiti	100	44.4	44.3	**35.8**	25.9	53.1	29.3	**35.0**
	LaHNet	100	44.5	**44.5**	30.9	11.6	**56.7**	**30.9**	24.5
30%	LINE	17.5	17.5	17.5	15.5	14.3	20.2	12.2	15.1
	HLP	98.2	50.0	50.0	41.8	31.5	58.4	37.9	39.3
	Graffiti	100	53.9	54.0	43.7	**31.9**	64.5	38.2	40.2
	LaHNet	99.8	57.3	**56.9**	**45.4**	27.6	**66.8**	**45.0**	**42.1**
50%	LINE	17.5	17.6	17.5	15.4	14.2	20.2	12.2	15.1
	HLP	98.2	51.7	51.9	43.6	33.2	60.4	39.9	41.0
	Graffiti	100	56.8	57.0	46.5	**34.4**	67.7	41.8	42.4
	LaHNet	100	59.6	**59.5**	**47.8**	29.0	**69.3**	**47.7**	**45.1**

TABLE 4.10: P@k results of the model LaHNet and baselines on LastFM2

Train + Val. size	Model	Train	**Val**	Test
10%	LINE	38.7	38.6	38.6
	HLP	100	39.6	39.7
	Graffiti	100	44.9	**44.9**
	LaHNet	94.1	44.9	44.7
30%	LINE	38.6	38.5	38.6
	HLP	100	47.7	47.7
	Graffiti	100	49.6	**49.6**
	LaHNet	99.4	49.0	48.8
50%	LINE	38.6	38.6	38.6
	HLP	100	50.4	50.4
	Graffiti	100	51.3	**51.3**
	LaHNet	99.5	50.3	50.1

TABLE 4.11: P@k results of the model LaHNet and baselines on IMDB

Dataset	Model	TVS 10%	TVS 10 to 30%	TVS 30 to 50%
DBLP	LaHNet	32.1	+9.1	+3.2
	Graf.	30.9	+7.6	+2.7
LastFM1	LaHNet	39.0	+16.6	+3.7
	Graf.	40.0	+10.3	+3.2
LastFM2	LaHNet	55.6	+12.4	+2.5
	Graf.	50.3	+9.7	+3.0
FlickR	LaHNet	31.9	+17.1	+5.3
	HLP	26.3	+22.5	+7.0
IMDB	LaHNet	44.7	+4.1	+1.3
	Graf.	44.9	+4.7	+1.7

TABLE 4.12: Comparison of the evolution of P@k performance between LaHNet and the best baseline when increasing the training + validation size (TVS).

Train size	Model	Train	**Val**	Test			
		Micro		**Micro**	Macro	User	Photo
10%	Without	99.6	29.6	29.3	29.1	28.6	29.5
	With	99.3	31.8	**31.9**	**31.9**	**31.8**	**32.0**
30%	Without	99.9	49.2	48.4	43.6	32.5	54.7
	With	100	50.1	**49.0**	**44.3**	**33.3**	**55.3**
50%	Without	99.4	55.1	54.0	47.9	33.7	62.0
	With	99.9	55.8	**54.3**	**48.2**	**33.9**	**62.3**

TABLE 4.13: Effect of learning the relation-specific weights w_r for FlickR. Performance is expressed as micro and macro precisions as indicated, performance for individual relations is micro precision - see text for further explanation.

Importance Of The Relations' Weights

Let us now analyze the role of the relation-specific weights w_r in the model. When there is only one type of relation like DBLP, there is nothing to learn. For the other datasets, learning w_r significantly improves the performance. We discuss below each dataset in turn and we denote $w_{A\to B}$ the weight of the relation from node type A to node type B.

FlickR After convergence the weights are almost equals on average: $w_{user\to photo} = 0.53$ and $w_{user\to user} = 0.47$ for the two relation types involving users. Actually, one can expect close weights as users follow users who publish photos they like, which might be on average the same type as their own photos. The results of the two models are reported in Table 4.13. Even with such a small average difference between the two types of weights, learning w_r improves the results. It has a significant impact at small training set size (10%) with an increase of +2.6 compared to a model with no weights and the difference tends to vanish when the training set is increased.

LastFM The LastFM datasets are the more interesting: there are 4 types of nodes with several relations, the number of labels is larger than for the other datasets. Let us focus on the relations involving users since this is the richer and most diverse group of relations with 4 relation types.

Learning the w_r has an important impact on the user-nodes performance. For LastFM1 (see Table 4.14), there is an improvement of +0.5 (train + validation size 10%), +5.6 (train + validation size 30%) and +5.8 (train + validation size 50%) when learning the w_r. The same holds for LastFM2 (see Table 4.15), where an improvement of +2.2 (train + validation size 10%), +6.3 (train + validation size 30%) and +6.2 (train + validation size 50%) can be observed. Concerning user scores, the more training data, the bigger the improvement.

The role of w_r for the other relations is less important, however, it still makes a difference. On the LastFM1 data set there is a global improvement of +2.5 on average for all the train + validation sizes. The same holds for the LastFM2 dataset, with an improvement of +3.5/+3.6 for train + validation sizes 30% and 50%, and a greater improvement for a train size of 10% (+6.5).

Train size	Model	Train	**Val**	Test					
		Micro		**Micro**	Macro	User	Track	Album	Artist
10%	Without	99.9	36.4	36.3	27.2	9.0	48.4	26.2	25.3
	With	100	39.5	**39.0**	**29.2**	**9.5**	**50.7**	**30.8**	**25.9**
30%	Without	99.8	54.2	53.3	40.3	9.7	65.8	42.7	42.9
	With	100	56.6	**55.6**	**43.6**	**15.3**	**67.1**	**44.7**	**47.3**
50%	Without	99.7	56.6	56.7	43.2	11.0	68.8	45.6	47.6
	With	100	59.2	**59.3**	**46.9**	**16.8**	**70.4**	**47.8**	**52.7**

TABLE 4.14: Effect of learning the relation-specific weights w_r for LastFM1. Performance is expressed as micro and macro precisions as indicated, performance for individual relations is micro precision - see text for further explanation.

Train size	Model	Train	**Val**	Test					
		Micro		**Micro**	Macro	User	Track	Album	Artist
10%	Without	99.8	38.1	38.0	26.1	9.4	48.9	22.1	24.1
	With	100	44.5	**44.5**	**30.9**	**11.6**	**56.7**	**30.9**	**24.5**
30%	Without	99.9	53.7	53.4	40.4	21.3	64.9	40.1	35.3
	With	99.8	57.3	**56.9**	**45.4**	**27.6**	**66.8**	**45.0**	**42.1**
50%	Without	99.8	55.9	55.9	42.8	22.8	67.1	42.9	38.5
	With	100	59.6	**59.5**	**47.8**	**29.0**	**69.3**	**47.7**	**45.1**

TABLE 4.15: Effect of learning the relation-specific weights w_r for LastFM2. Performance is expressed as micro and macro precisions as indicated, performance for individual relations is micro precision - see text for further explanation.

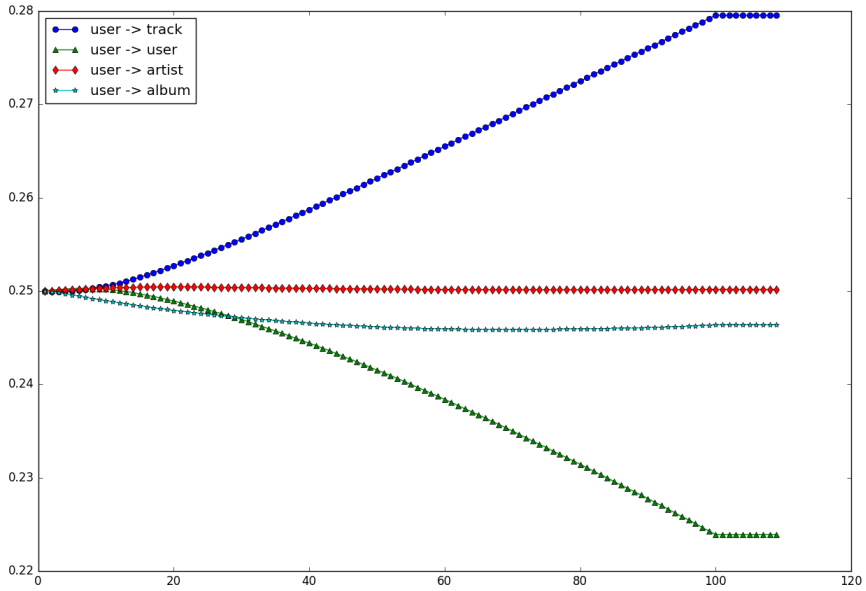


FIGURE 4.7: Evolution of w_r values over training steps for all relations r involving users for LastFM2. At convergence $w_{\text{user}\to\text{user}} = 0.22$, $w_{\text{user}\to\text{track}} = 0.28$, $w_{\text{user}\to\text{album}} = 0.25$, $w_{\text{user}\to\text{artist}} = 0.25$.

If we make the parallel between the learned w_r and the conditional entropy (see Figure 4.1 and Table 4.3) one observes that, for relations of type $user \to \bullet$, the lower the entropy of a user label Y_{user} conditioned on the label of its neighbor $X_\bullet$ is, the higher the coefficient of the corresponding relation type is. The proposed model then captures this dependency. For example, since the value (0.22) of the w_r coefficient for $user \to user$ is rather low, one can conclude that this relation is less relevant than, for example, the relation $user \to track$ (0.28). One can explain this result by the fact that the $user \to user$ relationship is more related to friendship and less to tastes compared to $user \to track/album/artist$. Looking at the music a user listens to will be more informative to infer what kind of music a user likes than looking at what his friends listen to.

The evolution of $w_{user\to\bullet}$ during the learning phase is reported on Figure 4.7. It clearly shows that the proposed procedure is able to differentiate the weights according to their importance for the classification task.

As a conclusion, learning the relation specific weights clearly improves the classification results. Comparing the performances on FlickR and LastFM, one can see that, learning the relation weights w_r allows capturing the most important relation types for the inference task.

Label correlation on the LastFM2 Dataset

As mentioned in Section 4.2.4, labels for different types of nodes are supposed to be distinct even when they have the same name (e.g. the label *pop* is not considered to be the same label for a track and an artist). It is interesting to get some hints on the relations between

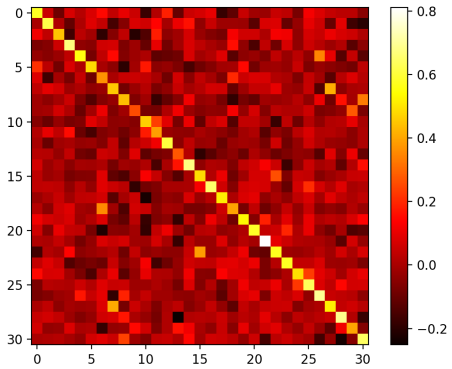


FIGURE 4.8: LastFM2 corpus: Graph of the cosine between the 30 classifiers having the same string-wise label (e.g. "pop") for artists and tracks. For example, line 15 corresponds to *pop_artist* and column 15 to *pop_track*. The lighter the square, the higher the scalar product.

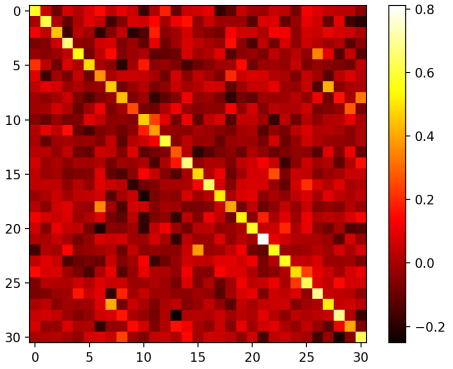
the representations of nodes from different types with the same labels. Since these labels are strongly correlated, intuitively the corresponding nodes should be in the same subspaces of the representation space. Since in our experiments the classifier for each label is linear, measuring the similarity of the two separating hyperplanes gives some indication on this correlation. We have plotted in Figure 4.8 the dot product of the linear classifier weights learned for labels with the same name for two different node types of the LastFM2 corpus. On Figure 4.8, the track label on the $i-th$ column corresponds to its artist counterpart on the $i-th$ row. We first notice that the inner product is non zero in most cases, showing existing correlations between the different classifiers, and further emphasizing the importance of learning a common latent representation space for nodes of different types.

Second, the diagonal values are all close to 1 for this example, so that the model has learned these correlations and the latent representations of tracks labeled *pop* are in the same region of the latent space as artists labeled *pop*, the same goes for the other labels. Other examples for the LastFM2 dataset are provided in Figure 4.9 for different pairs of node types.

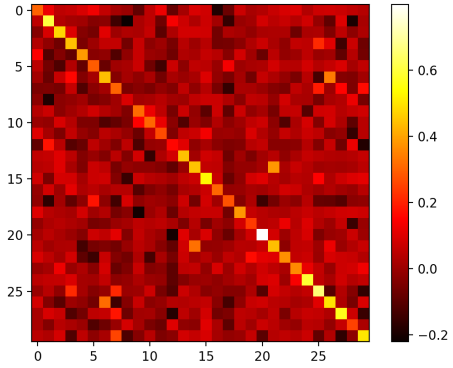
4.3 Conclusion

We have introduced a representation-based model for the challenging task of heterogeneous graph node classification. This model, **Labeling Heterogeneous Networks (LaHNet)**, learns, for each node and label, a deterministic latent representation in a vector space. It also learns relation specific weights that determine how informative the relations between different types of nodes are for the classification task. We introduced a continuous optimization hyperparameter scheme for simultaneously learning the different parameters of the model.

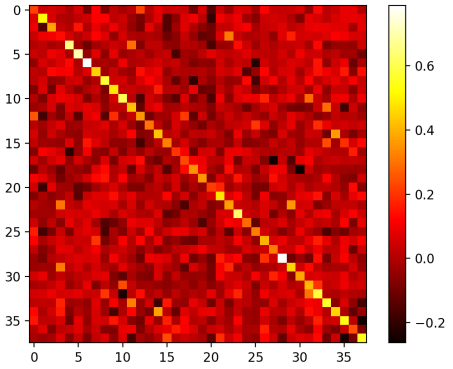
Experimental results show that LaHNet was able to learn correlations between labels of different types of nodes, thus notably improving the class label inference. We performed



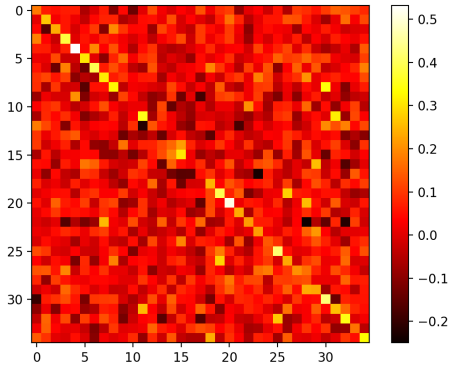
(A) Artists/Tracks labels.



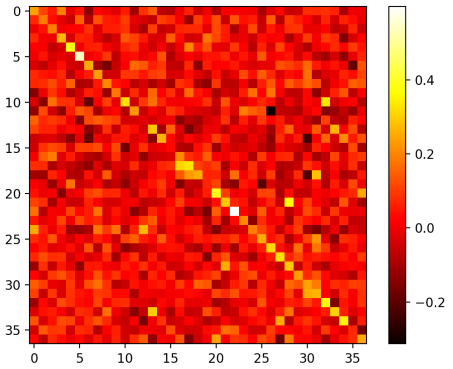
(B) Albums/Tracks labels.



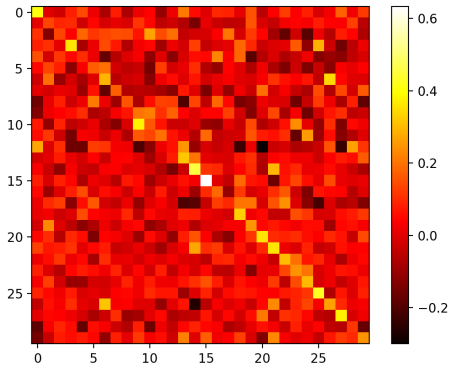
(C) Artists/Albums labels.



(D) Albums/Users labels.



(E) Artists/Users labels.



(F) Users/Tracks labels.

FIGURE 4.9: LastFM2 corpus: Graph of the cosine between the classifiers having the same string-wise label (e.g. "pop") for two different node types.

extensive experiments and showed that LaHNet works well compared to competing methods (Angelova et al., 2006; Zhu et al., 2002; Tang et al., 2015) on five different real word datasets (DBLP, FlickR, LastFM, and IMDB).

Part II

Learning Gaussian Representations and Applications

In the previous part we have used the deterministic representation learning framework to tackle the node classification task. We have extended this work to probabilistic representations and present this contribution in Chapter 5.

Since using probabilistic representations produce good results on node classification, we developed this framework to tackle other tasks. Thus, we present our third and fourth contributions on two different tasks both using the Gaussian representations learning framework. Our third contribution focuses on relational time series forecasting (Chapter 6). Our fourth contribution focuses on the ranking task (Chapter 7). Since the goal of the last two contributions was to explore the interesting properties of this new framework, we do not give a complete state of the art for those tasks, but rather we give an overview of related works in both corresponding chapters.

Like for Chapter 4, these three chapters are reproduction of published or submitted articles. As a consequence, there are some redundancies in the introductions and states of the art.

Chapter 5

Learning Gaussian Representations for Classification

Abstract

In this chapter (as in Chapter 4), we consider the problem of node classification in heterogeneous graphs. The approach is similar to the one used in Chapter 4: we propose a transductive approach to tackle this task based on learning graph embeddings. However, the proposed model, namely **Heterogeneous Classification with Gaussian Embeddings (HCGE)**, instead of learning deterministic representations, learns Gaussian representations to model the uncertainty associated to the node representations. A comparison with representative baselines is provided on three heterogeneous datasets defined in Chapter 4.

Contents

5.1 Introduction 75
5.2 Graph Node Classification with Gaussian Embeddings 76
5.2.1 Model 76
5.2.2 Experiments 80
5.3 Conclusion 84

5.1 Introduction

Classification of nodes in graphs is a relational classification problem where the labels of each node depend on its neighbors. Many problems in domains like image, biology, text or social data labeling can be formulated as graph node classification and this problem has been tackled with different approaches like collective classification (Sen et al., 2008), random walks (Angelova et al., 2012), and transductive regularized models (Jacob et al., 2014). Most approaches consider homogeneous graphs, where all the nodes share the same set of labels, propagating labels from seed nodes to their neighbors. Many problems in domains like biology or social data analysis involve heterogeneous networks where the nodes and the relations between nodes are of different types, each node type being associated to a specific set of labels. For example, the LastFM social network, one of the datasets used in our experiments, links users, tracks, artists and albums via seven different types of relations such as *friendship*, *most listened tracks*, and *authorship*. In such a network, nodes of different types influence each other and their labels are interdependent. The dependency is, however, more complex than with homogeneous networks and depends both on the nodes type

and on their specific relation. Classical methods for homogeneous graphs, based for example on label propagation, usually relies on a simple relational hypothesis like homophily in social networks. They cannot be easily extended to heterogeneous networks, and new methods have to be developed for dealing with this relational classification problem.

In this chapter, we consider the problem of node classification in heterogeneous graphs. We propose a transductive approach based on graph embeddings where the node embeddings are learned so as to reflect both the classification objective for the different types of nodes and the relational structure of the graph. When most embedding techniques consider deterministic embeddings where each node is represented as a point in a representation space, we focus in this chapter on density-based embeddings which capture some form of uncertainty about the learned representations. Uncertainty can have various causes related to the lack of information (isolated nodes in the graph) or because of the contradiction between neighboring nodes (different labels). Our hypothesis is that, because of these different factors, training will result in learned representations with different confidence, and that this uncertainty is important for this classification problem. For that, we use Gaussian embeddings which have been recently proposed for learning word (Vilnis et al., 2015) and knowledge graph (He et al., 2015) embeddings in an unsupervised setting. More precisely, each graph node representation corresponds to a Gaussian distribution where the mean and the variance are learned. The variance term is a measure of uncertainty associated with the node representation. The objective function is composed of two terms, one reflecting the classification task and the other one reflecting the relations between the nodes. Both mono and multi-label classification can be handled by the model. For the experiments, we focus on classification in social network data. This type of data offers a variety of situations which allows us to illustrate the behavior and the performance of the model for different types of heterogeneous classification problems.

To summarize, our contributions are as follows:

- We propose a new method for learning to classify nodes adapted to heterogeneous graph data;
- We model the uncertainty associated with the nodes representation;
- We provide a comparison with state of the art baselines on a series of social data classification problems representative of different situations.

5.2 Graph Node Classification with Gaussian Embeddings

5.2.1 Model

In this section we present our model, namely **Heterogeneous Classification with Gaussian Embeddings (HCGE)**.

We first introduce the notations used throughout this chapter. We also recall the basis graph and classification notations introduced in Chapter 4.

A heterogeneous network is modeled as a directed weighted graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \mathcal{W})$ where $\mathcal{N}$ is the set of nodes, $\mathcal{E}$ the set of edges and $\mathcal{W}$ the weights associated to the edges. Each node $x_i \in \mathcal{N}$ of the graph has a type $t_i \in \mathcal{T}$, where $\mathcal{T} = \{1, 2, \ldots, T\}$. We denote N_i the neighbors of x_i.

Regarding the classification task, let $\mathcal{Y}^t$ denote the set of categories associated with nodes of type t, and $\#\mathcal{Y}^t$ the cardinality of $\mathcal{Y}^t$. $\mathcal{L} \subset \mathcal{N}$ is the set of indexes of labeled nodes. For $i \in \mathcal{L}$, y_i is the class vector associated to x_i: node x_i belongs to category c if $y_i^c = 1$ and does not belong if $y_i^c = -1$.

In our model, each node x_i is mapped onto a representation which is a Gaussian distribution over the space $Z_i \sim \mathcal{N}(\mu_i, \Sigma_i)$ in $\mathbb{R}^d$. The latent space is common to all nodes. In this chapter, we compare two different parameterizations of Σ. We experimented with a spherical ($\Sigma_i = \sigma_i Id$) and a diagonal ($\Sigma_i = diag\left(\sigma_i^p\right)_p$) covariance matrix. We use a weight w_r for each type of relation, $w_{r_{ij}}$ refers to the weight of the type of edge linking node i to node j with a relation r_{ij}.

Loss Function

As in Chapter 4, we learn the representations of nodes and classifiers parameters by minimizing an objective loss function. It takes the general form of transductive regularized loss (Ji et al., 2010; Zhou et al., 2003), with a classification (Δ_C) and a regularization term (Δ_G), both being detailed later:

$$L(Z, \theta) = \sum_{i \in \mathcal{L}} \Delta_C(f_{\theta^{t_i}}(Z_i), y_i) + \lambda \sum_{i \in \mathcal{N}} \sum_{j \in N_i} w_{r_{ij}} \Delta_G(Z_i, Z_j) \tag{5.1}$$

As for classical transductive graph losses, the minimization in Equation 5.1 aims at finding a trade-off between the difference between observed and predicted labels in $\mathcal{Y}^t$, and the amount of information shared between two connected nodes. There are however major differences, since here Z is not a label as in classical formulations, but a node embedding. Finally, the function $f_{\theta^t}(.)$ is a parametric classifier for a node of type t – there is one such classifier for each node type. Since we are using Gaussian embeddings, the Zs are random variables and the regularization term is a dissimilarity measure between distributions.

To avoid over-fitting, following (Vilnis et al., 2015), we regularize the mean and the covariance matrix associated to each node. We add two constraints to prevent means and covariances to be too large and to keep the covariance matrices positive definite (this also prevents degenerate solutions):

$$||\mu_i|| \leq C \text{ and } \forall p,\ m \leq \sigma_i^p \leq M \tag{5.2}$$

where the different parameters C, m and M have been set manually after some trials on a subset of the DBLP training set to respectively 10, 0.01 and 10 (and not changed after that), but any other reasonable value will do.

The two following paragraphs refer to the respective parts of Equation 5.1.

Classifier

The mapping onto the latent space is learned so that the labels of each type of node can be predicted from the (Gaussian) embedding. For that, we use a parametric classification function f_{θ^t} depending on the type t of the node. This multivariate function takes as input a node representation and outputs a vector of scores for each label corresponding to the node

type. The parameters θ^t of the classifier are learned by minimizing the following loss on labeled data:

$$L_{Classification} = \sum_{i \in \mathcal{L}} \Delta_C(f_{\theta^{t_i}}(Z_i), y_i) \tag{5.3}$$

where $\Delta_C(f_{\theta^{t_i}}(Z_i), y_i)$ is the loss associated to predicting labels $f_{\theta^{t_i}}(Z_i)$ given the observed labels y_i. We recall that in this equation $f_{\theta^{t_i}}(Z_i)$ and y_i have values in $\mathbb{R}^{\#\mathcal{Y}^t}$.

In our experiments, we used different losses for Δ_C. We first considered the case where a class decision is simply the expectation of the classifier score together with a hinge loss, adapting the loss proposed in (Jacob et al., 2014). For a given node x of type t with an embedding Z, this gives:

$$\Delta_C(f_{\theta^t}(Z), y) = \Delta_{EV}(f_{\theta^t}(Z), y) \stackrel{\text{def}}{=} \sum_{k=1}^{\#\mathcal{Y}^t} \max\left(0; 1 - y^k \mathbb{E}_Z[f^k_{\theta^t}(Z)]\right) \tag{5.4}$$

where y^k is 1 if x belongs to category k and -1 otherwise, and $f^k_{\theta^t}(Z)$ is a random variable for category k.

Alternatively, the density based formulation allows us to leverage the density-based representation through a probabilistic criterion, even in the case of linear classifiers. We used here for Δ_C the $\log$-probability that $y^k f_{\theta^t}(Z)$ take a positive value. In this case, the variance will be influenced by the two loss terms: if the two terms act in opposite directions, one solution will be to increase variance. As we will see this is confirmed by the experiments.

$$\Delta_C(f_{\theta^t}(Z), y) = \Delta_{Pr}(f_{\theta^t}(Z), y) \stackrel{\text{def}}{=} -\sum_{k=1}^{\#\mathcal{Y}^t} \log \mathbb{P}\left(y^k f^k_{\theta^t}(Z) > 0\right) \tag{5.5}$$

In our experiments and for both costs, we used a linear classifier for $f^k_{\theta^t}$, which allows to easily compute the different costs and gradients, since the random variable $f^k_{\theta^t}(Z)$, being a linear combination of Gaussian variables, is Gaussian too. A basic derivation shows that:

$$\mathbb{P}\left(y^k f^k_{\theta^t}(Z) > 0\right) = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{\mu \cdot \theta^t}{\sqrt{2\sum_p (\theta^t_p \sigma^p)^2}}\right)\right) \tag{5.6}$$

where erf is the Gauss error function.

There are some notable differences between the two classification losses during learning. In the case of a linear classifier f_{θ^t}, $\mathbb{E}_Z[f^k_{\theta^t}(Z)] = \mu \cdot \theta^t_k$. Thus, minimizing Δ_{EV} only updates the mean of the Gaussian embedding: the covariance matrix of the embedding does not interfere with the classification term, and is only present in the second term of Equation 5.1.

For the Δ_{Pr} loss, the probability is proportional to $\operatorname{erf}\left(\frac{\mu \cdot \theta^t}{\sqrt{2\sum_p (\theta^t_p \sigma^p)^2}}\right)$ where the variance is present. When the graph regularization and classification cost pull the representation mean in opposite directions (opposite gradients), the model will respond by increasing

the variance for the spherical variance model[1]: this behavior is interesting since it transforms an opposition between regularization and classification costs into increased uncertainty.

Graph Embedding

We make the hypothesis that two nodes connected in the graph should have similar representations, whatever their type is. Intuitively, this will force nodes of the same type which are close in the graph to be close in the representation space. The strength of this attraction between nodes of the same class will be proportional to their closeness in the graph and to the weight of the path(s) linking them. We use the asymmetric loss proposed in (Vilnis et al., 2015; He et al., 2015):

$$L_{Graph} = \sum_i \sum_{j \in N_i} w_{r_{ij}} D_{KL}(Z_j || Z_i) \tag{5.7}$$

where $\Delta_G(Z_i, Z_j) = D_{KL}(Z_j||Z_i)$ is the Kullback-Leibler divergence between the distributions of Z_i from Z_j:

$$\begin{aligned} D_{KL}(Z_j||Z_i) &= \int_{x \in \mathbb{R}} \mathcal{N}(x; \mu_j, \Sigma_j) \log \frac{\mathcal{N}(x; \mu_j, \Sigma_j)}{\mathcal{N}(x; \mu_i, \Sigma_i)} dx \\ &= \frac{1}{2} \left(\mathrm{tr}(\Sigma_i^{-1}\Sigma_j) + (\mu_i - \mu_j)^T \Sigma_i^{-1}(\mu_i - \mu_j) - d - \log \frac{\det(\Sigma_j)}{\det(\Sigma_i)} \right) \end{aligned} \tag{5.8}$$

The loss L_{Graph} is a sum over the neighbors N_i of i, where $w_{r_{ij}}$ is the weight of the type of edge between x_i and x_j. Other similarity measures between distributions could be used as well, the Kullback-Leibler divergence having the advantage of being asymmetric, which fits well the social network datasets used in the experiments since relations are not necessarily symmetric.

Prior Parameters and Learned Relation Specific Parameters

For ease of reading, and to emphasize the Gaussian representations learning framework, we omitted in Equation 5.1 the prior graph related parameters ϕ_{ij} and ψ_i. In this contribution, we used the same prior parameters as in Chapter 4 (see Section 4.2.2).

Similarly, we did not discuss learning the w_r. The same reasoning as Chapter 4 holds for this model. For instance, when using the Δ_{EV} variant, we need to express μ_i as an explicit function of w by zeroing $\nabla_z L$ on $\mathcal{L}_w$. As in Chapter 4, we obtain closed-form equations for the parameters (mean and covariance in our case). This is the equivalence of Equation 4.6:

$$\mu_i^\star = \left[\sum_r w_r D_i^r \right]^{-1} \cdot \left(\sum_r w_r N_i^r \right) \tag{5.9}$$

[1] the increase will be in the direction of the normal to the classifier hyperplane for the diagonal variance model

with

$$D_i^r = \text{Diag}\left(\sum_{j:i\xrightarrow{r}j} \frac{\phi_{ij}}{\sigma_i^p} + \sum_{j:j\xrightarrow{r}i} \frac{\phi_{ji}}{\sigma_j^p}\right)_{p\in[1..d]}$$

$$N_i^r = \text{Vect}\left(\sum_{j:i\xrightarrow{r}j} \frac{\phi_{ij}\mu_j^p}{\sigma_i^p} + \sum_{j:j\xrightarrow{r}i} \frac{\phi_{ji}\mu_j^p}{\sigma_j^p}\right)_{p\in[1..d]}$$

where $j : i \xrightarrow{r} j$ corresponds to the set of r-neighbors of i.

For the other cases (Δ_{Pr}), the formulas are larger and for the sake of clarity we omit them here.

Algorithm

Learning the Gaussian embeddings $Z \sim \mathcal{N}(\mu, \Sigma)$ and the classifiers parameters θ consists in minimizing loss function in Equation 5.1. We used here a Stochastic Gradient Descent Method to learn the latent representations, i.e. the μ_i, Σ_i as well as the parameters θ of the classifiers.

Our algorithm samples a pair of connected nodes and then makes a gradient update of the nodes parameters. For each sampled node Z_i that is part of the labeled training set $\mathcal{L}$, the algorithm performs an update according to the first term of Equation 5.3. This update consists in successively modifying the parameters of the classification functions θ^{t_i} and of the latent representations μ_i and Σ_i so as to minimize the classification loss term. Then, the model updates its parameters with respect to the smoothness term of Equation 5.7. Note that, while we use a stochastic gradient descent, other methods like mini-batch gradients or batch algorithms could be used as well.

5.2.2 Experiments

Datasets

Experiments have been performed on three datasets respectively extracted from DBLP, FlickR and LastFM. For all but the first dataset (DBLP), each node can have multiple labels. The three datasets are described in Section 4.2.4 and summarized in Table 4.2.4.

We compare our approach with four state-of-the-art models:

- **LINE** (Tang et al., 2015), which is representative of unsupervised learning of graph embeddings suitable for various tasks such as classification. We performed a logistic regression with the learned representations as inputs.
- **HLP** (Zhu et al., 2002), which is representative of transductive graph algorithms developed for semi-supervised learning. As HLP is designed for homogeneous graphs, we perform as many random walks as the number of node types, considering each time that all the nodes are of a same given type.
- **Graffiti** (Angelova et al., 2012), which is a state of the art model for the task of classification with random walk in heterogeneous graph.

Train size	Model	Train **Micro**	Val **Micro**	Test **Micro**	Test Macro	Test Author	Test Paper
10%	LINE	25.1	18.9	19.5	23.0	29.1	16.8
	HLP	100	24.7	24.1	27.2	32.6	21.8
	Graffiti	100	32.4	30.9	38.1	50.8	25.3
	LaHNet	99.8	33.8	**32.1**	**40.0**	**53.9**	**26.0**
	HCGE(Δ_{EV},S)	99.7	33.1	30.9	38.5	52.1	24.9
	HCGE(Δ_{EV},D)	95.6	31.4	30.4	37.4	49.9	24.9
	HCGE(Δ_{Pr},S)	83.8	29.0	27.9	34.3	45.6	22.9
	HCGE(Δ_{Pr},D)	92.9	29.0	28.3	34.3	45.1	23.6
30%	LINE	24.0	21.5	21.9	24.8	30.1	19.5
	HLP	100	35.8	36.0	41.9	52.4	31.4
	Graffiti	100	39.6	38.5	46.6	61.1	32.1
	LaHNet	99.7	43.0	41.2	52.9	73.8	31.9
	HCGE(Δ_{EV},S)	98.5	44.4	**42.3**	52.6	71.0	**34.3**
	HCGE(Δ_{EV},D)	98.8	42.9	41.2	50.8	68.0	33.6
	HCGE(Δ_{Pr},S)	97.5	41.8	41.3	52.1	71.4	32.8
	HCGE(Δ_{Pr},D)	97.4	43.8	**42.3**	**54.1**	**75.0**	33.1
50%	LINE	24.2	21.1	22.3	25.0	29.8	20.2
	HLP	100	39.7	39.4	46.5	59.3	33.7
	Graffiti	100	41.5	41.2	49.4	64.1	34.8
	LaHNet	99.9	45.5	44.4	56.8	**79.2**	34.5
	HCGE(Δ_{EV},S)	99.3	45.6	44.6	55.2	74.1	36.3
	HCGE(Δ_{EV},D)	98.1	44.7	43.9	53.7	71.0	36.3
	HCGE(Δ_{Pr},S)	99.4	45.8	45.5	57.1	77.8	**36.4**
	HCGE(Δ_{Pr},D)	97.6	45.9	**45.7**	**57.7**	**79.2**	36.2

TABLE 5.1: P@1 results of the model HCGE and baselines on DBLP

- **LaHNet** (Chapter 4), which is our previous contribution, a state of the art model for the task of classification with deterministic vector representations in heterogeneous graph. This is the deterministic version of this contribution.

For a state of the art of the graph node classification task please refer to Chapter 3.

Concerning the evaluation measures and protocol, we use and follow the same measures and protocol as in Chapter 4 (see Section 4.2.4 for details).

Results

In this section we present the results of four variants of our Gaussian embedding model, and compare to LINE (Tang et al., 2015), Graffiti (Angelova et al., 2012), HLP (Zhu et al., 2002) and LaHNet (Chapter 4). The experiments are performed on the three datasets described in Table 4.2.4 and the results are described in Tables 5.1 (DBLP), 5.2 (FlickR) and 5.3 (LastFM). The best performing classifier (on the test set) is presented in bold.

Concerning the four variants of our model, HCGE($\Delta_\bullet$,X) refers to the HCGE model with the classification loss $\Delta_\bullet$ (Δ_{EV} or Δ_{Pr}) and a spherical (X=S) or diagonal (X=D) covariance matrix.

For micro P@k, our model generally outperforms the others on all the datasets. Supervised models (HLP, Graffiti, LaHNet and HCGE) using the class information outperform unsupervised representation learning, which matches the results reported in Chapter 4 and in (Jacob et al., 2014). On all datasets, the performances of HLP are below the performances of Graffiti, LaHNet and HCGE. This clearly shows that modeling the heterogeneity of the

Train size	Model	Train **Micro**	**Val** **Micro**	Test **Micro**	Macro	User	Photo
10%	LINE	24.4	19.4	20.7	23.2	29.1	17.3
	HLP	100	26.0	26.3	27.8	31.3	24.3
	Graffiti	100	24.3	24.5	27.0	32.7	21.2
	LaHNet	99.3	29.6	29.3	29.1	28.6	29.5
	HCGE(Δ_{EV},S)	98.9	33.5	**32.7**	**32.6**	32.4	**32.8**
	HCGE(Δ_{EV},D)	99.1	33.4	32.6	**32.6**	32.7	32.5
	HCGE(Δ_{Pr},S)	96.0	30.4	29.7	29.2	28.1	30.3
	HCGE(Δ_{Pr},D)	98.7	31.7	31.9	32.2	**33.0**	31.5
30%	LINE	23.0	21.6	21.5	24.2	30.6	17.9
	HLP	100	47.6	47.7	43.7	34.5	53.0
	Graffiti	100	47.5	47.0	43.7	**36.1**	51.3
	LaHNet	100	49.2	48.4	43.6	32.5	54.7
	HCGE(Δ_{EV},S)	99.1	51.5	50.0	45.6	35.4	55.8
	HCGE(Δ_{EV},D)	98.7	51.6	**50.1**	45.7	35.3	**56.0**
	HCGE(Δ_{Pr},S)	98.3	50.1	49.0	44.4	33.8	55.1
	HCGE(Δ_{Pr},D)	98.5	50.6	50.0	**45.8**	**36.1**	55.5
50%	LINE	23.2	21.8	21.8	24.6	31.0	18.2
	HLP	100	54.2	54.1	48.6	35.8	61.4
	Graffiti	100	54.4	54.0	48.8	36.9	60.8
	LaHNet	99.9	55.1	54.0	47.9	33.7	62.0
	HCGE(Δ_{EV},S)	97.9	56.7	55.8	50.0	36.5	**63.4**
	HCGE(Δ_{EV},D)	97.3	56.6	55.8	50.0	36.5	**63.4**
	HCGE(Δ_{Pr},S)	98.8	55.7	54.8	49.0	35.5	62.5
	HCGE(Δ_{Pr},D)	98.4	56.4	**55.9**	**50.3**	**37.2**	63.3

TABLE 5.2: P@k results of the model HCGE and baselines on FlickR

graph brings noteworthy improvements. Comparing the heterogeneous models, both LaHNet and HCGE outperform Graffiti on all datasets. On average, comparing to Graffiti, LaHNet is 2.4 better on DBLP, 2.1 better on FlickR and 2.5 better on LastFM. We observed the same behavior for HCGE, with +2.8 on DBLP, +4.4 on FlickR and +6.0 on LastFM. We can note that the more complex the dataset, the higher the gap compared to the baselines. This also shows that the use of representations can clearly improve the performances.

On each dataset, our model outperforms LaHNet (and the other competitors) 8 times over 9, with on average +1.0 points for DBLP, +2.3 for FlickR, and +3.8 for LastFM over the second ranked model. According to the results, introducing uncertainty in representations clearly improves results when comparing to LaHNet.

Let us also point out that, according to our initial intuition, the effect of using uncertainty has more impact when the amount of training data is lower: the difference between LaHNet and HCGE decreases in general when more training data is available (except for DBLP).

Let us compare the performance of the variants Δ_{EV} and Δ_{Pr}. Globally, Δ_{Pr} seems to be disadvantaged by a low number of training examples, when Δ_{EV} seems to be more stable in comparison to other baselines. However, the more training data, the closer the Δ_{Pr} variant is to Δ_{EV}. For example, on the DBLP dataset, moving from 10% to 30% improves on average Δ_{Pr} results by +13.7 but only by +11.1 for Δ_{EV}. For a training set size of 50%, the difference between Δ_{Pr} and Δ_{EV} is +1.1 on DBLP, and +0.1 on FlickR. For LastFM, the difference is resp. -14.6 for 10%, -6.5 for 30% and -1.5 for 50% of the dataset used for training. On the three datasets, the lower the training set size, the better Δ_{EV} seems to be comparing to Δ_{Pr}. We could not explain this difference in the behavior between Δ_{EV} and Δ_{Pr}, but believe that this is due to the fact that the covariance matrix is only optimized

Train size	Model	Train **Micro**	**Val** **Micro**	Test **Micro**	Macro	User	Track	Album	Artist
10%	LINE	20.8	20.6	20.4	15.9	5.6	26.0	14.5	17.4
	HLP	98.7	38.1	38.4	30.0	9.9	47.8	27.2	35.1
	Graffiti	100	40.1	40.0	31.4	**10.6**	49.0	28.1	38.1
	LaHNet	99.9	36.4	36.3	27.2	9.0	48.4	26.2	25.3
	HCGE(Δ_{EV},S)	99.8	44.4	**44.0**	**34.1**	9.6	**52.3**	**35.0**	**39.7**
	HCGE(Δ_{EV},D)	99.3	44.0	43.6	34.0	10.5	52.2	34.4	38.7
	HCGE(Δ_{Pr},S)	97.6	27.7	27.8	20.7	4.1	34.9	21.0	23.0
	HCGE(Δ_{Pr},D)	96.0	30.3	29.4	21.9	6.7	38.7	22.1	20.2
30%	LINE	20.5	20.9	20.5	17.0	10.1	25.9	14.4	17.5
	HLP	98.9	50.2	49.7	40.0	**17.2**	60.5	37.7	44.8
	Graffiti	100	50.8	50.3	40.4	**17.2**	61.7	36.2	46.5
	LaHNet	99.8	54.2	53.3	40.3	9.7	65.8	42.7	42.9
	HCGE(Δ_{EV},S)	99.6	58.2	**57.3**	45.0	14.8	**68.2**	**45.9**	51.2
	HCGE(Δ_{EV},D)	99.5	57.9	57.0	**45.3**	16.8	67.5	45.7	**51.3**
	HCGE(Δ_{Pr},S)	97.5	50.5	50.4	37.7	9.9	66.4	32.6	42.0
	HCGE(Δ_{Pr},D)	96.9	51.5	50.8	38.5	13.2	65.0	41.4	34.4
50%	LINE	20.5	20.5	20.5	17.0	10.3	26.0	14.4	17.5
	HLP	98.8	51.9	52.1	42.3	19.4	63.1	40.2	46.4
	Graffiti	100	53.2	53.5	43.2	19.1	65.4	39.5	48.7
	LaHNet	99.7	56.6	56.7	43.2	11.0	68.8	45.6	47.6
	HCGE(Δ_{EV},S)	99.4	60.3	**60.4**	**48.7**	**20.4**	**71.2**	**48.8**	**54.4**
	HCGE(Δ_{EV},D)	99.9	60.1	60.3	48.6	20.1	71.1	48.7	54.3
	HCGE(Δ_{Pr},S)	99.2	58.6	58.5	45.0	11.8	69.8	47.4	51.0
	HCGE(Δ_{Pr},D)	99.9	58.9	58.9	47.2	18.9	70.2	46.4	53.4

TABLE 5.3: P@k results of the model HCGE and baselines on LastFM

in the graph regularization term in the case of Δ_{EV}. Let us now compare the use of a spherical and a diagonal covariance matrix. For the Δ_{EV} variant, it looks like moving from a spherical covariance matrix to a diagonal one brings no improvement. It even decreases the performance on DBLP. Concerning the Δ_{Pr} variant, for which the covariance matrix plays a role in the classification cost, conclusions are reversed and using diagonal covariance matrices improves the results. On the FlickR dataset, using a diagonal variance improves the results by 1.4 on average. However, it looks like the more training data, the less the improvement, with +2.2 improvement for a training set size of 10%, +1.0 for 30% and +1.1 for 50%.

Qualitative Discussion

In this section, we focus on studying qualitatively the representations found by HCGE. We consider the most robust variant of our model (Δ_{EV},S), and the most challenging dataset, LastFM (similar observations were made on the other datasets). We will examine the respective role of regularization and classification costs on labeled training nodes, and the relationship between the learned variance of a node and the local node properties (like its number of neighbors).

We first examined the respective role of classification and regularization costs. In Equation 5.1, the max-margin classification cost implies that the gradient of a node x is 0 if $y^k \mathbb{E}_z[f^k_{\theta^t}(z)]$ is above 1. In this case, the only constraints on the node are due to the graph regularization cost. We can see how many of the nodes are used by the classification cost by looking at the number of cases for which $y^k \mathbb{E}_z[f^k_{\theta^t}(z)]$ is below or equal to 1. In Figure 5.1a, we have shown an histogram of $y^k \mathbb{E}_z[f^k_{\theta^t}(z)]$ for labeled nodes in the training set (after

convergence). For around one third of the nodes, the value of the classifier is above 1.1 – they could be removed from the labeled set without harming the solution (however, these could have been useful in early stages of optimization). It means that; similarly to kernel vectors in Support Vector Machine, a few nodes in the graph play the role of "kernel" nodes which preserve the boundaries in the graph.

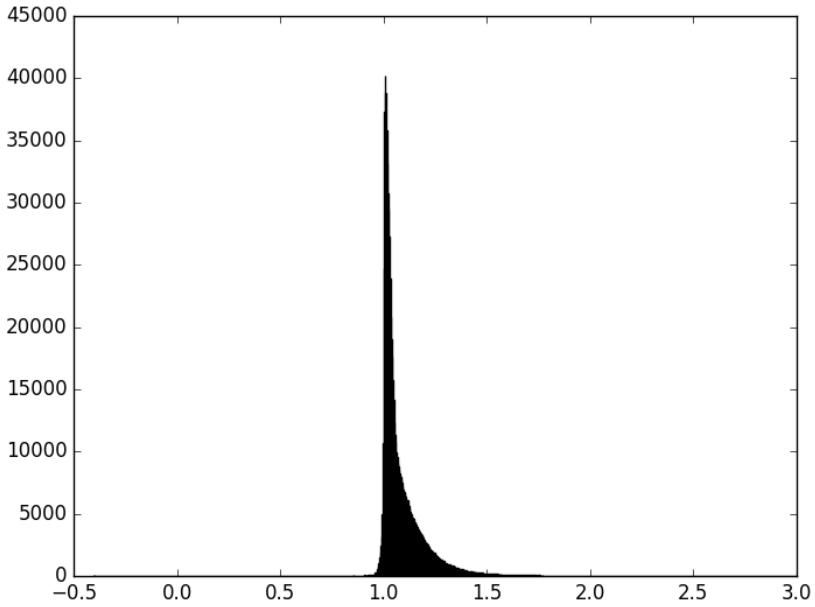
Regarding the relationship between the learned variance and the local properties of each node, we looked at the relationship between the PageRank[2] (Page et al., 1999) of a node and its variance. We expected central nodes to have a certain representation, i.e. a small variance. Figure 5.1b shows that high PageRank implies a small variance, which means that for central nodes, representations are less uncertain. This matches our expectations. However, the reverse implication is not true.

5.3 Conclusion

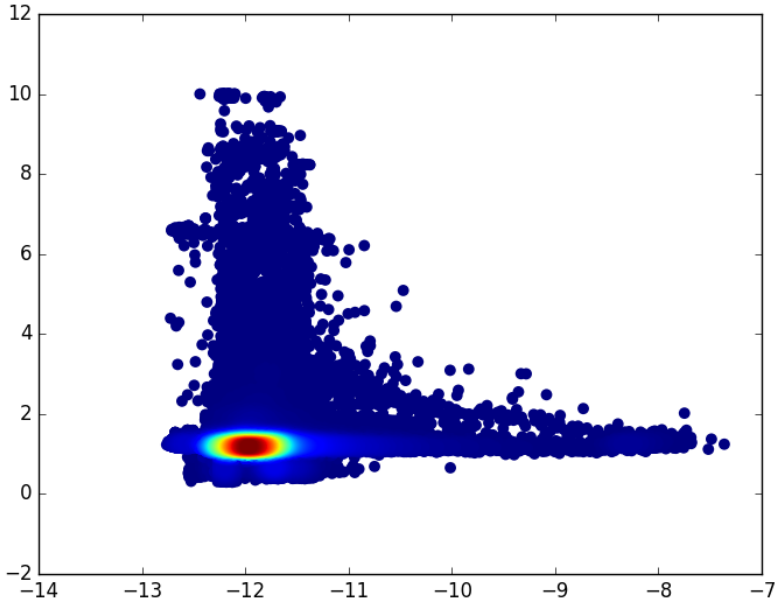
We have explored the use of uncertainty for learning to represent nodes in the challenging task of heterogeneous graph node classification. The proposed model, Heterogeneous Classification with Gaussian Embeddings (HCGE), learns for each node a Gaussian distribution over the representation space, parameterized by its mean and covariance matrix, by optimizing a loss function that includes a classification loss and graph regularization loss. We have examined four variants of this model, by using either spherical and diagonal covariance matrices, and by using two different loss functions for classification. Our model can easily be extended to inductive learning by defining the Gaussian representation z as a parameterized function of the input features.

Based on the experimental results obtained on datasets representative of different situations, our main findings are that (i) integrating uncertainty in representations improved classification (ii) according to our initial intuition, the effect of using uncertainty has generally more impact when the amount of training data is lower and (iii) according to our expectation, highly central nodes seem to have less variance associated to their representation.

[2]Using a standard damping factor of 0.15



(A) Histogram of $y^k \mathbb{E}_z \left[f^k_{\theta^t}(z) \right]$.



(B) σ against the log-PageRank.

FIGURE 5.1: Qualitative results for the model HCGE(Δ_{EV},S) on the LastFM dataset with 50% of the dataset used for train. In Figure 5.1b, we computed Gaussian kernel density to show high density regions in the plot.

Chapter 6

Learning Gaussian Representations for Relational Time Series Forecasting

Abstract

In this chapter, we address the problem of modeling multiple simultaneous time series generated by different sources, where the observations are not only correlated inside each series, but also across the different series. This problem happens in many domains such as ecology, meteorology, etc. We propose a new dynamical state space model, based on representation learning, for modeling the evolution of such series. The joint relational and temporal dynamics of the series are modeled as Gaussian distributions in a latent space. A decoder maps the latent representations to the observations. The two components (dynamic model and decoder) are jointly trained. Using stochastic representations allows us to model the uncertainty inherent to observations and to predict unobserved values together with a confidence in the prediction.

Contents

6.1 Introduction 87
6.1.1 Relational Time Series 87
6.1.2 Contribution 88
6.2 Related Work 89
6.3 Relational Time Series Forecasting with Gaussian Embeddings 91
6.3.1 Model 91
6.3.2 Informal description 91
6.3.3 Experiments 97
6.4 Conclusion 101

6.1 Introduction

6.1.1 Relational Time Series

Relational time series, i.e. multiple time series where the observations are correlated between series occur in many domains such as ecology, medicine, biology, earth observation by satellite imagery or local measurements, multimedia or even social data analysis. The correlations between the different observed series can correspond to a proximity (e.g. earth observation or epidemic diffusion) or to a similarity of behavior (e.g. user traces

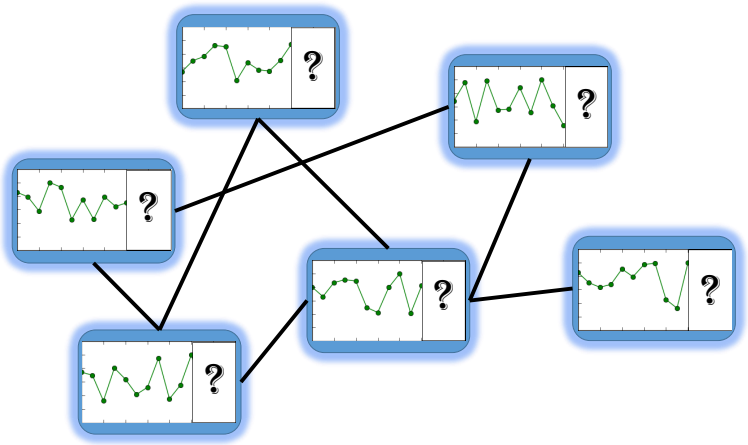


FIGURE 6.1: Representation of relational time series as a a graph, where nodes correspond to series, and edges correspond to a prior relation between two series. In this relational setting the task considered in this chapter is forecasting.

in social data). In the statistical literature, the modeling of relational time series is the topic of a dedicated field: spatio-temporal statistics (Cressie et al., 2011; Wikle et al., 2010). Different methodologies have been developed in the field for handling a large variety of spatio-temporal phenomena, with an emphasis on the analysis of natural observations like weather prediction, ecology or remote sensing. In the machine learning domain, there exists a vast literature dedicated to sequence or time series prediction. Recently, deep recurrent neural networks have witnessed notable successes in different sequence and time series modeling tasks leading to an increasing number of publications, e.g. (Barbounis et al., 2006; Hsieh et al., 2011; Cao et al., 2012; Hermans et al., 2013). Despite a large number of recent developments, the modeling and analysis of relational time series has not been considered, up to our knowledge, in the field of representation learning. In addition, most of the models are deterministic in the sense that they are trained to predict future values from the past of the series.

6.1.2 Contribution

We propose a new state space model for relational time series forecasting, able to model the uncertainty both at the observation and at the modeling levels. This approach associates each point of a time series with a Gaussian distribution in a latent space, the distribution over the observed values being directly computed from these latent distributions. The model has two main components. One is responsible for the dynamics in the latent space and is trained to model the evolution of the Gaussian distribution considering both the temporal intra-series and the relational inter-series dependencies. A second component acts as a decoder and maps the latent representations associated with each series to the corresponding observations in the output space. Throughout the chapter, we will consider that the inter-series relations are provided as prior knowledge, since we are mainly interested into series coming from physical sources for which this evidence is generally available. Note that this is not restrictive and the proposed model can be extended to learn the relations

too. Although the focus in the paper is on forecasting, the model can be straightforwardly extended to solve the imputation problem that aims at predicting missing values.

Our model is built on top of the model in (Ziat et al., 2016) which proposes a deterministic dynamical process model but does not consider any explicit modeling of uncertainty. The new model makes use of Gaussian embeddings instead of simple vectors, and extends the dynamics and loss functions of the model in (Ziat et al., 2016).

The contributions presented in this chapter are thus:

- A new dynamical model for relational time series inspired by representation learning;
- A stochastic component for modeling the uncertainties at the observation and dynamic levels;
- Predictions provided by the proposed approach correspond to distributions over possible outputs.

This chapter is organized as follows. In Section 6.2, we introduce some related work on forecasting in time series, representation learning for time series, and recent deep learning works focusing on modeling uncertainty. We do not provide a complete state of the art of the task since the purpose of this contribution was to see how good Gaussian representations can deal with this task. The model proposed for this contribution is presented in Section 6.3.1 together with four different variants. Section 6.3.3 presents experimental results on four datasets, and Section 6.4 concludes this work and gives some perspectives.

6.2 Related Work

The classical topic of time series modeling and forecasting has given rise to an extensive literature. In statistics, classical linear models include many variations around auto-regressive and moving average models (De Gooijer et al., 2006). In machine learning, non linear extensions of these models based on neural networks have been proposed as early as the 90s, opening the way to many other non linear models including kernel methods (Muller et al., 1999).

Relational time series have mainly been studied in the field of spatio-temporal statistics (Cressie et al., 2011; Wikle et al., 2010). The traditional method first relied on a descriptive approach using the first and second-order moments of the process for modeling the spatio-temporal dependencies. More recently, dynamical state models, where the current state is conditioned on the past, have been explored (Wikle, 2015). These models have been considered both for continuous/discrete space and time components. However, the most common way is to consider discrete time, leading to the modeling of time series of spatial processes as we do here. When space is discrete, the model comes down to a general vectorial autoregressive formulation. These models face a curse of dimensionality in the case of a large number of sources. Different strategies have been adopted to solve this problem such as embedding the spatio-temporal process in a low-dimensional manifold or parameter reduction (Wikle, 2015), leading to model families quite similar to the ones used in machine learning for modeling dynamical phenomena. Also, for complex underlying processes, observations only provide an incomplete description of the process dynamics so that modeling uncertainty at the data and model levels is an important topic.

In the last 10 years, there has been a growing interest in learning latent representations for example through neural networks and deep learning. Dynamical state space models such as recurrent neural networks (RNN), which have been used for time series forecasting in different contexts since the early nineties (Connor et al., 1994), have recently witnessed important successes in different areas for general sequence modeling problems, leading to breakthroughs in domains like speech (Graves et al., 2013), language generation (Sutskever et al., 2011), translation (Cho et al., 2014), and many others. Among this family, the model closest to ours is the dynamic factor graph model of (Mirowski et al., 2009) designed for multiple series modeling for the tasks of forecasting and imputation. However this model does not consider relational dependencies which is the focus of our approach.

Most of the above models make use of pointwise representations and do not model explicitly the uncertainties present in the process and/or in the observations. Recently, in the learning representation community, there has been a growing interest in using distributions as latent representations instead of points. (Vilnis et al., 2015; He et al., 2015) and our previous contribution, all make use of Gaussian distributions for representing different items like words (Vilnis et al., 2015), nodes in knowledge graphs (He et al., 2015) or nodes in graphs for transductive classification (Chapter 5). Note that Gaussian processes have also been used for time series prediction, but they have mainly been considered for univariate time series prediction (Hachino et al., 2011; Brahim-Belhouari et al., 2004) and they do not use a state space formulation.

Recent techniques that combine variational inference and stochastic back propagation (Kingma et al., 2014; Rezende et al., 2014) deal with uncertainty by modeling distributions in the observation space, mapping random variables within a latent space to observations with a deep neural network. The motivation for these models is the modeling and approximation of complex distributions. Extension of these ideas to the modeling of sequences and time series has been proposed very recently (Fraccaro et al., 2016; Krishnan et al., 2017; Karl et al., 2016; Chung et al., 2015). Many of these models propose to combine ideas coming from recurrent neural networks where the latent (or hidden) units are deterministic functions of the past observations with ideas from state space systems where the latent states are modeled as random variables. The objective is generally to bypass the limitations of each model family (absence of uncertainty modeling for RNNs and limitations to simple dynamics for state space models). Although this is a very recent line of work, several variations of these ideas have been proposed for modeling complex temporal dynamics while considering the uncertainty of the dynamical process. Like for their static counterpart (Kingma et al., 2014), these models mainly focus on estimating complex distributions. The difference is that they consider sequence models instead of vector spaces for the former models. They are then mainly used as generative models and evaluated on their ability to recover these distributions. Our model has been designed for forecasting and is not used as a generative model. Also, our formulation gives us direct access to the output distribution with no need to sample or work with intractable distributions. Our goal is not to capture complex data distributions but to predict future or missing values. Another difference is that all these models do have as input either past observations (like in classical RNNs) or a command (like in classical state space systems), which is not the case for our model where the Z representations are learned directly. Finally none of these models considers relational series as we do here. Another related line of work is the extension of classical latent linear state space models (Kalman filters) by neural networks (Krishnan et al., 2015). Here again the

goal is different, and like in classical Kalman filters, the objective is to recover the latent signal. However, even if there are clear differences between all these systems both in their formulation and in their use, they are related and clarifying the nature of this relation is an interesting open topic.

6.3 Relational Time Series Forecasting with Gaussian Embeddings

6.3.1 Model

Notations and Task Formal Description

Let us consider a set of n temporal sequences[1] $\mathbf{x_1}, .., \mathbf{x_n}$, coming from different sources, such that $x_i^{(t)} \in \mathbb{R}$ is the value of the ith sequence at time t defined by $\mathbf{x_i} = (x_i^{(1)}, .., x_i^{(T)})$, T being the number of observed time steps. For simplification, we consider that all the series have the same length, but again this is not restrictive.

We model the dependencies between the different series through a graph, the different sources being the graph nodes and the links modeling explicit dependencies between the sources. This is illustrated in Figure 6.1. The links can reflect a spatial proximity between the sources of the series, a similarity of behavior between users or any other predefined relation. These explicit relations are modeled in the latent space, through our hypothesis that links represent some prior information about a potential dependency between two sources. If the observations confirm this dependency, the model will learn latent representations for the sources dynamics that will be close one to the other. The proximity of the representations is controlled by the strength of the link between the two time series when provided, and denoted $e_{i,j}$. We assume that the graph structure is static through time.

Let us denote τ the size of the prediction horizon. The **forecasting** problem considered here is to compute, knowing $\{x_i^{(t)}\}_{i,t}$ and for all series i the values $x_i^{(T+k)}$ for all k in $[1; \tau]$.

6.3.2 Informal description

The proposed model is a dynamic state space model: the dynamics are modeled in a continuous latent state space and the observations are generated from states in this latent space. State space models have already been considered for multiple time series (e.g. (Mirowski et al., 2009)) and for spatio-temporal processes (e.g. (Wikle et al., 2010)).

Both the observations and the dynamics are subject to uncertainties. Usually, the observations correspond to a partial view of the underlying generating process and the dynamics being hidden is not directly accessible and should be modeled as a stochastic process.

To handle this uncertainty, we propose a model, denoted here Relational Dynamic model with Gaussian representations (**RDG**), that represents latent factors as **distributions** in a latent space and learns the series dynamics in this latent space. The distributions themselves are estimated using observations like for any other representation learning model. Besides being more adapted to handling the noise inherent to the process and to the observations, the model can be used to predict the posterior distribution of the variables associated to the series and in particular the confidence or variance associated to the predictions.

[1] For simplicity, each sequence is an univariate time series, but the model can be trivially extended to multivariate series.

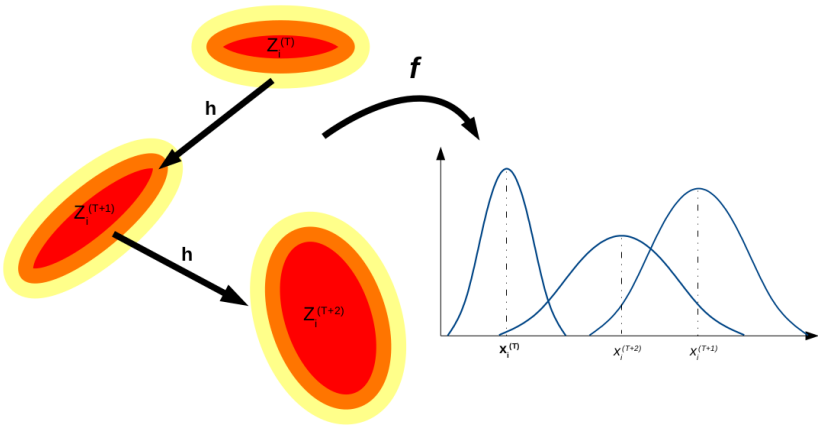


FIGURE 6.2: The Gaussian embedding $Z_i^{(t)}$, the dynamic function h and the decoding function f are learned. We compute $Z_i^{(t+1)}$ and $Z_i^{(t+2)}$ using h and $Z_i^{(t)}$. Then we compute $X_i^{(t+1)}$ and $X_i^{(t+2)}$ from $Z_i^{(t+1)}$ and $Z_i^{(t+2)}$ respectively using f.

The model is an extension of the deterministic model of (Ziat et al., 2016) and has two main components:

- **Decoding component f:** we consider that each series corresponds to a particular *trajectory* in an unknown latent space. Each series $x_i^{(1)}, ..., x_i^{(T)}$ is thus associated to a series of random variables in $\mathbb{R}^d$ denoted $Z_i^{(1)}, ..., Z_i^{(T)}$. $Z_i^{(t)}$ is the latent factor explaining the observed value of the series i at time t, and d is the size of the latent space. We model each $Z_i^{(t)}$ as a multivariate normal variable $\mathcal{N}(\mu_i^{(t)}, \Sigma_i^{(t)})$. The observation can be computed from this latent distribution by using a *decoding function* f mapping $Z_i^{(t)}$ to $X_i^{(t)} = f(Z_i^{(t)})$.
- **Dynamic component h:** The second component models the series dynamics in the latent space. We suppose that dynamics can be captured for all series through a function h that maps the latent random variable $Z_i^{(t)}$ to the next latent variable $Z_i^{(t+1)} = h(Z_i^{(t)})$. In addition, constraints are introduced to reflect prior knowledge about the relational dependency structure of the series. For any couple of series i and j with a known dependency, i.e. such that $e_{i,j} > 0$, we add a corresponding constraint on $Z_i^{(t)}$ and $Z_j^{(t)}$.

Figure 6.2 summarizes this description.

In the following, we explain how the distributions corresponding to the random variables $Z_i^{(t)}$ are learned, jointly to the functions f (decoder component) and h (dynamic component).

Model Definition

We suppose that the random variables $Z_i^{(t)}$ follow a Gaussian distribution $Z_i^{(t)} \sim \mathcal{N}(\mu_i^{(t)}, \Sigma_i^{(t)})$ where $\mu_i^{(t)}$ and $\Sigma_i^{(t)}$ have to be estimated from known observations. For simplicity, we consider in the following that $\Sigma_i^{(t)}$ is a diagonal matrix, with $\sigma_{i,j}^{(t)}$ denoting the jth value of the

diagonal of $\Sigma_i^{(t)}$. Since the representations $Z_i^{(t)}$ themselves are learned in the training phase, this diagonal hypothesis imposes a simple constraint on the $Z_i^{(t)}$ and the model remains very flexible.

We define a global loss function $\mathcal{L}(\mu, \mathbf{\Sigma}, f, h)$ where μ and $\mathbf{\Sigma}$ denote the means and covariance matrices for all the series and for all the time steps between 1 and T. The loss is a sum of three terms: (i) a decoding loss Δ_{De}, (ii) a dynamical loss Δ_{Dy} and (iii) a structural loss Δ_{R}:

$$\mathcal{L}(\mu, \mathbf{\Sigma}, f, h) = \sum_{i=1}^{n}\sum_{t=1}^{T} \Delta_{\text{De}_1}(f(Z_i^{(t)}), x_i^{(t)}) + \lambda_{\text{Dy}} \sum_{i=1}^{n}\sum_{t=1}^{T-1} \Delta_{\text{Dy}_1}(Z_i^{(t+1)}, h(Z_i^{(t)})) + \lambda_{\text{R}} \sum_{j=1}^{n}\sum_{t=1}^{T} e_{i,j}\Delta_{\text{R}}(Z_i^{(t)}, Z_j^{(t)}) \quad (6.1)$$

where λ_{Dy} and λ_{R} are hyperparameters weighting the importance of the different elements in the loss function.

The first term of Equation 6.1 corresponds to the *decoding component*, and forces both f and the learned distributions of variables Z to "explain" the observations, the second term, the *dynamic component*, encourages h to model the time dynamics in the latent space, while the third term captures the relations between the pairs of series. In the following, we use for f a linear function and for h either a linear or non-linear function. The reason for using a linear f is that the model flexibility mainly comes from the possibility to learn the Z representations and that adding a non linear decoder would not improve much its potential. Besides, it allows to derive closed form expressions for the predicted outputs and thus offers simple scalable solutions, whereas predicting the distributions for a non linear transformation of the latent variables would involve more complex approximate solutions.

Learning Learning the model is performed through the minimization of the loss function $\mathcal{L}(\mu, \mathbf{\Sigma}, f, h)$ with respect to μ, $\mathbf{\Sigma}$, f and h. To simplify the notations, the parameters of f and h are not made explicit in the notations – f and h are only supposed to be differentiable. At the end of the learning process, all the latent distributions for each of the time steps are known for the training data, as well as the decoding function f and the dynamical one h. We used *ADAM* (Kingma et al., 2015) as a stochastic gradient descent technique. This optimization can be easily made on a large scale dataset, and/or by using GPUs. We have considered different variants of the model. For clarity we first describe below the model that provided the best results on the majority of experiments. The other variants are briefly described at the end of the section and will then be used as baselines for comparison.

From Latent Space to Observations The mapping onto the latent space is learned so that the values $x_i^{(t)}$ of each series can be predicted from its respective Gaussian embedding $Z_i^{(t)}$ through the f function. We use, as the loss functions Δ_{De_1} for measuring the error between the predicted distribution $f(Z_i^{(t)})$ and the observation $x_i^{(t)}$, the expectation of the

mean squared error between the predictions and the observations:

$$\Delta_{\text{De}_1}(f(Z_i^{(t)}), x_i^{(t)}) \stackrel{\text{def}}{=} E\left[(f(Z_i^{(t)}) - x_i^{(t)})^2\right] \tag{6.2}$$

When f is a linear function, this loss can be explicitly written as:

$$\Delta_{\text{De}_1}(f(Z_i^{(t)}), x_i^{(t)}) = \sum_{k=1}^{d} \theta_k^2 \sigma_{i,k}^{(t)} + \left(< \theta, \mu_i^{(t)} > - x_i^{(t)}\right)^2 \tag{6.3}$$

Note that minimizing Δ_{De_1} updates both the mean and the variance of the distribution. More specifically, an observed value x_i with Δ_{De_1} will pull all the variances $\sigma_{i,k}^{(t)}$ down. This is an interesting property since observing values should reduce the variance of the representation. Moreover, variance is reduced for the dimensions that are important for the prediction.

Modeling Dynamics The loss function Δ_{Dy_1} aims at finding values $Z_i^{(\cdot)}$ and a dynamic model h, that will be used to predict the representation of the next state $Z_i^{(t+1)}$ of the time series i. The function h maps a distribution $\mathcal{N}(\mu_i^{(t)}, \Sigma_i^{(t)})$ to $\mathcal{N}(\mu_i^{(t+1)}, \Sigma_i^{(t+1)})$. Based on (Vilnis et al., 2015) and our previous contribution (Chapter 5), we use a Kullback-Leibler divergence (noted $D_{KL}(\cdot||\cdot)$) to compare the distribution of $Z_i^{(t+1)}$ to the distribution predicted by h, $f(Z_i^{(t)})$.

We consider that the latent representation at time $(t+1)$ is a non linear transformation of the latent distribution at time t. Practically, we used two non linear multilayer perceptrons (MLP), h^m and h^c; h^m predicts the means and h^c variances: more precisely the next mean is given by $\mu_i^{(t+1)} = h^m(\mu_i^{(t)}, \Sigma_i^{(t)})$, and the next variance by $\Sigma_i^{(t+1)} = h^c(\mu_i^{(t)}, \Sigma_i^{(t)})$. This gives:

$$\Delta_{\text{Dy}_1}(Z_i^{(t+1)}, h(Z_i^{(t)})) \stackrel{\text{def}}{=} D_{KL}(Z_i^{(t+1)} || \mathcal{N}(h^m(\mu_i^{(t)}, \Sigma_i^{(t)}), h^c(\mu_i^{(t)}, \Sigma_i^{(t)}))) \tag{6.4}$$

We make the hypothesis that the resulting distribution (for $Z_i^{(t+1)}$) is Gaussian. The KL divergence between the two Gaussian distributions has a simple analytic form from which the gradient can be easily computed (Vilnis et al., 2015).

Structural regularization term At last, Δ_{R} corresponds to a **structural regularization** over the graph structure that encourages the model to learn similar representations for time series that are interdependent. This forces the model to learn representations that reflect the structural dependencies between the series. Recall that these dependencies are supposed to be provided as priors for this model. We define this regularization loss as:

$$\Delta_{\text{R}}(Z_i^{(t)} || Z_j^{(t)}) = D_{KL}(Z_i^{(t)} || Z_j^{(t)}) \tag{6.5}$$

which again has, for Gaussian random variables, a simple analytical form that can be used for learning[2]. This is similar to what was done in Chapter 5 to define the relationships

[2] $\frac{1}{2}(\text{tr}({\Sigma_j^{(t)}}^{-1}\Sigma_i^{(t)}) + (\mu_j^{(t)} - \mu_i^{(t)})^T {\Sigma_j^{(t)}}^{-1}(\mu_j^{(t)} - \mu_i^{(t)}) - d - \log(\frac{|\Sigma_i^{(t)}|}{|\Sigma_j^{(t)}|}))$

between two nodes in the graph.

Impact of minimizing the KL-divergence on predicted values

Here, we show how the structural regularization term Δ_R between two connected time series bounds the difference between predicted distributions w.r.t. the total variation distance.

Definition 1 *Let X and Y be two real random variables of density distribution respectively $\mathcal{D}_X$ and $\mathcal{D}_Y$. Let $\mathcal{B}$ be the Borel set of $\mathbb{R}$. The total variation distance between X and Y noted $d_{TV}(X, Y)$ is defined via:*

$$d_{TV}(X, Y) = \sup_{A \in Borel} (|\mathcal{D}_X(A) - \mathcal{D}_Y(A)|) \tag{6.6}$$

where d_{TV} is a distance between distributions. It is used here because of its simple relation with D_{KL} through Pinsker inequality. d_{TV} represents the largest possible difference between the probabilities that the two random variables can assign to the same event. It is greater or equal to the ℓ_1 distance between the density functions.

Property 1 (Pinsker's inequality for Gaussian vectors) *Let Z_i and Z_j be two Gaussian random variables of $\mathbb{R}^d$ with both diagonal co-variance matrices. Let $\theta \in \mathbb{R}^d$. We have:*

$$d_{TV}(\theta \cdot Z_i, \theta \cdot Z_j) \leq \sqrt{\frac{d \cdot D_{KL}(Z_i||Z_j)}{2}} \tag{6.7}$$

Proof 1 *Since we consider here diagonal covariance matrices and that the KL-divergence is invariant by multiplying both random variables by the same scalar, we can show that:*

$$D_{KL}(Z_i||Z_j) = \sum_{k=1}^{d} D_{KL}(Z_{i,k}||Z_{j,k}) = \sum_{k=1}^{d} D_{KL}(\theta_k Z_{i,k}||\theta_k Z_{j,k}) \tag{6.8}$$

with $Z_{i,k}$ being the k-th component of the Gaussian vector Z_i.

Using Pinsker's inequality one can see that minimizing the KL-divergence also minimizes the total variation norm, leading to:

$$2 \sum_{k=1}^{d} (d_{TV}(\theta_k Z_{i,k}, \theta_k Z_{j,k}))^2 \leq \sum_{k=1}^{d} D_{KL}(\theta_k Z_{i,k}||\theta_k Z_{j,k}) \tag{6.9}$$

Then using the Cauchy–Schwarz inequality, we have:

$$\frac{1}{d} \left(\sum_{k=1}^{d} d_{TV}(\theta_k Z_{i,k}, \theta_k Z_{j,k}) \right)^2 \leq \sum_{k=1}^{d} (d_{TV}(\theta_k Z_{i,k}, \theta_k Z_{j,k}))^2 \tag{6.10}$$

Finally, each component of the random vectors Z being pairwise independent, we have:

$$d_{TV}(\sum_{k=1}^{d} \theta_k Z_{i,k}, \sum_{k=1}^{d} \theta_k Z_{j,k}) \leq \sum_{k=1}^{d} d_{TV}(\theta_k Z_{i,k}, \theta_k Z_{j,k}) \tag{6.11}$$

Combining the above inequalities, we can straightforwardly show the following one:

$$d_{TV}\left(\theta \cdot Z_i, \theta \cdot Z_j\right) \leq \sqrt{\frac{d \cdot D_{KL}(Z_i||Z_j)}{2}} \tag{6.12}$$

Thus, the closer the $\mathbb{R}^d$-Gaussian variables (w.r.t. the KL-divergence), the closer their inner product distribution with a vector of $\mathbb{R}^d$ (w.r.t. the total variation distance).

This means that having similar latent representations constrains the predicted values to be similar, and the above result provides a bound on this closeness.

Inference and Time Complexity

The inference step is straightforward. Having learned f, h and the distribution parameters of Z using past values for all series i, the last $Z_i^{(T)}$ distribution values will be used as initial values for predicting the latent distributions at $(T+1)$ using transfer function h: the predicted distribution then follows $\mathcal{N}(h(\mu_i^{(T)}, \Sigma_i^{(T)}))$. The time series prediction at time $T+1$ can then be computed via the decoding function f and follows $\mathcal{N}(h(\mu_i^{(T)}, \Sigma_i^{(T)}))$. To predict at horizon 2, we just iterate the procedure, starting from the distribution of the latent variables at time $T+1$, leading to distribution $\mathcal{N}(h \circ h(\mu_i^{(T)}, \Sigma_i^{(T)}))$ for $Z_i^{(T+2)}$, and then again using h to get the predicted distribution of the series at $T+2$, etc.

Inferring the value of the i-th time series at time $(T+N)$ requires to compute $f(h^N(Z_i^T))$, f being linear if h is a MLP the time complexity of the inference procedure is then in $\mathcal{O}(Ndd')$ with d' the size of the MLP hidden layer, with $\mathcal{O}(nTd + dd')$ parameters to learn, if h is linear the complexity of the inference is in $\mathcal{O}(Nd^2)$ with $\mathcal{O}(nTd + d^2)$ parameters to learn.

Variants

We have experimented with other variants of the model by considering alternative losses and h functions. For the loss, we also considered a simple difference between the expected value of f and the observation using a mean-square error:

$$\Delta_{\text{De}_2}(f(Z_i^{(t)}), x_i^{(t)}) \stackrel{\text{def}}{=} \left(E\left[f(Z_i^{(t)})\right] - x_i^{(t)}\right)^2 \tag{6.13}$$

When considering a linear decoding function such as $f(\cdot) =< \theta, \cdot >$, θ being the set of parameters of f, Δ_{De_2} can be rewritten as as:

$$\Delta_{\text{De}_2}(f(Z_i^{(t)}), x_i^{(t)}) = (< \theta, \mu_i^{(t)} > - x_i^{(t)})^2 \tag{6.14}$$

This loss only models the mean of the distribution whereas the loss in Section 6.3.1 considers both the mean and the variance.

For h, we also considered a linear function, instead of a non linear one as described before. This is again a simplification of the model. We then consider that the latent representation at time $(t+1)$ is a linear transformation of the latent distribution at time t. The

transformed variable is also a Gaussian and its parameters can be easily computed. h is a function from $\mathbb{R}^d$ to $\mathbb{R}^d$ which is represented by a matrix $\gamma \in \mathcal{M}_{d,d}(\mathbb{R})$:

$$\Delta_{\mathrm{Dy}_2}(Z_i^{(t+1)}, h(Z_i^{(t)})) \stackrel{\text{def}}{=} D_{KL}(Z_i^{(t+1)}||\gamma Z_i^{(t)}) = D_{KL}(Z_i^{(t+1)}||\mathcal{N}(\gamma\mu_i^{(t)}, \gamma\Sigma_i^{(t)}\gamma^T)) \quad (6.15)$$

6.3.3 Experiments

Datasets

Experiments have been performed on four datasets respectively extracted from Google Flu Trends[3], WHO[4] and from two datasets from Grand Lyon[5] (GL) (respectively data from traffic conditions and from car parks occupancy). All the series are normalized. For all datasets, we used binary dependency relations indicating whether two series are related or not.

- The **Google Flu Trend** (GFT) dataset is composed of an aggregation of weekly Google search queries related to the flu in 29 countries. This dataset spans about ten years of time. The relations between series are defined a priori so that the series of two countries i and j are linked, i.e. $e_{i,j} = 1$ in Equation 6.1, only if the countries have a common frontier. There are 96 relations.
- The **GL Traffic** (GL-T) dataset corresponds to the traffic conditions of the 50 busiest roads of the city of Lyon (France). Data is aggregated on 20 minutes windows spanning 15 days. The binary relations between series are based on the geographical proximity of roads. There are 130 relations.
- The **GL Park** (GL-P) dataset represents the occupancy of public car parks in Lyon. The series correspond to the occupancy of the 30 busiest car parks. It has the same window and period of time as the previous dataset, and the binary relations between series are based on the geographical proximity of car parks. There are 74 relations.
- The **WHO** dataset provides the number of deaths caused by diphtheria over 91 different countries, giving rise to 91 time series. The binary relations between series are defined so that two series are linked if the corresponding countries share a common frontier. There are 228 links.

Baselines

We compare our approach with five baselines:

- Auto-Regressive (**AR**), a monovariate linear auto-regressive model. It computes its predictions based on a learned linear function of a fixed number p of past values of the series. The order p of the model is a hyperparameter of the model selected by grid search.

[3]http://www.google.org/flutrends
[4]http://www.who.int
[5]http://data.grandlyon.com

- Feed Forward Neural Network (**FFNN**), representative of non-linear auto-regressive models of order p where the non-linear function is modeled as a feed-forward neural network with one hidden layer of size s. In this case, p and s are hyperparameters selected by grid search.
- **RNN**, a recurrent neural network, with one hidden layer of size s, of recurrent units and $tanh$ non-linearities. The RNN model is a state space non-linear auto-regressive model with exogenous inputs (the past values of the series). Note that this model should in principle be able to learn the inter-series dependencies, but the dependencies are not modeled explicitly as they are in our model. Also the RNN does not introduce explicit modeling of uncertainties. We used here a simple RNN, experiments with GRU and LSTM giving similar results.
- **KF** (Kalman, 1960), is a classic Kalman Filter with linear transformations from one state to another.
- **DFG** (Mirowski et al., 2009), a state of the art model that learns continuous deterministic latent variables by modeling the dynamics and the joint probabilities between series. All the hyperparameters of the baselines have been set using a validation set by grid search, including the best architectures for the dynamic model h when it is a multi-layer perceptron with one hidden layer or a linear model.

The different variants of our model are denoted $\mathbf{RDG}_{i,j}$ with $i, j \in \{1, 2\}$. $\mathbf{RDG}_{1,1}$ thus corresponds to the main model introduced in Section 6.3.1, $\mathbf{RDG}_{1,2}$ to the model with decoding loss Δ_{De_1} and linear h, i.e. dynamic loss Δ_{Dy_2}.

Experimental protocol

The different model hyper-parameters have been selected using a time series cross-validation procedure called rolling origin as in (Ben Taieb et al., 2014; Ganeshapillai et al., 2013). This protocol makes use of a sliding training window of size T. The value of T has been fixed so that it is large enough to capture the main dynamics of the different series. Each series has been normalized such that its values are between 0 and 1.
For the evaluation metric, we have considered a root-mean-square error (RMSE) criterion. We evaluate the models by considering prediction errors at horizon $t + 1$.

Results

Let us first present the performance of our model w.r.t. the baselines for prediction at horizon 1 in Figure 6.3. We have tested the four variants of our approach i.e. combinations of Δ_{De_1} or Δ_{De_2} with Δ_{Dy_1} or Δ_{Dy_2}. The proposed model obtains the best results on all the datasets except GFT where KF performs better. It outperforms the baselines on two datasets (GL-P -Grand Lyon Parks- and GFT -Google Flu Trends- on the table) and gets results similar to the RNN on the two others (GL-T -Grand Lyon Traffic- and WHO). $\mathbf{RDG}_{1,1}$ usually gets better results than the other models, i.e. the best combination is a MSE expectation error for the decoder and a non-linear model for the dynamics.

Figure 6.4 shows the prediction quality (RMSE) at $(T + 1)$, $(T + 2)$, $(T + 3)$, $(T + 4)$ and $(T + 5)$ and illustrates the ability of RDG to predict correctly at different horizons. One can remark that at $(T + 5)$ KF performs badly in comparison to other baselines.

Model	GL-T	GL-P	GFT	WHO
AR	0.0752	0.0892	0.0626	0.0832
FFNN	0.0751	0.0894	0.045	0.0838
RNN	0.0709	0.0890	0.0431	**0.0795**
KF	0.0711	0.0833	**0.0388**	0.0799
DFG	0.0712	0.0911	0.0592	**0.0795**
RDG$_{2,2}$	0.0742	0.0902	0.0607	0.0848
RDG$_{2,1}$	**0.0707**	0.0834	0.0434	0.0796
RDG$_{1,2}$	0.0765	0.0896	0.0589	0.0831
RDG$_{1,1}$	0.0718	**0.0828**	0.0429	**0.0795**

FIGURE 6.3: Comparison of RMSE at T+1 on the four datasets between baselines and our proposed model (RDG) on the prediction task. RDG$_{k,l}$ corresponds to the variant with losses (Δ_{De_k},Δ_{Dy_l}).

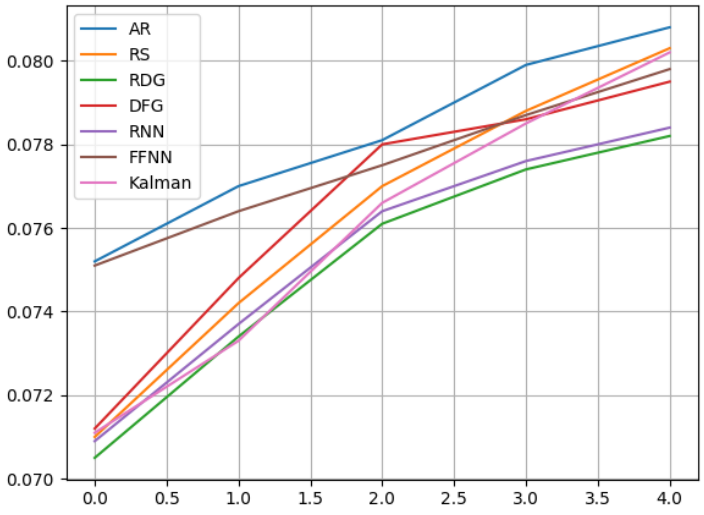


FIGURE 6.4: Comparison of RMSE from T+1 to T+5 on GL-T between baselines and our proposed model (RDG) on the prediction task.

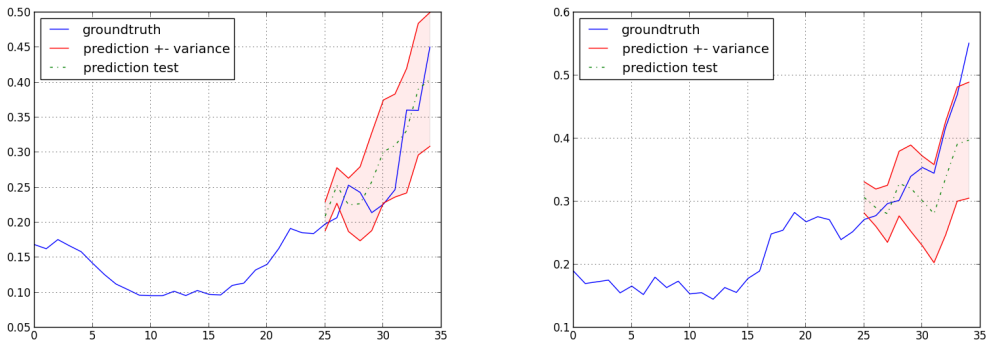


FIGURE 6.5: Forecasts on GFT (two different time series of the dataset) with the $\text{RDG}_{2,2}$ model showing its range of confidence: $E(f(Z^{(t)})) \pm \text{var}(f(Z^{(t)}))$. Prediction at $25 + n$ corresponds to $f(h^n(Z^{(25)}))$.

RDG has the additional property of modeling the uncertainty associated to its predictions, which is not the case for a RNN. Let us consider the curves presented in Figure 6.5. They illustrate, the predictions made by our model together with their associated variance computed through the Gaussian embeddings. First, one can see that the ground truth values are always within the confidence interval provided by our model, which means that RDG computes relevant minimum and maximum possible values. Another aspect is that the size of the interval increases with the prediction horizon, which is what is expected from such a model. The latter is then able to predict relevant confidence values for its predictions.

Comparison between RDG with/without structural regularization or uncertainty We compare in Table 6.1 the results between our model when taking into account the neighborhood graph ($\lambda_R \neq 0$) or not ($\lambda_R = 0$): forecasts are uniformly worse for all datasets when we do not take into account the neighborhood graph, it suggests that the regularizer improves the model when the input graph is relevant. Furthermore, we give the results obtained without uncertainty, which corresponds to the model described in (Ziat et al., 2016) (denoted Rainstorm): here again, our model outperforms this deterministic model for all the datasets.

Model	GL-T	GL-P	GFT	WHO
Rainstorm	0.0710	0.0886	0.0440	0.0804
RDG ($\lambda_R = 0$)	0.0719	0.900	0.0441	0.0807
RDG	**0.0707**	**0.0828**	**0.0388**	**0.0795**

TABLE 6.1: RMSE at $T + 1$ on the four datasets.

6.4 Conclusion

In this contribution, we have proposed a model for relational time series forecasting. Our model (RDG) is based on latent Gaussian embeddings, and has shown competitive performance on four different datasets compared to state-of-the-art models. Moreover, RDG allows us to model the uncertainty of predictions, providing for example confidence intervals for each prediction.

Chapter 7

Learning Gaussian Representations for Collaborative Filtering

Abstract

Most collaborative filtering systems, such as matrix factorization, use vector representations for items and users. Recommendation datasets consist in a matrix of item ratings from various users. This matrix can also be seen as a weight matrix for a bipartite graph between users and items which motivated our interest in tackling this task. Those representations are deterministic, and do not allow modeling the uncertainty of the learned representation. Uncertainty can be useful when a user has a small number of rated items (cold start), or when there is conflicting information about the behavior of a user or the ratings of an item. In this chapter, we leverage recent works in learning Gaussian embeddings and apply it to the task of recommendation. In this contribution, we show that our proposed model performs well on three representative collections (Yahoo, Yelp and MovieLens) and analyze learned representations.

Contents

7.1 Introduction 103
7.1.1 Recommender Systems and Uncertainty 103
7.1.2 Contribution 105
7.2 Learning Gaussian Embeddings for Collaborative Filtering 105
7.2.1 Model 105
7.2.2 Experiments 109
7.3 Conclusion 117

7.1 Introduction

7.1.1 Recommender Systems and Uncertainty

Recommendation is the last task involving relational data that we studied during this thesis. Recommendation datasets consist before all in a matrix of item ratings from various users. This matrix can also be seen as a weight matrix for a bipartite graph between users and items. Note that more complex recommendation datasets exist, which might involve multiple relation types between users and products (e.g. *buy*, *look at*) or associate attributes with users and items (e.g. *ratings*, *reviews*). Recommender systems help users find items

they might like in large repositories, thus alleviating the information overload problem. Methods for recommending items can be broadly classified into collaborative filtering (CF), content-based, and hybrid methods (Ricci et al., 2011). Among recommender systems, collaborative filtering systems are the most successful and widely used because they leverage all past ratings from users and items, while content-based approaches typically build users (or items) profiles from items (or users) predefined representations. In the context of the thesis, exploring collaborative filtering methods is interesting since the task can be cast into a problem involving relational data where entities are users and products, and relationships include some form of relation like *friendship*, *buying*, *rating*, *reviewing* or *searching for a product*.

This contribution is a first step in this direction: we explore the use of Gaussian representations for the recommendation task. Among collaborative recommender systems we are interested in those that represent users and items as vectors in a common latent "recommendation" space. Matrix factorization derivatives are a typical example of such models. The main assumption behind these models is that the inner product between a user and an item representation is correlated with the rating the user would give to the item. The main interest of these models is their potential for integrating different sources of information (Zhang et al., 2016) as well as their good performance, both in efficiency and effectiveness (Ricci et al., 2011). The most successful approaches are based on learning-to-rank approaches such as (Rendle et al., 2009; Weimer et al., 2007).

Despite the success of deterministic representation learning, one limitation of these models is their inability for handling uncertainty. Even when they are based on a probabilistic model, e.g. (Salakhutdinov et al., 2007; Rendle et al., 2009), they only suppose a Gaussian prior over the user and item embeddings but still learn a deterministic representation. The prior is thus only used as a regularization term on the user and item representations.

Uncertain representations can have various causes, either related to the lack of information (new users or items, insufficient number of ratings for a user or an item, users that did not rate any movie of a given genre), or to contradictions between user or item ratings. To illustrate the latter, let us take a user that likes only a part of Kung Fu movies, but with no clear pattern, i.e. no sub-genre. With usual approaches, such a user will have a component set to zero for the direction of the space corresponding to Kung Fu. With our proposed approach, we would still have a zero mean, but with a high variance. Our hypothesis is that, because of these two factors, namely cold start and conflicting information, training will result in learned representations with different confidences, and that this uncertainty is important for recommending.

Moreover, using a density rather than a fixed point has an important potential for developing new approaches to recommendation. First, instead of computing a score for each item, the model can compute a probability distribution of the score, or the covariance between the scores given by users for two different item. This can be interesting when trying to diversify result lists, since the variance of the representations can be leveraged e.g. by Portfolio approaches[1] (Wang et al., 2009): the model could propose diversified lists taking into account the variance of the representation (i.e. with different degrees of risk). Secondly, such models are interesting because they can serve as a basis for integrating different

[1]Economic theory dealing with investment in financial markets. The goal is to select a set (portfolio) of financial products based on their uncertainty/risk. This theory suggests that diversification is an effective way to reduce the risk of the portfolio (Markowitz, 1952).

sources of external information, and thus serve as a better bridge between content-based approaches and collaborative filtering ones. Thirdly, time could be modeled by supposing that the variance of the user or item representation increases with time if no new information (i.e. user ratings) are provided.

7.1.2 Contribution

In this chapter, we describe the model proposed as our fourth contribution. This model uses Gaussian embeddings which have been recently and successfully proposed for learning word (Vilnis et al., 2015), knowledge graph (He et al., 2015), nodes in a graph (Chapter 5) or time series (Chapter 6) embeddings. More precisely, we suppose that each user or item representation corresponds to a Gaussian distribution where the mean and the variance are learned. Said otherwise, instead of a fixed point in space, a density represents each item or user. The variance term is a measure of uncertainty associated to the user/item representation.

To learn these Gaussian embeddings, we rely on a learning to rank approach. More precisely, we use the same starting point as (Rendle et al., 2009), that aims at maximizing the probability of ranking pairs of items for individual users. While the starting point is the same, we can leverage the probabilistic framework to directly estimate this probability – while (Rendle et al., 2009) use a transformation of the difference of the items scores to compute it.

In the following, we first describe the proposed model, and then show experimentally that it performs very well compared to state-of-the-art approaches on three different datasets. The contributions of this chapter are

- The use of Gaussian representations for items and users in collaborative filtering;
- Extensive experimental work that shows the good performance of the model;
- Qualitative analysis to show how the variance component is used in the model.

7.2 Learning Gaussian Embeddings for Collaborative Filtering

7.2.1 Model

Our model is learned through a pairwise learning-to-rank criteria that was first proposed by Rendle et al. (Rendle et al., 2009) for collaborative filtering (Bayesian Personalized Ranking, BPR). Formally, they optimize the maximum a posteriori of a ranking model, a training dataset $\mathcal{D}$, i.e.

$$p\left(\Theta|\mathcal{D}\right) \propto p\left(\mathcal{D}|\Theta\right) p\left(\Theta\right)$$

where $\mathcal{D}$ represents the set of all ordered triples (u, i, j) (the user $u \in \mathcal{U}$ prefers item $i \in \mathcal{I}$ to item $j \in \mathcal{I}$) and Θ, the model parameters. The factor $p\left(\Theta\right)$ corresponds to the prior on the item and user representations (a Gaussian prior in (Rendle et al., 2009)). Then, using the standard hypothesis of the independence of samples given the model parameters, we have

$$p\left(\mathcal{D}|\Theta\right) = \prod_{(u,i,j)\in\mathcal{D}} p\left(i >_u j|\Theta\right) \tag{7.1}$$

In (Rendle et al., 2009), the probability that user u prefers item i to item j, noted by $i >_u j$, is modeled by the sigmoid function of the difference of the inner products, that is:

$$p\left(i >_u j|\Theta\right) = \sigma\left(x_i \cdot x_u + b_i - x_j \cdot x_u - b_j\right) \tag{7.2}$$

where b_i (resp. b_j) is the (popularity) bias for item i (resp. j)[2]. The popularity bias is the tendency for popular items to be recommended more frequently, i.e. it corresponds to the mean score that should be given to the corresponding item.

The Gaussian Embeddings Ranking Model

In this contribution, while we start with Equation 7.1, the different variables $x_\bullet$ ($\bullet$ is either a user u or an item i) are random variables, denoted $\mathbf{X}_\bullet$, and not elements of a vector space. We can hence estimate directly the probability $p\left(i >_u j|\Theta\right)$ for the inner product $\mathbf{X}_u \cdot \mathbf{X}_i$ to be higher than $\mathbf{X}_u \cdot \mathbf{X}_j$.

We start by supposing that user and item representations follow a normal distribution, that is for any user u and item i,

$$\mathbf{X}_i \sim \mathcal{N}\left(\mu_i, \Sigma_i\right) \text{ and } \mathbf{X}_u \sim \mathcal{N}\left(\mu_u, \Sigma_u\right)$$

We suppose that $\Sigma_\bullet = \mathrm{diag}(\sigma_{\bullet 1}, \ldots, \sigma_{\bullet N})$ is a diagonal covariance matrix[3] to limit the complexity of the model (N is the dimension of the latent space): in practice, we have $2N$ parameters for each user, and $2N + 1$ for each item ($+1$ because of the bias). The variance is associated with each element of the standard basis of the vector space. In practice, this helps the model to learn meaningful dimensions, since it forces the use of the standard latent space basis.

Since $\mathbf{X}_i$ and $\mathbf{X}_u$ are random variables, their inner product is also a random variable, and we do not need to use the sigmoid function of Equation 7.2 as for BPR to compute the probability that user u prefers i to j. We can instead directly use the random variables defined above, that is:

$$\begin{aligned} p\left(i >_u j|\Theta\right) &= p\left(\mathbf{X}_i \cdot \mathbf{X}_u + b_i > \mathbf{X}_j \cdot \mathbf{X}_u + b_j|\Theta\right) \\ &= p\Big(\underbrace{\mathbf{X}_u \cdot \left(\mathbf{X}_j - \mathbf{X}_i\right)}_{\mathbf{Z}_{uij}} < b_i - b_j \Big| \Theta\Big) \end{aligned}$$

where b_i and b_j are the item biases[4]. To compute the above equation, we use two following simplifications:

1. Item representations are independent given the model parameters. This implies that the difference $\mathbf{X}_j - \mathbf{X}_i$ follows a normal distribution

$$\mathcal{N}\left(\mu_j - \mu_i, \Sigma_i + \Sigma_j\right)$$

[2]Since this equation represent the difference between the expected scores of items i and j, there is no user bias.

[3]Note that we denote the variance σ.

[4]In this contribution, we did not consider them to be random variables, but they could be.

2. We approximate the inner product distribution by a Gaussian[5] using moment matching (mean and variance), using the results from (Brown et al., 1977) who defined the moments of the inner product of two Gaussian random variables; with our notations, the two first moments of $\mathbf{Z}_{uij}$ are defined as:

$$\mathbb{E}\left[\mathbf{Z}_{uij}\right] = \mu_u^\top(\mu_j - \mu_i) \tag{7.3}$$

$$\begin{aligned}\mathrm{Var}\left[\mathbf{Z}_{uij}\right] &= \mu_u^\top\left(\Sigma_i + \Sigma_j\right)\mu_u + \left(\mu_j - \mu_i\right)^\top \Sigma_u(\mu_j - \mu_i) \\ &\quad + \mathrm{tr}\left(\Sigma_u\left(\Sigma_j + \Sigma_i\right)\right) \\ &= \sum_k \left(\sigma_{ik} + \sigma_{jk}\right)\left(\mu_{uk}^2 + \sigma_{uk}\right) + \sigma_{uk}\left(\mu_{jk} - \mu_{ik}\right)^2\end{aligned} \tag{7.4}$$

The above assumptions allow us to write:

$$p\left(i >_u j|\Theta\right) = \int_{-\infty}^{b_i - b_j} \mathcal{N}\left(x; \mathbb{E}\left[\mathbf{Z}_{uij}\right], \mathrm{Var}\left[\mathbf{Z}_{uij}\right]\right)\, \mathrm{d}x \tag{7.5}$$

which is the Normal cumulative distribution function, it can be rewritten using the error function ($\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$):

$$\begin{aligned}p\left(i >_u j|\Theta\right) &= \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{(b_i - b_j) - \mathbb{E}\left[\mathbf{Z}_{uij}\right]}{\sqrt{2\mathrm{Var}\left[\mathbf{Z}_{uij}\right]}}\right)\right] \\ &= \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{b_i - b_j - \mu_u^\top(\mu_j - \mu_i)}{\sqrt{2\sum_k \left(\sigma_{ik} + \sigma_{jk}\right)\left(\mu_{uk}^2 + \sigma_{uk}\right) + \sigma_{uk}\left(\mu_{jk} - \mu_{ik}\right)^2}}\right)\right]\end{aligned} \tag{7.6}$$

This function can be easily differentiated with respect to the $\mu_{\bullet k}$ and $\sigma_{\bullet k}$ ($\bullet$ is a user or an item):

$$\begin{aligned}\frac{\partial p\left(i >_u j|\Theta\right)}{\partial \mu_{ik}} &= -\frac{\partial p\left(i >_u j|\Theta\right)}{\partial \mu_{jk}} = -\Delta_{uij}\frac{\mu_{uk}D_{uij}^2 + 2\sigma_{uk}(\mu_{jk} - \mu_{ik})N_{uij}}{D_{uij}^3} \\ \frac{\partial p\left(i >_u j|\Theta\right)}{\partial \sigma_{ik}} &= \frac{\partial p\left(i >_u j|\Theta\right)}{\partial \sigma_{jk}} = \Delta_{uij}N_{uij}\frac{\mu_{uk}^2 + \sigma_{uk}}{D_{uij}^3} \\ \frac{\partial p\left(i >_u j|\Theta\right)}{\partial \mu_{uk}} &= \Delta_{uij}\frac{(\mu_{ik} - \mu_{jk})D_{uij}^2 + 2\mu_{uk}(\sigma_{ik} + \sigma_{jk})N_{uij}}{D_{uij}^3} \\ \frac{\partial p\left(i >_u j|\Theta\right)}{\partial \sigma_{uk}} &= \Delta_{uij}N_{uij}\frac{\sigma_{ik} + \sigma_{jk} + (\mu_{jk} - \mu_{ik})^2}{D_{uij}^3}\end{aligned} \tag{7.7}$$

with

$$\begin{aligned}N_{uij} &= b_i - b_j - \mathbb{E}\left[\mathbf{Z}_{uij}\right] \\ D_{uij} &= \sqrt{2\mathrm{Var}\left[\mathbf{Z}_{uij}\right]} \\ \Delta_{uij} &= \frac{1}{\sqrt{\pi}}\exp\left[-\left(\frac{N_{uij}}{D_{uij}}\right)^2\right]\end{aligned} \tag{7.8}$$

[5] This is a common approximation because of ease of computation.

From Equation 7.6, and ignoring the biases, we can see that the difference between the inner products $\mu_u \cdot \mu_i$ and $\mu_u \cdot \mu_j$ controls which is the preferred item, i.e. if the difference is negative (resp. positive) then erf returns a positive (resp. negative) value. Furthermore, the variances of the user and the item control the confidence. The larger the variances, the closer to 0.5 the probability (since $\mathrm{erf}(0) = 0$). This is the main difference with other matrix factorization-based approaches.

Finally, we have to define the prior distribution over the parameters Θ on the parameters, i.e. the means and variances. As we consider only a diagonal covariance matrix, we can consider an independent prior for each mean and variance, i.e. for any $\mu_{\bullet k}$ and $\Sigma_{\bullet kk}$. In this case, using a NormalGamma (NG) prior is a natural choice since it is the conjugate distribution of a normal distribution with unknown mean and variance. More specifically, we suppose that

$$\left(\mu_{\bullet k}, \sigma_{\bullet k}^{-1}\right) \sim \mathrm{NG}\left(\nu, \lambda, \alpha, \beta\right) \tag{7.9}$$

where $\bullet$ is a user or an item and:

$$\mathrm{NG}\left(x, \tau |\, \nu, \lambda, \alpha, \beta\right) = \frac{\beta^\alpha \sqrt{\lambda}}{\Gamma(\alpha)\sqrt{2\pi}} \tau^{\alpha - \frac{1}{2}} e^{-\beta\tau} e^{-\frac{\lambda\tau(x-\nu)^2}{2}} \tag{7.10}$$

where Γ is the gamma function. We choose standard values for the parameters: $\nu = 0$, $\lambda = 1$, $\alpha = 2$ and $\beta = 2$ – these parameters do not change much the solution, and were chosen so that user and item representations are not too much constrained. With this choice of parameters the mode of the mean is 0 and the mode of the variance is 1.

Loss function

Plugging Equations 7.9 and 7.1 gives[6]:

$$\begin{aligned}
\mathcal{L} &= \log p\left(\Theta\right) + \sum_{(u,i,j)\in D_S} \log p\left(i >_u j |\Theta\right) \\
&\stackrel{+}{=} \sum_{u,k} \left[\left(\frac{1}{2} - \alpha\right)\log\left(\sigma_{uk}\right) - \frac{2\beta + \lambda\mu_{uk}^2}{2\sigma_{uk}}\right] + \sum_{i,k} \left[\left(\frac{1}{2} - \alpha\right)\log\left(\sigma_{ik}\right) - \frac{2\beta + \lambda\mu_{ik}^2}{2\sigma_{ik}}\right] \\
&\quad - c\sum_k b_k^2 + \sum_{(u,i,j)\in\mathcal{D}} \log p\left(i >_u j |\Theta\right)
\end{aligned}$$

where $\bullet$ is a user or an item, $\stackrel{+}{=}$ means equal up to a constant, and $p\left(i >_u j|\Theta\right)$ is given by Equation 7.6. We also suppose that biases are distributed through a normal prior probability distribution (to avoid over-fitting), which results in an L2 regularization over biases. The parameters $\Theta = \{\Sigma_\bullet, \mu_\bullet\}_{\bullet\in\mathcal{U}\cup\mathcal{I}}$ are optimized with stochastic gradient update, by picking on a triple at each step and updating the correspond means and variances (for user u, item i and j). The experimental protocol is detailed in Section 7.2.2.

[6]Note that $\log NG(\mu_{\bullet k}, \sigma_{\bullet k}^{-1}) \stackrel{+}{=} \log\exp\left[(1/2-\alpha)\log(\sigma_{\bullet k}) - \beta/\sigma_{\bullet k} - \lambda(\mu_{\bullet k}-\nu)^2/(2\sigma_{\bullet k})\right]$

Dataset	Users			Items			Ratings		
	10	20	50	10	20	50	10	20	50
Yahoo!	3,386	1,645	286	1,000	1,000	995	158,868	99,915	32,759
MovieLens	743	643	448	1,336	1,330	1,307	94,491	91,053	80,643
Yelp	13,775	8,828	3,388	44,877	39,355	27,878	980,528	790,859	466,522

TABLE 7.1: Datasets statistics by truncation level (10, 20 or 50 ratings by user in the training set).

Ordering items

To evaluate, we need to produce a ranked list of items for each user. To do so, we could define a distribution probability over ranked list of items from our model using pairwise comparison of items. We could, for instance, consider that $i >_u j$ if $p(i >_u j|\Theta) > 0.5$, since it defines total ordering over items. However, this ranking does not preserve the variance information since $p(i >_u j|\Theta) > 0.5$ if and only if $(b_i - b_j - \mu_u^\top(\mu_j - \mu_i)) > 0$. So, to preserve the variance information of the representations when ordering items, we chose a simpler and more efficient way of computing a single total ordering. More precisely, we order items by their probability of having a positive score, i.e.:

$$s_{ui} = p(\mathbf{X}_u \cdot \mathbf{X}_i + b_i > 0) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{b_i - \mu_u^\top \mu_i}{\sqrt{2\sum_k \sigma_{ik}\left(\mu_{uk}^2 + \sigma_{uk}\right) + \sigma_{uk}\mu_{ik}^2}}\right)\right] \tag{7.11}$$

This ordering preserves the variance information – a score ($\mathbf{X}_u \cdot \mathbf{X}_i + b_i$) with a high variance will be associated with a score s_{ui} close to 0.5 (either greater if $\mu_i \cdot \mu_u + b_i > 0$ or smaller otherwise). Other ranking functions, based on Portfolio theory for instance, can be used.

7.2.2 Experiments

We experimented with three different datasets extracted respectively from the *Yahoo! Music ratings for User Selected and Randomly Selected songs, version 1.0* [7], the MovieLens 100k dataset [8] and the Yelp Dataset Challenge [9]. The **Yahoo!** dataset contains ratings for songs. The original rating data includes 15,400 users, and 1,000 songs for approximately 350,000 ratings. The **MovieLens** dataset contains ratings for movies. The original rating data includes 943 users, and 1,682 movies for approximately 100,000 ratings. The **Yelp** dataset contains ratings for businesses. The original rating data includes 1,029,432 users, and 144,072 businesses for approximately 4.1 million ratings.

We experimented with the following models, which are all based on a learning-to-rank criterion (except for MP):

(MP) Most popular ranking where items are ranked based on how often they have been seen in the past – while simple, this baseline is always competitive. There is no hyper-parameter to optimize for this procedure;

[7] This dataset is available through the Yahoo! Webscope data sharing program `http://webscope.sandbox.yahoo.com/`.

[8] Available at `http://grouplens.org/datasets/movielens/100k/`.

[9] See `https://www.yelp.com/dataset_challenge` for more information.

(SM) Soft margin (hinge) ranking loss, using stochastic gradient descent (Weimer et al., 2008) using the implementation from the library (Gantner et al., 2011) and optimizing all possible hyper-parameters;

(BPRMF) Bayesian Personalized Ranking (Rendle et al., 2009) on which our model cost function is based on. We use the implementation from (Gantner et al., 2011) optimizing all possible hyper-parameters;

(CofiRank) Collaborative Filtering Ranking (Weimer et al., 2007) is a state-of-the art ranking algorithm for recommender systems that optimizes an upper bound of nDCG[10]. We chose the hyper-parameters based on their paper (Weimer et al., 2007);

(GER) The model presented in this contribution, named Gaussian Embeddings Ranking. The number of dimensions was set among 5, 10, 20, 50, 100, 200, 500 and 1000.

We followed mostly the experimental procedure of (Weimer et al., 2007). In each case, the dataset has been preprocessed as follows. For each dataset a given amount (10, 20, or 50) of items are kept from each user for training the model (the truncation level), 10 for the validation set, and the remaining ratings were used for testing. Users with fewer than 30, 40 or 70 (depending on the amount of items kept for the training set) ratings were removed to ensure that we could evaluate on at least 10 ratings for each user. We also removed items that were not rated by at least 5 users. The statistics of the training datasets are shown in Table 7.1. The validation set is used to select the best hyper-parameters for GER, SM and BPRMF. Finally, for evaluation, we re-rank the judged items using the evaluated model, and use a metric for ranked lists of items, namely nDCG at ranks 1, 5 and 10. Several ranking measures have been proposed in the literature to evaluate the performance of a ranking function. The nDCG measure (Järvelin et al., 2002) is a popular measure used in information retrieval and learning to rank models, as for instance in (Weimer et al., 2007). We used nDCG rather than MAP since nDCG handles multiple levels of relevance, whereas MAP is designed for binary relevance. The reported results are the average of 5 different experiments, with 5 different train/validation/test splits.

Results are reported in Table 7.2. There are roughly three groups of models: (1) Soft Margin (SM) that performed worse; (2) BPRMF and most popular (MP); (3) GER and CofiRank (CR). Most popular (MP) performs well because of the chosen experimental evaluation where only items *seen* by the user are evaluated and ranked.

Our model (GER) outperforms the others on the MovieLens and Yelp dataset. It is above CofiRank (nDCG difference between 0.01 and 0.08), and is only below CofiRank on the Yahoo dataset for train sizes 20 and 50 (nDCG difference between 0.001 and 0.02). Overall, GER is performing very well on three datasets with different characteristics (in terms of rating behavior, and number of ratings). Finally, a dimension of 50 generally gives good results independently of the dataset for GER, and c (bias regularization) $\approx 10^{-5}$ give good results on MovieLens and Yelp, but needs to be higher ($\approx 10^{-3}$) on Yahoo to achieve good results.

[10] `https://github.com/markusweimer/cofirank`

Train Size →	10			20			50		
Model ↓	N@1	N@5	N@10	N@1	N@5	N@10	N@1	N@5	N@10
Yahoo!									
MP	53.0	59.1	67.3	52.5	58.3	66.4	53.6	57.8	64.0
BPRMF	52.8	59.0	67.2	52.2	58.3	66.4	52.2	57.7	63.5
SM	50.9	56.7	65.4	49.7	55.6	64.2	49.9	54.1	60.3
CR	**53.5**	**60.3**	68.2	**57.8**	**61.7**	**68.9**	**56.0**	**60.0**	**65.6**
GER	**53.5**	**60.3**	**68.3**	53.8	60.7	68.2	54.3	59.6	65.3
MovieLens									
MP	66.0	64.7	65.8	68.4	65.3	66.3	69.1	67.4	67.5
BPRMF	66.1	64.6	65.7	66.3	64.3	65.8	66.9	65.0	66.2
SM	55.9	57.5	60.3	58.3	59.6	61.6	58.6	60.4	62.5
CR	69.0	67.3	68.6	69.7	68.5	69.5	71.4	69.4	70.6
GER	**70.3**	**67.7**	**70.0**	**72.0**	**69.5**	**71.1**	**72.5**	**71.3**	**71.5**
Yelp									
MP	40.7	41.5	46.9	39.5	39.9	44.7	37.4	37.6	41.4
BPRMF	40.8	41.3	46.8	39.6	39.8	44.6	37.3	37.2	40.9
SM	37.3	38.3	44.4	35.8	36.9	41.9	33.4	34.1	38.0
CR	47.1	46.9	51.1	46.5	46.6	50.4	46.2	45.8	48.6
GER	**55.2**	**52.2**	**56.2**	**57.4**	**53.5**	**56.4**	**58.2**	**53.7**	**55.3**

TABLE 7.2: Collaborative Ranking results. nDCG values ($\times 100$) at different truncation levels are shown within the main columns, which are split based on the amount of training ratings.

Analysis

We now turn to the analysis of results, based on the study of the learned representations for the MovieLens dataset[11]. We used the hyper-parameters that were selected on the validation set, and looked at the set of mean-variance couples, i.e. the $(\mu_{\bullet k}, \sigma_{\bullet k})$ for $k \in \{1, \ldots, N\}$ and where $\bullet$ is a user or an item.

In figure 7.1, we plotted the different couples $(\mu_{\bullet 1}, \sigma_{\bullet 1})$, as well as the histogram of mean and variance for the MovieLens dataset. We can see that the model exploits the different ranges of mean (-0.3 to 0.3) and, more importantly, variance (0.4 to 1.6). We recall that we use a NormalGamma prior for $(\mu_{\bullet k}, \sigma_{\bullet k})$. Thus, according to the chosen parameters, the mode of the means should be near 0 and the mode of the variances near 1. The ranges exploited by the model are then around the priors. This was satisfying since it was not obvious that the variance component would be used by the model. The same holds for the other datasets, components and training sizes. Other examples are given in Figures 7.2 for users and 7.3 for items.

To have a deeper insight on the use of the variance, in Figures 7.4 and 7.5, we plotted violin (density) plots of the variance of users and items, for different latent space dimensions (5 to 1000), and training set sizes (i.e. truncation levels of 10, 20 and 50). We can observe that when the amount of training data increases, the median of the $\sigma_{\bullet k}$ decreases while the variance of the $\sigma_{\bullet k}$ increases. This corresponds to our hypotheses. The decrease of the median of the $\sigma_{\bullet k}$ shows that more evidence (i.e. more training examples) reduces the uncertainty

[11]The same holds for the other datasets.

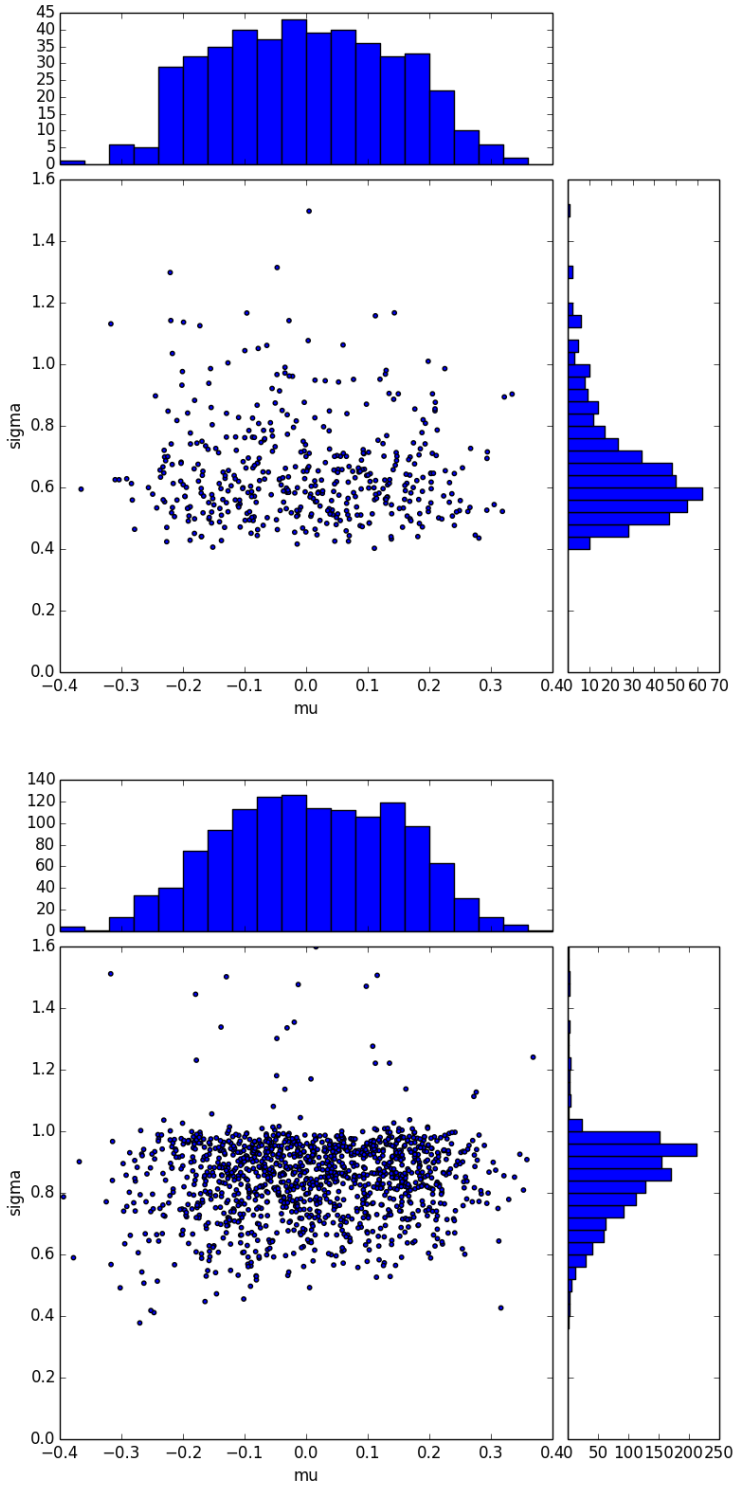


FIGURE 7.1: User (top) and item (bottom) learned first component representation plots ($\sigma_{\bullet 1}$ vs $\mu_{\bullet 1}$) for the training set with 50 items per user from the MovieLens dataset and a representation dimension of 50.

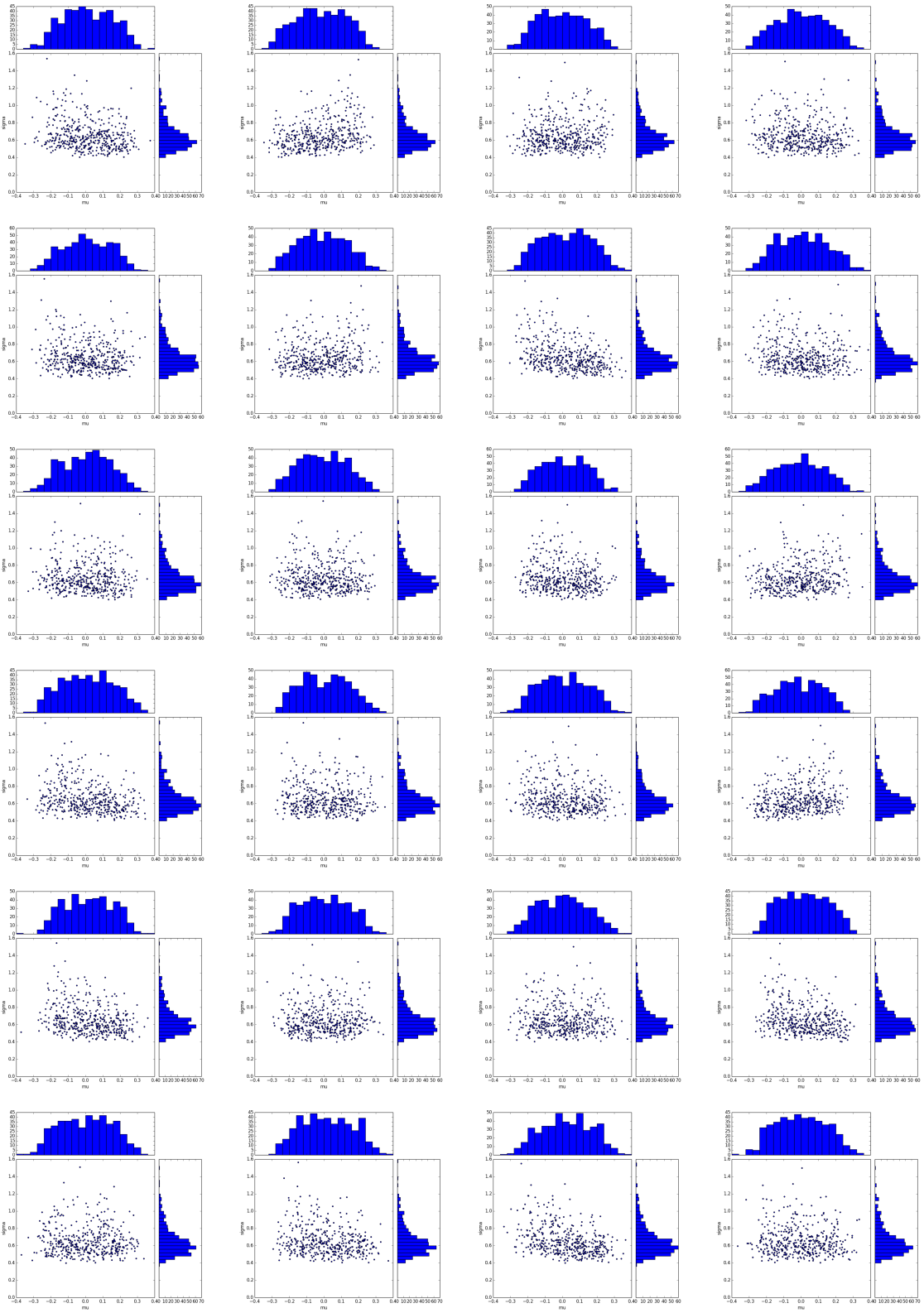


FIGURE 7.2: Subset of user learned components representation plots ($\sigma_{\bullet k}$ vs $\mu_{\bullet k}$ with different k) for the training set with 50 items per user from the MovieLens dataset and a representation dimension of 50.

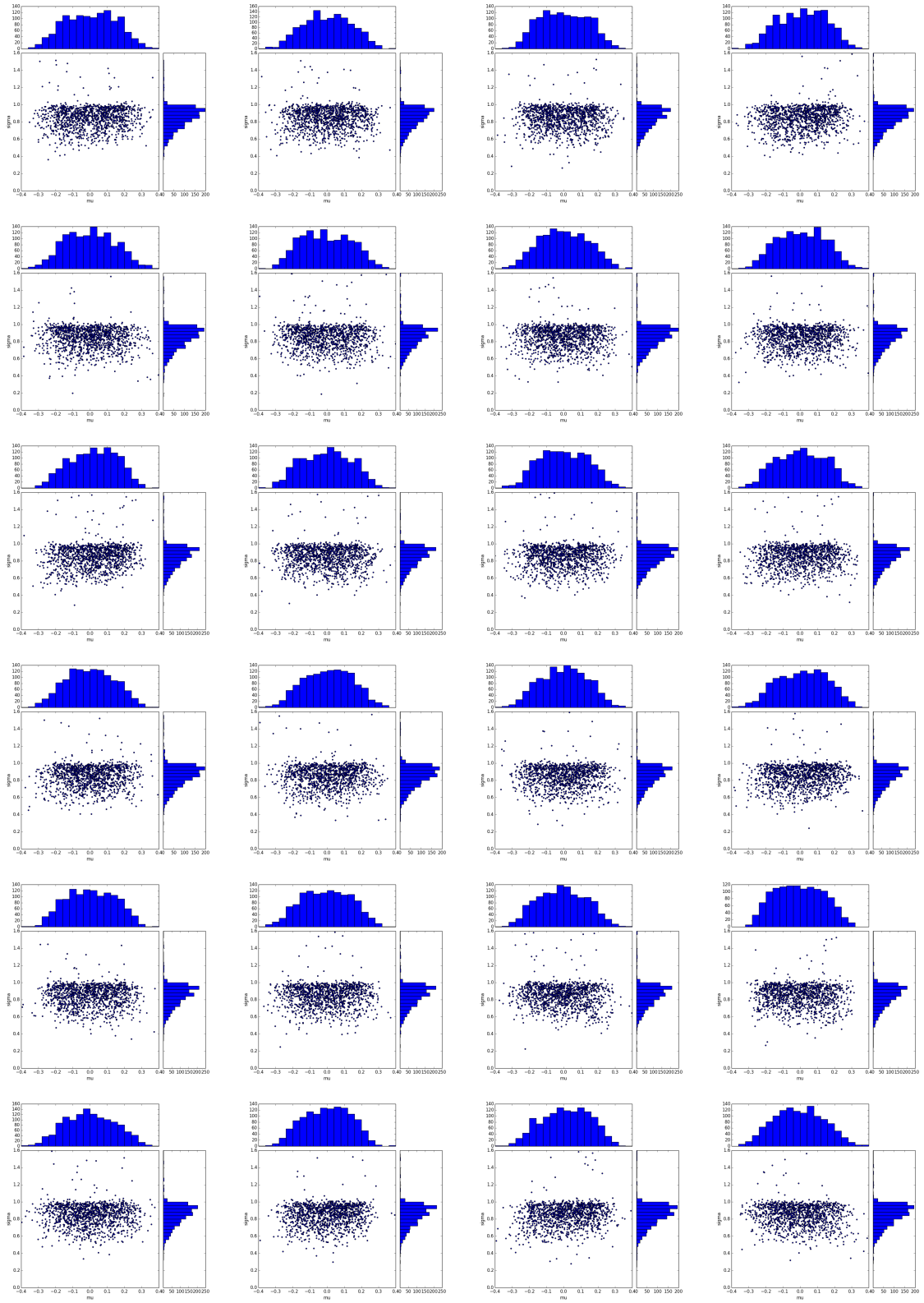


FIGURE 7.3: Subset of item learned components representation plots ($\sigma_{\bullet k}$ vs $\mu_{\bullet k}$ with different k) for the training set with 50 items per user from the MovieLens dataset and a representation dimension of 50.

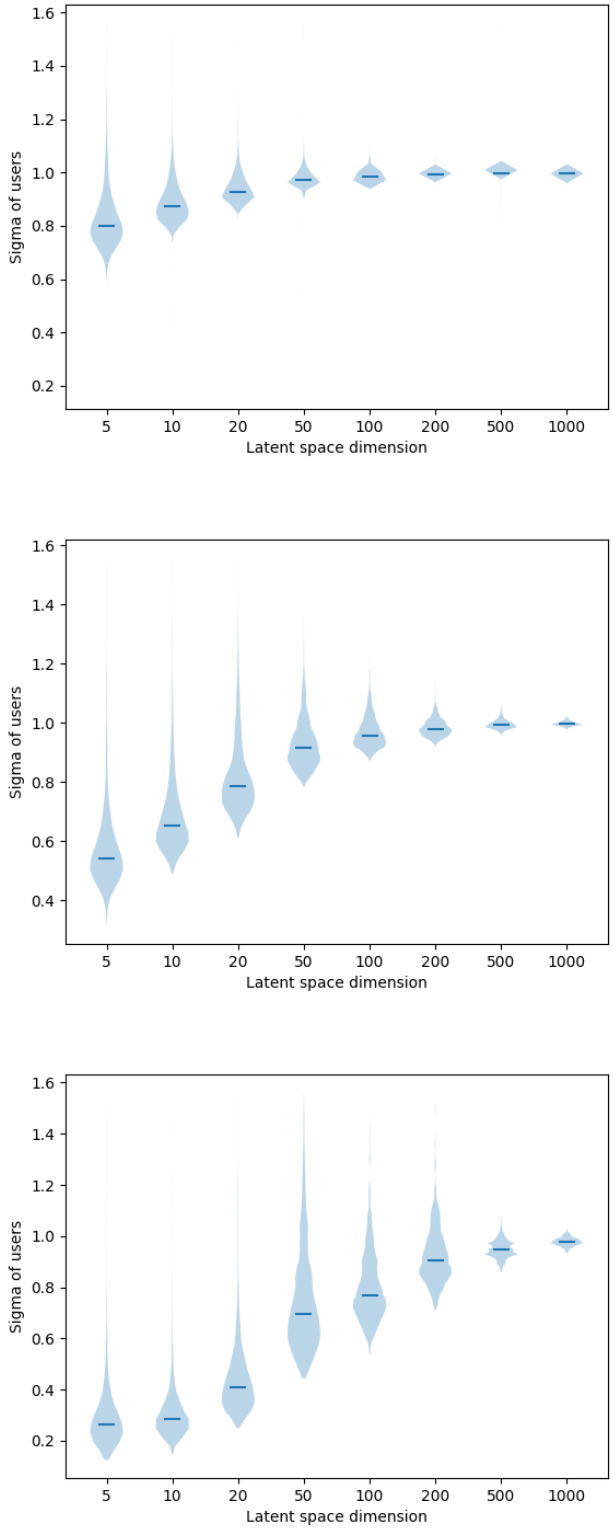


FIGURE 7.4: Violin (density) plots for the variance of users ($\sigma_{\bullet k}$) for different latent space dimensions on the MovieLens dataset. Three training set sizes settings (truncation levels) were used: 10 (top), 20 (middle) and 50 (bottom)

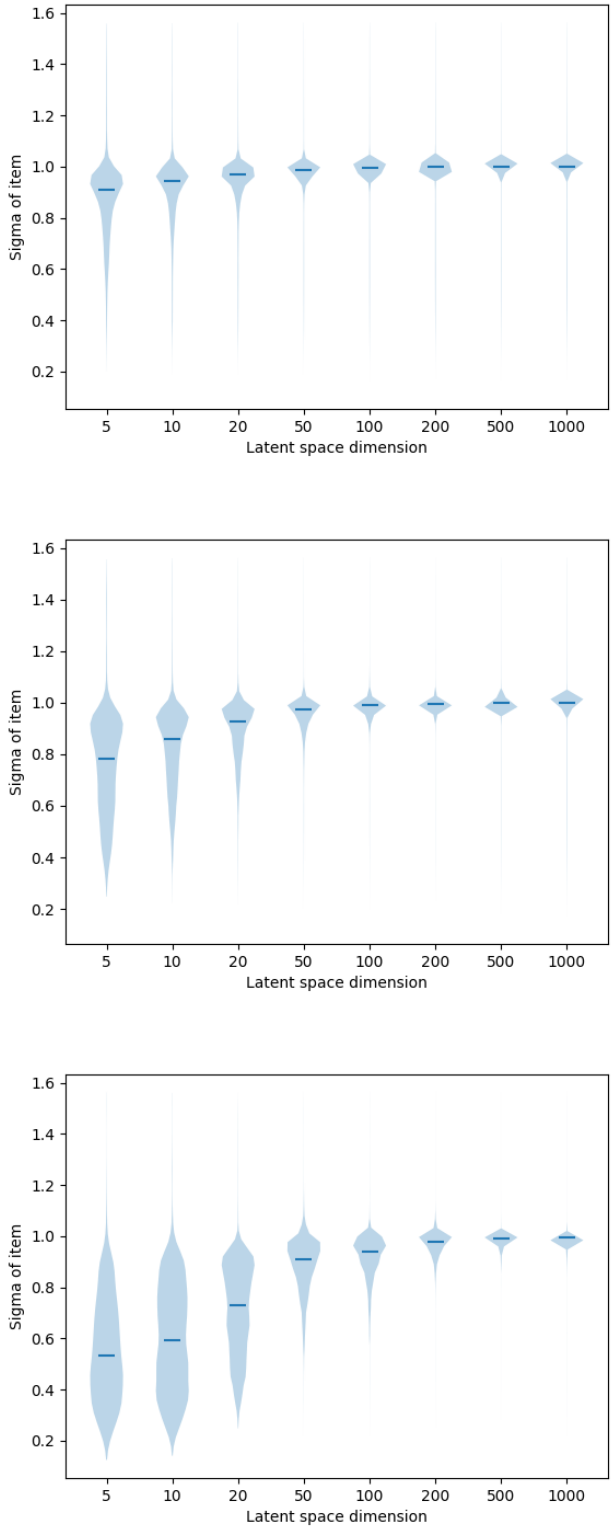


FIGURE 7.5: Violin (density) plots for the variance of items ($\sigma_{\bullet k}$) for different latent space dimensions on the MovieLens dataset. Three training set sizes settings (truncation levels) were used: 10 (top), 20 (middle) and 50 (bottom)

on the representation. The increase of the variance of the $\sigma_{\bullet k}$ can be the consequence that more training data induces more possible conflicting ratings and hence an increase in some case the variances. The latter is confirmed by the case of item representations that receive more ratings. Finally, we can observe that when the dimension increases too much, most of the density of the $\sigma_{\bullet k}$ is centered around the prior – which could imply that most of the dimensions were simply not used.

7.3 Conclusion

In this chapter, we presented a collaborative recommender system where user and items correspond to distributions in a latent space, rather than to a deterministic point for other representation-based models, allowing to cope with the inherent uncertainty due to lack of information, or to conflicting information. We show on three datasets that our model outperforms state-of-the-art works. We analyzed qualitatively the results showing that the variance distribution was correlated with space dimension and train size.

Our work is a first step in the direction of exploiting a social graph along with the ratings, and opens hence the following possible research directions:

- The integration of different sources of information (e.g. sellers, reviews), exploiting the uncertainty of each source;
- The use of the variance for handling time learning a dynamic function modeling its evolution through time (the representation becomes more and more uncertain if no new information is provided);
- The diversification of proposed rankings (e.g. Portfolio theory can be used to produce rankings with various risks).

Part III

Conclusion

Chapter 8

Conclusion et perspectives

Contents

8.1 Conclusion 121
8.1.1 Contributions 121
8.1.2 Learning Hyperparameters 122
8.1.3 From Deterministic to Gaussian Representations 122
8.2 Perspectives 123
8.2.1 Classification Task 123
8.2.2 Forecasting Task 123
8.2.3 Ranking Task 124
8.2.4 Learning Gaussian Embeddings for Knowledge Bases 124

8.1 Conclusion

In this thesis we have explored representation learning techniques in the context of relational data. We have studied three different tasks: graph node classification, relational time series forecasting and collaborative filtering. Whatever the studied task, the goal of our work was to make the best use of the information contained in the graph to improve the learned representations.

We first focused on learning deterministic representations for the classification task, and then explored a new framework consisting in exploiting Gaussian representations of entities, i.e. of nodes in a graph.

8.1.1 Contributions

Graph Node Classification

The classification task is the first task we were interested in. We proposed two different contributions for the challenging task of heterogeneous graph node classification.

In our first contribution (Chapter 4), we proposed a model that learns deterministic representations for each node. This is the only contribution where we use deterministic representation learning. Our proposed model also learns relation specific weights that determine how informative the relations between different types of nodes are. Experimental results show that this model is able to learn correlations between labels of different types

of nodes, thus notably improving the class label inference. We performed extensive experiments and showed that this model works well compared to competing methods.

In our second contribution (Chapter 5), we have adapted the previous model to learn Gaussian representations and have explored the use of uncertainty in this context. We have examined four variants of this model, by using either spherical and diagonal covariance matrices, and by using two different loss functions for classification. According to the results, introducing uncertainty in representations clearly improves results when comparing to its deterministic counterpart.

Relational Time Series Forecasting

In our third contribution (Chapter 6), we proposed to explore quantitative and qualitative results of learned Gaussian representations for the relational time series forecasting task.

We proposed a model for relational time series forecasting based on latent Gaussian embeddings. This model has shown competitive performance on four different datasets compared to state-of-the-art models. Using Gaussian representations has allowed us to model the uncertainty of predictions, and to provide confidence intervals for each prediction.

Collaborative Filtering

In our last contribution (Chapter 7), we have pursued the exploration of quantitative and qualitative results of learned Gaussian representations for a different task, namely collaborative filtering.

We have proposed a recommender system where users and items correspond to Gaussian representations. This has allowed us to cope with the inherent uncertainty due to lack of information or conflicting information. We have shown that our model outperforms state-of-the-art works. We have analyzed qualitatively the results showing that the variance distribution was influenced with space dimension and train size.

8.1.2 Learning Hyperparameters

In both Chapters 4 and 5, we used the framework of continuous optimization of hyperparameters (Bengio, 2000; Luketina et al., 2016). This framework learns regularization hyperparameters simultaneously to the model parameters. There is no formal proof that such procedures converge to an optimal choice of hyperparameters. We derived in these contributions our own version for the problems at hand. While continuous optimization of hyperparameters is still an open problem, since this method is unlikely to find the best set of hyperparameters, we have observed that this framework was key to find an approximate solution which performs well for our classification task, both with deterministic and non-deterministic embeddings. Handling graphs with a variety of relationships dictate the use of such techniques.

8.1.3 From Deterministic to Gaussian Representations

The main goal of this thesis was to apply representation learning techniques to relational data. Our first contribution explored the use of deterministic representation learning for

the classification task. Based on these results, we extended the model to use Gaussian representations, based on the intuition that it would help the model by allowing to explicitly model the uncertainty of representation of nodes – especially those isolated in the graph, or with conflicting information (e.g. belonging to different communities). We found that this modeling clearly improves the results, which encouraged us to explore different tasks, namely forecasting and ranking, to see if those advantages would still hold.

Integrating uncertainty in representations improved the results compared to state of the art models. Besides quantitative results and based on the learned Gaussian representations of the experiments of our last three contributions, our conclusions were that:

- The effect of using uncertainty has generally more impact when the amount of training data is lower;
- Highly central nodes seem to have less variance associated to their representation;

In our last three contributions we applied this method to represent different types of entities (e.g. nodes in a graph) into the space of Gaussian distributions, and learn the representations directly in that space. This allowed us to represent entities not as vectors, but as densities over a latent space, directly representing notions of uncertainty and enabling a richer geometry in the embedded space. We have shown the effectiveness of Gaussian representations compared to vector representation on various tasks, namely classification, forecasting and ranking.

8.2 Perspectives

Finally, we detail some of the perspectives open by this thesis, dealing with each task in turn.

8.2.1 Classification Task

The contributions presented in Chapters 4 and 5 use a simple modeling of the interactions between nodes. Besides the learned hyperparmeters, no distinction is made between the different types of relations that exist in heterogeneous networks.

Thus, modeling interactions in a more complex way, such as the ones presented in Chapter 2 for knowledge bases, could help take advantage of such diversity.

8.2.2 Forecasting Task

The contribution presented in Chapter 6 aims at applying the Gaussian representation framework for the forecasting task. We firstly choose to use it based on the model presented in (Ziat et al., 2016), which is a simple but effective model.

But more sophisticated models exist using the deterministic representation learning framework. For example, it would be interesting to adapt the model presented in (Ziat et al., 2017) which learns more complex interactions between time series. In this work, the prediction of a time series at time $T + 1$ uses the latent representations of all linked time series at time T. In this case, applying the Gaussian framework to this model should allow us to capture richer dynamics across multiple time series. This would allow studying how

the variances of the learn representations at time T will influence the variances of Gaussian representations at time $T + 1$.

8.2.3 Ranking Task

The contribution presented in Chapter 7 is a first step that aimed at assessing whether the Gaussian representation framework could be applied to recommendation in the context of relational data. Since we got promising results, proceeding with this work by proposing new models for relational data with various types of nodes (other than users and items) and integrating different sources of information (link between users or items, content information, etc.) is interesting.

8.2.4 Learning Gaussian Embeddings for Knowledge Bases

As shown in Section 2.3.2, lot of works have focused on learning deterministic representations for knowledge bases. To the best of our knowledge, the presented models have not been adapted to the Gaussian framework. (He et al., 2015) have proposed a model to learn Gaussian representation, the score they use for a triplet (s, r, o) is $\mathcal{S}(s, r, o) = D_{\mathrm{KL}}(Z_s - Z_o || Z_r)$. Thus, one can imagine adapting the deterministic models presented in Section 2.3.2, i.g. TransE (Bordes et al., 2013) with deterministic or Gaussian translations over Gaussian representations.

Bibliography

Abernethy, Jacob, Olivier Chapelle, and Carlos Castillo (2008). "WITCH: A New Approach to Web Spam Detection". In: *In Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*.

Airoldi, Edoardo, David Blei, Eric Xing, and Stephen Fienberg (2005). "A latent mixed membership model for relational data". In: *Proceedings of the 3rd international workshop on Link discovery*. ACM, pp. 82–89.

Angelova, Ralitsa and Gerhard Weikum (2006). "Graph-based text classification: learn from your neighbors". In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pp. 485–492.

Angelova, Ralitsa, Gjergji Kasneci, and Gerhard Weikum (2012). "Graffiti: graph-based classification in heterogeneous networks". In: *World Wide Web* 15.2, pp. 139–170.

Barbounis, TG, JB Theocharis, MC Alexiadis, and PS Dokopoulos (2006). "Long-term wind speed and power forecasting using local recurrent neural network models". In: *IEEE TEC*.

Bay, Herbert, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool (2008). "Speeded-up robust features (SURF)". In: *Computer vision and image understanding* 110.3, pp. 346–359.

Belkin, Mikhail, Partha Niyogi, and Vikas Sindhwani (2006). "Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples". In: *J. Mach. Learn. Res.* 7, pp. 2399–2434. ISSN: 1532-4435. URL: http://portal.acm.org/citation.cfm?id=1248547.1248632.

Belmouhcine, Abdelbadie and Mohammed Benkhalifa (2015). "Implicit Links based Web Page Representation for Web Page Classification". In: *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*. ACM, p. 12.

Ben Taieb, Souhaib and Rob Hyndman (2014). "Boosting multi-step autoregressive forecasts". In: *ICML*.

Bengio, Yoshua (2000). "Continuous optimization of hyper-parameters". In: *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*. Vol. 1. IEEE, pp. 305–310.

— (2012). "Deep learning of representations for unsupervised and transfer learning". In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pp. 17–36.

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Jauvin (2003). "A neural probabilistic language model". In: *Journal of machine learning research* 3.Feb, pp. 1137–1155.

Bengio, Yoshua, Olivier Delalleau, and Nicolas Le Roux (2006). "Label propagation and quadratic criterion". In: *Semi-supervised learning* 10.

Bengio, Yoshua, Aaron Courville, and Pierre Vincent (2013). "Representation learning: A review and new perspectives". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35.8, pp. 1798–1828.

Bhagat, Smriti, Graham Cormode, and S Muthukrishnan (2011). "Node classification in social networks". In: *Social network data analytics*. Springer, pp. 115–148.

Blei, David M, Alp Kucukelbir, and Jon D McAuliffe (2017). "Variational inference: A review for statisticians". In: *Journal of the American Statistical Association* just-accepted.

Bojchevski, A. and S. Günnemann (2017). "Deep Gaussian Embedding of Attributed Graphs: Unsupervised Inductive Learning via Ranking". In: *ArXiv e-prints*. arXiv: 1707.03815 [stat.ML].

Bordes, Antoine, Jason Weston, Ronan Collobert, and Yoshua Bengio (2011). "Learning structured embeddings of knowledge bases". In: *Conference on Artificial Intelligence*.

Bordes, Antoine, Xavier Glorot, Jason Weston, and Yoshua Bengio (2012). "Joint learning of words and meaning representations for open-text semantic parsing". In: *Artificial Intelligence and Statistics*, pp. 127–135.

Bordes, Antoine, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko (2013). "Translating embeddings for modeling multi-relational data". In: *Advances in Neural Information Processing Systems*, pp. 2787–2795.

Bordes, Antoine, Xavier Glorot, Jason Weston, and Yoshua Bengio (2014). "A semantic matching energy function for learning with multi-relational data". In: *Machine Learning* 94.2, pp. 233–259.

Boulanger-Lewandowski, Nicolas, Yoshua Bengio, and Pascal Vincent (2011). "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription". In: *ISMIR*.

Bourigault, Simon, Cedric Lagnier, Sylvain Lamprier, Ludovic Denoyer, and Patrick Gallinari (2014). "Learning social network embeddings for predicting information diffusion". In: *Proceedings of the 7th ACM international conference on Web search and data mining*, pp. 393–402.

Brahim-Belhouari, Sofiane and Amine Bermak (2004). "Gaussian process for nonstationary time series prediction". English. In: *Computational Statistics & Data Analysis* 47.4, pp. 705–712. DOI: 10.1016/j.csda.2004.02.006. URL: http://linkinghub.elsevier.com/retrieve/pii/S0167947304000301.

Brown, Gerald G and Herbert C Rutemiller (1977). "Means and Variances of Stochastic Vector Products with Applications to Random Linear Models". In: *Management Science* 24.2.

Cao, Qing, Bradley T Ewing, and Mark A Thompson (2012). "Forecasting wind speed with recurrent neural networks". In: *European Journal of Operational Research* 221.1, pp. 148–154.

Cao, Shaosheng, Wei Lu, and Qiongkai Xu (2015). "GraRep: Learning Graph Representations with Global Structural Information". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, pp. 891–900.

Chapelle, Olivier, Bernhard Scholkopf, and Alexander Zien (2009). "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]". In: *IEEE Transactions on Neural Networks* 20.3, pp. 542–542.

Che, Zhengping, David Kale, Wenzhe Li, Mohammad Taha Bahadori, and Yan Liu (2015). "Deep computational phenotyping". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 507–516.

Chen, Minmin, Zhixiang Xu, Kilian Weinberger, and Fei Sha (2012a). "Marginalized denoising autoencoders for domain adaptation". In: *arXiv preprint arXiv:1206.4683*.

Chen, Shuo, Josh L Moore, Douglas Turnbull, and Thorsten Joachims (2012b). "Playlist prediction via metric embedding". In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 714–722.

Cho, K, B Van Merriënboer, C Gulcehre, D Bahdanau, F Bougares, H Schwenk, and Y Bengio (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *EMNLP*.

Chung, Junyoung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C. Courville, and Yoshua Bengio (2015). "A Recurrent Latent Variable Model for Sequential Data". In: *NIPS*.

Ciregan, Dan, Ueli Meier, and Jürgen Schmidhuber (2012). "Multi-column deep neural networks for image classification". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, pp. 3642–3649.

Collobert, Ronan, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa (2011). "Natural language processing (almost) from scratch". In: *Journal of Machine Learning Research* 12.Aug, pp. 2493–2537.

Connor, Jerome T, R Douglas Martin, and Les E Atlas (1994). "Recurrent neural networks and robust time series prediction". In: *Neural Networks, IEEE Transactions on*.

Cressie, Noel A. C. and Christopher K. Wikle (2011). *Statistics for spatio-temporal data*. Wiley series in probability and statistics. Hoboken, N.J. Wiley. ISBN: 978-0-471-69274-4.

Dahl, George, Abdel-rahman Mohamed, Geoffrey E Hinton, et al. (2010). "Phone recognition with the mean-covariance restricted Boltzmann machine". In: *Advances in neural information processing systems*, pp. 469–477.

Dahl, George E, Dong Yu, Li Deng, and Alex Acero (2012). "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition". In: *IEEE Transactions on audio, speech, and language processing* 20.1, pp. 30–42.

Dauphin, Grégoire Mesnil Yann et al. (2012). "Unsupervised and transfer learning challenge: a deep learning approach". In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pp. 97–110.

De Gooijer, Jan G and Rob J Hyndman (2006). "25 years of time series forecasting". In: *International journal of forecasting*.

Deng, Li, Michael L Seltzer, Dong Yu, Alex Acero, Abdel-rahman Mohamed, and Geoff Hinton (2010). "Binary coding of speech spectrograms using a deep auto-encoder". In: *Eleventh Annual Conference of the International Speech Communication Association*.

Denoyer, Ludovic and Patrick Gallinari (2010). "A Ranking Based Model for Automatic Image Annotation in a Social Network". In: *ICWSM*.

Devooght, Robin, Amin Mantrach, Ilkka Kivimäki, Hugues Bersini, Alejandro Jaimes, and Marco Saerens (2014). "Random walks based modularity: application to semi-supervised learning". In: *Proceedings of the 23rd international conference on World wide web*. ACM, pp. 213–224.

Duan, Yajuan, Furu Wei, Ming Zhou, and Heung-Yeung Shum (2012). "Graph-based collective classification for tweets". In: *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, pp. 2323–2326.

Fraccaro, Marco, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther (2016). "Sequential Neural Models with Stochastic Layers". In: *Advances in neural information processing systems 2016*.

Freeman, William T and Michal Roth (1995). "Orientation histograms for hand gesture recognition". In: *International workshop on automatic face and gesture recognition*. Vol. 12, pp. 296–301.

Gallagher, Brian, Hanghang Tong, Tina Eliassi-Rad, and Christos Faloutsos (2008). "Using ghost edges for classification in sparsely labeled networks". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 256–264.

Gammerman, Alexander, Volodya Vovk, and Vladimir Vapnik (1998). "Learning by transduction". In: *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., pp. 148–155.

Ganeshapillai, Gartheeban, John Guttag, and Andrew Lo (2013). "Learning connections in financial time series". In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 109–117.

Gantner, Zeno, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme (2011). "MyMediaLite: A Free Recommender System Library". In: *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys 2011)*.

Getoor, Lise and Christopher P Diehl (2005). "Link mining: a survey". In: *ACM SIGKDD Explorations Newsletter* 7.2, pp. 3–12.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (2011). "Domain adaptation for large-scale sentiment classification: A deep learning approach". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 513–520.

Graves, Alan, Abdel-rahman Mohamed, and Geoffrey Hinton (2013). "Speech recognition with deep recurrent neural networks". In: *IIIE ICASSP*.

Grover, Aditya and Jure Leskovec (2016). "Node2Vec: Scalable Feature Learning for Networks". In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864. ISBN: 978-1-4503-4232-2.

Gu, Quanquan, Charu Aggarwal, Jialu Liu, and Jiawei Han (2013). "Selective sampling on graphs for classification". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 131–139.

Hachino, Tomohiro and Visakan Kadirkamanathan (2011). "Multiple Gaussian process models for direct time series forecasting". In: *IEEJ Transactions on Electrical and Electronic Engineering* 6.3, pp. 245–252. DOI: 10.1002/tee.20651.

Hamel, Philippe, Simon Lemieux, Yoshua Bengio, and Douglas Eck (2011). "Temporal Pooling and Multiscale Learning for Automatic Annotation and Ranking of Music Audio." In: *ISMIR*, pp. 729–734.

He, Shizhu, Kang Liu, Guoliang Ji, and Jun Zhao (2015). "Learning to Represent Knowledge Graphs with Gaussian Embedding". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, pp. 623–632.

Hermans, Michiel and Benjamin Schrauwen (2013). "Training and analysing deep recurrent neural networks". In: *Advances in Neural Information Processing Systems*, pp. 190–198.

Hinton, Geoffrey et al. (2012). "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups". In: *IEEE Signal Processing Magazine* 29.6, pp. 82–97.

Hsieh, TJ, HF Hsiao, and WC Yeh (2011). "Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm". In: *Applied soft computing*.

Hwang, Taehyun and Rui Kuang (2010). "A heterogeneous label propagation algorithm for disease gene discovery". In: *SDM*, p. 12. URL: http://www.siam.org/proceedings/datamining/2010/dm10_051_hwangt.pdf.

Jacob, Yann, Ludovic Denoyer, and Patrick Gallinari (2014). "Learning latent representations of nodes for classifying in heterogeneous social networks". In: *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, pp. 373–382.

Järvelin, Kalervo and Jaana Kekäläinen (2002). "Cumulated gain-based evaluation of IR techniques". In: *ACM Transactions on Information Systems (TOIS)* 20.4, pp. 422–446.

Jenatton, Rodolphe, Nicolas L Roux, Antoine Bordes, and Guillaume R Obozinski (2012). "A latent factor model for highly multi-relational data". In: *Advances in Neural Information Processing Systems*, pp. 3167–3175.

Jensen, David, Jennifer Neville, and Brian Gallagher (2004). "Why collective inference improves relational classification". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 593–598.

Ji, Ming, Yizhou Sun, Marina Danilevsky, Jiawei Han, and Jing Gao (2010). "Graph regularized transductive classification on heterogeneous information networks". In: *European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 570–586.

Jordan, Michael I, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul (1999). "An introduction to variational methods for graphical models". In: *Machine learning* 37.2, pp. 183–233.

Kalman, Rudolph Emil (1960). "A New Approach to Linear Filtering and Prediction Problems". In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D, pp. 35–45.

Karl, Maximilian, Maximilian Sölch, Justin Bayer, and Patrick van der Smagt (2016). "Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data". In: *CoRR* abs/1605.06432.

Kim, Yong-Deok and Seungjin Choi (2014). "Bayesian binomial mixture model for collaborative prediction with non-random missing data". In: *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, pp. 201–208.

Kingma, Diederik and Jimmy Ba (2015). "Adam: A method for stochastic optimization". In: *ICLR*.

Kingma, DP and M Welling (2014). "Auto-Encoding Variational Bayes". In: *ICLR*.

Kolda, Tamara G and Brett W Bader (2009). "Tensor decompositions and applications". In: *SIAM review* 51.3, pp. 455–500.

Koren, Yehuda, Robert Bell, and Chris Volinsky (2009). "Matrix factorization techniques for recommender systems". In: *Computer* 42.8.

Krishnan, Rahul G, Uri Shalit, and David Sontag (2015). "Deep Kalman Filters". In: *NIPS 2015 Workshop*.

Krishnan, Rahul G., Uri Shalit, and David Sontag (2017). "Structured Inference Networks for Nonlinear State Space Models". In: *Proceedings of the 31 AAAI Conference on Artificial Intelligence*.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Lowe, David G (1999). "Object recognition from local scale-invariant features". In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Vol. 2. Ieee, pp. 1150–1157.

Lu, Qing and Lise Getoor (2003). "Link-based classification". In: *ICML*. Vol. 3, pp. 496–503.

Luketina, Jelena, Tapani Raiko, Mathias Berglund, and Klaus Greff (2016). "Scalable Gradient-Based Tuning of Continuous Regularization Hyperparameters". In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 2952–2960.

MacQueen, James et al. (1967). "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA., pp. 281–297.

Mann, Gideon S and Andrew McCallum (2010). "Generalized expectation criteria for semi-supervised learning with weakly labeled data". In: *The Journal of Machine Learning Research* 11, pp. 955–984.

Markowitz, Harry (1952). "Portfolio selection". In: *The journal of finance* 7.1, pp. 77–91.

McDowell, Luke and David Aha (2012). "Semi-Supervised Collective Classification via Hybrid Label Regularization". In: *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*. Ed. by John Langford and Joelle Pineau. ICML '12. New York, NY, USA: Omnipress, pp. 975–982. ISBN: 978-1-4503-1285-1.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013). "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*, pp. 3111–3119.

Mirowski, Piotr and Yann LeCun (2009). "Dynamic factor graphs for time series modeling". In: *Machine Learning and Knowledge Discovery in Databases*. Springer.

Mnih, Andriy and Geoffrey E Hinton (2009). "A scalable hierarchical distributed language model". In: *Advances in neural information processing systems*, pp. 1081–1088.

Mohamed, Abdel-rahman, George E Dahl, and Geoffrey Hinton (2012). "Acoustic modeling using deep belief networks". In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.1, pp. 14–22.

Monner, Derek D and James A Reggia (2013). "Recurrent Neural Collective Classification". In: *Neural Networks and Learning Systems, IEEE Transactions on* 24.12, pp. 1932–1943.

Mukherjee, Tanmoy and Timothy M Hospedales (2016). "Gaussian Visual-Linguistic Embedding for Zero-Shot Recognition." In: *EMNLP*, pp. 912–918.

Muller, KR, A J Smola, G Ratsch, B Scholkopf, J Kohlmorgen, and V Vapnik (1999). "Using support vector machines for time series prediction". In: *Kernel methods—support vector learning*.

Myaeng, Sung Hyon and Mann-ho Lee (2000). "A practical hypertext categorization method using links and incrementally available class information". In: *In Proceedings of SIGIR00, 23rd ACM International Conference on Research and Development in Information Retrieval (Athens, GR*. Citeseer.

Namata, Galileo Mark, Ben London, and Lise Getoor (2016). "Collective Graph Identification". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 10.3, p. 25.

Nandanwar, Sharad and M. N. Murty (2016). "Structural Neighborhood Based Classification of Nodes in a Network". In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. New York, NY, USA: ACM,

pp. 1085–1094. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939782. URL: http://doi.acm.org/10.1145/2939672.2939782.

Neville, Jennifer and David Jensen (2000). "Iterative classification in relational data". In: *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pp. 13–20.

Nickel, Maximilian, Volker Tresp, and Hans-Peter Kriegel (2011). "A three-way model for collective learning on multi-relational data". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 809–816.

Nie, Lan, Brian D Davison, and Xiaoguang Qi (2006). "Topical link analysis for web search". In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pp. 91–98.

Niepert, Mathias, Mohamed Ahmed, and Konstantin Kutzkov (2016). "Learning Convolutional Neural Networks for Graphs". In: *ICML 2016*.

Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd (1999). *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Previous number = SIDL-WP-1999-0120. Stanford InfoLab. URL: http://ilpubs.stanford.edu:8090/422/.

Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena (2014). "Deepwalk: Online learning of social representations". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 701–710.

Peters, Stephane, Ludovic Denoyer, and Patrick Gallinari (2010). "Iterative annotation of multi-relational social networks". In: *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*. IEEE, pp. 96–103.

Pfeiffer III, Joseph J, Jennifer Neville, and Paul N Bennett (2015). "Overcoming Relational Learning Biases to Accurately Predict Preferences in Large Scale Networks". In: *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pp. 853–863.

Pimplikar, Rakesh, Dinesh Garg, Deepesh Bharani, and Gyana Parija (2014). "Learning to Propagate Rare Labels". In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, pp. 201–210.

Rendle, Steffen, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme (2009). "BPR: Bayesian personalized ranking from implicit feedback". In: *Uncertainty in Artificial Intelligence*. AUAI Press, pp. 452–461. ISBN: 978-0-9749039-5-8.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic backpropagation and variational inference in deep latent Gaussian models". In: *International Conference on Machine Learning*.

Ricci, Francesco, Lior Rokach, Bracha Shapira, and Kantor Paul B. (2011). *Recommender Systems Handbook*. Vol. 532. ISBN: 978-0-387-85819-7. DOI: 10.1007/978-0-387-85820-3. arXiv: arXiv:1011.1669v3.

Riedel, Sebastian, Limin Yao, Andrew McCallum, and Benjamin M Marlin (2013). "Relation Extraction with Matrix Factorization and Universal Schemas." In: *HLT-NAACL*, pp. 74–84.

Rifai, Salah, Yann N Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller (2011). "The manifold tangent classifier". In: *Advances in Neural Information Processing Systems*, pp. 2294–2302.

Rudolph, Maja, Francisco Ruiz, Stephan Mandt, and David Blei (2016). "Exponential family embeddings". In: *Advances in Neural Information Processing Systems*, pp. 478–486.

Sacca, Claudio, Michelangelo Diligenti, and Marco Gori (2013). "Collective Classification Using Semantic Based Regularization". In: *Machine Learning and Applications (ICMLA), 2013 12th International Conference on*. Vol. 1. IEEE, pp. 283–286.

Salakhutdinov, R and A Mnih (2007). "Probabilistic Matrix Factorization". In: *Proceedings of Advances in Neural Information Processing Systems*. Vol. 20, pp. 1257–1264.

Seide, Frank, Gang Li, and Dong Yu (2011). "Conversational speech transcription using context-dependent deep neural networks". In: *Twelfth Annual Conference of the International Speech Communication Association*.

Sen, Prithviraj, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad (2008). "Collective Classification in Network Data". In: *AI Magazine* 29.3, pp. 93–106.

Socher, Richard, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng (2011). "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection". In: *Advances in Neural Information Processing Systems*, pp. 801–809.

Socher, Richard, Danqi Chen, Christopher D Manning, and Andrew Ng (2013). "Reasoning with neural tensor networks for knowledge base completion". In: *Advances in Neural Information Processing Systems*, pp. 926–934.

Srivastava, Nitish and Ruslan R Salakhutdinov (2012). "Multimodal learning with deep boltzmann machines". In: *Advances in neural information processing systems*, pp. 2222–2230.

Stern, David H, Ralf Herbrich, and Thore Graepel (2009). "Matchbox: large scale online bayesian recommendations". In: *Proceedings of the 18th international conference on World wide web*. ACM, pp. 111–120.

Sun, Yizhou, Yintao Yu, and Jiawei Han (2009). "Ranking-based clustering of heterogeneous information networks with star network schema". In: *KDD*, pp. 797–806.

Sutskever, Ilya, James Martens, and Geoffrey E. Hinton (2011). "Generating Text with Recurrent Neural Networks". In: *Proceedings of ICML*.

Tang, Jian, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei (2015). "LINE: Large-scale Information Network Embedding". In: *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pp. 1067–1077.

Vilnis, Luke and Andrew McCallum (2015). "Word representations via gaussian embedding". In: *ICLR*.

Vu, Thuy and D Stott Parker (2015). "Node Embeddings in Social Network Analysis". In: *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*. ACM, pp. 326–329.

Wang, Jun and Jianhan Zhu (2009). "Portfolio theory of information retrieval". In: ACM Press. DOI: 10.1145/1571941.1571963.

Wang, Xi and Gita Sukthankar (2013). "Multi-label relational neighbor classification using social context features". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 464–472.

Weimer, Markus, Alexandros Karatzoglou, Quoc Viet Le, and Alex Smola (2007). "CofiRank Maximum Margin Matrix Factorization for Collaborative Ranking". In: *Advances in Neural Information Processing Systems*, pp. 1–3.

Weimer, Markus, Alexandros Karatzoglou, and Alex Smola (2008). "Improving maximum margin matrix factorization". In: *Machine Learning* 72.3, pp. 263–276.

Weston, Jason, Samy Bengio, and Nicolas Usunier (2010). "Large scale image annotation: learning to rank with joint word-image embeddings". In: *Machine learning* 81.1, pp. 21–35.

— (2011). "Wsabie: Scaling up to large vocabulary image annotation". In: *IJCAI*. Vol. 11, pp. 2764–2770.

Wikle, Christopher K. (2015). "Modern perspectives on statistics for spatio-temporal data". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 7.1, pp. 86–98.

Wikle, Christopher K and Mevin B Hooten (2010). "A general science-based framework for dynamical spatio-temporal models". In: *Test* 19.3, pp. 417–451.

Winn, John and Christopher M Bishop (2005). "Variational message passing". In: *Journal of Machine Learning Research* 6.Apr, pp. 661–694.

Wu, Zuxuan, Xi Wang, Yu-Gang Jiang, Hao Ye, and Xiangyang Xue (2015). "Modeling spatial-temporal clues in a hybrid deep learning framework for video classification". In: *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*. ACM, pp. 461–470.

Xiang, Rongjing and Jennifer Neville (2013). "Collective inference for network data with copula latent markov networks". In: *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, pp. 647–656.

Yang, Zhilin, William Cohen, and Ruslan Salakhutdinov (2016). "Revisiting Semi-Supervised Learning with Graph Embeddings". In: *ICML 2016*.

Ye, Junting and Leman Akoglu (2015). "Robust Semi-Supervised Classification for Multi-Relational Graphs". In: *arXiv preprint arXiv:1510.06024*.

Zhang, Fuzheng, Nicholas Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma (2016). "Collaborative Knowledge Base Embedding for Recommender Systems." In: *KDD*. ACM Press, pp. 353–362.

Zhang, Junbo, Guangjian Tian, Yadong Mu, and Wei Fan (2014). "Supervised deep learning with auxiliary networks". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 353–361.

Zheng, Q and DB Skillicorn (2015). "Spectral Embedding of Signed Networks". In: SIAM.

Zhou, Dengyong, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf (2003). "Learning with Local and Global Consistency". In: *Proceedings of the 16th International Conference on Neural Information Processing Systems*. NIPS'03. Cambridge, MA, USA: MIT Press, pp. 321–328. URL: http://dl.acm.org/citation.cfm?id=2981345.2981386.

Zhou, Dengyong, Jiayuan Huang, and Bernhard Schölkopf (2005). "Learning from labeled and unlabeled data on a directed graph". In: *Proc. of the 22nd intern. conf. on Machine learning*. ICML '05. New York, NY, USA: ACM, pp. 1036–1043. ISBN: 1-59593-180-5. DOI: http://doi.acm.org/10.1145/1102351.1102482. URL: http://doi.acm.org/10.1145/1102351.1102482.

Zhou, Sijun, Shizhou Zhang, and Jinjun Wang (2015). "Deep sparse coding network for image classification". In: *Proceedings of the 7th International Conference on Internet Multimedia Computing and Service*. ACM, p. 24.

Zhou, Yang and Ling Liu (2014). "Activity-edge centric multi-label classification for mining heterogeneous information networks". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 1276–1285.

Zhu, Xiaojin and Zoubin Ghahramani (2002). *Learning from labeled and unlabeled data with label propagation*. Tech. rep. Citeseer.

Ziat, Ali, Gabriella Contardo, Nicolas Baskiotis, and Ludovic Denoyer (2016). "Learning Embeddings for Completion and Prediction of Relationnal Multivariate Time-Series". In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning-ESANN*.

Ziat, Ali, Edouard Delasalles, Ludovic Denoyer, and Patrick Gallinari (2017). "Spatio Temporal Neural Networks for Space-Time Series Forecasting and Relations Discovery". In: *Data Mining, (ICDM), 2017 IEEE 17th International Conference on*. IEEE.