HAL open science

Representation Learning for Large Scale Classification

Thomas Gerald

▶ To cite this version:

Thomas Gerald. Representation Learning for Large Scale Classification. Machine Learning [cs.LG]. Sorbonne Université, 2020. English. NNT : 2020SORUS316 . tel-03987588

HAL Id: tel-03987588

https://theses.hal.science/tel-03987588

Submitted on 14 Feb 2023

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.

THÈSE DE DOCTORAT DE
SORBONNE UNIVERSITÉ

Spécialité
Informatique
École Doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Thomas Gerald

Pour obtenir le grade de
DOCTEUR de SORBONNE UNIVERSITÉ

Sujet de la thèse :

Representation Learning for Large Scale Classification

Apprentissage de représentation pour la classification large échelle

devant le jury composé de :

M. Patrick Gallinari	Sorbonne Université – LIP6	Directeur
M. Nicolas Baskiotis	Sorbonne Université – LIP6	Superviseur
M. Massih-Reza Amini	Université Grenoble Alpes – LIG	Rapporteur
Mme. Pascale Kuntz-Cosperec	Polytech'Nantes – LS2N	Rapporteur
M. Julien Tierny	Sorbonne Université – LIP6	Examinateur
Mme. xiangliang zhang	KAUST – CEMSE	Examinateur

Thomas Gerald: *Representation Learning for Large Scale Classification*, © 2020

Remerciements

Je voudrais premièrement remercier mon encadrant, Mr Nicolas Baskiotis pour les conseils prodigués, les discussions et les relectures. Je souhaiterais aussi adresser mes remerciements à Mr Patrick Gallinari mon directeur de thèse. Aussi, j'aimerais remercier Mr Hatem Hajri et Mr Hadi Zatiti, qui m'ont proposé de collaborer sur des articles d'apprentissage de représentation dans des espaces hyperboliques (chapitre 4 de cette thèse).

Pour nos débats et leurs soutiens, je souhaiterais remercier les trois personnes avec qui j'ai partagé mon bureau ces quatre dernières années: Mickaël Chen, Édouard Delasalles et Artuhur Pajot ainsi que toute l'équipe MLIA.

Dans le cadre privé, j'aimerais remercier ma conjointe Adène, ma famille proche, Bertrand, Claire, Joséphine et Jeanne et mes amis Hugo, Alexis, Evrim ainsi que tous les autres m'ayant apporté leurs supports.

CONTENTS

CONTENTS v
LIST OF FIGURES vii
LIST OF TABLES xi
ACRONYMS xiii
1 INTRODUCTION 1
1.1 Machine learning and Big data . . . 1
1.2 Extreme Classification . . . 2
1.3 Contributions . . . 4
1.4 Structure of the document . . . 6
2 EXTREME CLASSIFICATION PREVIOUS AND CURRENT APPROACHS 9
2.1 The large scale classification task . . . 10
2.1.1 Notations . . . 10
2.1.2 Corpus . . . 10
2.1.3 Discussion on distribution and corpora size . . . 11
2.1.4 Evaluation metrics . . . 12
2.2 Adressing classification challenges . . . 14
2.2.1 Early multi-class classification . . . 15
2.2.2 Classification and current applications . . . 18
2.2.3 Efficient time complexity approaches for XML . . . 19
2.3 Learning embeddings for eXtreme classification . . . 31
2.3.1 Faster proccessing and discret representations . . . 31
2.3.2 XML based continuous representation . . . 33
2.3.3 Representation learning and hyperbolic space . . . 37
2.4 An uncompleted challenge . . . 41
3 DSNC AND ATOMS NETWORK : LEARNING CODES 43
3.1 DSNC : learning binary codes for fast classification . . . 44
3.1.1 Proposed approach . . . 46
3.1.2 Deep Stochastic Neural Code : Encoding . . . 47
3.1.3 Deep Stochastic Neural Code : Decoder . . . 48
3.1.4 Learning procedure . . . 50
3.1.5 Protocol and experiments . . . 52
3.1.6 DSNC: Discussion . . . 57
3.2 Atoms Network : Learning class code-words . . . 57
3.2.1 Model Description . . . 58
3.2.2 Results and Analysis . . . 62
3.2.3 Constraint analysis . . . 63
3.2.4 Atoms Network : Discussion . . . 64
3.3 Better capture structured data . . . 65

4 HYPERBOLIC EMBEDDING FOR STRUCTURED DATA 67
4.1 Embedding Graph Data within Poincaré Ball Model 67
4.1.1 Riemannian GMM 70
4.1.2 EM for Hyperbolic *Gaussian Mixture Model* : 72
4.2 Hyperbolic Embedding of Structured Data 74
4.2.1 Learning Embedding with communities 75
4.3 Community detection 79
4.3.1 Complexity and Scalability 81
4.4 Community classification 83
4.4.1 Community Classification Results 84
4.4.2 Discussion on Hyperbolic graph embeddings 86
4.5 Hyperbolic embeddings and structured data 88
5 HYPERBOLIC MULTI-LABEL CLASSIFICATION 91
5.1 Classification based continuous embeddings 92
5.1.1 Learning representation for classification 92
5.1.2 Hyperbolic embeddings for classification 93
5.2 A joint labels/examples hyperbolic embedding 96
5.2.1 Evaluation 99
5.2.2 Discussion 99
5.3 Examples based hyperbolic embedding 100
5.3.1 Prediction performances 103
5.3.2 Ensembling 103
5.3.3 Discussion 105
5.4 Discussion and perspective 106
6 CONCLUSION 109
6.1 Contributions and adressed challenges 109
6.2 Remaining challenges 110
6.3 Discussion on the advance of the domain 112
BIBLIOGRAPHY 113
Appendices 123
1 Decoding Algorithm DSNC 125
2 Riemannian Manifold and Hyperbolic space 125
2.1 Riemannian Manifold 125
3 Community embedding additional visualization 130

LIST OF FIGURES

Chapter 1: Introduction 1

Chapter 2: Extreme Classification Previous and Current Approachs 9

Figure 2.1 Percentage of examples annotated by each label of the dataset, sorted by ascending frequency for the *delicious* and the *wiki10* datasets. 12

Figure 2.2 Examples of forest for multi-label classification as performed in *MLRF* algorithms, each leaf corresponding to a subset of labels. The prediction is then obtained by merging labels of each leaf reached in the tree classifiers (typically a vote) in order to obtain a rank over the labels. 23

Figure 2.3 Examples of Bloom filters process for multi-label classification. Each labels get an associated code, classifiers from features space returns a code vector. The prediction is depicted in the figure in removing successively classes where bits are positive (gray circles) in labels representation and negative (white circles) for example representation. 25

Figure 2.4 The initial concept of ECOC, learning more classifier than classes to handle the non-separability of documents according to their classes. 32

Figure 2.5 Principle of representation methods to face the *eXtreme classification*. First partitioning the input space and then learn representation on each partition. 35

Figure 2.6 Same geodesic (orange line) in the hyperboloid and Poincaré disk model 38

Figure 2.7 Figure depicting similarity between Poincaré distance and graph distance. The first figure (2.7a) giving the partial geodesics between two points of similar norms, showing curvature. The second figure (2.7b) compares the ratio $\frac{d(\mathbf{x},\mathbf{y})}{d(0,\mathbf{x})+d(0,\mathbf{y})}$ using *Hyperbolic* distance (blue curve) and *Euclidean* distance (orange curve). Notice that this figure is inspired by the paper *Representation Tradeoffs for Hyperbolic Embeddings* (Sa et al. 2018). 39

Figure 2.8 Representation of hierarchical data using the Poincaré disk representation. 39

Chapter 3: DSNC and ATOMS network : learning codes 43

Figure 3.1 The main principle of LSH algorithms, each point belongs to a bucket depending on the sign of linear functions h_1 and h_2. K-NN is then performed only on samples from the same bucket. In the original work h_i are considered as random *hyperplane*. 45
Figure 3.2 DSNC model architecture, $\mathbf{x}$ the input features vector, ϕ the modeled Bernoulli distribution according to $\mathbf{x}$, $\mathbf{b}^{\mathbf{x}}$ the binary code sampled from the Bernoulli distribution and d_θ the decoder of binary codes (see section 3.1.3) 46
Figure 3.3 Linear decoding of binaries code, we learn a linear multi-class classifier taking as input code parametrized by a matrix $W \in \mathbb{R}^{c \times K}$. Prediction is then given for a code $\mathbf{b}^x$ by $l = \arg\max W^t \mathbf{b}^x + \mathbf{a}$ 48
Figure 3.4 Hashing decoding of binaries code, each code is associated with one class only by forwarding each example using the linear decoding method, then all couples (code, label) are stored in a hash map (or code being interpretable as integers a table could also work). Predicting examples is then performed by looking at the labels associated with the code of an example to label. 49
Figure 3.5 K-NN decoding of binaries code, for each training examples we stored couples (code, label). For prediction we compute the distance of a testing code to stored ones. With a bucket-based methods (Mohammad Norouzi et al. 2013) the number of comparisons is drastically lowered. 49
Figure 3.6 Complexity comparison for DMOZ-12K 55
Figure 3.7 t-SNE representation of the class codes for a sub-sample of 60 classes of ALOI dataset. Codes of a same class are represented by the same color. 56
Figure 3.8 Architecture of the dictionary atoms model. 1) From input get a probability vector which corresponds to the probability of selecting a subset of classes corresponding to the example; 2) sample which subset will be selected according to the previous distribution; 3) Merge the labels corresponding to the labels of selected subset . 59
Figure 3.9 Architecture of the atom dictionary which models the positive and negative atoms 60
Figure 3.10 Precision (@1,@5) and sparsity of the output depending on the selection constraint α_s (*delicious*) 64

CHAPTER 4: HYPERBOLIC EMBEDDING FOR STRUCTURED DATA 67
Figure 4.1 Representation of hierarchical data using the Poincaré disk representation. 69
Figure 4.2 Normalization coefficient $\sigma \mapsto \zeta_m(\sigma)$ for different values of the dimension m of the Poincaré ball 71

Figure 4.3 Visualization of two Gaussians within the Poincaré ball model, with the red points being the Gaussian mean. 72
Figure 4.4 Visualize the impact of the O_3 loss over embeddings considering two Gaussian. For this figure we took two random hyperbolic Gaussian and sample point into hyperbolic space. We then compute for each point their probability to belong to each Gaussian and finally applied the O_3 loss until convergence. 76
Figure 4.5 The distributions $\hat{y}$ of communities for *DBLP*, *Wikipedia* and *BlogCatalog* dataset 88
Figure 4.6 Hyperbolic embeddings colored according to regression logistic prediction for *karate*, *polblog*, *books*, *football* and *dblp* graphs . 89

CHAPTER 5: HYPERBOLIC MULTI-LABEL CLASSIFICATION 91
Figure 5.1 Labels distribution on *wiki10* 94
Figure 5.2 Embedding wiki10 subset of labels into hyperbolic space 95
Figure 5.3 Learning and overfitting for the mapping part. Validation decrease whereas the training performances still increase. 105

CHAPTER 6: CONCLUSION 109
Figure 1 Illustrating the principle of tangent space with the two operators exponential and logarithmic map. 126
Figure 2 Compareason of Riemannian gradient descent algorithm proposed in Bonnabel 2013 for figure 2a and Nickel et al. 2017 for figure 2b. In both figure we minimize the squared hyperbolic distance between two points. 128
Figure 3 Visualization of exact and approximated RSGD 128
Figure 4 Visualization of parallel transport of a vector (red) along a geodesic (blue) on the Poincaré disk. 128
Figure 5 Hyperbolic embeddings colored according to ground truth for *karate*, *polblog*, *books*, *football* and *dblp* graphs 130
Figure 6 Hyperbolic embeddings colored according to gmm prediction for *karate*, *polblog*, *books*, *football* and *dblp* graphs 131
Figure 7 Hyperbolic embeddings colored according to regression logistic prediction for *karate*, *polblog*, *books*, *football* and *dblp* graphs . 132

LIST OF TABLES

CHAPTER 1: INTRODUCTION 1

CHAPTER 2: EXTREME CLASSIFICATION PREVIOUS AND CURRENT APPROACHS 9

Table 2.1 Statistics of mono-label corpora 11

Table 2.2 *Extreme* multi-label corpus characteristics 11

Table 2.3 Examples of frequent/infrequent labels for the wiki10 dataset . . 12

Table 2.4 Methods summary with the type of approach and their prediction time. 30

Table 2.5 Example of trichotomy based ECOC 32

CHAPTER 3: DSNC AND ATOMS NETWORK : LEARNING CODES 43

Table 3.2 DMOZ-1K distance intra-class (test-dataset) 53

Table 3.1 Accuracy of the proposed model DSNC and the two baselines on the different datasets. The gray background indicates a constant decoding time. 54

Table 3.3 Multi-label results for Mediamill using multi-label DSNC 56

Table 3.4 Proposed approaches versus SLEEC (One classifier) and OVA . . 62

Table 3.5 Dictionary learning with ensemble learning versus SLEEC . . . 63

Table 3.6 Sparsity/Precision depending on the threshold of the matrix a^+ and a^- on delicious dataset (2 learners on *delicious*) 63

Table 3.7 Atoms Network comparison performances (p@1/p@3/p@5) on wiki10 dataset with our model without sparsity constraint (AN), the SLEEC baseline and most common label baseline (MC) . . . 64

CHAPTER 4: HYPERBOLIC EMBEDDING FOR STRUCTURED DATA 67

Table 4.1 Characteristics of the datasets used for experimental evaluation. $|V|$ the number of nodes, $|E|$ the number of edges, K the number of communities and ML whether or not the dataset is multi-label (whether or not a node can belong to several communities). . . 80

Table 4.2 Unsupervised community detection performances for Hyperbolic K-Means (H-KM) and Expectation-Maximisation (H-EM) in comparison with state-of-the-art method *ComE*. 83

Table 4.3 Results for the different classification methods. With : H-B the supervised hyperbolic K-Means based on hyperbolic barycenter; H-GMM the supervised hyperbolic Gaussian mixture model; H-LR the regression logistic in hyperbolic space; *ComE* Cavallari et al. 2017. 85
Table 4.4 Performances obtained by our method compared to Hyperbolic-SVMCho et al. 2018 (H-SVM) for small-scale datasets for 2-dimensional embedding. Results are the means for 5-folds cross-validation for 5 experiments. 85
Table 4.5 Precisions at 1, 3, 5 for HLR (Hyperbolic Logistic Regression) for 2 and 10 dimensional embeddings, MCC-CV (mean of the most common community using 5-cross validation sets) and MCC-A (most common community for the entire dataset) 87

CHAPTER 5: HYPERBOLIC MULTI-LABEL CLASSIFICATION 91
Table 5.1 Results of our approach (Hyp) compared to several state-of-the-art approaches. For *SLEEC* and *AnnexML*, the predictions are made by aggregating the label vote of 15 different models. *FastXML* and *PfastreXML* aggregate 50 trees prediction. *AnnexML* merging 1 to 3 classifiers results. All reported results come from the *Extreme Classification repository*, except for *AnnexML* for which the reported results are those of the original paper. 100
Table 5.2 Influence of the size of the representations and the number of aggregated models for the *Wiki10-31K* corpus for the precision @1,3,5. Hyperbolic refers to our approach, Euclidean refers to a Euclidean embedding similar to the AnnexML algorithm without pre-clustering. 101
Table 5.3 Precision scores at 1/3/5 for our method compared to AnnexML (State-of-the-Art XML representation). All the scores are given for one learner except for last column where Tagami 2017a results are reported for 15 learners. EE meaning example/example loss and O locality loss 103
Table 5.4 Results obtained by ensembling multiple learners and different embeddings dimensionality. To evaluate we use different metrics, the precision, the nDCG and the PsnDCG described in the section 2.1.4 104
Table 5.5 Comparison of *Mean Rank* between hyperbolic and Euclidean Space on the *Wiki10* dataset 107

CHAPTER 6: CONCLUSION 109

ACRONYMS

XMoL	eXtreme Mono-Label
XMuL	eXtreme Multi-Label
XMiL	eXtreme Missing-Label
SVP	Singular Value Projection
RSGD	Riemannian Stochastic Gradient Descent
SVM	Support Vector Machine

CHAPTER 1

INTRODUCTION

Contents

1.1 Machine learning and Big data 1
1.2 Extreme Classification 2
1.3 Contributions 4
1.4 Structure of the document 6

1.1 Machine learning and Big data

Sharing videos, photographs or more generally multimedia content is today a common practice. This behavior is promoted by information technologies progress and the generalized access to the greatest number to multiple devices, such as smartphones or laptops. For the past twenty years, the stream of content did not stop increasing, besides, most of the time content is stored and in free access for a large number of users. For instance, during the year 2019, on average, six thousand messages (*tweets*) seconds were published on the twitter platform. Also, website content has widely grown, such as the online encyclopedia *Wikipedia*, which today counts more than 51 million articles. Therefore, one problem remains, how can users efficiently retrieve contents?

Today, seeking for an article or more globally content, is obtained by entering queries consisting of relevant keywords in search engines. For the most specific searches, search engines will suggest that users pick topics from a list of keywords.

Unfortunately, there are few documents where keyword information is explicitly given, thus in most of the search engines an upstream algorithm has to process images, text or other types of data to retrieve relevant keywords. Retrieving keywords is at the origin of many applications, such as previously stated for information retrieval but also in information organization tasks: for instance sorting content thematically.

However, the excessive number of data to process prevents human annotation because of cost and efficiency reasons. For this aim, many automatic annotation approaches were developed to deal with the classification task. Despite the efforts, the diversity, the size and more largely the complexity of the data prevent previous approaches to scale to the size of corpora.

To achieve the development of efficient methods, the last ten years have seen the rise of machine learning approaches which, thanks to a previously annotated set of documents train a parametric function that is able to retrieve these keywords or categories.

Today, machine learning methods produce the state-of-the-art in image recognition or segmentation, automatic translation, recommendation systems, sentiment analysis or text classification. Nevertheless, the classical techniques do not pass the scale when considering huge number of possible keywords.

In this thesis, we, therefore, propose to address the multiple challenges of automatic classification when the number of possible categories is numerous. The associated domain which aims to address this challenge is known as *extreme classification*, which characterizes the task of classification when we consider more than several thousand or millions of classes possible. We addressed those challenges through the lens of representation learning, an emerging approach to address challenges associated with *extreme classification*. Before going further and introducing the contributions and the flow of the thesis, let first describe *extreme classification* challenges and its different associated tasks.

1.2 Extreme Classification

Extreme classification aims to efficiently retrieve correct annotations for complex and large corpus. Consequently, the following challenges are addressed:

- **Storage complexity** : All automatic classification approaches rely on storing data or function parameters. Thus, elements stored have to be contained in classical contemporary devices.

- **Prediction time complexity** : The time consumed by algorithms to get classes has to be shorter as possible. Typically, in industries the objective is to reduce the computing resources while obtaining accurate annotation. Today, classical classification algorithms mainly classify in linear time according to the number of labels, *extreme classification* oriented methods have to develop sub-linear time methods.

- **Learning time complexity** : Similarly to prediction time the learning time should be as fast as possible. However, contrary to prediction, computing capacities are frequently greater.

- **Accuracy** : Predicted labels must be accurate and relevant for each document. Typically, accuracy of the model will be evaluated by its capacity to retrieve correct annotations on an already annotated set named test set (different from the one use for training the classifier).

A major drawback with standard methods is the prediction time according to the number of labels which mainly scale linearly with it. This issue can be handled using additional constraints to reduce prediction time and storage. For instance, ensuring sparsity using linear classifiers reduces the number of parameters and consequently the number of operations. Despite reducing the number of parameters, such methods still consider as many predictors (or classifiers) as there are classes. To alleviate processing time one can consider "divide and conquer" paradigm. The main idea of divide and conquer paradigm relying on partitioning whether the features or label space, thus, classifiers only focus on a sub-part of the data.

On the other hand, having accurate prediction is as important as prediction time complexity. However, in many cases reducing complexity negatively impact accuracy. Therefore, the objective is generally to consider a trade-off between those two challenges, optimizing both accuracy and inference time.

The classification tasks

If approaches mainly differ according to the addressed challenge, it also depends on the considered tasks (or schemes of classification). In extreme classification different objectives are addressed depending on the task. We can define three classification schemes:

- **eXtreme Mono-Label (XMoL) classification :** Each document is annotated with a unique label. For instance, for the *ImageNet* classification task, a dataset of pictures, the objective is to predict the class describing at the best each image.

- **eXtreme Multi-Label (XMuL) classification:** Each document is annotated with one or several labels. For instance, retrieve all items on an image, or finding all topics of a document.

- **eXtreme Missing-Label (XMiL) classification :** Each document is annotated with one or several labels, but the paradigm differs from XMuL because of the corpus partially labeled. For instance, in the context of images where only a small subset of items already labels images, the objective is then to predict the missing items. Thus labels are considered as potential features contrary to XMuL where none of the labels are given.

Take care of distribution

Each of *extreme classification* schemes have to deal with specific problems. In mono-label setting, occurrences of labels could drastically differ, such that some labels are observed much more than others. We refer to this issue by unbalanced labels distribution. Labels distribution has a direct impact on performances, particularly if the real distribution of labels differs from the one used to train the classifier. Fortunately, on considered

datasets it is rarely the case and training set, evaluation set and test set have similar distributions of labels.

The real issue caused by labels distribution balance is when we address multi-label classification. In this last setting we often observe important unbalance, leading to have in real corpus labels annotating more than half of the training examples and others only a few number of examples. Classifiers should be thus carefully designed to ensure not only predicting most frequent classes.

Representation Learning

Neighbor search algorithm consists in finding closest neighbors considering a specific metric and space. In classification, documents are lying in a space named features space, one assertion is that close documents considering a distance or a similarity in the space implies that those document are closely related in terms of content. With this last prior one could expect that documents having close representation will have similar annotation. However, it is not necessarily the distance associated with the features space (such as l_2 distance in Euclidean space) that is the most relevant. Besides, depending on the size of the feature space, nearest neighbor search can be time consuming, for instance images can be represented with pixel values vectors that could lead to very high prediction time.

Representation learning methods aim to tackle those two issues by embedding the features space in a low-dimensional one and ensuring that closely embedded documents are similarly labeled. Moreover, neighborhood-based search prediction still could predict rare labels since a document may lie close to a document labeled by underrepresented labels. Although often reaching the top performances, greedy neighbor search is time consuming even when embedded in lower space. Indeed, prediction suggests comparing an embedded example to predict with a large set of already known representations. However, this issue can be addressed using an approximate neighbor search.

1.3 Contributions

The objective of the thesis is the study of representation approaches with the objective to find efficient solutions to the extreme classification challenges. We mainly focused on two of these challenges, the prediction time complexity and the prediction accuracy ones.

At the beginning of the PhD few methods proposed modern approaches to deals with large-scale classification with representation based methods. Representation approaches were relying on the labels/features space dimensionality reduction paradigm. The objective consists in finding the best low size representation of documents or labels and consequently reducing the time complexity. Once a document is projected, one should retrieve labels through an algorithm or a function involving at least as many operations as there are labels. We can split embedding methods into two parts, the embedding process also

named encoding and the reconstruction one often referred as the decoding process or the decoder.

Mostly, previous approaches considered *Euclidean* embeddings of the feature space. At the beginning of the *PhD*, this principle was known as the *Labels/Features* space dimensionality reduction. It generally reduces the complexity from $O(NK)$ to $O(Nc + cK)$ with N the number of features, K the number of classes and c the size of the representation (NC for the encoder and cK for the decoder). However, one should notice that considering a finite number of representations and a hash function from representation space to an index would lead to a very fast decoding process. In this last eventuality storing for each representation the label associated will allow a constant decoding time. Thus, the only time complexity bottleneck is the encoding function and the application of the hash function. Learning a binary representation of documents is a possible application of the previous principle. Indeed, considering a space of low dimension, would allow enumerating and storing all possible combinations (considering c as the binary vectors size will lead to 2^c possible codes). Ou first contribution in section 3.1 studies binary embeddings to fasten prediction time according to the stated principle. However, when the dimension of the representation space becomes too large, this last decoding method is intractable as the number of codes increases exponentially with c. Fortunately, it is still possible to obtain sub-linear time classification. For instance, considering the frame of representation learning and neighbor search could lead to time efficiency prediction using buckets based neighbor search. Then, the last remaining question relies on how to learn those binary vectors. Designing it randomly would obviously lower performances particularly in neighbor search decoding. To handle this, we propose an end-to-end learning procedure that learn both encoder and decoder jointly.

Following the same principle of finding efficient structure representation for time-efficient label prediction, we proposed an end-to-end approaches to address multi-label setting. In multi-label corpus, subsets of labels are co-dependent i.e. labels that are correlated because they always or often occur together. Consider those subsets to learn a classifier should lead to improve performances. The main idea of the second contribution is thus to discover those subsets and for a given document to retrieve subsets that most likely labels it. Similarly to the previous proposal, learning the encoder (part of the model that aims to select subsets) and label partitions (subsets of co-dependent labels) together would rely on better performances. Under certain conditions, it is then possible to retrieve the annotations in sub-linear time by aggregating the selected label sets.

Today the accuracy challenge is mainly tackled considering continuous representations learning approaches. Particularly recent works such as *SLEEC* (Kush Bhatia et al. 2015) or *AnnexML* (Tagami 2017a) reached top performances on a vast majority of corpora. Those algorithms embed closely in the representation space documents which are similar w.r.t their labels. Subsequently, inference is based on fast neighbor search decoding by

comparing documents representation of the training set and representation of documents to predict. One should thus imagine using labels and document representation together.

However, embed labels and documents together need to ensure that each of the representations of the document is close to their labels representations. If labels are organized as a hierarchy, intuitively we would organize labels embedding as a graph and thus consider a graph distance. Hyperbolic embeddings is known as a great choice to correctly represent graph organized data, particularly due to the similarity between graph distance and hyperbolic distance. In a first study, we proposed to verify ability of hyperbolic embeddings in describing correctly structured types of data. To this end, the contribution deal with hyperbolic embedding applied to graph community detection and classification. For this aim, we introduce an *expectation-maximization* algorithm to fit a hyperbolic *Gaussian Mixture* model, having two objectives: agglomerate node representation of the same communities and retrieve communities.

Nonetheless, classification of communities does not allow to assess it efficiency to deal with large-scale classification but to validate its ability to embed structured data. We introduce hyperbolic embedding for large-scale corpus by embedding labels and documents together. Prediction considering neighbor search on labels and examples remains difficult. We will show that even optimizing a cost having this objective will lead to lower performances than relying on a neighbor search only based on documents. However we clearly show that it helps to structure the documents representation. In this last contribution, we proposed several costs for learning representations into hyperbolic space and different approximation of the documents embedding function. We show through many experiences that hyperbolic space is relevant for embedding *extreme classification* types of data.

1.4 Structure of the document

In the first chapter 2, the section 2.1.3 introduces corpus characteristics and discusses of their sizes, their features and the organization of labels to figure out what the *extreme classification* challenges rely on. In the same section, we also describe the different ways classification tasks are evaluated and discuss the pros and the cons of the different metrics. The section 2.2.1 will provide an overview of the different classification schemes and usual methods to face it. Then, the bibliography of the approaches developed to face *extreme classification* challenges is examined. Representation learning type of approach is particularly interesting for us since we developed three contributions based on it. Thus, the section 2.3.1 describes current state-of-the-art representation based approaches to deal with the tasks : firstly by describing those based on fastening prediction such as *ECOC* and then those based on prediction accuracy with mainly continuous representation

approaches. This chapter concludes introducing recent approaches to learn hyperbolic embedding and gives tools to understand optimization process in section 2.1.2.

The chapter 3 presents the first two contributions of this thesis. The first contribution learning binary embedding for fast prediction is described in the section 3.1: its modelization is given in section 3.1.1, its evaluation on different large corpus in section 3.1.5 and in the section 3.1.6 a discussion on the advantages of the method. Addressing the multi-label setting in proposing the atoms network that learns subset of co-dependent labels is described in 3.2. The section 3.2.2 proposes evaluation of experiments to validate the model's performances and a discussion on future improvements to improve the trade-off between prediction performances and prediction time complexity .

In chapter 4, we introduce the second objectives of this thesis: obtaining better structured representation by introducing hyperbolic embeddings for graph communities detection and classification. The chapter is organized as follows : 1)The description of the tools used to develop the methods in section 4.1.1; 2)The presentation of the *Expectation/Maximization* algorithm for learning *Gaussian Mixture Model* in hyperbolic space in section 4.1.2; 3)The presentation of the embedding process and the joint community and node embedding optimization in section 4.2.
In the subsequent section, we evaluate the approach over two tasks the first dealing with community detection (section 4.3), and the second addressing community classification (section 4.4.1) on real graphs. We finally discuss the advantages of hyperbolic manifolds to embed structured data and their usefulness in multi-label application tasks (section 4.4.2).

With the validation obtained in previous chapter, we now address the *extreme classification* challenge using hyperbolic manifolds in chapter 5. We present two tasks, the first one being the classification using K-*NN* approaches by : 1) proposing a framework that embeds both documents and labels in the hyperbolic space in section 5.2; 2) proposing an approach for documents embedding within hyperbolic manifold relying on label vector similarities (section 5.3).

We finally in chapter 6 conclude by reminding the different contributions and discussing of their advantages and drawback (section 6.1). To conclude in section 6.2), we discuss potential improvement and impacts of the proposed approaches.

CHAPTER 2

EXTREME CLASSIFICATION PREVIOUS AND CURRENT APPROACHS

Contents

2.1 The large scale classification task 10
2.1.1 Notations 10
2.1.2 Corpus 10
2.1.3 Discussion on distribution and corpora size 11
2.1.4 Evaluation metrics 12
2.2 Adressing classification challenges 14
2.2.1 Early multi-class classification 15
2.2.2 Classification and current applications 18
2.2.3 Efficient time complexity approaches for XML 19
2.3 Learning embeddings for eXtreme classification 31
2.3.1 Faster proccessing and discret representations 31
2.3.2 XML based continuous representation 33
2.3.3 Representation learning and hyperbolic space 37
2.4 An uncompleted challenge 41

Before going further in the contributions and introducing the three main contribution of this thesis, we will first introduce the extreme classification task by presenting corpus, challenges and evaluations. We then introduce historical methods to handle binary classification (two classes classification), multi-class classification and multi-label classification. We continue the discussion of related works by describing more specific methods that address the different challenges of the *Extreme Classification*: the prediction time efficiency, the storage efficiency and, of course, the prediction accuracy. As we particularly address the *Extreme Classification* tasks over the representation learning paradigm, we dedicate an entire section to present with deeper precision those approaches, detailing particularly methods concerning hyperbolic representations.

We finally conclude by a discussion on the current state-of-the-art and what would be interresting to explore to tackle *Extreme Classification* remaining challenges.

2.1 The large scale classification task

In this first section, the objective will be to make the reader aware of *extreme classification* challenges. In order to make an idea of the challenges, understanding the characteristics of corpora will be the first matter of interest for us. The description below will spot the difficulties encountered when facing such task, such as those related to the data distribution or to the extremely large number of labels. We then introduce metrics to evaluate classification performances and how it differs from usual classification metrics, we particularly outline the multi-label setting where evaluation using simple accuracy metric is irrelevant.

However, before discussing those two points it is necessary to provide notations for corpus, labels and features. Thus, the following section will formalize notations which we will endeavor to maintain all along this document.

2.1.1 Notations

For all classifications methods whether addressing mono or multi-label tasks, we define a set referring to the considered corpus as $\mathcal{D}$. In supervised classification $\mathcal{D}$ is a set of tuples $\mathcal{D} \subset F \times Y$ where F is called the features space and $Y = \{0,1\}^K$ the label space.

For each tuple $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ we refer as $\mathbf{x}$ being the features of an example and the label vector of an example as $\mathbf{y}$. We say that an example $(\mathbf{x}, \mathbf{y})$ is labeled or annotated with j only if the jth component of the label vector is equal to one, i.e. $y_j = 1$. However the nature of labels space differs between mono and multi-label dataset, in mono label corpus an example can be annotated with only one unique class thus each label vector get only one component to one, i.e. $\forall(\mathbf{x}, \mathbf{y}) \in \mathcal{D}, \sum_{i=1}^{K} \mathbf{y}_i = 1$, in multi-label there is no such restriction. A classifier is then a function from the features space to the label space $f : F \to \{0,1\}^K$.

For convenience in the following, we also define the label matrix being the concatenation of label vectors for the dataset $\mathcal{D}$ such that :

$$L_{\mathcal{D}} = \begin{pmatrix} \mathbf{y}_{1,1} & \cdots & \mathbf{y}_{1,K} \\ \vdots & \ddots & \vdots \\ \mathbf{y}_{|\mathcal{D}|,1} & \cdots & \mathbf{y}_{|\mathcal{D}|,K} \end{pmatrix} \tag{2.1}$$

2.1.2 Corpus

An extreme classification corpus is firstly characterized by the large number of labels to discriminate, generally we start considering a corpus to be "*extreme*" from ten thousand different labels. It could be, moreover, pertinent to consider lower corpus sharing common properties with *extreme* corpus (such labels distribution). Firstly, we should remark that

mono-label and multi-label corpus do not address the same challenges, mono-label corpus examples are not explicable with many labels, thus, do not make use of correlation between labels. Thus the principal difficulty in mono-label remains to make accurate prediction with scalable time and storage.

Dataset	Avg documents/labels	#labels	#train
MNIST	6000	10	60,000
IMAGENET	1281.2	1,000	1,281,167
ALOI	108	1000	108,000
DMOZ	7.6	12294	93805

Table 2.1 – Statistics of mono-label corpora

In the table 2.1, we show statistics of mono-label corpus, showing that the main difficulty relies on handling the size of the labels space and the number of documents for each label (particularly for ALOI and DMOZ corpora).

However, the real challenging task is to address multi-label classification where taking into account distribution and dependencies over labels are mandatory. According to the table 2.2 describing different multi-label corpora, we can notice that a corpus could get more examples than labels (as *Wiki10-31K*) and similarly only a few examples for each label (for instance in the *Amazon-670K* corpus).

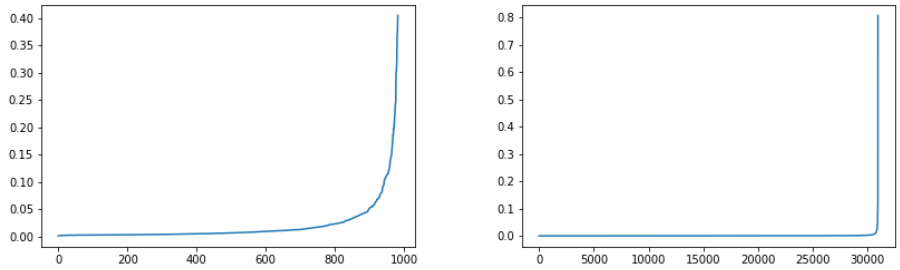
Dataset	Avg documents/labels	Avg labels/documents	#labels	#train
Delicious	312	19.03	983	12,920
RCV1-2K	1,218	4.79	2,456	623,847
EURLex-4K	449	5.37	3,993	15,539
Wiki10-31K	8.52	18.64	30,938	14,146
Delicious-200K	72.29	75.54	205,443	196,606
WikiLSHTC-325K	17.46	3.19	325,056	1,778,351
Wikipedia-500K	24	4.77	501,070	1,813,391
Amazon-670K	4	5.45	670,091	490,449

Table 2.2 – *Extreme* multi-label corpus characteristics

In addition to the low number of examples for each label, labels are rarely represented equally in each corpus, leading to have over and underrepresented classes.

2.1.3 Discussion on distribution and corpora size

Figure 2.1 shows the percentage of examples in the dataset labeled with each label, sorted by frequency from the most infrequent to the most common ones, in the *wiki10* and *delicious* datasets. In each dataset, the distribution seems to follow a power law distribution with few labels concerning a high percentage of examples, on the contrary



(a) delicious (b) wiki10

Figure 2.1 – Percentage of examples annotated by each label of the dataset, sorted by ascending frequency for the *delicious* and the *wiki10* datasets.

label name	frequency
wikipedia	80.7%
wiki	41.9%
reference	28.3%
history	18.7%

(a) head labels

label name	occurences
musc	1
articulation	3
boards	5
banality	7

(b) tail labels

Table 2.3 – Examples of frequent/infrequent labels for the wiki10 dataset

most labels having very few examples. A direct consequence is that predicting head labels - labels that occur frequently - is an easy task (e.g. in the *wiki10* dataset, more than 80% of the examples are annotated with the label *wikipedia*), however, predict with accuracy tail labels is much more difficult.

Therefore, some algorithms such as *AnnexML* (Tagami 2017a) focus on tail labels classification. Moreover, heads or tails labels are frequently meaningful (see table 2.3), head labels generally correspond to more general concepts (history is a global concept) and at the contrary tail labels represent more specific ones. This fact underlies that it should exist in those corpora a hierarchy or more generally a taxonomy over labels. Thus trained classifiers must be robust to overfitting and aware of labels correlation and distribution to accurately annotate documents.

2.1.4 Evaluation metrics

For mono-label classification, evaluation is mainly performed by the accuracy metric. The Accuracy measures the percentage of classes correctly predicted :

$$accuracy(\mathcal{D}, f) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{D}} \mathbb{1}_{f(\mathbf{x})=\mathbf{y}} \tag{2.2}$$

with

$$\mathbb{1}_{f(\mathbf{x})=\mathbf{y}} = \begin{cases} 1 & f(\mathbf{x}) = \mathbf{y} \\ 0 & f(\mathbf{x}) \neq \mathbf{y} \end{cases}$$

Contrary to mono-class classification task, the multi-label one cannot be evaluated using accuracy as in general finding accurately all the labels that annotate an example will fail and thus give only little information on the performances of the classifier. In multi-label classification, using *precision* (percentage of correctly predicted labels over a number of inferred labels), *recall* (percentage of correctly predicted label over the amount of truth labels) can reflect the algorithm performances. Unfortunately, those metrics are not straight forward for the *extreme multi-label classification* task for at least two reasons: 1) the majority of algorithms are not designed to output a set of predicted labels but only a score for each label; 2) the number of labels is too large to expect to predict the exact label set, which usually leads to poor performances indicators having a very low and meaningless score. In order to have a good estimation of algorithms performances, the community mainly prefers ranking metrics as *precision@k* or *Discounted cumulative gain@k*. In this section we present different evaluation metrics with their advantages and their inconvenience. Moreover we now consider as output of a classifier f a score on each label and no more a set of labels, i.e. $f : F \to \mathbb{R}^K$ and denote $\hat{\mathbf{y}}$ the resulting vector. Those evaluation metrics often consider ranking in their formula, let's define the rank function for a vector $\hat{\mathbf{y}} \in \mathbb{R}^K$ returning the set of indexes corresponding to the labels with the highest score:

$$rank_n(\hat{\mathbf{y}}) = \underset{S \subset \mathbb{N}, |S|=n}{\arg\max} \sum_{i \in S} \hat{\mathbf{y}}_i \tag{2.3}$$

Precision at n (`P@k`): supposing that there is at least n labels to predict for a given example, the precision at n indicates the average number of correct labels among the first n predicted labels.

$$p@k(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n} \sum_{l \in rank_n(\hat{\mathbf{y}})} y_l \tag{2.4}$$

Propensity Scored at n (`PS@k`): Very similar to precision, this metric takes into account the label distribution and the hierarchy of labels :

$$PS@k(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n} \sum_{l \in rank_n(\hat{y})} \frac{y_l}{p_l} \tag{2.5}$$

With p_l the propensity score for a label l (Jain et al. 2016) which is a factor mainly depending on the number of occurrences of the labels in the dataset.

Discounted cumulative gain at k (`DCG@k`): The discounted cumulative gain takes into account the rank order of the prediction, intuitively error on top of the ranked vector will lead to lower the score more importantly than errors at the end of the ranked vector:

$$DCG@k(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n} \sum_{l \in rank_n(\hat{y})} \frac{y_l}{log(1 + rank(l))} \tag{2.6}$$

With $rank(l)$ the rank of the lth label.

normalized Discounted Cumulative Gain (`nDCG@k`):

$$nDCG@k(\hat{\mathbf{y}}, \mathbf{y}) = \frac{DCG@k}{\sum_{l=1}^{\min(k,l)} \frac{1}{log(1+l)}} \tag{2.7}$$

(`PSDCG@k`):

$$PSDCG@k(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n} \sum_{l \in rank_n(\hat{y})} \frac{y_l}{p_l log(1 + rank(l))} \tag{2.8}$$

(`PSnDCG@k`):

$$PSnDCG@k(\hat{\mathbf{y}}, \mathbf{y}) = \frac{PSDCG@k}{\sum_{l=1}^{\min(k,l)} \frac{1}{log(1+l)}} \tag{2.9}$$

Many other metrics are commonly used in *large scale* multi-labels classification, particularly by mixing the objectives of the metrics previously presented such as *nDCG@k*, `PSDCG@k` (merge propensity score with `DCG` or `PSnDCG@k` (merge propensity and `nDCG`). For more details reader can refer to the *Extreme Classification Repository*(K. Bhatia et al. 2016).

2.2 Adressing classification challenges

Addressing classification tasks has been widely studied: in this section, we will describe algorithms and previous methods having as objective classification. Firstly, we will introduce historical algorithms and classical schemes for learning multi-class classifiers. We then introduce today's approach and recent improvements to deal with classification tasks, particularly with the task-oriented models addressing classification for specific data such as images or raw texts.

Thirdly, we will describe approaches that directly address large-scale challenges such as taking into account prediction time or label correlation. We finally conclude and introduce the next section on the representation approaches to face large-scale classification.

2.2.1 Early multi-class classification

Binary classification. One of the first historic cases of classification involving machine learning was the mono-label binary classification task: datasets contain exactly two different possible annotations, each corresponding to one or the other class. In a large majority of case the representation of examples (image, text) is characterized by a vector, for instance we could consider an image as the vector containing value of pixels; for a text document, we can consider each word as an index and represent a document by a one-zero vector of words (one for index of words occurring in the document).

Thus, to handle binary classification one may consider a separator represented by a hyperplane with on one side of the hyperplane examples annotated with one class and on the other side examples annotated with the other. Retrieving a hyperplane with such property is the aims of the *perceptron* proposed by Rosenblatt 1958, one of the first machine learning approach for classification.

However, this approach suffers from low generalization ability, generalization being the capacity of the model to adapt to new data that has never been seen during the training phase. Finding the best hyperplane that guarantees that the separator is as far as possible from both classes is a solution to handle the generalization issues. Thankfully, the approach proposed by Boser et al. 1992 allowed to find the optimal margin between examples and thus, retrieve the best separator. This last approach known as Support Vector Machine (SVM) is still considered up to date and led to many applications this last decade.

More generally, classical scheme of classification consists of optimizing an objective function based on a loss function l and a regularization terms $\Omega(f)$ to minimize:

$$E(f) = \sum_{(\mathbf{x},\mathbf{y}))}^{N} l(f(\mathbf{x}), \mathbf{y}) + \Omega(f)$$

Where f could be a linear function ($f(\mathbf{x}) = W.\mathbf{x}^t + b$) corresponding thus to a hyperplane and $\Omega(f)$ regularizations on the weight of the function f. Moreover a wide range of loss has been studied such as *hinge loss*, *mean square error* or *logistic loss*. Considering y equal to 1 or -1 for the first and respectively the second class, we can define the usual losses:

- *hinge loss* : $l(f(\mathbf{x}), y) = max(0, 1 - y.f(\mathbf{x}))$
- *mean square error* : $l(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$
- *logistic based loss* : $l(f(\mathbf{x}), y) = log(1 + e^{-y\dot{f}(\mathbf{x})})$

To optimize those error functions different optimization process could be used, when the error function is convex one can void the derivative and look for a closed-form solution.

However, in many cases the objective function does not get a closed-form solution and an approximation of the solution can be obtained by using a gradient descend algorithm.

Multi-class classification. If those approaches work initially for binary classification, we can easily apply minor transformation in order to make them work for the multi-class setting.

The "one versus" scheme of classification is one of the solutions: learning a set of binary classifiers each one classifying one class against one or several others. Mostly, linear or *SVM* classifiers are used as those base classifiers. Two different approaches of this scheme are often designed to handle multi-class classification : the *one-versus-one* and the *one-versus-all* approaches. The first learns for each possible couple of classes a binary classifier (total of K^2 classifiers) and the second learns for each class one classifier that discriminates the considered class against all others (total of K classifiers). For the first approach, *one-versus-one* prediction is obtained by aggregating results of classifiers and then select the predicted class according to a criterion (for instance using a majority vote : the class that get the highest number of votes is the one predicted). For the *one-versus-all*, the prediction is mostly obtained by selecting the class which the classifier respond with the most confidence. In this last scheme learning binary classifiers with a joint error function is more pertinent, for instance optimizing log-likelihood based on *softmax* modelization of probabilities:

$$l(f(\mathbf{x}), y) = \sum_{i=0}^{K} -y_i log \left(\frac{e^{f(\mathbf{x})_i}}{\sum_{k=1}^{K} e^{f(\mathbf{x})_k}} \right) \tag{2.10}$$

On specific multi-class corpus labels may derive from a taxonomy and correspond to a hierarchy over labels such that each class corresponds to a leaf in the hierarchy. Thus, learning classifiers (using binary classifiers) that correspond to nodes in the taxonomy is a relevant approach. This last approach is still in use and particularly time efficient, we will latter review those methods in the section 2.2.3.

Nevertheless, those methods mainly consider learning linear separators, and in the general case we do not observe linearly separable feature space (representation of data in a vector space). To handle this issue one can use neighborhood based classifiers such as K Nearest Neighbors (K-NN), where the prediction function f relies on finding closest neighbors in the feature space. Similarly, one could associate to each part of the space a class using non-linear regions, for instance the K-Means algorithm that associate each example in the feature spaces to a centroid (barycenter of vectors). The centroids are thereafter associated with specific classes (or eventually sets of classes).

Modeling a probability function $P(Y|X)$ when noticing that data follow a specific distribution is viable, for instance using a *Gaussian Mixture* prior to model this probability. Another common way to handle non-linearity rely on modifying the feature space, i.e.

applying on original feature space a function $g : F \rightarrow F'$ such that data are embedded in a space where they are linearly separable. Recent approaches trending is to consider the prediction function f as a more complex function not necessarily linear, the domain of learning non-linear functions is often referred to deep learning and those functions as neural networks. The g function can be thus handcrafted for kernel approaches or learned such its output get the desired properties for representation approaches.

We surely could continue to cite methods; however, it is not the aims of the current section. Readers would notice the rich panel of approaches to face the different automatic classification issues, in this document we could not give an extensive description of the domain. However, some of the methods mentioned above will be described with further details later (K-NN, neural networks and representation methods).

Multi-label classification. In multi-label setting, each example can be annotated with many classes, fortunately multi-class approaches could easily be derived for this last scheme. For instance, one could use the *one-versus-all* or the *binary relevance* (one classifier for each label stating if it labels a document or not) scheme of classification selecting all classes that respond positively in each classifier (using for instance a threshold over scores returned by classifiers). However, another information is important, some classes appear more a less frequently together. The idea of co-dependent classes refers as the labels correlation in the rest of this document. However for linear based algorithms the loss function cannot be the same since many classes must be selected in the output vector. Losses such as *log-softmax* (equation 2.10) is no more relevant as its aim relies on setting one component to one and the others close to zero. Multi-label loss such as *binary cross entropy logits* (equation 2.11) will be thus preferred; seamlessly optimizing likelihood but with a different model for estimate probabilities:

$$l(f(\mathbf{x}), y) = -\sum_{i=0}^{K} y_i log(\sigma(f(\mathbf{x})_i)) + (1 - y_i) log(1 - \sigma(f(\mathbf{x})_i)) \tag{2.11}$$

Similarly to multi-class mono-label scheme, handling non-linearity could rely on equivalents approaches: K-NN, K-Means or Gaussian Mixture Model (*GMM*). However, different aggregations must be designed to select no more one but many classes (do not only predict the most likely).

Robust classification. Another difficulty when addressing classification is to ensure robustness of algorithms. Considering any of the above scenarios, a solution would rely on learning several times each classifier and then aggregate predictions. This principle is implemented in *bagging* methods or in *boosting* one's (Schapire et al. 2000): *Bagging* relies on learning a set of weak classifiers and subsequently aggregate classifiers prediction; *Boosting* methods rely on a similar principle in the way of learning a set of weak classifiers, but contrary to *bagging* classifiers are learned sequentially, thus allowing designing classifiers that sequentially correct previous classifiers weaknesses.

Additionally, ensure classifier robustness can be handled easily in an *OVA* scheme in learning more classifier than classes. Instead of learning for each class a unique classifier, one can learn classifiers for a subset of classes against another such that the number of classifiers is higher than the number of classes, and each class can be retrieved by merging selected subsets (for instance using the intersection of selected subsets). The framework proposing to handle classification using this process refer to *Error Correcting Output Coding* (see section 2.3.1).

Robust classifications for generic corpus (data independent) is largely and still studied yet. For instance, an extensive review of general classification approaches in *An up-to-date comparison of state-of-the-art classification algorithms* (C. Zhang et al. 2017) is given. However, some frameworks or architectures are more effective to deal with specific types of data. As today most approaches rely on deep learning approaches (learning non-linear functions), we think introduce those approaches is mandatory to understand the current state-of-the-art.

2.2.2 Classification and current applications

The Multi-Layer Perceptron Multilayer perceptron (or neural networks) offers an alternative to one-layer linear models described above, having several advantages over linear methods. Particularly, the ability to capture non-linear separation and contrary to kernel methods learning parametrized non-linearity. The principle remains simple, it relies on stacking linear function and non-linear ones (considering the perceptron as the linear function):

$$\begin{aligned} f\colon F &\to \mathbb{R}^K \\ \mathbf{x} &\mapsto W.\mathbf{x} + b \end{aligned}$$

The multilayer perceptron relies on learning a more complex function which is the composition of basic functions : $f = g_n \circ g_{n-1} \circ \cdots \circ g_2 \circ g_1$ with g_i parametric functions (g_i will refer as a layer of the neural network). Moreover, g_i function cannot be only linear functions, since composition of linear functions still is linear. Generally, those functions are defined by a non-linearity ψ named activation and a linear function such that:

$$\begin{aligned} g_i\colon \mathbb{R}^{d_{i-1}} &\to \mathbb{R}^{d_i} \\ \mathbf{x} &\mapsto \psi_i(W_i.\mathbf{x} + \mathbf{b}_i) \end{aligned}$$

A main drawback of learning such function is the optimization that largely relies on the gradient descent algorithm using chain rules in order to back propagate gradient through layers. As generally learning such function introduce a non-convex objective, minimization of a loss function rarely converge on a global minimum. Despite this last issue, plenty of classification applications rely on neural networks and in a vast majority of case it participates to largely improve state-of-the-art performances.

Neural networks and applications. Handwritten digits mono-class classification with the design of layer function based on learning convolution weights named *Convolutional Neural Networks* (CNN by Fukushima et al. 1982; LeCun et al. 1989; Lecun et al. 1998) largely participate in the popularity of those approaches in tackling *kernel* based methods. However, it is in the last decade that neural networks raised in interest with scalable methods for scene image classification with the *ImageNet* classification challenge (Krizhevsky et al. 2012). This last one achieved a huge improvement compared to the other methods. In multi-label setting, CNN approaches have succeeded to great improvement by using bounding box (Wei et al. 2014) or more generally region based CNN instead of whole images.

Image classification is not the only challenge that neural networks tackled, those last years numerous approaches emerged to face many diverse tasks. For instance, raw text classification or more generally sequential data is today addressed with neural networks. In text classification, neural networks are largely dominating the classification task, more particularly raw data can be processed in a sequential way using *Recurent Neural Network* RNN (P. Liu et al. 2016), CNN (Kim 2014) or using both (Lai et al. 2015; J. Liu et al. 2017).

Moreover, those approaches do not solely tackle the classification task, they also provide state-of-the-art in image generation (Kingma et al. 2013; Goodfellow et al. 2014), video prediction, automatic translation (Bahdanau et al. 2014) or even in text generation (Vaswani et al. 2017). It would be too long to be extensive in the descriptions of neural methods and applications. As our contributions do not focus predominately on neural networks architecture, we redirect the interested reader to literature. However, if those methods remain effective in a large range of applications and also for classification purpose, the cost for prediction (time, memory) is important. Indeed, since in most of the case the last layer relies on a linear function, the complexity is frequently at least linear to the number of labels. Thus, neural networks approaches based on classical scheme of classification still are costly addressing the time complexity challenge.

2.2.3 Efficient time complexity approaches for XML

Even if *multi-labels* classification task has been a widely studied field of research, provided methods suffer many issues, with mainly the time complexity facing a large number of data. Competitions to tackle time complexity and performance issues such as the *Large-Scale Hierarchical Text Challenge* (LSHTC Partalas et al. 2015) provided several benchmarks to deal with classification considering a huge number of labels. In this section we briefly describe algorithms designed to face large-scale multi-class classification challenges (prediction times, storage and accuracy). Those approaches can be split in three main paradigms:

1. *One-versus-All* paradigm : Using sparse weight/representation computing for fastening prediction.
2. *Divide and Conquer* paradigm : Dividing the task such that each classifier address only subsets of classes or providing specific classifier organization (intuitively tree of classifiers).
3. *Representation Learning* paradigm : Learning an embedding function from F to F' such that $dim(F) >> dim(F')$ given a richer or task-specific representation of data.

In this section we will successively explicit the different approaches, we first start with the *OVA* based approaches. We then present the Divide and Conquer approaches (also named hierarchical approaches) and gives their strength and weakness. Finally, we present representation approaches and mixed ones that attempted to merge the advantages of each. As our works are mainly contributing to the later one type of approach, we deeply analyze previous works in representation learning for classification in section 2.3.3.

2.2.3.1 OVA based approaches

In order to reduce processing time as well as storage, many works rely on sparse parameters learning algorithms. The principle is the following: make a maximum of weight in the parametrized function to zero, hence, only non-zero parameters are stored and used for calculation. Learning sparse linear models can be performed using restriction on weights, in that way two types of approaches lead, one learning sparse weights through regularization terms (such as l_1 based losses) or those pruning weights after the training step.

For examples the PD-Sparse algorithm (I. E. H. Yen et al. 2016) proposes an optimization algorithm to leverage sparsity called *Fully-corrective Block Coordinate Frank Wolf* algorithm. This algorithm reduces training time by taking into account sparsity induced by an l_1, l_2 regularization (using *elastic net* methods (Zou et al. 2005)). The authors proposed an improvement of *PD-Sparse* named *PPD-Sparse* (I. E. Yen et al. 2017) taking advantage multi-cores for efficient training in adapting loss function to be optimized using parallel computation.

The method named *DISMEC* (Babbar et al. 2016) aims to solve the task in the `OVA` framework improving storage/time/performance efficiency by offering the following contributions: 1) Exploiting parallelization during training, leading to decrease significantly training time; 3) Faster prediction and lower storage by inducing sparsity into the model, with a method to prune weights. The authors noticed that an extremely large number of weights are in the neighborhood of zero, and show that pruning them do not lead to drastically lower prediction performances while broadly decreasing storage and processing time.

Learning subset of classifiers. The *Error correcting output coding* (`ECOC` (Kong et al. 1995)) approach can be considered as an OVA derived algorithm or a time-efficient

extension of OVA. The principle is the following: each label is represented by a predefined code (discrete representation), then, the framework relies on learning C (the size of the representation) classifiers such that the concatenation of the classifiers outputs lead to retrieve the labels code. If the original aim of the approach was to perform better multi-class classification by using codes size larger than the labels space, at a later stage the objective become to be time efficient by using smaller latent space (Puget et al. 2015). This last kind of approach can be interpreted as *OVA* but also as a representation approach, thus we will deeper describes this framework in section 2.3.1. Those approaches led to the *Ground Testing* (Ubaru et al. 2017) framework (GT) where each bit corresponds to a set of labels allowing to efficiently deals with multi-label classification.

If OVA framework is often associated with prediction accuracy performances, in *Extreme Classification* those methods are time consuming (excepting ECOC framework) as most of the time involves as many linear functions as there are classes (K). On the contrary methods relying on the *Divide and Conquer* paradigm offer much better time complexity, often having a prediction time being logarithmic w.r.t the number of classes. In the following section, we present those methods and their main advantages to leverage *extreme classification* challenges.

2.2.3.2 Divide and conquer paradigm

Divide and Conquer framework consists in dividing classification into low costly classification tasks.

One way to achieve this purpose is to use hierarchical structures as binary tree structure allowing a theoretical prediction time logarithmic according to K the number of classes.

This framework relies on dividing recursively the label or feature space, learning classifier for each of those subdivisions and inferring an example applying those classifier sequentially until reaching a leaf.

The hierarchical classification scheme. More formally, let consider $G(V, E)$ a directed graph with V the nodes, E the edges and $|V| = N$, for each node :

- $L = \{l_1, l_2, \dots l_N\}$ correspond to a set of labels subset associates to each node i.e. $l_i = \{l_{i,1}, l_{i,2}, \dots l_{i,|l_i|}\}$ with $l_{i,k}$ is a set of labels.
- $A = \{f_1, f_2, \dots f_N\}$ such that $f_i \colon F \to \{0, 1\}^{|l_i|}$

We add the following constraints:

$$(v_i, v_j) \in E \iff l_j \in l_i \tag{2.12}$$

$$l_i = \{\bigcup_j l_j | (v_i, v_j) \in E\} \text{ and } \{\bigcap_j l_j | (v_i, v_j) \in E\} = \emptyset \tag{2.13}$$

Intuitively we just describe the parents' children relation such that each child has a label subset that is included in the parent label's subset and that all children have disjoint labels set. We also said that the hierarchy is strict when $\forall f_i \in A, \forall x \in F \, \|f_i(\mathbf{x})\| = 1$ meaning classifiers select one unique child. We also define the arity of hierarchical classifiers, the classifier is said having arity k when $\forall f_i \in F, |l_i| = k$.

Hierarchical classifiers. One of the main challenges when addressing classification within the hierarchical frameworks is finding the partition of labels i.e. finding an appropriate set L. In few applications the hierarchical information or the label taxonomy is well defined such that the set L is explicitly given in the data information. However it is rarely the case, thus the main objective is to find an adequate criterion in order to partition the labels or feature space. One can use criteria based on entropy (*C4.5* Quinlan 2014). Different criteria have been developed each one correcting or adding information to others, for instance Rokach et al. 2008 proposed criterion measuring incorrect labeled examples and correct the sets accordingly. However if tree classifiers are rather fast in prediction according to the number of classifiers, each classifier can be time costly according to the number of considered features. To accelerate that process, one can use the projection of data to learn a less costly classifier i.e. using a feature space F' such that $dim(F') << dim(F)$. In this way, S. Bengio et al. 2010 proposed to embed features into a low-dimensionality space that preserve label similarities.

Node balance. Another difficulty is to ensure the balance between edges of the vertices. In certain cases having unbalanced trees can lead to use a linear number of classifiers to get prediction. In order to fit to the data distribution and ensure balance in each node, the *LOMTree* algorithm Choromanska et al. 2014 proposes an online building tree algorithm based on entropy based criteria with logarithmic depth. To avoid to create orphan nodes (nodes with none or few examples/classes associated) the authors proposed a swapping node algorithm allowing to swap nodes to a more convenient path.

The limitation of binary trees. Previous tree architecture was mainly defined using binary trees i.e. each node get exactly two children, thus labels subset for a node l_i is decomposed into two subset $l_i = \{l_i^l, l_i^r\}$. Involving to mainly create balanced binary tree (number of right sub-nodes equals more or less number of left) and allow to mostly focus on the partition criterion. Thus possibly obtain a non-natural distribution i.e. cannot ensure that we can equally divide into two equal partitions with respect to the number of labels or examples.

If mainly hierarchical approaches are based on binary trees, recent works proposed to extend to more complex structure such as Jernite et al. 2016. This works propose to learn trees with any chosen arity suggesting maximizing likelihood of examples representation.

In all previous methods leaf nodes are corresponding to one unique label, otherwise multi-label classification cannot use those straight forward structure.

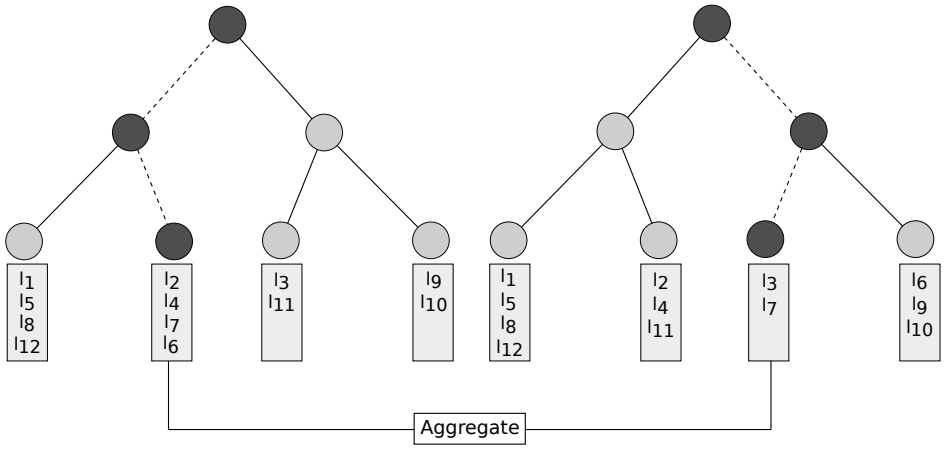


Figure 2.2 – Examples of forest for multi-label classification as performed in *MLRF* algorithms, each leaf corresponding to a subset of labels. The prediction is then obtained by merging labels of each leaf reached in the tree classifiers (typically a vote) in order to obtain a rank over the labels.

Multi-Label hierarchical classification. Dealing with multi-label setting, hierarchical approaches need some adaptation. One can imagine to associate to each node a label and then predicts according to the path from the root to the leaf. However this method gets some limitation particularly when the hierarchy is not strict and is unknown, and learning ensemble of hierarchical classifier is frequently preferred (named forest). It is the way addressed by *MLRF* hierarchical classifier (Agrawal et al. 2013) a multi-label large-scale algorithm using in addition a forest of trees. In addition, *MLRF* considers associating to each leaf of the hierarchy not a unique label but a set of labels (illustrate in figure 2.2). In this works each tree is learned until a node contains less than $log(K)$ labels. For inference, authors proposed to aggregate each leaf reached to get score on each label and thus produce a ranking. Indeed, as describe in section 2.1.4, ranking metrics are often preferred to strict accuracy. For tree construction, the criterion remains simple, left side and right side are decided by a separator over the features by minimizing left-hand side and right-hand side probability to get a label i.

In a more general way, *divide and conquer* paradigm mainly rely on finding partitions. Indeed, reducing for each example the number of labels candidates will speed up the prediction. The general framework *LPSR* introduced by Weston et al. 2011b address the partitioning problems considering any type of classifiers. The purpose of this works is to reduce the number of labels to compare with. In each node the objective consists of partitioning the feature space F such that examples lie in different partitions, thus each partition is associate to only a small number of labels. To partitions the space the authors consider a *label scorer*, i.e. a function taking an example and a label and produces a score for a label (can be a classifier or directly the ground truth). This last approach relies on new criteria for partitioning, authors proposed a hierarchical application of the partitioning method using a linear *label-scorer* (Weston et al. 2011a), which allows fastening prediction

time up to $61\times$ without loss of significative performance loss compared to the others approaches.

The ranking objective. One of the issues of optimized objective functions rely on optimizing an accuracy cost and not a ranking one. Thus, an error on the first-ranked label and on the nth one will get the same meaning according to the cost function. Intuitively, there is less consequences make an error for a label ranked at the tail of the predicted label vector compared to one in the head of the predicted label vector. Some metrics take into account the rank of the prediction such as the `nDCG` (see section 2.1.4). It is one of the issues stated by (Prabhu et al. 2014). This issue has been thus addressed in (Y. Wang et al. 2013) proposing a new partitioning formulation directly optimizing a ranking loss leading to improve significatively prediction performances.

Tail labels prediction. Multi-label corpus often relies on two types of labels, the head labels, labels that occur in numerous examples and tail labels occurring in only few ones. The vast majority of approaches are thus relying on predicting accurately head labels that are much easier to retrieve since classifiers are merely better at generalizations when considering a lot of examples to learn with. Thus, an improvement of previous methods have been presented in *PfastreXML* (Jain et al. 2016) promoting *tail labels* (few represented ones) classification and using loss based on propensity score function (refer section 2.1.4 for details on the metric).

If *PfastreXML* and *FastXML* tree building algorithms allow obtaining a good purity in each leaf in maximizing ranking loss function, the prediction procedure is not directly equivalent to the optimized loss. In *FastXML* the prediction procedure is a K-nearest neighbor search using training documents reaching the same leaf as support for the prediction. In order to tackle the issues, a recent work offers a new partition method named *Graph Partitioning Tree GPT* (Tagami 2017b) leading to partitioning nodes while keeping most of the neighbors (with respect to the labels) in the same partition. The approach relies on building a graph of tail labels nearest neighbors and learning the linear partitioner in each node in maximizing probability of neighbors reaching the same child node. This process led to improve performances on an important number of large-scale datasets.

The recent proposed method *CRAFTML* (Siblini et al. 2018) reached the top performances of hierarchical multi-label classification by using forest like classifiers using both, label projection (using k-means for children construction) and feature projection to partition the nodes and learns efficient classifiers.

Recently the method *PARABEL* suggests learning an algorithm that halfway rely on trees and One-vs-all approaches by learning hierarchical model where examples can fall in several children (relaxing the 2.13 constraint). Each leaf corresponds to a set of labels and examples can join several of them. They also introduce a new building tree algorithm based on hierarchical k-means *LPSR*. In order to get final prediction, a *multi-class SVM* is learned on each leaf then they aggregate the score obtained by the SVM leafs.

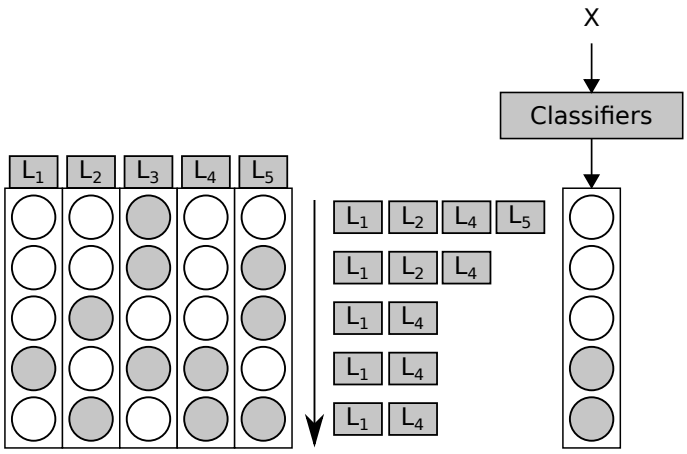


Figure 2.3 – Examples of Bloom filters process for multi-label classification. Each labels get an associated code, classifiers from features space returns a code vector. The prediction is depicted in the figure in removing successively classes where bits are positive (gray circles) in labels representation and negative (white circles) for example representation.

Close to the *Divide and Conquer* paradigm, *Bloom filters* (Bloom 1970) offers another approach based on dividing label space: this approach consists similarly to *ECOC* learning a set of binary classifiers such that all positive bits appear on the label representation (see figure 2.3). The work proposed in (Cisse et al. 2013) relies on Bloom Filter for extreme classification. Authors introducing a new hash function for label representation based on label co-occurrences instead of randomized ones and show performance improvements compared to *Binary Relevance* approaches.

Tree cascade error. In missing-label classification tree architecture is similarly employed. The work *SwiftXML* (Prabhu et al. 2018a) provides a tree construction to leverage multi-label classification issues using in each node classifiers features and labels as input.

Although that all hierarchical approaches efficiently address prediction time complexity challenges, many of them are not robust to internal nodes misclassification. Indeed, any error on a node will often lead to wrong prediction (excepting in *PARABEL* where classifiers redundancy is allowed between same depth nodes). Mostly, hierarchical method classifiers making a unique error will frequently lead to a "cascade of errors". To reduce this risk and improve robustness some hierarchical methods rely on computing nearest neighbors on the leaves such as the *FastXML* approach or rely on learning large forest of classifiers. Indeed, *nearest neigbhor* based algorithms are often considered as robust and still in use in classification in important industrial application. To improve those neighbors based classifiers, ensuring that similar examples (according to their labels) are lying close (according to a chosen metric) in neighbor search space should provide better performances. Hopefully, this is the aim of representation based approaches, those approaches are consequently reaching top accuracy performances. In the following section,

we will give an overview of representation classification methods (we will introduce them deeper in section 2.3) .

Representation learning for XML

In search of solving the miss-classification issues caused by the "cascade error" in hierarchical classification, representations learning is identified as an effective solution. Representation learning aims to map an input space (or feature space F) to a latent one such that the embedding space gets better properties in order to perform classification. In most cases, the objective is to find a m mapping function from the input space to another space, the label's prediction is then processed using for example *Nearest Neigbhor Search* or *Linear Classifier* (for linear classification, an objective should be to find a mapping function to a space where documents are linearly separable). More formally let have $m : F \to F'$ with F' the representation space (typically but not only $\mathbb{R}^M$), $s_Y : Y \times Y \to \mathbb{R}$ a similarity between label vector, and $s_R : F' \times F' \to F'$ a similarity function on the representation space. In most cases the objective rely on finding m such that $\forall(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), (\mathbf{x}_3, \mathbf{y}_3) \in \mathcal{D}$:

$$s_Y(\mathbf{y}_1, \mathbf{y}_2) \geq s_Y(\mathbf{y}_1, \mathbf{y}_3) \implies s_R(m(\mathbf{x}_1), m(\mathbf{x}_2)) \geq s_R(m(\mathbf{x}_1), m(\mathbf{x}_3)) \tag{2.14}$$

With this constraint, $\arg\max_{y_j} s_R(m(\mathbf{x}_i), m(\mathbf{x}_j))$ ensure that $\mathbf{y}_j$ is the closest label vector from $\mathbf{y}_i$. A legit question is why preferring a representation space instead of the feature space F. The objective is twofold :

- Two examples in the feature space are not necessarily close in terms of labels similarity. Thus constraining a representation space to respect the constraint 2.14 will lead to better prediction performances.
- Generally, the size of the feature space is large (see section 2.1.2), thus learning classifiers from a lower representation space if $dim(F') << dim(F)$ will drastically reduce processing time.

Those two objectives are addressed in the vast majority of works. Early approaches to decrease the dimension are mainly focused on finding an embedding of label vectors preserving a label similarity (named *label space dimension reduction*). To deal with, a first idea would be to project label vectors using a linear function that preserve label similarity.

$$\begin{aligned} m_l \colon \{0,1\}^K &\to F' \\ \mathbf{y} &\mapsto W^t\mathbf{y} \end{aligned}$$

To retrieve labels one should first map input (from feature space) to label representation ($m_f \colon F \to F'$) and finally design a reconstruction method to get labels from representation space (m_l^{-1} inverse of the m_l function). This approach is proposed in (Hsu et al. 2009)

in sampling a random linear function under several constraints. If this last method works efficiently, it is due to the low dimensionality of representation space. Moreover, the distribution is chosen according to the sparsity of the label space Y to guarantee an isometry between Y and F'. Despite reducing the time computational cost, a main drawback relies on processing m_l^{-1} which is based on a time costly matching pursue algorithm.

Low rank labels representation for fast reconstruction. A simple increase of efficiency could be to design faster reconstruction method such as linear function. Particularly, as the label matrix $L_{\mathcal{D}}$ is very sparse, we can consider it low rank i.e. the label matrix can be written on a basis of size $|F'| << K$. As it is low rank it should be possible to find a singular value decomposition $L_{\mathcal{D}} = U\Sigma V^t$ and $U^t L_{\mathcal{D}} = \Sigma V^t$ (U is $K \times K$, and V is $N \times N$, both are orthogonal) with null singular value. This assumption has been largely studied those last two decades in numerous works. Particularly, the methods *PLST* (Tai et al. 2012) address representation through this principle encoding labels based on the main labels space directions.

The main idea of the paper relies on using instead of random projections the M singular vector associate to the M largest singular values such embedding is given by $m_l = U^t_{1:M} L_{\mathcal{D}}$. To decode the embedded vector one can use the inverse linear transformation $(U^t_{1:M})^{-1}$ which in this particular case is $U_{1:M}$.

Despite a well grounded approaches nothing guarantee that the regression to learn m_f should accurately approximate the representation space. Thus the improvement *CPLST* (Y.-n. Chen et al. 2012) proposed to learn representation being aware of the feature space in learning the projection of the label matrix and a regressor jointly. Those approaches refer to the domain known as label/feature space dimensionality reduction. Recently an extensive review of those methods has been proposed by Siblini et al. 2019.

Overfitting. Overfitting when addressing extreme classification is a common issue to a large range of model, particularly in embedding methods. Thus restraining feature space to only relevant features is a key point to avoid overfitting. Similarly to previous approaches, one can approximate the feature space with a low rank assumption, and thus only considering principal components of F. This solution is proposed in the method *LEML* (Yu et al. 2013) using additionally to the decomposition a regularization terms on the trace of the low rank matrix. In addition authors showed that the approach can efficiently work in missing labels setting thanks to the increase of stability and generalization.

Tail label awareness. Moreover the storage still is costly and *SVD* computation for such a large matrix remains a real bottleneck to perform the training step. A final drawback is the tail label awareness. Indeed projecting the label space while keeping only the largest eigenvectors associate to largest eigenvalue leads to discard the tail label information. This drawback is stated by Akbarnejad et al. 2016, showing that it is particularly problematic in the missing-label setting.

Time consuming methods. Despite reducing the dimensionality of the feature space, all related methods still apply a reconstruction which is at least linear to the number of labels. To tackle this issue actual approaches are considering using different embeddings for a sub-region of the feature/label space in addition to decoding approximation to ensure low time complexity (Kush Bhatia et al. 2015).

Divide and conquer and representation approaches. Recent representation based methods rely on *Label Space Dimensionality Reduction* but also address divide and conquer paradigm jointly. Indeed, first splitting the input or output space and then learn representation for each of the generated subset should fasten the prediction.

The *SLEEC* (Kush Bhatia et al. 2015) algorithm proposal relies on dividing the feature space, allowing performing classification on extremely large label space. It also contributes to learn representation where instead of using linear reconstruction function, K-NN based algorithm is preferred leading to better prediction performances. The main contribution of SLEEC are the following: 1) Drastically reduce the learning/inference time by using *K-Means* algorithms to split data; 2) Using K-NN on the learned representation for examples in the same partition in order to improve performances; 3) Design a new method for learning labels-features dependent embedding using label neighborhood. Moreover, to ensure prediction robustness the method makes use of several learners which prediction are aggregated. Most of recent approaches rely on the following principle: 1) split the input/label space; 2) learning embedding on document/label subsets; 3) process prediction based on nearest neighbors search.

Using the K-*NN* for prediction does not directly lead to lower time complexity. The complexity depends on the amount of training examples rather than the number of labels. Fortunately, the number of examples is not exceeding much the number of labels in multi-label extreme classification corpora (see section 2.1.2), moreover, speed up can be obtained with approximated nearest neighbors search.

Thus, the paper *AnnexML* (Tagami 2017a) does not consider the low rank assumption as it is rarely the case for large label space, particularly when the number of labels per examples is large. The approach uses *stochastic gradient* to optimize the constraint 2.14, efficients neighbor searches and labels based partitions. Contrary to *SLEEC*, partitions depend on the correlation of labels based on the cosine similarity between label vectors and it learns classifiers to associate each example to a partition. The speed up is then obtained by neighbors search approximation in the representation space such as the tree search or graph search. Prediction complexity is additionally lowered by the post-partitioning of the features space.

Learning efficiency. If learning hierarchical classifiers is pretty fast due to the partition of the label space (classifiers are considering only a small number of classes), on the contrary the training step in representation approaches could be time consuming considering the loss optimized over all possible labels.

To tackle the learning time complexity issue one could use the negative sampling principle : considering a random batch of negative labels for loss processing (used in Tagami 2017a). More recently the works *SNM* (Reddi et al. 2018) proposed a new loss family (named *Ordered Weighted Loss*) closely related to negative sampling approaches. This loss family involves many theoretical guarantee such being a valid surrogate of the $0/1$ retrieval loss. Moreover, it empirically demonstrates the effectiveness of the approach in obtaining state-of-the-art results in terms of prediction performances and convergence speed.

Deep Neural Network for XML. If mainly linear functions are used to map input to representation space, deep learning framework offer the possibility to learn more complex function. Such a neural network *DXML* (W. Zhang et al. 2017) proposes to learn examples representation using a multilayer perceptron (`MLP`) or *Deep Walk* (Perozzi et al. 2014) proposes networks for graph representation.

Moreover, deep neural networks can directly deal with raw features such as a sequence of words. Thus methods based on *CNN* or *RNN* can lead to better prediction performances since they consider sequences. This last kind of approach is proposed in *BOW-CNN* (J. Liu et al. 2017) using `CNN` for classification and an adapted loss function for training. Although very good prediction performances and representation allowing lower computational cost, those algorithms still are linear wrt. the number of labels.

Structure of the representation space. Obtained pertinent representation structure should be characterized by how well the constraint 2.14 is respected. Recent works showed that for different data structure, embedding using specific manifold and metrics could help to get better structure of the representation space. Particularly, *hyperbolic* manifold (Nickel et al. 2017; B. Chen et al. 2019; Ganea et al. 2018) theoretically and empirically proved their robustness for embedding data such as graph structured ones or hierarchical ones. Moreover, a large amount of *eXtreme* classification corpus respect a hierarchical structure regarding labels (see section 2.1.4) we thus believe that using such embedding in the multi-label classification task would improve performances.

Method	Main Principle	Inference time depends on
ECOC	Learning a binary code for each labels	Number of examples/size of codes
DISMEC (Babbar et al. 2016)	OVA, Sparse weight	Sparsity/Number of labels
PDSparse (I. E. Yen et al. 2017)	OVA, Sparse weight	Sparsity/Number of labels
Slice (Jain et al. 2019)	OVA	logarithmic to the number of labels
K-NN	Nearest Neighbors	Number of examples
KD-TREE	Nearest Neighbors	Logarithmic WRT number of examples
LSH	Nearest Neighbors	Sub linear to the number of examples
FastXML (Prabhu et al. 2014)	Hierarchical	Logarithmic w.r.t labels/Number of trees
PfastreXML (Jain et al. 2016)	Hierarchical	Logarithmic w.r.t labels/Number of trees
SwiftXML (Prabhu et al. 2018a)	Hierarchical MLML	Logarithmic w.r.t labels/Number of trees
CRAFTML (Siblini et al. 2018)	Hierarchical	Logarithmic w.r.t labels/Number of trees
PARABEL (Prabhu et al. 2018b)	Hierarchical OVA	Sublinear w.r.t labels/Number of trees (few needed)
PLST (Tai et al. 2012)	Representation	Number of labels
CPLST (Y.-n. Chen et al. 2012)	Representation	Number of labels
LEML (Yu et al. 2013)	Representation, MLML	Number of labels
SLEEC (Kush Bhatia et al. 2015)	Representation, clustering	linear w.r.t Number of examples
AnnexML (Tagami 2017a)	Representation, clustering	sub-linear w.r.t Number of examples
DEFRAG (Jalan et al. 2019)	Representation, clustering	

Table 2.4 – Methods summary with the type of approach and their prediction time.

2.3 Learning embeddings for eXtreme classification

In the contributions we focused on representations approaches for *extreme classification* type of corpora. We indeed believe that properties of representation based methods make it a good challenger to efficiently tackle *extreme classification* challenges. In addition, this kind of approach today stands at the state of the art in prediction performances. In this section we present the different approaches to face large-scale classification through embedding methods. We first introduce the error correcting output coding (ECOC) based approaches which despite do not perform the best propose many interesting qualities particularly regarding the prediction time. We then present large scale embedding methods using Euclidean space and subsequently explain and give insight on the construction of the representation space. Finally, we present hyperbolic space as two of our works rely on this kind of embedding. In the meantime, we present recent hyperbolic approaches and explain their advantages in order to embed multi-label data.

2.3.1 Faster proccessing and discret representations

A particular type of representation approach relies on learning discrete examples representation. The benefits of those approaches mainly rely on the decoding time. Their low embeddings size and the properties of discrete and finite representation space make it faster at the decoding step. If in this document we refer to those methods as embedding based approaches those approaches could be similarly viewed as *one-versus-all* type of approaches.

Error Correcting Output Codes :

At the origin of the *ECOC* in machine learning, the aim was to associate with each labels a large binary vector to ensure robustness. Later approaches have reduced the representation size such that using representation size to be smaller than the number of labels to discriminate.

To understand the main principle, consider a partition formed by multiple-linear separators such that in each partition there is only elements annotated with one unique class. In some case the number of separator is insufficient to ensure this property, thus more of them are needed to respect it. Figure 2.4 depicts the concept, where neither class is separable using two linear separators but are using three.

However, we should remark that to encode K labels only $log(K)$ partitions are necessary. Hence, subsequent work relied on this property by learning few classifiers. The principle in this case relies on choosing for each class a code word of size T with $T << K$, K being the number of classes. After that for each "bits" of the code word learns a binary classifier that split the data into two or more groups. For prediction, one can compute the output of the T classifiers and associate the closest class code. Different methods are thus proposed to associate the tuples made up of classes and codes such as maximizing

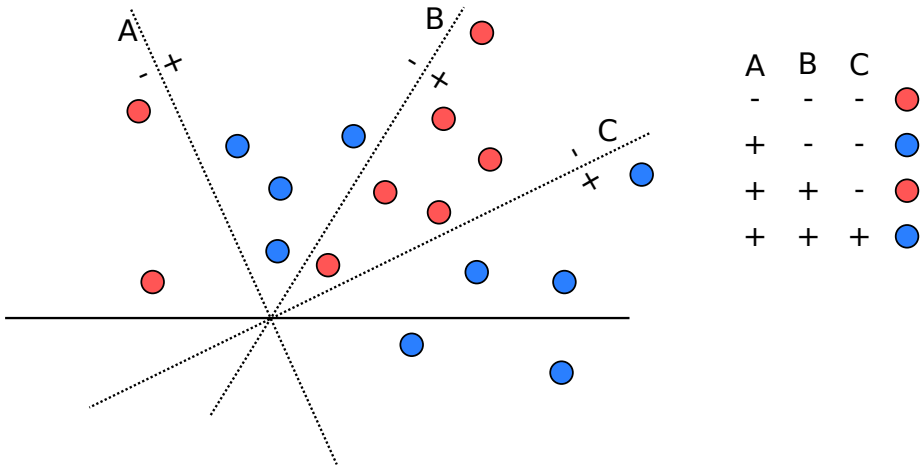


Figure 2.4 – The initial concept of ECOC, learning more classifier than classes to handle the non-separability of documents according to their classes.

Table 2.5 – Example of trichotomy based ECOC

Label	C_1	C_2	...	C_T
1	1	0	...	-1
2	0	0	...	-1
3	-1	1	...	-1
⋮	⋮	⋮	⋱	⋮
K	1	-1	...	-1

the hamming distance between code vectors. To sum up the principle of *ECOC* learning we can decompose the procedure as following :

1. For each class chose a code word.
2. Learn for each component of the code word a classifier.
3. Predict using similarity search the closest codeword.

Lowering complexity. Theoretically, a low size for representation is sufficient since only a logarithmic number of components are required to encode K elements. However, the ideal case does not really apply facing real data. In practice the methods involve $C \times log(K)$ classifiers keeping the complexity in $O(log(K))$. Compared to one vs. all classifiers that need $O(K)$ learning ECOC is clearly efficient.

Extend to trichotomies. A variation of previous ECOC could be to consider ternary codes, two values for partitioning and one for unconsidered examples, this ECOC approach is proposed by Allwein et al. 2000 considering only a subset of classes for each classifier (a partition is not covering all the space). In this case,this partitioning is named trichotomy and the code matrix is of the form $M = \{-1, 0, 1\}^{C \times K}$.

Multi-label and the ECOC principles :

In multi-label setting the *ECOC* counterpart could be seen as the Group Testing methods. Similarly, the objective is to train M classifiers where each of them code for a subset of potential labels. The objective is then to learn classifiers where each one code for a small number of labels, i.e., let $A \in \{0,1\}^{M \times K}$ (called the ground testing matrix) with $A_{ij} = 1$ meaning that the ith group gets the labels j. The lines of the matrix are k-sparse (having at most k non-zero entries). Thus the objective is to learn a classifier which is positive for a group if it contains the labels, the representation is then the selection vectors (concatenation of classifiers prediction). The prediction is thus obtained by aggregating selected groups, this method is typically implemented in Ubaru et al. 2017, who randomly constructing the ground testing matrix (under certain guarantees on the structure).

If those methods are seldom used, we believe that it could lead to address multi-label challenges with efficiency in time and precision. Despite discrete based learning approaches are appealing, optimization can be difficult since it is not possible to use standard optimization methods such as gradient descents based. Moreover, considering low representation size or a few number of classifiers will not lead to a sufficient expressiveness since only two decisions are possible (-1, 1). On the counterpart using continuous representation methods (learning representation for each example in $\mathbb{R}^n$) is much more informative and methods based on it are more robust to errors.

2.3.2 XML based continuous representation

Although reducing time complexity is quite necessary when facing the large-scale challenge, learning correlations between documents could turn out to be important especially for multi-label settings. Capturing dependencies between labels could be difficult with the expressiveness of binary based representation. Today, multi-label classification is led by representational approaches using Euclidean space for representations in low-dimensional space. The objectives of those methods rely on capturing similarities between documents to ensure similarly labeled documents to have a close representation according to the optimized metric. To this end, those last years many methods have been developed such as label space reduction, sparse representation or low dimensionality embedding.

Label space reduction :

The label space reduction consists of learning compressed representations of labels. If $Y \subset \{0,1\}^{N \times K}$ correspond to the label matrix for all elements of the considered dataset, then label space reduction methods assume that it exists in a matrix $U \subset \mathbb{R}^{K \times C}$ where $C << K$ that describe the labels. Then, the representation of each label's vectors is given by $r_i = U^t Y_i$ and decoded by $(U^t)^{-1} r_i$. The objective of classification is now to learn regressor $f : \mathbb{R}^M \to \mathbb{R}^C$ that best approximate label representations of $U^t Y$.

To ensure that this decomposition is possible, previous works consider that the label matrix is low rank, i.e $rank(Y) < K$ (PLST Tai et al. 2012). The principle relies on SVD decomposition where Y is expressed as $Y = U\Sigma V^t$ where instead selecting the entire matrix U to get labels representation, only the M first vectors of the matrix corresponding to the eigenvectors with the highest eigenvalues Σ are selected. Finally, the prediction for an example x_i is given by $U_{1:M}^{t^{-1}} f(x_i)$.

However, although those methods success to obtain label representation that well describes labels space, no guarantee are given to learn the regression successfully (function f). The CPLST (Y.-n. Chen et al. 2012) proposes to improve PLST through the addition of feature information. The approach introduces a method to learn both together, the representation of label space and the regressor. In this work the representation is given by $V_{1:m}Z$ where $Z = Y - \frac{1}{N}\sum_{i=1}^{N} Y_i$ and V is obtained by SVD $Z^t HZ = U\Sigma V$ with $H = XX^+$ (X^+ the pseudo-inverse of X).

Applying this method to get representation consists of minimizing the following problems:

$$\min_{W, VV^t = I} \left\| XW^t - ZV^t \right\|_F^2 + \left\| Z - ZV^t V \right\|_F^2$$

Ensuring to find a compromise between label representation and regressor. Finding best representation for labels is not the only objective of XML, but also ensuring that decoding time is low. Partitioning the feature space can lead to better representation and faster decoding. This last drawback is addressed in the *SLEEC* approach (see further).

The K-NN algorithms :

Another approach to face classification relies on K-*nearest neighbor* algorithm: finding for each sample its neighbors in features space and associate labels with respect to the neighborhood. Those approaches are originally suited for mono-label algorithms (Dasarathy 1991). The *Multi-label* approach based on the K-NN algorithm is based on aggregating labels of the neighborhood (M.-L. Zhang et al. 2005; M.-L. Zhang et al. 2007). Those methods are particularly robust when the number of examples is sufficient. Moreover, this classification approach does not need any learning or optimization process to produce rank or labels. However, those models can be particularly inefficient in time complexity with large features space or excessive numbers of examples. Meanwhile, a large number of works have studied this issue and many proposals allow decreasing processing time. To this end, those methods are today often coupled with *Divide and Conquer* approaches or considering low-dimensional embeddings.

In addition to K-NN in standard Euclidean space, we can reduce complexity in specific space such as hamming ones. For instance, (M. Norouzi et al. 2012) proposed a fast approximated search for hamming space based on splitting data into buckets. Same authors latter introduce construction of buckets producing an exact search (Mohammad Norouzi et al. 2013) in root square complexity regarding the number of examples. Typically,

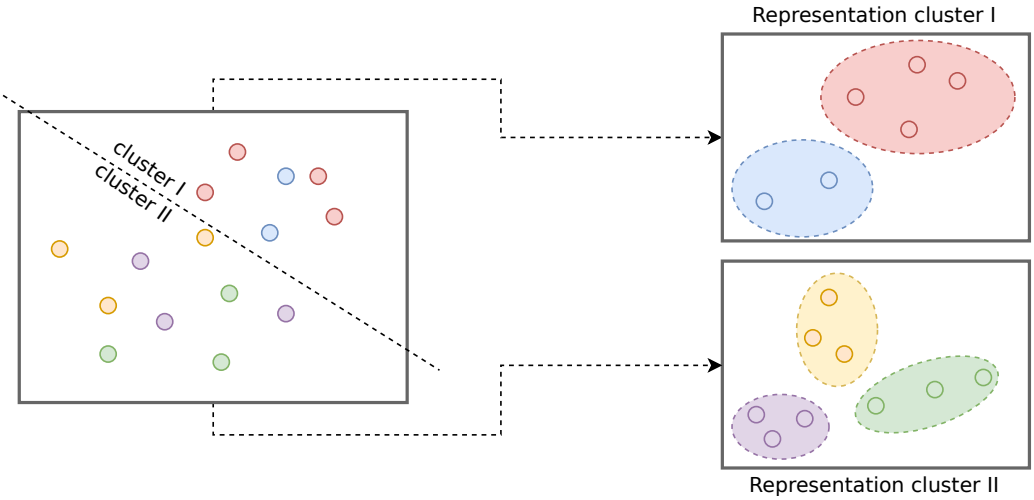


Figure 2.5 – Principle of representation methods to face the *eXtreme classification*. First partitioning the input space and then learn representation on each partition.

this last approach can be used in order to boost decoding time for *ECOC* based approaches using K-NN as decoder.

SLEEC Algorithm :

The SLEEC algorithm (Kush Bhatia et al. 2015) addresses the multi-label representation challenges by: 1) taking into account co-occurrences of document labels; 2) make faster the prediction steps using a K-Means partitioning based algorithms. This framework obtained the state-of-the-art results in 2015 and is still competitive with most recent approaches. The principle is the following :

1. Partitioning the space using K-means
2. Finding embeddings by minimize for each partition $\left\|Y^tY - Z^tZ\right\|_F^2 + \lambda \left\|Z\right\|_1$. Then $Z \in R^{n\times m}$ is the embedding matrix (e.g. each row of the matrix corresponds to the embedding of a unique document)
3. Find a mapping linear function defined by the matrix V such that :

$$\min_V \left\|Z - VX\right\|_F^2 + \lambda \left\|V\right\|_F^2 + \mu \left\|VX\right\|_1$$

For the first minimization (finding embedding matrix), the authors proposed to learn it using a Singular Value Projection (SVP) approach (low rank gradient descent algorithm). In addition Z is designed to be sparse by optimizing the L_1 regularization involving set to zero non-decisive weights. The second minimization is provided by the *ADMM* algorithm (Sprechmann et al. 2013). To predict examples, the authors rely only on the naive K-NN approaches, the prediction time being drastically reduced by the upstream partitioning : a K-Means algorithm on feature space.

Similarly to the *PLST* algorithm reviewed in the previous section, the procedure to learn embedding and regressor involves two different steps. potentially leads to find a suboptimal

regression function. However, accumulating sparse linear functions and partitioning the feature space leads to drastically lowering the time complexity. As partitioning is performed with the K-Means algorithm on the feature space, clustering do not necessarily regroup examples that are similarly labeled. This issue is tackled in the *AnnexML* framework.

AnnexML :

Tagami 2017a propose to address *SLEEC* main issues stated in the previous section : the partitioning considering only the documents features and the low rank prior. It proposes a new supervised algorithm for partitioning and introduce a relevant loss to preserve documents similarities. One of the main contributions is a new partitioning algorithm taking into account labels by learning a clustering classifier which aims to regroup similarly labeled examples (with respect to cosine distance between label vectors). Thus, it ensures that similarly labeled examples lie in the same partition. Authors also provide adapted approximate neighbors search to fasten the prediction time merging Graph and Tree search.

Once partitioned, optimization is done using a gradient descent maximizing the cosine similarity on embedding. To this end, authors propose to model the conditional probability of two examples :

$$P(\mathbf{x}_j|\mathbf{x}_i) = \frac{e^{cos(\mathbf{z}_i,\mathbf{z}_j)}}{\sum\limits_{k\in\mathcal{N}_c^i\cup\{j\}} e^{cos(\mathbf{z}_i,\mathbf{z}_k)}}$$

With $\mathcal{N}_c^i$ the negative set of examples for the partition c, $\mathbf{z}_i$ representation of documents and $\mathbf{x}_i$ associate document in features space. At the best of our knowledge, *AnnexML* still produces the state-of-the-art performances on several large-scale corpora. We will accurately describe the method in the last chapter (chapter 5) since we propose a similar embedding construction in the last contribution.

Despite embedding approaches produce state-of-the-art performances on a large variety of corpus, it remains unclear which paradigm is the best for large-scale classification. Indeed, the performance of the different type of approaches will depend on the structure of label/example space in the corpus. Typically, tree-based classifiers better perform on corpus within a tree structure among labels. However, recent approaches state the relevance of hyperbolic embedding for structured data representation, particularly in low dimensions compared to its Euclidean counterparts.

Indeed we deeply believe that those manifolds are fitted to capture complex data structure, particularly when a taxonomy among labels or examples exists. In this thesis, we proposed two works based on this geometry with the aim of dealing with structured representations. In the following section, we dispense hyperbolic geometry basics and describe improvement and application of the machine learning community in this field.

2.3.3 Representation learning and hyperbolic space

Learning representation is intimately connected to a prior of data structure. It has been assumed that some representation spaces are better to capture some structure than others. Particularly, recent research has allowed representation learning within *Riemannian* manifolds, thanks to recent advances in projected gradient descent. More particularly, recent works demonstrated the usefulness of hyperbolic manifold to accurately represent data structure as graphs or hierarchies.

2.3.3.1 The hyperbolic space

We call hyperbolic manifolds Riemannian manifolds with constant negative curvature usually denoted as H^n for the n-dimensional *Hyperbolic* manifold. One of the representations of the manifold is the hyperboloid defined by $z^2 - x^2 - y^2 = -R^2$ in $\mathbb{R}^3$. In this thesis we will work on the hyperbolic space through the *Poincaré Disk (Ball) model* (isometry of the hyperboloid model). The Poincaré Ball model is defined by an n-coordinate system such that the manifold is an open ball i.e $x \in \{y| \sum_{i=0}^{n} y_i^2 < c\} = \mathbb{B}^n(0,c)$ (in the following we fix $c = 1$) and the associated metric tensor $g_h = \frac{2}{1-||x||} g_x$. This *Riemmannian* manifold is denoted by the couple $(\mathbb{B}^n, g_h)$ (manifold and its Riemmannian metric). To go further in *Riemannian* geometry and understand concepts more deeply, readers could take a look at the introduction Rouvière 2016.

In this document we propose to make use of the *Poincaré ball* model, where distance, logarithmic and exponential map are defined in closed form.

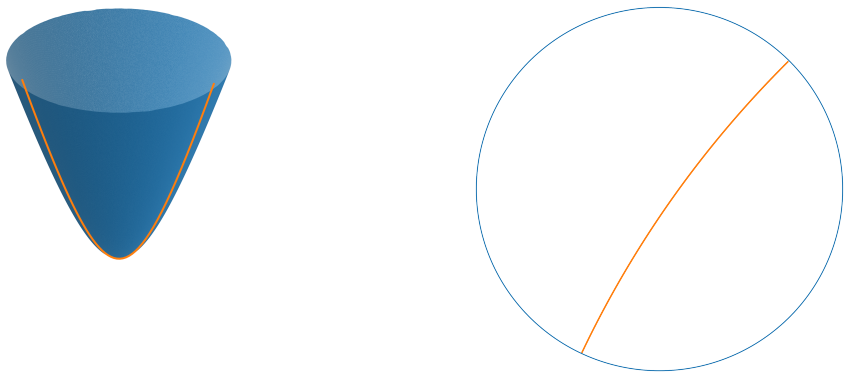
From hyperboloid model to Poincaré ball: To convert a point from Poincaré ball model representation to the hyperboloid one, the following formula is defined:

$$x_i = \begin{cases} \frac{2y_i}{1-\sum_{i=0}^{n} y_i^2} & \forall i \in \{1, 2, \ldots, n-1\} \\ \frac{1+\sum_{i=0}^{n} y_i^2}{1-\sum_{i=0}^{n} y_i^2} & \text{for } i = n \end{cases}$$

And reversely transformation from the *Hyperboloid* to the *Poincaré ball* is processed by :

$$\forall i \in \{1, 2, \ldots, n-1\} \; y_i = \frac{x_i}{1 + x_n}$$

The figure below 2.6 shows geodesics in the hyperbolic space, the first one using the hyperboloid model, the second using the Poincaré ball model.



(a) Geodesic in the Hyperboloid model (b) Geodesic in the Poincare disk model

Figure 2.6 – Same geodesic (orange line) in the hyperboloid and Poincaré disk model

Poincaré Ball Distance : The distance in the Poincaré ball model is defined for two points (vectors) $\mathbf{x}, \mathbf{y} \in \mathbb{B}^n(0,1)$ by :

$$d(x,y) = cosh^{-1}\Big(1 + 2\frac{\|\mathbf{x}-\mathbf{y}\|^2}{(1-\|\mathbf{x}\|^2)(1-\|\mathbf{y}\|^2)}\Big) \tag{2.15}$$

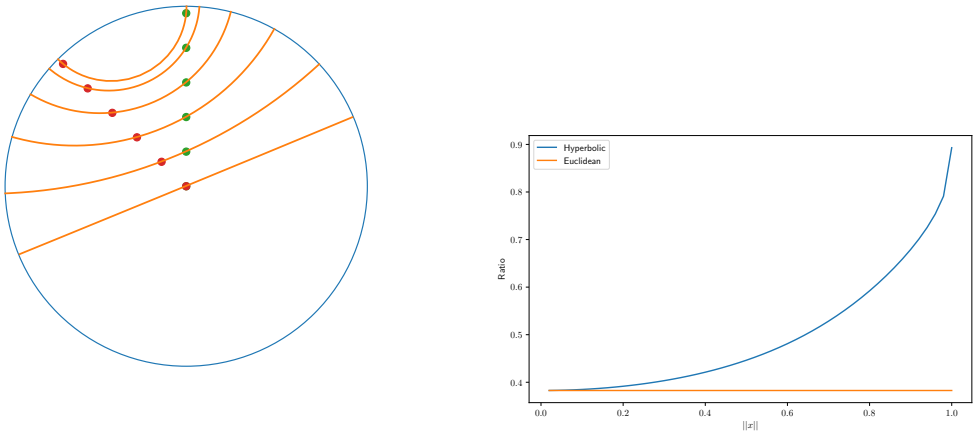
The distance below is intimately similar to trees graph distance, considering the example given by Sa et al. 2018, let have $\mathbf{x}, \mathbf{y} \in \mathcal{B}^n(0,1)$ and the ratio of distance $\frac{d(\mathbf{x},\mathbf{y})}{d(\mathbf{x},0)+d(\mathbf{y},0)}$, when the norm of either x or y tends to one, then the ratio increase. We depict the ratio according to the norm in the figure 2.7b. In a case of trees, this ratio is equals to one when considering the central point at the root of the two other points. In hyperbolic space when the point is close to the boundary, we get closer to the graph distance.

In figure 2.8 the hyperbolic embedding of a simple tree graph is depicted, in the hyperbolic representation we would notice that embedded point preserve the hierarchy order in low dimensionality (i.e. point at a distance d are closer to $d \pm 1$ than $d \pm 2$ in the graph is preserved considering Poincaré ball distance).

Thus, hyperbolic space is a great choice to embed data such as graph structured ones, hyperbolic distance having similarities with graph distance for extremum points.

2.3.3.2 Application Background

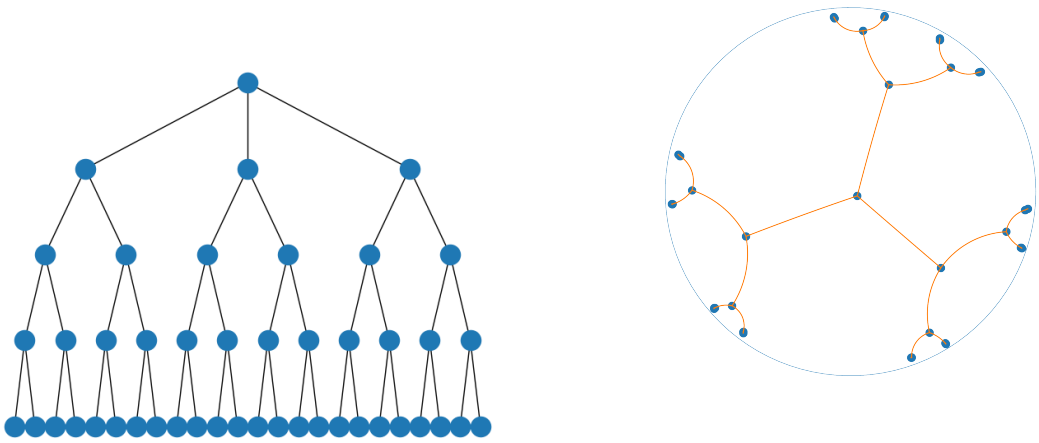
Using hyperbolic space to capture structured information is not a new idea. Several works from the beginning of the year 2017 have provided methods to capture hierarchical information in the Riemannian manifolds. Embedding *WordNet* (a hierarchical corpus of



(a) Poincare Disk

(b) Distance ratio

Figure 2.7 – Figure depicting similarity between Poincaré distance and graph distance. The first figure (2.7a) giving the partial geodesics between two points of similar norms, showing curvature. The second figure (2.7b) compares the ratio $\frac{d(\mathbf{x},\mathbf{y})}{d(0,\mathbf{x})+d(0,\mathbf{y})}$ using *Hyperbolic* distance (blue curve) and *Euclidean* distance (orange curve). Notice that this figure is inspired by the paper *Representation Tradeoffs for Hyperbolic Embeddings* (Sa et al. 2018).



(a) Tree to embed

(b) Embedding of tree in hyperbolic space

Figure 2.8 – Representation of hierarchical data using the Poincaré disk representation.

English words) has been provided by Nickel et al. 2017 proposing a new approximation of gradient descent algorithm using retraction (refer to figure 2b in the appendix). One of the main conclusions of the study is the ability of hyperbolic space to capture hierarchical information using low-dimensional embedding contrasting with Euclidean requirements for same performances.

In the continuity of those works (Sa et al. 2018) proposed a new approach for embeddings hierarchical data, extending the 2-dimensional embedding algorithm to larger dimensions in addition to the hyperbolic counterpart of *Principal Component Analysis* (reducing dimensions with respect to the main direction of data). Although this method outperforms previous state-of-the-art ones, the full hierarchy of the data is required thus restricting its application field.

Works mentioned above evaluate on toy data. Real life tasks have recently been addressed such as the *GloVe* language model adapted for the hyperbolic space in *Poincaré GloVe* (Tifrea et al. 2018). Empirical results tackled *GloVe* Euclidean embeddings for words similarity (looking to the neighborhood of embeddings), analogy (which d is c what b is to a) and hypernym classification (labeling embeddings e.g. *"amphibians, alligators ..."* correspond to *"reptile"*) showing that even in non-hierarchical application hyperbolic embedding could provide competitive performances.

Deep neural architectures and their optimizations on the *Poincaré* model have been recently introduced in *Hyperbolic Neural Network* (Ganea et al. 2018), given a method to perform logistic regression and to embed sequential data by proposing hyperbolic counterpart of many neural layers.

In the mean times, embeddings approaches mapping usual neural layers to *Poincaré* Ball Model have been developed such as in question answering tasks in (Tay et al. 2017).

Mathieu et al. 2019a proposes the hyperbolic counterpart of the `VAE` defining consequently the hyperbolic Gaussian distribution.

A recent trending task using hyperbolic representation is the embedding of graph structured data, such as transposing graph neural networks for node representation (Q. Liu et al. 2019) or convolutional neural networks (Chami et al. 2019).

In classification, the *Hyperbolic Interaction Model* (B. Chen et al. 2019) has provided efficient methods for multi-label classification learning metric pairing labels and documents hyperbolic representation. To embed text sequential data Ganea et al. 2018 make use of hyperbolic recurrent neural networks.

With a large panel of works, hyperbolic space proofs its efficiency in many applications over the last two years. Programming libraries for optimization and visualization recently emerged such as geoopt [1] and geomstats [2] easing the machine learning applications relying on non-Euclidean manifolds.

However, approaches for classification or embeddings large corpus of structured data has not been efficiently established yet.

1. https://geoopt.readthedocs.io/en/latest/index.html
2. https://github.com/geomstats/geomstats

2.3.3.3 Learning representation within hyperbolic manifolds

Thanks to various theoretical advances, the optimization in Riemannian manifolds is possible today. The current optimization approaches are based on projected gradient descent. The general principle relies on computing a gradient on the tangent space at a point and then to project it via the exponential map to the manifold. It will also be necessary to introduce new concepts such as parallel transport when one wants to introduce a "memory" of previous gradient updates. For instance, Bécigneul et al. 2018 developed an adaptive gradient descent such as *Adam* or *AMSGrad* for hyperbolic space . We will give in the appendix of this document the definitions and formulas necessary for these optimizations (in appendix 2). In the associated works, we will mostly use this last kind of optimization to perform the optimization of the representations.

2.4 An uncompleted challenge

Extreme classification has been a challenging task those previous years. In the curse to tackle those challenges, three different paradigm exist : One-versus-all with sparsity or some versus some with the ECOC; the divide and conquer paradigm and hierarchical classification; the representation approaches. Today, methods from all those paradigm get pro and con arguments, OVA strengths being the prediction accuracy, hierarchical ones naturally tackling time complexity challenge and representation methods relying on low learning complexity and prediction accuracy.

However, few representation methods proposed approaches to efficiently capture the structure of labels or examples taking into account co-dependencies of labels or the label hierarchical prior. Representation of data in a compliant space can be one of the solutions, particularly in involving different spaces fitting the structure. In this thesis we propose to tackle those challenges introducing first a stochastic method to face the representation challenge by embedding feature space in a discrete space. To tackle the multi-label setting we proposed a method learning dictionary of co-dependent labels, with representation being the binary vector that aims to select the relevant label subsets. In the following chapters, we study the hyperbolic manifold and demonstrate its relevance to capture efficiently complex structured data. Finally, we study approaches addressing *eXtreme Multi Label classification* challenges through hyperbolic embedding, with these studies we showed the benefits of this Riemanian space to efficiently capture and represent multi-label data structure.

CHAPTER 3

DSNC AND ATOMS NETWORK : LEARNING CODES

Contents

3.1 DSNC : learning binary codes for fast classification 44
3.1.1 Proposed approach 46
3.1.2 Deep Stochastic Neural Code : Encoding 47
3.1.3 Deep Stochastic Neural Code : Decoder 48
3.1.4 Learning procedure 50
3.1.5 Protocol and experiments 52
3.1.6 DSNC: Discussion 57
3.2 Atoms Network : Learning class code-words 57
3.2.1 Model Description 58
3.2.2 Results and Analysis 62
3.2.3 Constraint analysis 63
3.2.4 Atoms Network : Discussion 64
3.3 Better capture structured data 65

The first work presented in this chapter addressed the extreme mono-label classification based on binary representations. Similarly to *ECOC*, the contribution is based on learning a mapping function from the feature space to a set of codes in $\{0,1\}^c$ corresponding to classes. Considering binary representations, we drastically lower the time complexity thanks to fast nearest neighbor search that could theoretically reach $\sqrt{N}$ binary vectors comparison (with N the number of training examples) to produce the prediction. Contrary to existing *Error Correcting Output Coding* (*ECOC*) approaches, we do not attribute each class code randomly but integrate the discovery of codes in the learning step. In this chapter, we first present the *DSNC* (Deep Stochastic Neural Code) model for mono-label classification. Then, we propose an extension of the algorithm for multi-label classification. However, in multi-label setting, the *DSNC* approach did not succeed getting competitive performances. In order to tackle this issue, we present leads we started to develop such as a binary dictionary model that is optimized similarly to *DSNC*. This approach relies on grouping co-dependent labels and classify by selecting most relevant set of labels.

For both proposed approaches, we evaluate the models on real large scale datasets and discuss their performances and drawbacks.

3.1 DSNC : learning binary codes for fast classification

When facing classification within a large number of classes, one challenge is to keep the inference complexity at a reasonable level. Classical approaches frequently have an inference complexity which is linear with the number of categories, leading to the prohibitive processing time for a large-scale corpus. Even if parallelized techniques or use of GPUs can drastically decrease this computation time, the complexity remains very high, especially when the final application gets low computational resources. One approach to lower complexity would rely on using fewer classifiers than the number of classes, as it is in the *ECOC* framework.

In the *ECOC* framework (Berger 1999), a binary code is associated with each category, and a function is learned to map each input to one possible code. Since defining binary codes of size $\log K$ is sufficient [1] to encode K classes, the resulting number of classifiers will be $\mathcal{O}(\log K)$. But those approaches suffer from two main drawbacks: (1) choosing which code to associate to each category is usually handcrafted, even by using random codes or by using complex heuristics (Zhong et al. 2010; Zhong et al. 2013) that need a heavy learning process. (2) Even if codes are carefully chosen, the performances of the resulting techniques are usually lower than classical one-vs-all approaches. Learning the mapping corresponds to train multiple binary classifiers such that each classifier output corresponds to a partition of the classes.

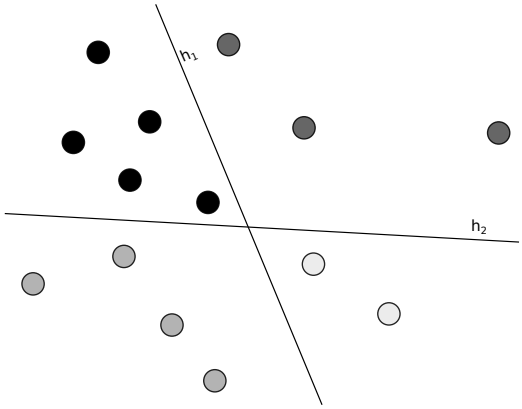
In this first contribution, we propose a new model named *Deep Stochastic Neural Codes* (*DSNC*). The objective is to address the two discussed drawbacks, by learning jointly codes to associate with each class and the mapping function from feature space to codes in an end-to-end fashion.

Initially the aim of learning such codes is to infer the labels in constant time. Indeed, considering a finite number of code vectors and the label associated to each code, one can decode (associate a code to a class) storing all possible combinations of the codes and classes. Thus inference consists to retrieve the code for a document and then find in the stored table the corresponding class. Unfortunately, having a too large binary space dimensionality makes the enumeration of codes intractable.

Thus, we propose several methods to retrieve class from codes. Using K-nearest neighbor search reported the best trade-off between performances and time complexity. If K-NN search is often time consuming as it involves comparing each example to a large set of other examples, many approximate algorithms have improved their efficiency. Particularly, hashing methods split the search space into several partitions (Indyk et al. 1998; Gionis et al. 1999) uses linear separators to partitions the space (refer to figure 3.1). To even go faster, one can recursively partition the search space such that final search space contains very few examples thus leading to a few number of comparisons

1. concrete experiments are usually made with codes of size $k \log K$ where k is a factor such that $k \approx 10$

(*kd-tree* Bentley 1975; Jo et al. 2017). Improving those principles to make data dependent partitions such as maximizing the collisions of close examples within the same hash is still studied to better approximate neighbor search (Andoni et al. 2013; Andoni et al. 2015).



$h_1(x)$	$h_2(x)$	Bucket
≤0	≤0	11→b_1
>0	≤0	01→b_2
≤0	>0	10→b_3
>0	>0	00→b_4

Figure 3.1 – The main principle of LSH algorithms, each point belongs to a bucket depending on the sign of linear functions h_1 and h_2. K-NN is then performed only on samples from the same bucket. In the original work h_i are considered as random *hyperplane*.

Moreover exact efficient methods have been developed for particular latent space such in hamming ones, useful for instance for the search in *Bag of Words* (BOW) representation. This being the case in M. Norouzi et al. 2012 which takes advantage of finite and discrete properties to accelerate K-NN processing using hashing index. Considering binary space to perform exact neighbor search based on dividing data into bucket codes (Mohammad Norouzi et al. 2013) drastically accelerate the decoding process.

Additionaly to the decoding the other problem is how to design the codes. At the contrary to most existing neural approaches using continuous derivation to learn the mapping, the proposed model integrated stochastic units to efficiently sample the code space. Our model is based on a deep neural network where one of the hidden layers is a stochastic layer in which each neuron can take the value 0 or 1 only. Since our architecture involves a discrete non-differentiable layer, we propose a learning algorithm based on the *Straight Through* estimator proposed in (Y. Bengio et al. 2013). Results on different datasets show the effectiveness of the approach in terms of inference time complexity along with accuracy performances making this algorithm competitive when facing a large number of classes.

The contributions are thus threefold:

- We propose a new family of discrete deep neural networks aiming at classifying when the number of categories is large by learning to map inputs to binary codes, and codes to categories.

- We present an end-to-end learning algorithm that do not need any *a priori* heavy process, the model being able to decide by itself which code to associate to which category.
- We show that this model is competitive with classical techniques in terms of accuracy while keeping a low inference complexity.

3.1.1 Proposed approach

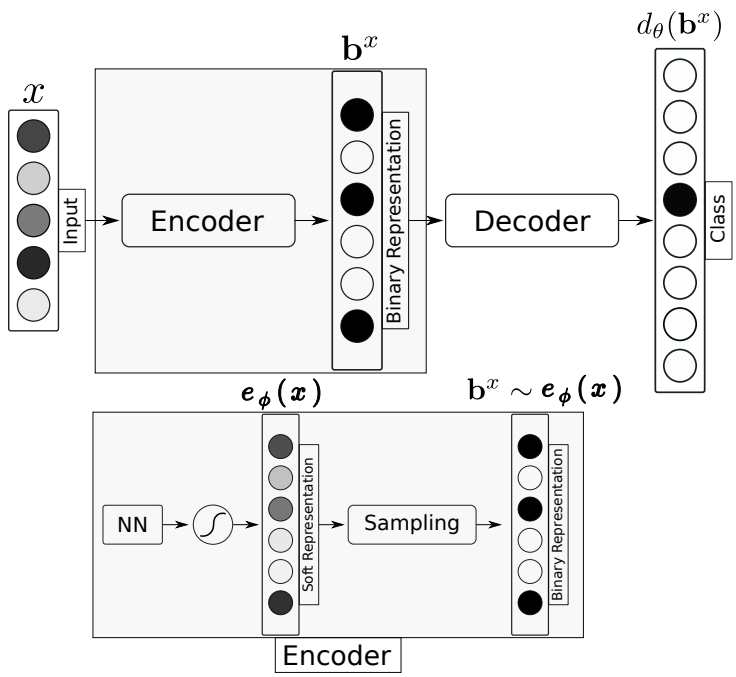


Figure 3.2 – DSNC model architecture, $\mathbf{x}$ the input features vector, ϕ the modeled Bernoulli distribution according to $\mathbf{x}$, $\mathbf{b}^{\mathbf{x}}$ the binary code sampled from the Bernoulli distribution and d_θ the decoder of binary codes (see section 3.1.3)

Usually in classification, large deep learning architectures represent the current state-of-the-art methods when facing a specific type of data. However these usual approaches are rarely time-efficient. In this section we present our model which can be compared to a multi-layer perceptron using a discrete latent space. This property leads to efficiently decoding the latent vectors, we will describe the different decoding approaches in section 3.1.3.

The main difficulty of the model remains in learning an encoder and a decoder considering a binary layer. In the main part of the literature, an usual way is to learn encoder and decoder independently. On the contrary we learn the two functions in a joint way. Moreover all along the learning step we consider the binary representation obtained by sampling over a distribution, thus it gives the advantage of the exploration of the representation

space. We describe the learning procedure inspired by *REINFORCE* (Williams 1992) in the section 3.1.4.

Let consider $\mathbb{R}^n$ and $Y = [0,1]^K$ as the input space and the output categories. We attempt to find a function f such that f gives the probability that for an input $\mathbf{x} \in \mathbb{R}^n$ we obtain a category $y \in Y$. The function f in our case relies on three different steps. The first one corresponds to the encoding step which from the feature space predicts a vector corresponding to the parameters of a binary code distribution. The second step consists in sampling a code from this distribution. The last step named decoding, takes a code as input and outputs a class.

Figure 3.2 summarize the different steps: the encoder that from an input feature vector firstly produce a vector in $[0,1]^c$ that correspond to Bernoulli distributions parameters; sampling according to the distributions a binary vector $\mathbf{b}^x$, the decoder which from $\mathbf{b}^x$ produces a class.

3.1.2 Deep Stochastic Neural Code : Encoding

The encoder transforms the input data from the input space $F \subset \mathbb{R}^n$ to a vector in $B \subset [0,1[^c$ corresponding to parameters p of c independant Bernoulli binary distributions. We denote in this paper e_ϕ the encoder parametrized by weights ϕ, $\mathbf{b}^x$ the binary vector drawn according to the distribution parametrized by $e_\phi(\mathbf{x})$. More precisely we model :

$$\begin{aligned} e_\phi : &\mathbb{R}^n \to [0,1[^c \\ &\mathbf{x} \mapsto \sigma(W \cdot \mathbf{x} + \mathbf{a}) \end{aligned}$$

With σ typically given by $\sigma(x) = \frac{1}{1+e^{-x}}$.

Once parameters obtained, we sample over the c defined Bernoulli distributions leading to get a binary vector in $\{0,1\}^c$ i.e. $\mathbf{b}^x \sim q_\phi(\mathbf{b}|\mathbf{x})$. We recall the Bernoulli distribution probability mass function is given by :

$$M(b,p) = \begin{cases} p & \text{if } b = 1 \\ (1-p) & \text{if } b = 0 \end{cases}$$

The probability of a code relying on the c Bernoulli distributions parametrized by encoder outputs is thus:

$$q_\phi(\mathbf{b}|\mathbf{x}) = \prod_{i=1}^{c} \mathbf{b}_i e_{\phi_i}(\mathbf{x}) + (1-\mathbf{b}_i)(1-e_{\phi_i}(\mathbf{x}))$$

However, during the inference step we replace the sampling by an *hard-sigmoid* function on the Bernoulli parameters as we so not need exploration in this step.

3.1.3 Deep Stochastic Neural Code : Decoder

Once the code obtained, the last step consists of decoding a binary vector to produce a label vector in Y, we refer this step as the decoding step. We introduce the three variants of the decoder function in this section. In the following, we will denote the decoder by d_θ.

Linear decoding. We refer our first method as *linear-decoding* which consist of using a linear function from codes to a probabilistic vector of classes. If this method still is linear according to the numbers of labels, it allows verifying the validity of learned codes.

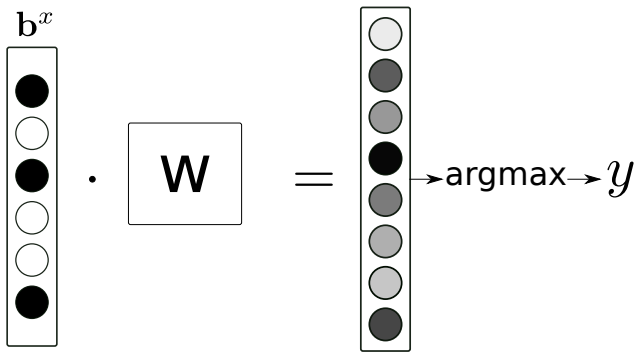


Figure 3.3 – Linear decoding of binaries code, we learn a linear multi-class classifier taking as input code parametrized by a matrix $W \in \mathbb{R}^{c \times K}$. Prediction is then given for a code $\mathbf{b}^x$ by $l = \arg\max W^t \mathbf{b}^x + \mathbf{a}$

Hash decoding. The second decoding method uses a dictionary containing all the possible codes (2^c) and associate each code to class. If we know classes and codes for the documents of the training set, we do not know the classes of codes associated with no training examples. To this end, the linear decoding method is used to get for each code the associated label vector. When the size of the codes is small, we can enumerate all possible codes and apply the linear decoder on them to get the associated classes. More formally, for each code $\mathbf{b} \in \{0,1\}^c$ we store the couple code/class $(\mathbf{b}, \arg\max W^t \mathbf{b} + \mathbf{a})$ into a table. However, considering too large code size involve too much memory needs, thus we consider this last decoding scheme available only when codes size is lower than 30. This decoding method is very fast as we just have to look at the table to directly obtain the prediction (see appendix .1).

K-NN decoding. The last decoder process involves the nearest neighbors search algorithm: in this context we will consider the training codes as the known codes, and the test set as a query set. We can notice that potentially some training documents can be encoded with same codes, in practice there are fewer codes to compare than training examples. To ensure that similarly labeled documents codes are close to each other, we will present in section 3.1.4 a regularization term in the loss. The K-NN decoding

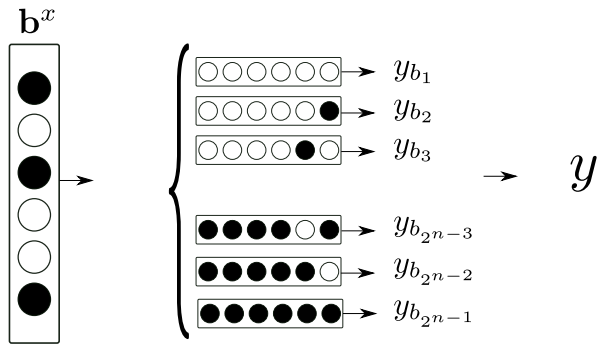


Figure 3.4 – Hashing decoding of binaries code, each code is associated with one class only by forwarding each example using the linear decoding method, then all couples (code, label) are stored in a hash map (or code being interpretable as integers a table could also work). Predicting examples is then performed by looking at the labels associated with the code of an example to label.

approach takes advantage of fast nearest neighbor search using the bucket-based algorithm (Mohammad Norouzi et al. 2013). In this particular case the decoding time complexity relies on the number of training examples. Depending on the dataset considered, the K-NN time complexity can be drastically lowered.

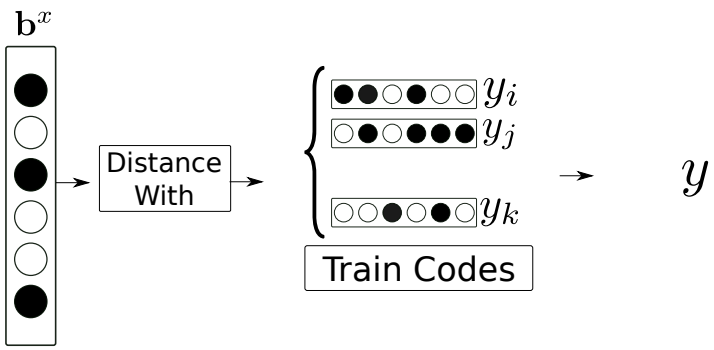


Figure 3.5 – K-NN decoding of binaries code, for each training examples we stored couples (code, label). For prediction we compute the distance of a testing code to stored ones. With a bucket-based methods (Mohammad Norouzi et al. 2013) the number of comparisons is drastically lowered.

Complexity. For all models, we consider using a linear encoder from the feature space $F \subset \mathbb{R}^n$ to the Bernoulli space $[0,1]^c$, thus encoder time complexity is in $O(nc)$. The first decoder we considered is a *one-against-all* based model, which consists of using k classifiers which take into account a class against all others. We implemented it using a linear layer using the loss presented in section 3.1.4. This type of classifier considers a matrix of size $c \times K$, the decoding time complexity is thus in $O(cK)$. This first decoding remains linear to the number of labels. In the end the complexity of this model is in $O(nc + cK)$.

In extreme classification, methods like ECOC use an efficient way to deal with the performance/time complexity trade-off. ECOC encodes the data into predefined bits codes (one by classes), and decodes them similarly to our methods using nearest-neighbors search with different metrics (as hamming distance). The encoder complexity involves c classifiers, and the decoding function get a maximum of k comparison between binary vectors of size c. However an important panel of methods to efficiently perform the nearest neighbors search has been previously proposed. Mohammad Norouzi et al. 2013 proposes a method using a hamming distance in $O(\frac{c\sqrt{k}}{\log_2 k})$. The complexity of the overall model is $O(nk + \frac{c\sqrt{k}}{\log_2 k})$.

With our approach three different complexities can be achieved according to the three different decoding approach. The linear one has similar computation time than an MLP considering the same numbers of layers (for one hidden units $O(nc + ck)$). This method does not allow a sub-linear inference, but can be performed faster due to the use of binary operation instead of floating ones.

The second approach involving a hashing function allowing our model to decode in $O(1)$ and have an overall complexity in $O(nc)$. However this method is not scalable when considering a large size for codes.

The last decoding function involves the nearest neighbor search, achieving at least similar complexity to ECOC. We get multiple codes for each class contrary to the ECOC which mainly considers one code for each class. Let r be the numbers of different codes used which admits the following upper bound $r < min(2^c, |\mathcal{D}|)$. Inferring the classes of the training set allows obtaining a decoding complexity of $O(\frac{c\sqrt{r}}{\log_2 r})$. At the end, the complexity of this method is $O(\frac{c\sqrt{r}}{\log_2 r}) + nc$.

3.1.4 Learning procedure

In this section we describe the procedure to learn the binary encoder, presenting the *REINFORCE* algorithm and the *Straight-Through* estimator.

Learning through gradient descent. Learning such a model needs to fit over the data e_ϕ and d_θ functions. Thus the error associates to this model can be written as :

$$J(\phi, \theta) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}} \mathbb{E}_{\mathbf{b}^x \sim q_\phi(\mathbf{b}|\mathbf{x})} \mathcal{L}(d_\theta(\mathbf{b}^x, \mathbf{y})) \tag{3.1}$$

With $\mathbf{b}^x$ the binary representation vector sampled over the Bernoulli distributions $q_\phi(\mathbf{b}|\mathbf{x})$ with parameters $e_\phi(\mathbf{x})$ and $\mathcal{L}$ the loss function. In our particular case the negative log likelihood through log-softmax modeling of the probabilistic label vector (see equation 2.10).

Considering the probability of a code according to its document features $q_\phi(\mathbf{b}|\mathbf{x}) = \prod_{i=1}^{c} \left[\mathbf{b}_i e_{\phi_i}(\mathbf{x}) + (1 - \mathbf{b}_i)(1 - e_{\phi_i}(\mathbf{x}))\right]$ we can now estimate the gradient of the error

function using the REINFORCE procedure. Thus, an approximation of the gradient of the error using an M trail Monte Carlo approximation is obtained by:

$$\nabla_{\phi,\theta} J(\phi,\theta) = \frac{1}{M}\sum_{i=1}^{M} \sum_{\substack{(\mathbf{x},\mathbf{y})\in\mathcal{D} \\ \mathbf{b}^x \sim q_\phi(\mathbf{b}|\mathbf{x})}} [\nabla_\phi(log(q_\phi(\mathbf{b}|\mathbf{x}))\mathcal{L}(d_\theta(\mathbf{b}^x), y) + \nabla_\theta \mathcal{L}(d_\theta(\mathbf{b}^x), \mathbf{y})] \tag{3.2}$$

With $\mathcal{D}$ the training dataset containing tuples $(\mathbf{x}, \mathbf{y}) \in F \times Y$ respectively the input data and the categories. The theoretical proof of the gradient approximation below will be found in (Williams 1992).

However REINFORCE approach involves a long learning time and does not scale well in a large action space. More recent alternative methods have been developed to approximate such gradient problem. To approximate the gradient we propose to use a variant of the Straight-Through-Estimator (*STE* Y. Bengio et al. 2013) which obtained better performances than reinforce approach over the handwritten number (*MNIST*) classification task.

The *STE* allows us to estimate the gradient over a non-differentiable function according to an error function L. The principle relies in considering sampling for the forward step and gradient of the identity during the backward one. Thus the update of the parameters is produced as follows:

$$\theta_{t+1} = \theta_t - \sum_{\substack{(\mathbf{x},\mathbf{y})\in\mathcal{D} \\ \mathbf{b}^x \sim q_\phi(\mathbf{b}|\mathbf{x})}} \nabla_{\theta_t} \mathcal{L}(d_{\theta_t}(\mathbf{b}^x), \mathbf{y}) \tag{3.3}$$

$$\phi_{t+1} = \phi_t - \sum_{\substack{(\mathbf{x},\mathbf{y})\in\mathcal{D} \\ \mathbf{b}^x \sim q_\phi(\mathbf{b}|\mathbf{x})}} \nabla_{\mathbf{b}_x} \mathcal{L}(d_\theta(\mathbf{b}^x), \mathbf{y}) \nabla_{\phi_t}(e_{\phi_t}(\mathbf{x})) \tag{3.4}$$

Although the *Straight-Through* estimator is a biased estimation of the gradient, convergence is obtained at a reasonable time while with *REINFORCE* algorithm optimization becomes intractable to obtain acceptable convergence (especially when the number of codes is large).

Structured Latent Space. Our third decoding method uses the nearest-neighbor search approach. It is better in this case to ensure similar labeled documents are close to each other in the representation space. Thus, we propose to use in addition a regularization function which ensures that similar examples in terms of label have similar codes. To guarantee it, we propose to minimize the *intra-class* distance and maximize the *inter-class* distance. Let consider the two following sets :

$$\mathcal{D}_{intra} = [((\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j)) \in \mathcal{D}^2 | y_i = y_j]$$

$$\mathcal{D}_{inter} = [((\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j)) \in \mathcal{D}^2 | y_i \neq y_j]$$

With this configuration we now consider the error function :

$$
\begin{aligned}
J(\phi,\theta) =& \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}\mathbb{E}_{\mathbf{b}^x\sim q_\phi(\mathbf{b}|\mathbf{x})}\mathcal{L}(d_\theta(\mathbf{b}^x,\mathbf{y})) \\
&+ \beta\mathbb{E}_{[(\mathbf{x}_i,\mathbf{y}_i),(\mathbf{x}_j,\mathbf{y}_j)]\sim\mathcal{D}_{intra}} L_2(e_\phi(\mathbf{x}_i), e_\phi(\mathbf{x}_j)) \qquad (3.5) \\
&- \gamma\mathbb{E}_{[(\mathbf{x}_i,\mathbf{y}_i),(\mathbf{x}_j,\mathbf{y}_j)]\sim\mathcal{D}_{inter}} L_2(e_\phi(\mathbf{x}_i), e_\phi(\mathbf{x}_j)) \qquad (3.6)
\end{aligned}
$$

Where the minimization of the intra-class distance is represented by the 3.5 and the maximization of the inter-class distance is represented by 3.6 and β, γ the coefficient associates to each of those additional loss functions. In the following of the section, we will refer equations 3.5 and 3.6 as structure losses.

3.1.5 Protocol and experiments

Datasets. In order to validate the proposed model, we selected three datasets with a large number of categories.

The first one named *ALOI* (Galar et al. 2013) is a dataset of 1000 classes of sift features extracted from image objects. It remains a simple dataset to fit but which is non-linearly separable. We split the dataset into train/validation/test randomly with respectively 80%, 10% and 10% of data.

The second ones namely the *DMOZ* dataset composed of short text description in a bag of word embedding. This dataset contains 12294 classes and a large vocabulary input size. We also split DMOZ into sub-dataset of 1000 classes. We name that last dataset *DMOZ-1K*. For both *DMOZ* datasets, for training performances we deleted few documents containing more than 500 words.

The last dataset is *IMAGENET* designed for the *imagenet-2012* classification contest of 1000 image categories. Instead of using raw images, we got the features from the pre-trained model *resnet-152* (He et al. 2016).

Optimizer and training methods. Poor performances have been shown using the *REINFORCE* approach for learning the models, thus all the experiments below are trained with the *STE* gradient estimator. We train all the models using an adaptive gradient descent optimizer namely Adam (Kingma et al. 2014) since it leads to a better convergence. The encoder and decoder functions experimented are linear ones. For the classification loss, we choose the negative log-likelihood criterion

$$
l(d_\theta(\mathbf{b}^x,\mathbf{y}) = -\sum_{i=0}^{k} \mathbf{y}_i log\left(\frac{e^{d_\theta(\mathbf{b}^x)_i}}{\sum_{j=0}^{K} e^{d_\theta(\mathbf{b}^x)_j}}\right)
$$

In mono-label setting, one should notice that only one of the terms is non-zero.

Evaluation. We report on this chapter the evaluation of the models through accuracy on the different datasets. We also compare our model to others usual methods with the MLP as the multilayer perceptron using one hidden units, the ECOC methods as the random ECOC classifier and the OAA as the one-against-all classifier. We also give performances depending on the use of the *structure* loss during the learning step.

Table 3.1 shows the accuracy obtained for the test set. We denote two variants of our model in the tables, the letter R means that we regularize to minimize the intra-class and maximize the inter-class distance, the letters NN means that we used to decode the nearest neighbors search. Using low dimensionality in the first two rows of each column can use the constant decoding time named *hash-decoding*. If the time is constant, it remains difficult in this particular setting to obtain competitive performances. To perform better we require larger dimensionality. In this setting all our experiments tackle the random-code based approach for a similar complexity excepting for *Imagenet* corpus. However none of the methods presented outperform the deep MLP model with the same configuration thought the gap is relatively small on *dmoz-12k*. As we can observe in the table, in the case of dmoz, using the regularization helps to improve the results of the two decoding approaches. On the contrary on *ALOI* data, performances tend to be a little lower with the regularization method. Because of the non-linear separability of *ALOI* classes, the baseline *ECOC* drastically fails, indeed contrary to others datasets only 128 features describe each image. In this last case our method benefits of having many codes for a class, as regularization lowering this effect, performances are better without using it.

DSNC Analysis

binary code quality In this section we describe the influence of the code size and the regularization over the latent learned binary space. The distance intra-class and inter-class are summarized in table 3.2 with the label yes/no informing about the use of the structure losses during the learning step.

distance	R/size	24	36	60	100	200
intra-class (train)	YES	1.1 ± 1	2 ± 2	2.8 ± 3	8.82 ± 6	18.4 ± 11
	NO	2.3 ± 1	4.9 ± 2	10.9 ± 3	23 ± 5	59.3 ± 10
inter-class (train)	YES	11.78 ± 0	17.5 ± 0	29.4 ± 0	47 ± 1	94.4 ± 2
	NO	11.6 ± 0	16.8 ± 0	26.6 ± 0	40.8 ± 1	72.3 ± 2
N codes	YES	$8k$	$13k$	$15k$	$24k$	$26k$
	NO	$15k$	$23k$	$28k$	$29k$	$30k$
intra-class (test)	YES	4.8 ± 3	7.4 ± 4	12.1 ± 6	20.1 ± 10	37 ± 19
	NO	5.9 ± 3	9.4 ± 4	16.6 ± 6	29.6 ± 9	63.9 ± 17
inter-class (test)	YES	$10.8 \pm 1.$	16.8 ± 1	27.0 ± 2	45.0 ± 3	90.8 ± 5
	NO	$10.5 \pm 1.$	15.5 ± 1	25.6 ± 2	41.6 ± 4	80.6 ± 9

Table 3.2 – DMOZ-1K distance intra-class (test-dataset)

Table 3.1 – Accuracy of the proposed model DSNC and the two baselines on the different datasets. The gray background indicates a constant decoding time.

	model	DSNC				MLP		ECOC
dataset	code size	linear	NN	linear+R.	NN+R.		Reg.	
DMOZ-1K	12	31.772	22.425	33.156	23.492	39.204	39.716	10.738
	24	39.736	36.779	41.326	37.028	48.496	48.748	22.144
	36	42.928	41.208	45.414	42.776	51.48	51.716	27.589
	60	46.84	44.734	48.742	47.41	53.532	54.084	33.850
	100	49.164	46.807	50.984	49.888	54.954	55.658	38.212
	200	51.058	49.065	53.052	52.355	56.262	56.908	41.33
DMOZ-12k	12	15.09	15.2	15.34	15.24	20.05	20.18	3.8
	24	21.15	18.79	21.64	19.27	28.74	29.3	17.591
	36	24.83	22.21	25.95	24.17	32.14	32.05	21.71
	60	28.36	25.97	29.71	29.96	35.41	35.36	25.079
	100	30.42	27.6	31.98	33.94	37.45	37.23	27
	200	32.22	29.32	33.95	35.95	39.12	38.25	27.91
	400	31.96	33.02	33.6	38.65	36.71	39.75	28.08
ALOI	12	34.918	34.84	33.328	33.366	82.04	81.992	1.53
	24	67.92	63.66	66.064	64.19	88.174	87.27	4.21
	36	76.73	74.19	75.84	79.94	89.91	88.18	5.81
	60	83.478	83.19	82.66	81.74	92.22	89.78	9.03
	100	88.014	88.58	87.606	88.72	99.96	90.79	13.8
	200	91.288	91.88	90.542	91.26	95.15	92.41	22.4
IMAGENET	12	1.5	0.82	1.49	0.805	14.6	13.11	12.32
	24	15.6	7.705	9.01	4.595	53.19	53.46	32.42
	36	53.82	36.46	45.74	25.14	59.11	58.85	45.03
	60	60.07	48.61	63	53.655	60.83	63.66	56.5
	100	68.66	63.405	68.81	63.515	65.9	66.81	64.5
	200	70.67	66.935	71.61	68.54	-	67.3	69.76

Table 3.2 shows that the regularization process allows us to get an accurate nearest neighbor decoding method. As a side effect of the loss, we show that the number of training codes is lower when using the structure losses, allowing us to get a better speedup for the decoding step.

Complexity Although our model did not reach the performance of MLP model, it has a much lower complexity. Figure 3.6 presents the theoretical time complexity to

decode using the nearest neighbor search for the DMOZ-12k dataset. However only codes of size lower than 24 allow storing the entire table of codes and classes and allow constant decoding times. Performances for such dimensionality do not allow to get relevant performances for both *random-code* method and *DSNC*.

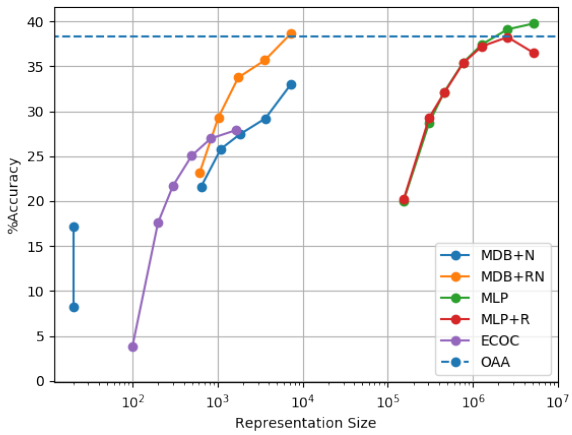


Figure 3.6 – Complexity comparison for DMOZ-12K

Using very short codes our model is equivalent to the *ECOC* approach, however, when considering codes size greater than 60 our model performs better results in a lower time than *ECOC* model. The MLP and OAA (dotted line) methods are close of ours models using a large code size. However, the K-NN decoding approach performs better in terms of prediction time. To conclude our model achieves a good trade-off between accuracy (higher than random *ECOC* method) and time complexity (better than all other tested models).

In figure 3.7 we show the *t-SNE* representation of a sub-sample of 60 classes of the ALOI dataset, using the latent binary representation learned by the DSNC model. Each color represents a class and a class can have multiple representations as more than one code is learned per class. The interesting fact is that codes seem to be grouped by color constituting small compact clusters. This figure tends to confirm the previous analysis of the latent space: the proposed model is able to discover codes which keeps intra-class distance low and it guarantees separability between classes.

DSNC for Multi-Label classification limits and results :

We now propose to evaluate the models in the multi-label setting. However some changes are required to make it work in this setting. We first change the learning decoder function that is now given by $d_\theta(x) = \sigma(W_\theta \mathbf{b}^x)$, where $\sigma(x) = \frac{1}{1+e^{-x}}$. Instead using the negative log-likelihood criterion, we use the binary cross entropy based loss:

$$\mathcal{L}(d_\theta(\mathbf{b}^x), y) = y.log(d_\theta(\mathbf{b}^x)) + (1-y).log(1-d_\theta(\mathbf{b}^x))$$

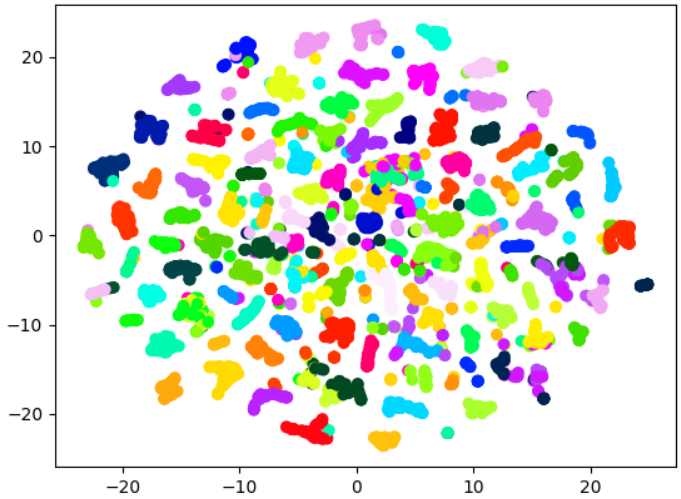


Figure 3.7 – t-SNE representation of the class codes for a sub-sample of 60 classes of ALOI dataset. Codes of a same class are represented by the same color.

In order to get the label vector using the K-NN decoding scheme, we merged all the label vectors of the 10 closest neighbors by summing them.

The results obtained with *DSNC* in multi-label settings are reported in the table 3.3, we added for comparison the algorithm *SLEEC* (Kush Bhatia et al. 2015), one of the state-of-the-art multi-label representation method. As reported our methods in multi-label setting do not perform well, the results did not succeed tackling the baselines in each configuration.

Table 3.3 – Multi-label results for Mediamill using multi-label DSNC

Precision@k	SLEEC	MLP	DSNC-STE
Linear decoding			
P@1	**87.4**	85.2	83.5
P@3	**73.45**	69.8	67.6
P@5	**59.17**	53.5	58.4
kNN decoding			
P@1	**87.4**	84.9	85.2
P@3	**73.45**	70.3	69.8
P@5	59.17	**61.3**	60.6

3.1.6 DSNC: Discussion

The presented contribution is a stochastic neural network architecture for multi-class classification, which learns jointly a function to map stochastically an input to a binary code and a decoder function which associates codes to classes. The stochastic mapping between the input space and the latent binary space allows to efficiently exploring the code space but introduces a non-differentiable layer.

The Straigh-Through estimator is used to approximate the gradient and to learn the parameters. In addition, a regularization is proposed to achieve a better structure of the latent space, with fewer and more compact codes. Thanks to the finite discrete property of the latent space, the proposed model is able to retrieve the class associated with each code with a negligible constant time for small code size and in the generic case with a sublinear time w.r.t. to the number of classes. Experiments show the benefits of our model in terms of accuracy and time complexity. However, today a vast majority of tasks rely on Multi-label data, where documents are annotated with a set of classes and the proposed methods did not obtain good results under multi-label setting. Thus, still using binary properties we propose a new model to fit the multi-label setting in the next section.

3.2 Atoms Network : Learning class code-words

In multi-label classification setting documents are annoted with many classes, thus, it often exists classes that are strongly correlated. Often, documents describing a similar topic will share a large number of labels without being labeled in exactly the same way. For instance, a document describing algorithm could be labeled with different words such as "science", "computer science", "algorithm", "machine learning", "optimization" and so on. It is obvious then that lots of documents will share set of labels, for instance to continue the previous example all algorithms will at least label with "science", "computer science", "algorithm". In the previous section we did not explicitly introduce those correlations, which leads to perform poorly in multi-label setting. Capturing these correlations is important. Moreover the vast majority of hierarchical approaches structure are precisely based on this dependency among labels. However, multi-label corpus with a strict hierarchical structure are uncommon in real life datasets, thus, most of the hierarchical methods are using many classifiers named forests of classifiers. On the other hand, a lot of documents share a fixed set of sub-labels. In this section we propose a framework that aims to predict the subset of correlated labels for a document. The method consists of transforming an example into a set of atoms that describe it, this approach shares similarities with dictionary learning methods where the objective is to subdivide information into many simple ones. More recently a similar framework named *Ground Testing Matrix* (Ubaru et al. 2017) has been proposed with as objectives finding subset of classes and learn classifiers to select the appropriate subsets. Those methods get several advantages, particularly in prediction speed where aggregating selected subsets can be performed time

efficiently. However, building the matrix of subsets is a complex task. In the following we introduce a new approach based on learning in a end-to-end way classifiers and atoms.

In this section we present the research conducted inspired by dictionary learning methods. The main idea consists to learn a set of atoms, such that each atom corresponds to a probability distribution over a subset of correlated labels. The process can be sum-up into two steps, the first one relies on selecting the most likely subsets of labels for a particular example, the second step consists of the aggregation of the subsets of labels. With this process we can ensure a faster prediction mainly correlated with the size of the support of the atoms, i.e., the number of labels encoded by the atoms with a non-zero probability. In order to allow better prediction, we also introduce two kinds of subsets, positive subsets which correspond to the most likely classes of an example, and negative subsets corresponding to the most unlikely classes for example. The main advantage of the approach is to provide control of the time complexity of the inference process by controlling the sparsity of the atoms and thus the number of labels for each atom. The following section presents the procedure to learn both the atom dictionary and the decomposition over the dictionary using a gradient descent based algorithm.

3.2.1 Model Description

Formulation. Let $\mathcal{D} = F \times Y$ the training set, with $\mathbf{x}_i \in F \subset \mathbb{R}^n$ the features of the example i and $y_i \in Y \subset \{0,1\}^K$ the associated label vector considering L labels such that $y_{il} = 1$ if the example i belongs to the lth class and 0 otherwise. For the classification task, we are looking for a function $f : F \to Y$ that predicts the label vector from the description of an example. Our method involves learning atoms: each atom a_j models a probability vector of labels considering only a certain number of labels. In practice, the atoms are grouped in a sparse matrix of dimensions $A \times K$ with A the number of considered atoms. We also consider a function $g : X \to [0,1]^A$ mapping example $\mathbf{x}$ to a distributions vector modeling $P(a_j|\mathbf{x})$ $\forall j \in \{0,1,...,A\}$. In practice we modeled $P(a_j|\mathbf{x})$ by a linear layer with a sigmoid activation :

$$P(\mathbf{a}_j|\mathbf{x}) = \sigma(W\mathbf{x})_j = (\frac{1}{1+e^{W\mathbf{x}}})_j$$

To predict, we first map the input to the probability vector using g, then we select the most likely subsets using a threshold or sampling according to the policy $P(a_j|\mathbf{x})$. Finally, we aggregate the label vectors in order to get a score for each label using the sum of label probabilities. In order to guarantee the positivity of the learned matrix, we use in practice different activation functions such as *rectifier linear units* (`relu`) or *sigmoid*. Once the model learned, the hard value of those activation can be directly replaced in the matrix.

To extend the expressivity of the model we use in practice two matrices containing atoms. The first one contains the set of labels which potentially belong to the example (positive set), and the second one on the contrary contains subset which has to be removed

Algorithm 3.1 Atoms Network prediction

```
procedure PREDICT(x, t)
    labels ← {}
    for j ∈ {1, 2, ... A} do
        if P(a_j^+ | x) = (1/(1+e^{W+x}))_j > t then
            labels ← labels ∪ l_j^+
        end if
    end for
    for j ∈ {1, 2, ... A} do
        if P(a_j^- | x) = (1/(1+e^{W-x}))_j > threshold then
            labels ← labels − l_j^-
        end if
    end forreturn labels
end procedure
```

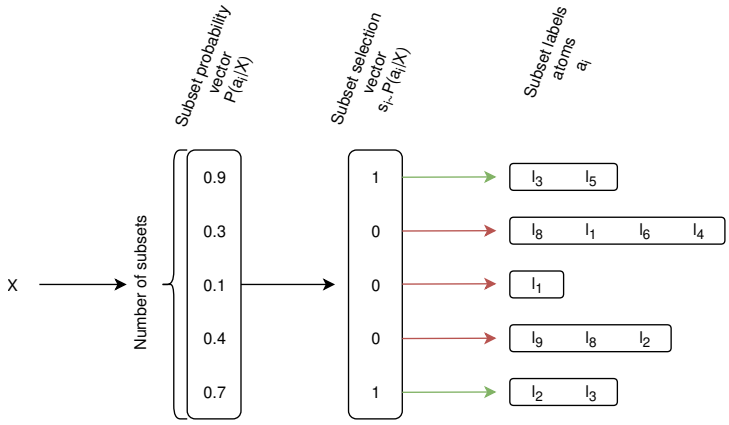


Figure 3.8 – Architecture of the dictionary atoms model. 1) From input get a probability vector which corresponds to the probability of selecting a subset of classes corresponding to the example; 2) sample which subset will be selected according to the previous distribution; 3) Merge the labels corresponding to the labels of selected subset

from the selected ones in the positive set. We denote as a^+ the matrix of potential labels and a^- the other one. We also consider two functions g^+ (resp. g^-) having as purpose to select the positive subset (resp. the negative subset). This process is summed up in the figure 3.9. The intuition behind this architecture is to predict potential labels with a^+ and offering potential correction with a^-, it thus allows to cover larger combination possibilities with a lower number of atoms. We also expect that splitting atoms into two categories will allow retrieving correlations over the labels once the model learned: in the positive a^+ atoms, the labels highly correlated will appear, and at the contrary negative

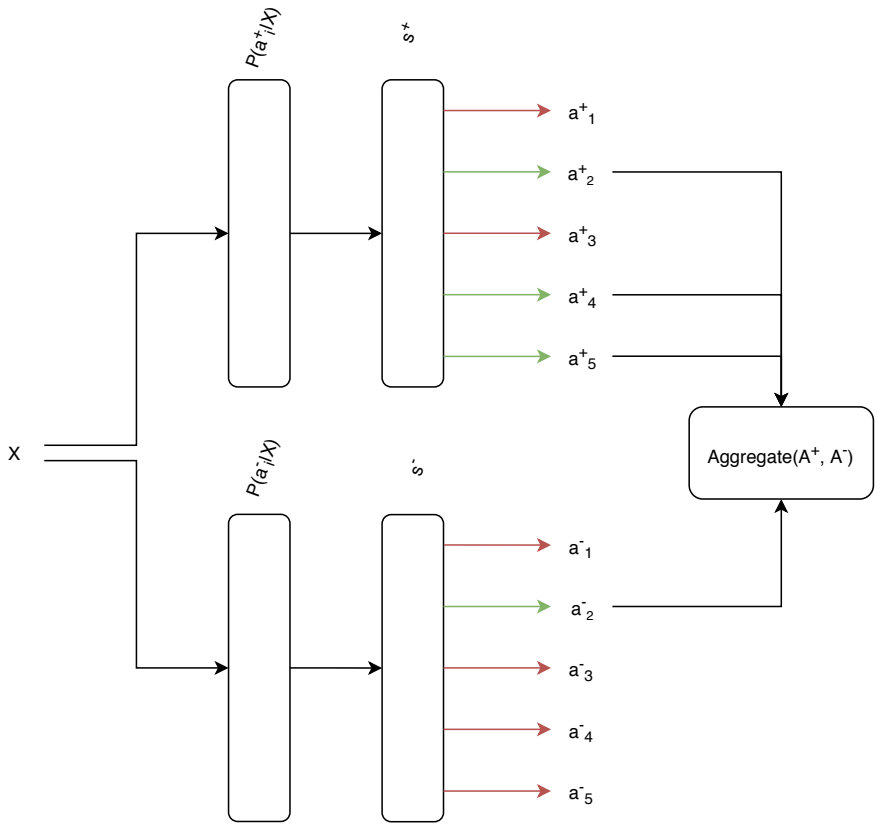


Figure 3.9 – Architecture of the atom dictionary which models the positive and negative atoms

atoms will more likely contains uncorrelated labels. All the prediction process is summed up in the algorithm 3.1 and the figure 3.8.

Controlling complexity through sparsity: The time complexity of the inference of the presented model is directly correlated to the number of selected subsets in average for each example, but also depend on the number of classes belonging to each subset. Let consider $S < A$ the average number of selected atoms, and K the average number of labels in each subset, considering that we use uniformly each atom, then the average complexity of the model is in $O(NA + SK)$. We consider that the real bottleneck complexity is due to the large number of labels, thus we need in the model to minimize K. In order to guarantee the complexity, we will ensure during the learning phase to have the less possible number of labels in each subset, which is equivalent to ensure sparsity of the matrix a^+, a^-.

Learning: Using a sampling procedure to construct the label vector, the function to optimize can be written as the minimization of the expected value over the training set of a loss function which depends on the sampled subset:

$$\underset{g^+,g^-,a^+,a^-}{\arg\min}\ \mathbb{E}_{(\mathbf{x},\mathbf{y})\in\mathcal{D}}\,\mathbb{E}_{\substack{s_i^+\sim g^+(\mathbf{x})\\ s_i^-\sim g^-(\mathbf{x})}}\,\Delta(r((s_i^+)^t a^+,(s_i^-)^t a^-),\mathbf{y}) \tag{3.7}$$

with r the aggregation function, s^+ (resp. s^-) the positive sampled selection vector (resp. the negative sampled selection vector). Different methods can be used to optimize this loss function, we selected according to the previous contribution (see section 3.1) the *Straight-Through* (Y. Bengio et al. 2013) gradient estimator. The STE allows processing a gradient for a Bernoulli sampling layer, thus the model can be optimized by the usual gradient descent algorithm. The error function associated with this optimization problem will be named E_c (for classification error).

Ensuring sparsity: The sub-linear time complexity according to the number of classes is obtained by controlling the sparsity of the dictionary. We experiment several methods to control the trade-off between sparsity and expressiveness.

In the case of *relu* activation for both matrices a^- and a^+, the values lower than 0 are set to zero; in case of the *sigmoid* activation, we use a threshold to replace low value by zeros in both matrices. However without any more constraints, nothing guarantee to get a large number of zero values. To resolve the issue, we propose different constraints based on different regularization terms. The first regularization term we use allows controlling values of atoms matrices by adding a penalization term on the matrices a^+/a^-, we refer to the cost as E_{a^+}/E_{a^-}:

$$E_{a_+} = \lambda_{a^+}\sum_{i=0}^{A}(\alpha_{a^+} - \frac{1}{L}\sum_{j=0}^{|L|} a_{ij}^+)^2$$

With $\alpha_{a^+} \in [0,1]$ a parameter corresponding to the mean value wished for atoms impacting directly the numbers of high values in the a^+ matrix. In a similar way, we can use a similar cost for the negative matrix a^- (parameter α_{a^-}).

Moreover, constraining the numbers of atoms used for example can influence the sparsity of atoms: indeed, limiting the number of selected atoms ensures that few atoms allow describing labels of an example. It does not allow easy correction using other atoms and thus encourages the sparsity. The corresponding error function is thus :

$$E_{s^+} = \lambda_{s^+}\sum_{\mathbf{x},y\in D}(\alpha_{s^+} - \frac{1}{|g^+(\mathbf{x})|}\sum_{i=0}^{|g^+(x)|} g^+(\mathbf{x})_i)^2$$

With $\alpha_s^+ \in [0,1]$ a hyper-parameter controlling the significance of the sparsity over the positive selection vector s (e.g number of selected atoms), we can apply similarly this regularization on g^-. We refer to these terms as E_{s^+} (res E_{s^-}).

Avoiding atoms redundancy: Our objective is to classify using a minimal number of possible atoms. Avoiding redundancy between atoms is thus an important feature. We need to find a loss which allows measuring correlation between atoms. An intuitive way to do it is to minimize the matrix product of atoms matrices with their-self, ensuring the orthogonality of the atoms. The constraint:

$$E_r = \lambda_r(a^+.(a^+)^t + a^-.(a^-)^t)$$

is added to the overall cost function.

3.2.2 Results and Analysis

In this section we evaluate our methods on different extreme multi-labels datasets (refer to section 2.1.2). However it is still difficult to apply this approach on very large datasets such as *DeliciousLarge* or *Wiki10* due to the number of classes, thus the presented experiments concern mainly middle size datasets such as *Delicious*. We compare our method to several other algorithms reported by the *Extreme Classification Repository* (K. Bhatia et al. 2016). Most of Extreme classification models are improved by considering ensemble methods, in the second part we look at the performances when using ensemble of atoms networks and compare performances with state-of-the-art ones. We also explore the specificity of the model and we study the influences of the different constraints in terms of performances but also on the structure of the inferred atoms. Finally, we discuss the weakness of the presented approach.

DSNC prediction performances. In order to leverage our model performances we compare the methods to different state-of-the-art methods, such as *SLEEC*, one-versus-all (*OVA*) and most common label baseline (*MC*) which associates to each example the most common labels of the training set. We also use the naive baseline consisting of associating for each example the k most common labels. We use for our model two kinds of configurations: the best without any sparsity constraints and the best with sparsity constraints which guarantee to have on average 95% of sparsity for a^+, a^-.

Dataset/Clf	SLEEC(15)	OVA	AN (No constraint)	AN (95% sparsity)	MC
Delicious	68/64/56	65/59/53	64/58/53	60/55/50	40/38/35
Mediamill	88/73/59	84/67/53	88/69/54	70/49/38	85/64/50

Table 3.4 – Proposed approaches versus SLEEC (One classifier) and OVA

Ensemble learning: State-of-the-art methods such as *SLEEC*, *FastXML* or *AnnexML* use ensemble classifiers in order to improve the results. The next table presents our approach using an ensemble of classifiers (AN for Atoms Network). In practice representation methods (*SLEEC*, *AnnexML*) use 15 classifiers.

0% sparsity (a^+, a^-)				
Dataset/Clf	SLEEC(15)	AN(1)	AN(5)	AN(10)
Delicious	68/64/56	64/58/53	65/59/54	65/59/54
Mediamill	88/73/59	88/69/54	89/69/54	89/70/56
95% sparsity (a^+, a^-)				
Dataset/Clf	SLEEC(15)	AN(1)	AN(5)	AN(10)
Delicious	68/64/56	60/55/50	64/58/54	64/58/53
Mediamill	88/73/59	70/49/38	74/50/37	70/50/38

Table 3.5 – Dictionary learning with ensemble learning versus SLEEC

Table 3.5 compares results of *Atoms Network* to *SLEEC* using different number of classifiers, it shows that increasing the number of classifiers up than 5 do not allow improvement of the results. We also compare our methods with and without sparsity, with sparsity we remove lot of labels in the atoms such each atoms get only few classes in it thus accelerate the merging of selected vectors, however, sparsity drastically lower the performances.

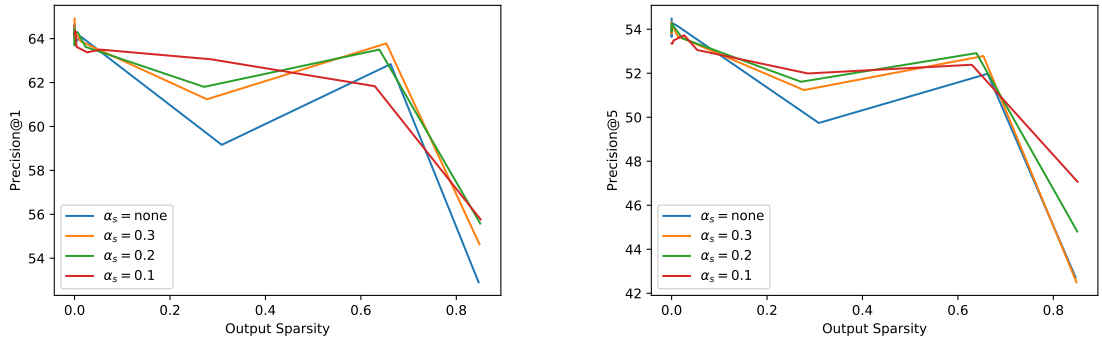
3.2.3 Constraint analysis

Threshold Dictionaries: In these experiments we show the robustness of our method on the sparsity constraints. In the table 3.6 we summarize results obtained by using a threshold on both atoms matrix a^+/a^- to guarantee a sparsity on those matrices. However, although sparsity of matrices allows decreasing time complexity it does not correspond to the overall sparsity: the number of used atoms can be extremely large such that there is no zero values in the final aggregated vector. The table shows the obtained results and the effective sparsity (corresponding to the sparsity of the output label vector) when we use only the constraints E_{a^+} and E_{a^-} with $\alpha_{a^+} = \alpha_{a^-} = 0.05$.

a^+/a^- sparsity	P@1	P@3	P@5	effective sparsity
0%	64	59	54	0%
20%	64	59	54	0%
50%	63	58	54	1.5%
95%	61	56	51	66%
99%	50	45	41	85%

Table 3.6 – Sparsity/Precision depending on the threshold of the matrix a^+ and a^- on delicious dataset (2 learners on *delicious*)

A parameter that can influence the sparsity is the use of the E_s constraint depending on the α_s factor, which limits the number of selected atoms for each example. In the figure 3.10 we show the results of the variation of the α_s parameter ($\alpha_{s^-} = \alpha_{s^+}$ in this experiments) with $\alpha_{a^+} = \alpha_{a^-} = 0.05$. We can especially observe that using higher hyper-parameter value α_s allows to be more stable when the sparsity increases (i.e., higher threshold).



(a) (b)

Figure 3.10 – Precision (@1,@5) and sparsity of the output depending on the selection constraint α_s (*delicious*)

Large unbalanced datasets. Although the model can be considered as competitive in terms of precision performances on *Delicious*, the performances are lower for large datasets especially when there are fewer examples per label. We show the issue with an experiment on the *Wiki10* dataset reported in the table 3.7 with the precision at 1,3 and 5, compared to *SLEEC* algorithm and the naive baseline which attributes to all examples the label vector containing the label scores sorted by label frequency. We notice that precision at 1 decreases with the increase of dimensionality when the precision at 5 increases, the first intuition is that using low dimensionality will enforce the learning procedure to mostly design atoms containing head labels.

	AN(unconstrained)	SLEEC	MC
50	80/50/37	86/73/63	
100	79/49/38	–	80/50/37
200	77/50/40	–	

Table 3.7 – Atoms Network comparison performances (p@1/p@3/p@5) on wiki10 dataset with our model without sparsity constraint (AN), the SLEEC baseline and most common label baseline (MC)

3.2.4 Atoms Network : Discussion

In this works we design a new approach to face multi-label challenges with the aims to create subsets of correlated labels. The process is thus two folds, first design two matrices containing positive and negative atoms i.e., subset of correlated labels and subsets of

uncorrelated labels, secondly learning classifiers to adequately select for each example atoms (label subset). Finally, prediction is then obtained by merging the selected atoms by aggregating positive atoms and withdraw negative labels according to negative atoms selection. To this end, we propose an end-to-end learning approach based on discrete optimization using the Straigh-Through estimator introduced by (Y. Bengio et al. 2013). With this contribution we tested the validity of the following intuition : can we learn the set of correlated labels for large-scale multi-label corpora. At the current state of the work, it is still unclear if it is feasible as the methods could be improved with recent optimization process, change in architecture and aggregation methods for atoms. On large corpus such as wiki10 it is still difficult to tackle the state-of-the-art methods and even to beat the most common baseline using only a restricted number of labels subset. However, this work needs to be taken further condering for instance optimization methods from the non-negative matrix factorization area or different architecture.

3.3 Better capture structured data

In this chapter we proposed two new approaches to tackle the time complexity challenge of extreme classification. The first approach aims to embed data into a binary space and proposes different methods to retrieve classes from discretes embeddings allowing very fast decoding processes in constant time or sub-linear time according to the number of training examples codes. However, the approach fails to reach competitive performances under the multi-label setting. This first work has been published at the ICONIP 2017 conference (Gerald et al. 2017).

The second contribution is still based on learning binary representation. Its objective is to learn subsets of correlated labels and to select for each example the appropriate subset. The selector is in this case a binary representation (1 if subset is selected else 0). To this end we proposed a gradient based optimization algorithm which aims to learn in end-to-end way both selector and subsets. However, this approach did not allow us to tackle the state-of-the-art. Capturing efficiently structured data and embedding is the key point to obtain higher performances. Generally, representation approaches efficiently capture dependencies over labels mostly considering continuous representation of documents. In the following chapters we study continuous embedding to efficiently capture correlation, particularly using hyperbolic representation space well adapted to capture structure we are interested in.

CHAPTER 4

HYPERBOLIC EMBEDDING FOR STRUCTURED DATA

Contents

4.1 Embedding Graph Data within Poincaré Ball Model 67
4.1.1 Riemannian GMM 70
4.1.2 EM for Hyperbolic *Gaussian Mixture Model* : 72
4.2 Hyperbolic Embedding of Structured Data 74
4.2.1 Learning Embedding with communities 75
4.3 Community detection 79
4.3.1 Complexity and Scalability 81
4.4 Community classification 83
4.4.1 Community Classification Results 84
4.4.2 Discussion on Hyperbolic graph embeddings 86
4.5 Hyperbolic embeddings and structured data 88

In the previous work we have highlighted the importance of considering the data structure for multi-label classification. However the proposed methods still lack in capturing labels correlations. If learning binary embeddings or dictionaries with binary atoms remains a difficult task regarding its optimization or its expressivness, recently hyperbolic embeddings have shown their strengh to represent efficiently structured data. As discussed in section 2.3.3, the hyperbolic space gets relevant properties to embed hierachical or graph structures since the metric preserves hierarchical or graph distance. In this chapter we propose to explore hyperbolic embeddings for capturing structured data by proposing to embed graph into hyperbolic space. We introduce an unsupervised approach learning a Gaussian Mixture Model in hyperbolic space with two objectives. The first is to be able to clusterize in the space and the second to help to structure the embedded space.

4.1 Embedding Graph Data within Poincaré Ball Model

The present contribution is concerned with learning *Graph Structured Data* (GSD). Examples of such data include social networks, hierarchical lexical databases such as Wordnet (Miller 1995) and Lexical entailment's datasets such as Hyperlex (Vulić et al.

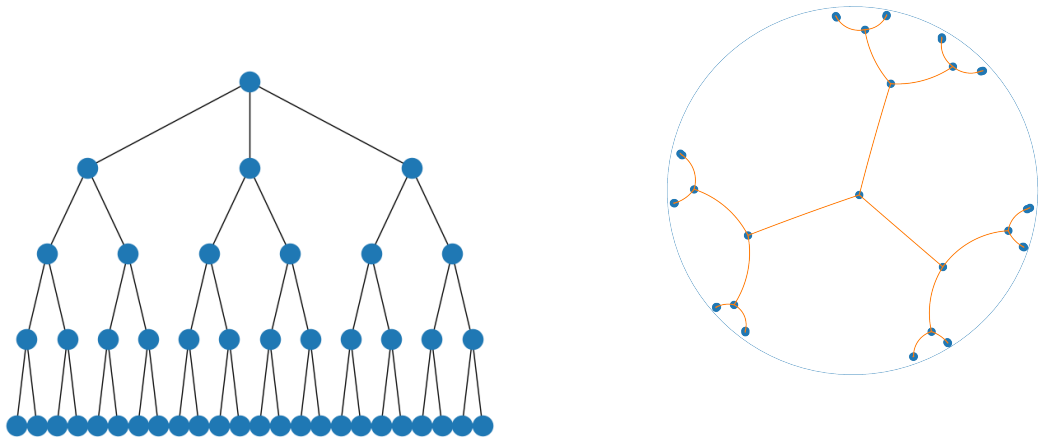
2016). In the state-of-the-art, one can distinguish two different approaches to cluster this type of data. The first one applies pure clustering techniques on graphs such as spectral clustering algorithms (Spielmat 1996), power iteration clustering (Lin et al. 2010) and label propagation (Zhu et al. 2002). The second one relies on two-step and may be called Euclidean clustering after (Euclidean) embedding. First, it embeds data in the Euclidean space using techniques such as Nod2vec, Graph2vec and DeepWalk and then applies traditional clustering techniques such as K-means algorithm. This approach appeared notably in (Zheng et al. 2016; Tu et al. 2016; D. Wang et al. 2016). More recently the *ComE* algorithm (Cavallari et al. 2017) achieved state-of-the-art performances in detecting communities on graphs. The main idea of this algorithm is to alternate between embedding and learning communities through Gaussian mixture model.

Learning GSD received significant achievement in recent years thanks to the discovery of hyperbolic embeddings. Although, for several years it has been speculated that hyperbolic spaces would better represent GSD than Euclidean space (Gromov 1987; Krioukov et al. 2010; Boguñá et al. 2010; Adcock et al. 2013), it is only recently that these speculations have been proven effective through concrete studies and applications (Nickel et al. 2017; Chamberlain et al. 2017; Sa et al. 2018). As outlined by (Nickel et al. 2017), Euclidean embeddings require large dimensions to capture certain complex relations such as the Wordnet noun hierarchy. On the other hand, this complexity can be captured by a simple model of hyperbolic geometry such as the Poincaré disc of two dimensions (Sa et al. 2018) due to the similarity of hyperbolic distance with graph distance (see figure 2.7 in section 2.3.3). Additionally, hyperbolic embeddings provide better visualization of clusters on graphs than Euclidean embeddings (Chamberlain et al. 2017). Recently, deep learning methods for graph structured data have been transposed using hyperbolic space. Hyperbolic Graph Neural Networks (Q. Liu et al. 2019) or Hyperbolic Convolutional Neural Networks (Chami et al. 2019) provided competitive results compared to their Euclidean counterpart.

Given the success of hyperbolic embedding in providing faithful representations, it seems relevant to set up an approach that applies pure hyperbolic techniques in order to learn nodes and more generally community representation on graphs.

Hyperbolic representation of hierarchies and graphs. Figure 4.1 presents an example of hyperbolic representation of a tree, with figure 4.1b the embeddings of the hierarchy depicted in figure 4.1a. As we can notice, contrary to Euclidean counterpart leaf nodes are far from each other. Secondly, norms of nodes embeddings have a specific meaning, deeper a node is in the graph greater its norm is.

This motivation is in line with several recent works that have demonstrated the effectiveness of hyperbolic tools for different applications in brain computer interfaces (Barachant et al. 2012), computer vision (Said et al. 2017) and radar processing (M. Arnaudon et al. 2011). In particular Barachant et al. 2012 used the concept of Riemannian barycenter on the space of covariance matrices to classify brain computer signals. Said et al. 2017 introduced Expectation-Maximization (EM) algorithms for symmetric positive



(a) Tree to embed (b) Embedding of tree in hyperbolic space

Figure 4.1 – Representation of hierarchical data using the Poincaré disk representation.

definite matrices space and then applied it to classify patches of images. Marc Arnaudon et al. 2013 used the Riemannian median on the Poincaré disc to detect outliers in Radar data.

Motivated by the success of hyperbolic embeddings on one hand and hyperbolic learning algorithms on the other hand, we propose in this work the adaptation of *ComE* approach to the hyperbolic space. Two different applications are targeted:

Unsupervised learning on GSD. We present a scheme to learn communities based on Poincaré embeddings (Nickel et al. 2017) and Gaussian mixture model. To this end we derive the theorical Riemannian EM algorithm proposed in Said et al. 2018 for the hyperbolic space.

Supervised learning on GSD. We propose a supervised framework that uses community-aware embedding of graphs in hyperbolic spaces. To evaluate this framework we propose three different tools to retrieve the communities: distance to the Riemannian barycenter, Riemannian Gaussian Mixture Models (GMM) derived from (Said et al. 2018) and a Riemannian logistic regression proposed in (Ganea et al. 2018). For both unsupervised and supervised frameworks, the proposed methods are evaluated on real-data social networks. We also make comparisons with the *ComE* approach (Cavallari et al. 2017) and recent geometric methods (Cho et al. 2018). Moreover, this last application concerns the multi-label classification task, thus allow us to validate the relevance of hyperbolic spaces for classification.

This work is organized as follows. In the first section, we describe tools giving the definition of *Gaussian Mixture Model* in Riemannian manifolds (section 4.1.1). In section 4.1.2 we describe and derive the hyperbolic *Gaussian Mixture Model Expectation-Maximization* algorithm relying on the theorical Riemannian GMM defined in Said et al.

2018. The introduction of the learning algorithm for embedding graph structured data in hyperbolic space is introduced in section 4.2. In section 4.3 and section 4.4.1, we provide experiments, comparisons with state-of-the-art and discussion of both supervised and unsupervised evaluation. Finally in section 4.5 we discuss perspectives of the proposed approach.

4.1.1 Riemannian GMM

Gaussian distributions have been extended to manifolds in various ways (Skovgaard 1984; Pennec 2006; Said et al. 2018). In this chapter we rely on the recent definition provided in (Said et al. 2018) as it comes up with an efficient learning scheme based on Riemannian mixture models. In the next section, these models will be applied to learn communities on graphs. The same distribution was particularly used in (Mathieu et al. 2019b) to generalize variational-auto-encoders to the Poincaré Ball. Moreover (Ovinnikov 2019) proposed an extension of Wasserstein auto-encoders to manifolds based on the same definition of Gaussian distributions. Those works rely on the calculation of a mean and a variance for defining Gaussians of the mixture. In this section, we first describe the formulation of the barycenter, then we introduce the expression of the *GMM* as well as the tools necessary for its calculation.

Barycenter

As a Riemannian manifold of negative curvature, $\mathbb{B}^m$ enjoys the property of existence and uniqueness of the Riemannian barycenter (Afsari 2011). More precisely, for every set of points $\{x_i, 1 \leq i \leq n\}$ in $\mathbb{B}^m$, the empirical Riemannian barycenter

$$\hat{\mu}_n = \text{argmin}_{\mu \in \mathbb{B}^m} \left(\sum_{i=1}^{n} d^2(\mu, x_i) \right)$$

exists and is unique. Several stochastic gradient algorithms can be applied to numerically approximate $\hat{\mu}_n$ (Bonnabel 2013; M. Arnaudon et al. 2011; M. Arnaudon et al. 2011; M. Arnaudon et al. 2014).

Gaussian Mixture Model

Given two parameters $\mu \in \mathbb{B}^m$ and $\sigma > 0$, respectively interpreted as theoretical mean (or barycenter) and standard deviation, the Riemannian Gaussian distribution $\mathcal{G}(\mu, \sigma)$ on $\mathbb{B}^m$, is given by its density:

$$f(x|\mu, \sigma) = \frac{1}{\zeta_m(\mu, \sigma)} \exp\left[-\frac{d^2(x, \mu)}{2\sigma^2} \right]$$

A first interesting and practical property is that $\zeta_m(\mu, \sigma)$ the normalization factor does not depend on μ (Said et al. 2018):

$$\zeta_m(\mu,\sigma) = \zeta_m(0,\sigma) = \zeta_m(\sigma) = \int_{\mathbb{B}^m} \exp\left[-\frac{d^2(x,0)}{2\sigma^2}\right] dv(x)$$

A more explicit expression of $\zeta_m(\sigma)$ has been given recently in Mathieu et al. 2019b as follows:

$$\zeta_m(\sigma) = \sqrt{\frac{\pi}{2}}\frac{\sigma}{2^{m-1}}\sum_{k=0}^{m-1}(-1)^k C_{m-1}^k e^{\frac{p_k^2\sigma^2}{2}}\left(1+\mathrm{erf}(\frac{p_k\sigma}{\sqrt{2}})\right)$$

with $p_k = (m-1)-2k$ and erf the error function with $erf(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$. Figure 4.2 displays some plots of the function $\sigma \mapsto \zeta_m(\sigma)$.

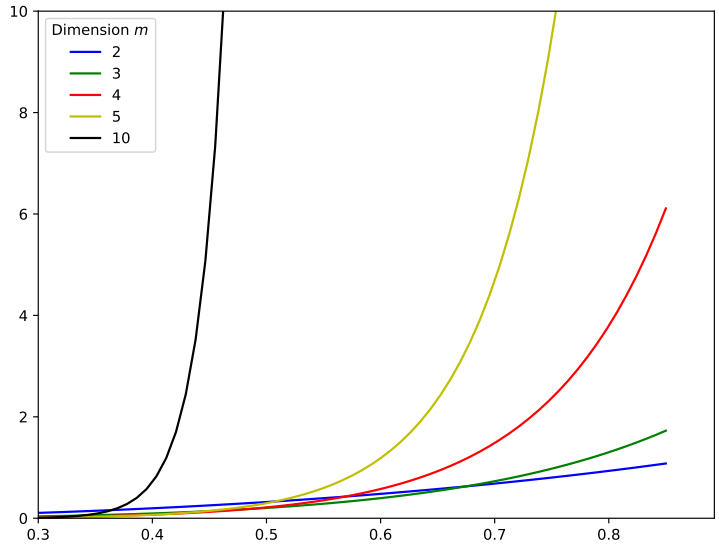


Figure 4.2 – Normalization coefficient $\sigma \mapsto \zeta_m(\sigma)$ for different values of the dimension m of the Poincaré ball

Recall Maximum Likelihood Estimation (MLE) of the parameters μ and σ, based on independent samples $x_1,\ldots,x_n$ from $\mathcal{G}(\mu,\sigma)$ given as follows:

- The MLE $\hat{\mu}_n$ of μ is the Riemannian barycenter of $x_1,\ldots,x_n$.
- The MLE $\hat{\sigma}_n$ of σ is

$$\hat{\sigma}_n = \Phi\left(\frac{1}{n}\sum_{i=1}^n d^2(\hat{\mu}_n, x_i)\right)$$

  where $\Phi : \mathbb{R}_+ \longrightarrow \mathbb{R}_+$ is a strictly increasing bijective function given by the inverse of $\sigma \mapsto \sigma^3 \times \frac{d}{d\sigma}\log\zeta_m(\sigma)$.

We show in figure 4.3 density of two hyperbolic Gaussians in the Poincaré ball model.

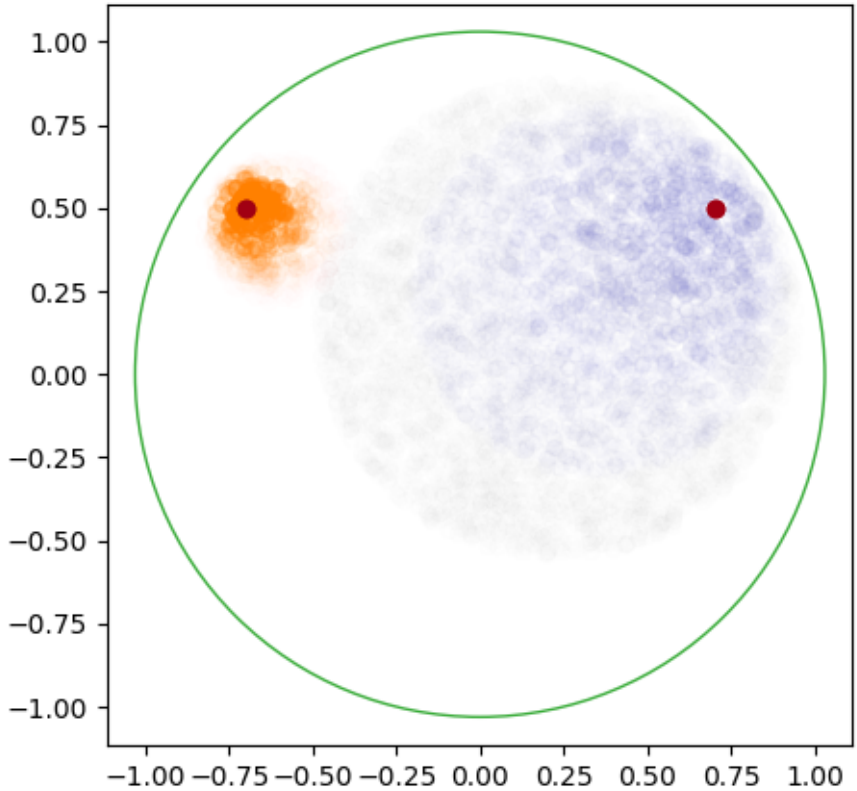


Figure 4.3 – Visualization of two Gaussians within the Poincaré ball model, with the red points being the Gaussian mean.

4.1.2 EM for Hyperbolic *Gaussian Mixture Model* :

In this section, we describe the proposed *Expectation-maximization* algorithm for learning a hyperbolic Gaussian mixture model, firstly by describing the Expectation procedure, and then the Maximization one. One should first remark its similarity with Euclidean GMM, except in the calculation of the normalization factor where no closed form is provided.

Estimate Gaussian Mixture :

Riemannian EM algorithm introduced in Said et al. 2018 has similarities with the usual EM on Euclidean spaces. It is employed to approximate distributions on manifolds by a mixture of standard distributions. We recall the density F of GMM on the Poincaré ball

$$F(x|\mu,\sigma) = \sum_{k=0}^{K} \pi_k f(x|\mu_k,\sigma_k)$$

where $x \in \mathbb{B}^m$, π_k the mixture coefficient, μ_k, σ_k the parameters of the k-th Riemannian Gaussian distribution. The Riemannian EM algorithm computes the GMM that best fits a set of given points $x_1, \ldots, x_N$. This is done by maximizing the log-likelihood of the joint probability of observing each point under the hypothesis that observations are independent. Given an initialization of π, μ and σ, performing EM numerically translates to alternating between Expectation and Maximization steps. Fitting a Gaussian mixture model on Riemannian manifolds have some similarities with its Euclidean counterpart. The main complexity is that there are no known closed forms for the log-likelihood maximization of the mean and standard deviation.

Expectation :

The expectation step estimates the probability of each sample x_i to belong to a particular Gaussian cluster. The estimation is based on the posterior distribution $\mathbb{P}(z_i = k|x_i)$, i.e., the probability that the sample x_i is drawn from the k-th Gaussian distribution:

$$w_{ik} = P(z_i = k|x_i) = \frac{\pi_k \times f(x_i|\mu_k, \sigma_k)}{\sum_{j=0}^{N} \pi_j \times f(x_i|\mu_j, \sigma_j)} \tag{4.1}$$

Maximization

The maximization step estimates the mixture coefficient π_k, mean μ_k and standard deviation σ_k for each Gaussian component of the mixture model.

Mixture coefficient. The mixture coefficients are computed by :

$$\pi_k = \frac{1}{N} \sum_{i=1}^{N} w_{ik} \tag{4.2}$$

Mean. Updating the μ parameter relies on estimating weighted barycenters. For this, it is required to approximate the weighted barycenter $\hat{\mu}_k$ of the k-th cluster:

$$\hat{\mu}_k = \arg\min_{\mu} \sum_{i=1}^{N} w_{ik} d^2(\mu, x_i) \tag{4.3}$$

via Riemannian optimization. We will use algorithm 4.1 (Marc Arnaudon et al. 2013) which has proven effective for radar applications. This algorithm returns an estimate of $\hat{\mu}_k$, it can be view as a gradient descent using the logarithmic map to get gradient on the tangent space and the exponential map to project it on the manifold (see appendix 2).

Standard deviation. The MLE of the standard deviation of the Gaussian distribution is previously defined in Section 4.1.1. Thanks to the property of Φ being strictly increasing and bijective, estimation σ_k can be given by solving the following problem :

Algorithm 4.1 Barycenter computation

Require : $W = (w_{ik})$ weight matrix, $\{x_1, \cdots, x_N\}$ a subset of $\mathbb{B}^m$, ϵ convergence rate (small), λ barycenter learning rate

1: Initialisation of μ_k^0
2: **do**
3: $\mu_k^{t+1} \leftarrow \mathrm{Exp}_{\mu_k^t}\left(\lambda \frac{2}{\sum_{i=1}^{N} w_{ik}} \sum_{i=1}^{N} w_{ik} \mathrm{Log}_{\mu_k^t}(x_i)\right)$
4: **while** $d(\mu_k^t, \mu_k^{t+1}) > \epsilon$ ▷ t is the iteration index **return** μ_k^{t+1}

$$\hat{\sigma_k} = \arg\min_{\sigma_s} \left| \left(\frac{1}{\sum_{i=0}^{N} w_{ik}} \sum_{i=0}^{N} d^2(\mu_k, x_i) w_{ik} \right) - \Phi^{-1}(\sigma_s) \right| \tag{4.4}$$

An inverted search is used to find an approximation of σ_k by computing its values for a finite number of σ_s. To compute $\Phi^{-1}(\sigma_s)$ which involves the term $\frac{d}{d\sigma} log\zeta_m(\sigma)$ we use automatic differentiation algorithms provided by the used pytorch library (Paszke et al. 2019).

4.2 Hyperbolic Embedding of Structured Data

Dataset in this chapter differs from the previous one used. In this chapter we will consider a graph $G(V, E)$, with V the set of nodes and E a set of edges such that $E \subset V \times V$. Each node $i \in V$ is labeled by one or many communities represented by a vector $y_i \in \{0, 1, \ldots, k\}$.

To embed hierarchical or graph data using Poincaré Ball representation on can use the following loss (Nickel et al. 2017):

$$\sum_{(i,j)\in E} \frac{e^{-d(\phi_i,\phi_j)}}{\sum_{k\sim U(V)} e^{-d(\phi_i,\phi_k)}}$$

With $U(V)$ the uniform distribution over nodes.

However, in this work we will take into account different order of proximity and considering communities with a GMM prior.

Learning Graph representation

This section generalizes community embedding and detection of graphs to the Riemannian setting. The goal of embedding GSD is to provide a faithful and exploitable representation of the graph structure. It is mainly achieved by preserving *first-order*

proximity that enforces nodes sharing edges to be close to each other. It can additionally preserve *second-order* proximity that enforces two nodes sharing the same context to be close (i.e. nodes that are neighbors but not necessarily directly connected).

First-order proximity. To preserve first-order proximity we adopt a loss function similar to (Nickel et al. 2017):

$$O_1 = -\sum_{(v_i,v_j)\in E} log(\sigma(-d^2(\phi_i,\phi_j))) \tag{4.5}$$

with $\sigma(x) = \frac{1}{1+e^{-x}}$ the sigmoid function and $\phi_i \in \mathbb{B}^m$ the embedding of the i-th node of V.

Second-order proximity. In order to preserve second-order proximity, the representation of a node has to be close to the representations of its context nodes. To this end we introduce ϕ'_i the embedding of the nodes in context. For this, we adopt the negative sampling approach (Mikolov et al. 2013) and consider the loss:

$$O_2 = -\sum_{v_i\in V}\sum_{v_j\in C_i}\left[log(\sigma(-d^2(\phi_i,\phi'_j))) + \sum_{v_k\sim\mathcal{P}_n} log(\sigma(d^2(\phi_i,\phi'_k)))\right] \tag{4.6}$$

with C_i the nodes in the context of the i-th node, $\phi'_j \in \mathbb{B}^m$ the embedding of $v_j \in C_i$ and $\mathcal{P}_n$ the negative sampling distribution over V given by $\mathcal{P}_n(v) = \frac{deg(v)^{3/4}}{\sum_{v_i\in V} deg(v_i)^{3/4}}$. Notice that we get two different representations one for node embeddings ϕ and another for context embeddings ϕ'. Differentiate nodes to context embeddings has firstly been introduced in Perozzi et al. 2014.

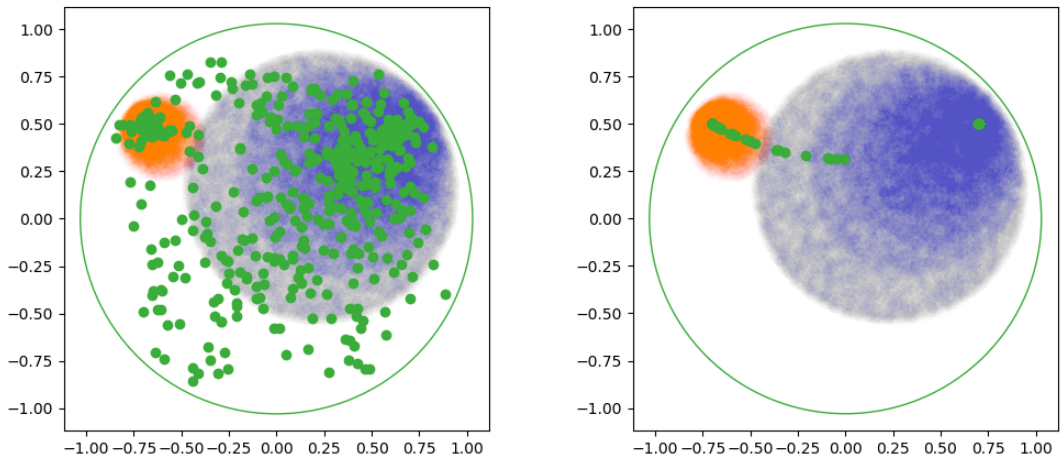
4.2.1 Learning Embedding with communities

Since we want to learn embeddings for the purpose of community detection and classification, we need to ensure that embedded nodes belonging to the same community are close. Therefore, similarly to the statement made in *ComE* Cavallari et al. 2017 we connect node embeddings (first and second-order proximity via the minimization of O_1 and O_2) together with community GMM prior using a loss function O_3. The latter is named the community loss, which we write as:

$$O_3 = -\sum_{i=1}^{|V|}\sum_{k=0}^{K} w_{ik} log\left(\frac{1}{\zeta_m(\sigma_k)} e^{-\frac{d^2(x_i,\mu_k)}{2\sigma_k^2}}\right)$$

This loss gets as objective to bring closer each point to its Gaussian which is associated during the *Expectation-maximization* procedure. As we can observe in the figure 4.4 all

embedded point will follow the geodesics between the two Gaussian if optimized until reaching convergence. However we do not want all point to be placed along the geodesics, thus we stop the O_3 optimization before reaching the convergence.



(a) Before learning (b) After convergence

Figure 4.4 – Visualize the impact of the O_3 loss over embeddings considering two Gaussian. For this figure we took two random hyperbolic Gaussian and sample point into hyperbolic space. We then compute for each point their probability to belong to each Gaussian and finally applied the O_3 loss until convergence.

To solve the community and graph embedding jointly we therefore minimize the final loss function:

$$L = \alpha.O_1 + \beta.O_2 + \gamma.O_3$$

with α, β, γ the weights of respectively first, second-order and community losses.

Optimization :

We now show the gradient computations for the loss functions O_1, O_2 and O_3. Following the idea of Ganea et al. 2018, we use the Riemannian Gradient Descent (RGD) algorithm, which first computes the gradient on the tangent space and then project the new values on the ball

$$\phi^{t+1} = \mathrm{Exp}_{\phi^t}\left(-\eta \frac{\partial O_{\{1,2,3\}}}{\partial \phi}\right) \tag{4.7}$$

where ϕ is a parameter, $t \in \{1, 2, \cdots\}$ is the iteration number and η is a learning rate. Marc Arnaudon et al. 2013 introduced the formula giving the gradient of $d^p(\phi_i, \phi_j)$ (p being the exponent applied on the distance generally $p = 2$) where $\phi_i, \phi_j \in \mathbb{B}^m$:

$$\nabla_{\phi_i} d^p(\phi_i, \phi_j) = -p \times d^{p-1}(\phi_i, \phi_j) \times \frac{\mathrm{Log}_{\phi_i}(\phi_j)}{d(\phi_i, \phi_j)} \tag{4.8}$$

Using the chain rule, the gradient of a loss function involving hyperbolic distance $h = g \circ d^p$ (g being a differentiable function) can be computed as follows considering ϕ_j fixed:

$$\nabla_{\phi_i} h = g'(d^p(\phi_i, \phi_j)) \nabla_{\phi_i} d^p(\phi_i, \phi_j)$$

where the expression $\nabla_{\phi_i} d^p(\phi_i, \phi_j)$ is given in Equation (4.7). In the attached code (discussed in the Appendix), optimization is performed by redefining the gradient of the distance and then using usual auto-derivation tools provided by PyTorch backend.

To avoid division by $d(\phi_i, \phi_j)$, which becomes computationally difficult when ϕ_i and ϕ_j are close, we adopt $p = 2$ which experimentally revealed to be numerically more stable. Explicit forms for the gradient of O_1, O_2 and O_3 are as follows:

Update based on O_1.

$$O_1 = -\sum_{(v_i, v_j) \in E} log(\sigma(-d^2(\phi_i, \phi_j)))$$

Recall that O_1 maintains first-order proximity by preserving the distance between directly connected nodes $(v_i, v_j) \in E$ with embeddings (ϕ_i, ϕ_j). The gradient of O_1 with respect to ϕ_i is

$$\nabla_{\phi_i} O_1 = -2 \times \mathrm{Log}_{\phi_i}(\phi_j) \times \sigma(d^2(\phi_i, \phi_j))$$

In this last formula, we used the fact that $log(\sigma(-x))' = -\sigma(x)$. By symmetry $\nabla_{\phi_j} O_1 = -2 \times \mathrm{Log}_{\phi_j}(\phi_i) \times \sigma(d^2(\phi_i, \phi_j))$.

Update based on O_2.

$$O_2 = -\sum_{v_i \in V} \sum_{v_j \in C_i} \left[log(\sigma(-d^2(\phi_i, \phi'_j))) + \sum_{v_k \sim \mathcal{P}_n} log(\sigma(d^2(\phi_i, \phi'_k))) \right]$$

The updates ϕ_i, ϕ'_j and ϕ'_k the embedding of v_i, v_j and v_k where v_i is a node, v_j belongs to the context C_i of v_i and v_k a negative sample v_i are given as follows:

- Update of ϕ'_j is done exactly as in O_1 since it occurs only in the first term of the sum.

- Update of ϕ'_k is based on the gradient computation

$$\nabla_{\phi'_k} O_2 = -2 \times \mathrm{Log}_{\phi'_k}(\phi_i) \times \sigma(-d^2(\phi_i, \phi'_k))$$

- Update of ϕ_i is based on the gradient computation

$$\nabla_{\phi_i} O_2 = \sum_{v_j \in C_i} \Big[-2 \times \mathrm{Log}_{\phi_i}(\phi'_j) \times \sigma(d^2(\phi_i, \phi'_j)) + \sum_{k=1}^{n_{neg}} \Big(2 \times \mathrm{Log}_{\phi_i}(\phi'_k) \times \sigma(-d^2(\phi_i, \phi'_k)) \Big) \Big]$$

where n_{neg} is the number of negative samples.

Update based on O_3.

$$O_3 = -\sum_{v_i \in V} \sum_{k=0}^{K} w_{ik} log \left(\frac{1}{\zeta_m(\sigma_k)} e^{-\frac{d^2(\phi_i, \mu_k)}{2\sigma_k^2}} \right)$$

The updates of w_{ik}, μ_k and σ_k were detailed in subsection 4.1.2. The update of ϕ_i uses the formula

$$\nabla_{\phi_i} O_3 = \sum_{k=0}^{K} \frac{w_{ik}}{2\sigma_k^2} \nabla_{\phi_i}(d^2(\phi_i, \mu_k))$$

To optimise the parameters we use the RGD algorithm (Equation 4.7) and its adaptive counterpart RAMSGrad (Algorithm .4 proposed in Bécigneul et al. 2018). We observed better performances using the adaptive methods which is used in the last section to perform experiments.

Hyperbolic community learning algorithm Algorithm 4.2 presents a high level scheme for learning community-aware embeddings of graph data on hyperbolic space.

Algorithm 4.2 Community learning and detection on hyperbolic space

Require : $G(V, E)$ graph data, K number of communities, m dimension of hyperbolic space, $lr, \alpha, \beta, \gamma$ learning rates, l random walk length, n_w number of walks from some node, n_c context size, n_{neg} number of negative samples, `max_iter1,2` node and community embedding, respectively, community detection maximum iterations

Ensures : ϕ node embeddings, (μ, σ, π) GMM components: mean, standard deviation and mixture coefficients

Initialisation of ϕ, μ, σ, π randomly from a uniform distribution with fixed bounds
for `iter<max_iter1` **do**
    for $(i, j) \in E$ **do**
        Update ϕ_i, ϕ_j via *RGD* based on $\alpha.O1$
    end for
    for $p \in P$ **do**
        for $i \in p$ **do**
            Update ϕ_i and all ϕ_j in the context of i via RGD of $\beta.O_2$
            ▷ Select context and negatively sampled nodes of size n_c and n_{neg} by generating n_w Random walks of length l from each node as in *DeepWalk* Perozzi et al. 2014
        end for
    end for
    for `iter<max_iter2` **do**
        Update μ, σ, π ▷ By alternating between Expectation and Maximisation steps as described in Subsection 4.1.2
    end for
end for

4.3 Community detection

The previous section showed how to learn community-aware embeddings of graph data where Gaussian mixture models on hyperbolic space are used to model communities. To assess the relevance of the learned embeddings we designed an unsupervised community detection and a supervised node classification experimental frameworks. We propose several ways to assign each node of the graph to one of the K communities. This will allow us in the next section to experimentally evaluate the usefulness of the learnt embeddings for prediction and node classification tasks.

In this section, we provide experimental results and compare them with recent works from the literature [1].

1. The package implementing our algorithms will be made publicly available online in the near future.

We present experiments on *DBLP*[2] a library of scientific papers graph, *Wikipedia*[3] a graph built on Wikipedia dump, *BlogCatalog*[4] a blog social network graph and *Flickr*[5] a graph based on Flickr users. We also experiment on several low scale datasets shown in Table 4.1 and provide visualisations of the learned embeddings when the dataset allows it.

Table 4.1 – Characteristics of the datasets used for experimental evaluation. $|V|$ the number of nodes, $|E|$ the number of edges, K the number of communities and ML whether or not the dataset is multi-label (whether or not a node can belong to several communities).

Corpus	$\|V\|$	$\|E\|$	K	ML
Karate	34	77	2	no
Polblogs	1224	16781	2	no
Books	105	441	3	no
Football	115	613	12	no
DBLP	13,184	48,018	5	no
Wikipedia	4,777	184,812	40	yes
BlogCatalog	10,312	333,983	39	yes
Flickr	80,513	5,899,882	195	yes

Hyper-parameters selection Some parameters used for learning the embedding are difficult to know a priori. Therefore, we performed several grid-search to tune the hyper-parameters producing the best cross-validation performances in terms of conductance, defined below, when running our algorithm and the baseline *ComE*. In particular the parameters λ, the learning rate, c, the size of the context window and t, the number of negative sampling nodes appeared to be the most influencing on results. For our algorithm[6], parameters c and t were selected from the set $\{5, 10\}$, β and $\alpha \in \{0.1, 1\}$, $\gamma \in \{0.01, 0.1\}$ and λ from $[1e-2, 1e-4]$. For all experiments, we generated, for each node context considering 10 random walks, each of length 80. For the first 10 epochs, embeddings are trained using only O_1 and O_2 and then using the complete loop as described in Section 4.2.1.

As for running the algorithm of *ComE*, the tested parameters are $\lambda \in \{0.1, 0.01\}$, $\beta \in \{1, 0.1\}$ and $\gamma \in \{0.1, 0.01\}$. All others parameters are the ones set by default.[7]

2. https://aminer.org/billboard/aminernetwork
3. http://snap.stanford.edu/node2vec/POS.mat
4. http://socialcomputing.asu.edu/datasets/BlogCatalog3
5. http://socialcomputing.asu.edu/datasets/Flickr
6. For Flickr dataset we only run one experiment for each parameter, moreover, the number of parameters tested is lower than those tested for others datasets
7. *ComE* code available at https://github.com/vwz/ComE

- Hyperbolic K-Means (H-K-Means): K-Means finds K cluster centroids and label points by minimising the intra-cluster variances. Based on the notion of Riemannian barycenter, Algorithm 4.3 illustrates Riemannian K-means.

Algorithm 4.3 Hyberbolic K-Means

Require : K number of clusters, $\{\phi_i\} \subset \mathbb{B}^{n\times m}$ node embeddings, `max_iter` maximum iterations

Ensures : I labels of each sample, μ cluster centroids

```
Initialisation of centroids μ ⊂ B^{K×m} randomly from a uniform distribution
for t ∈ {1,2,3,... max_iter} or convergence do
    for i ∈ {1,2,...,n} do
        I_i^t ← arg min_j d(μ_j, φ_i)
    end for
    for k ∈ {1,2,...,K} do
        Update barycenter μ_k ← RiemanianBarycenter({φ_j | I_j = k}
    end for
    convergence ← True iff ∀i ∈ {1,2,...,n}, I_i^{t-1} = I_i^t
end for
```

- Hyperbolic Expectation-Maximisation (H-EM): We apply several EM iterations to obtain a well converged GMM. Then, the probability for a node to belong to some community (modelled by one component of the GMM) is computed. Each node is then labelled with the community giving it the highest posterior probability.

4.3.1 Complexity and Scalability

Complexity With $|V|$ the number of nodes and $|E|$ the number of edges, the time complexity of embedding GSD for first and second-order proximity are respectively in $\mathcal{O}(|E|)$ and $\mathcal{O}(|V|)$. Assuming a fixed number of iterations for all gradient descent computations, performing embedding and Riemannian barycenter based K-means has $\mathcal{O}(|V|.m.K + |E|)$ complexity thus is in $\mathcal{O}(|V| + |E|)$ for large graphs. Similarly, the time complexity of community embedding with EM loop is in $\mathcal{O}(|V| + |E|)$.

Scalability The GSD embedding update process for O_1 and O_2 in $\mathbb{B}^m$ is a linear function of m. Therefore, embedding the GSD scales well to large datasets. To accelerate updates of O_1, O_2, O_3, we use batch gradient descent algorithm mainly for large datasets.

Although run-times of Riemannian K-means and EM are higher than their Euclidean versions the number of iteration remains similar. Notice that the computation of the normalisation factor $\zeta_m(\sigma)$ makes fitting hyperbolic GMM computationally challenging, this can be visually perceived in Figure 4.2. Thus, numerically we have to deal with the out of bound numbers due to the terms $e^{\frac{(m-1)^2\sigma^2}{2}}$. To keep the computation time-bounded,

we fixed a finite number of values for σ for which we pre-computed ζ_m. To avoid the out of bound issue we restricted our work to 10 dimensions while increasing the floating-point precision.

Evaluation metrics We use three performance metrics to assess the relevance of the proposed hyperbolic approach for learning communities on graphs.

• *Conductance:* measures the number of edges shared between separate clusters. The lower the conductance is, the less edges are shared between clusters. Let C_i be the set of nodes for the cluster i, A the adjacency matrix of the GSD, the mean conductance MC over clusters is given by:

$$MC = \frac{1}{K}\sum_{i=1}^{K} \frac{\sum\limits_{j\in C_i, k\notin C_i} A_{jk}}{\min\left(\sum\limits_{j\in C_i, k\in V} A_{jk}, \sum\limits_{j\notin C_i, k\in V} A_{jk}\right)} \tag{4.9}$$

• *NMI :* Let A_{ij} be the number of common nodes belonging to both the predicted cluster i and the real cluster j, $|V|$ being the number of nodes, A_i^p the number of elements in the i-th predicted cluster and A_i^t the number of elements in the i-th real community. The NMI is given by:

$$\text{NMI} = \frac{-2\sum\limits_{i=0}^{K}\sum\limits_{j=0}^{K} A_{ij} log\left(\frac{A_{ij}|V|}{A_i^p A_j^t}\right)}{\sum\limits_{i=0}^{K} A_i^p log\left(\frac{A_i^p}{|V|}\right) + \sum\limits_{j=0}^{K} A_j^t log\left(\frac{A_j^t}{|V|}\right)} \tag{4.10}$$

• *Precision@1*: Since the real labels of communities are known, we propose a supervised measure derived from the precision metric that uses the true community labels. A problem we encountered is that we do not know which label to associated with each community. For small number of communities, all possible combinations are computed and the best performance is reported. This solution is not tractable when the number of communities grows. A greedy approach is then used: the predicted labels of the largest cluster is first associated to the known true label of the dominant class, similarly the second largest cluster is associated to the second dominant class with known labels and so on. The process is repeated until all clusters are treated. Finally, for mono-label datasets, the *Precision@1* corresponds to the mean percentage of the correctly guessed labels. For multi-label datasets (node can belongs to several communities), an element is considered correctly labelled if the inferred community corresponds to one of its true known communities.

Community Detection Results. In this paragraph we present the unsupervised experiments based on K-Means and EM algorithm. For each dataset we perform 5 experiments. The average mean of the performance metrics are shown in Table 4.2.

For the three corpora DBLP, BlogCatalog and Flickr we obtained better results with Hyperbolic clustering methods with only few exceptions. Although hyperbolic embedding

Table 4.2 – Unsupervised community detection performances for Hyperbolic K-Means (H-KM) and Expectation-Maximisation (H-EM) in comparison with state-of-the-art method *ComE*.

		Precision@1			Conductance			NMI		
Dataset	m	H-KM	H-EM	ComE	H-KM	H-EM	ComE	H-KM	H-EM	ComE
DBLP	2	78.5±1.8	**78.6**±4.8	75.9	6.8±4.2	**6.7**±4.4	10.1	66.1±3.4	**66.2**±5.8	55.9
	5	79.6±2.4	81.2±2.1	**87.5**	**4.6**±3.4	4.8±3.8	5.8	**71.3**±2.4	69.7±2.1	62.0
	10	71.4±13.1	**81.5**±0.1	80.4	6.0±4.6	**5.2**±4.0	5.6	65.4±8.9	**69.3**±0.6	56.6
Wikipedia	2	8.6±0.8	**16.1**±4.0	9.3	96.6±3.0	96.6±5.1	**94.7**	**6.9**±0.8	5.5±2.1	6.3
	5	9.7±0.4	10.1±0.7	**10.7**	93.7±3.4	93.8±4.4	**91.2**	**8.8**±0.3	8.6±0.3	8.0
	10	9.3±0.3	**11.9**±1.1	11.1	91±4.1	90.5±4.7	**89.6**	**8.6**±0.0	**8.6**±0.1	7.7
BlogCatalog	2	8.1±0.2	**9.8**±0.3	7.5	**92.5**±6.0	93.1±8.1	93.6	**4.4**±0.0	4.1±0.0	3.4
	5	**13.4**±0.3	12.6±0.7	12.7	88.4±6.6	**87.8**±0.5	87.6	**10.4**±0.5	10.1±0.5	10.5
	10	**18.9**±0.6	16.5±0.8	15.8	85.9±7.5	**84.7**±7.8	86.7	**14.6**±0.2	14.0±0.3	13.3
Flickr	2	8.0±0.2	**12.9**±0.8	6.4	**93.5**±10.0	94.4±13.0	96.4	**24.8**±0.6	**24.7**±0.5	21.7
	5	13.0±0.1	**13.4**±0.1	10.8	**89.7**±12.8	**89.7**±13.9	91.6	**31.8**±0.1	**31.8**±0.1	29.5
	10	13.8±0.1	14.0±0.2	13.3	89.5±12.1	**88.2**±14.5	**89.3**	32.7±0.1	32.7±0.1	**33.1**

seems to perform better than Euclidean embedding, it remains inconclusive which of H-EM or H-KM is better suitable: for DBLP, Wikipedia and Flickr the H-EM approach showed better results, however for *BlogCatalog* K-Means algorithm performed better.

4.4 Community classification

Assuming to know the communities of some nodes, the objective is to predict the communities of unlabelled nodes while using the computed community-aware embedding. The nodes are split into a test set and a validation set. We will use three different methods to predict the labels of the validation set:

- Hyperbolic Barycenters (H-B): This method uses a supervised version of K-Means by computing the Riemannian barycenter of nodes known to belong to a given community. Each barycenter is considered as a cluster centroid representing the community. Then nodes from the validation set are associated to the communities of the n nearest barycenters (depending on the considered *Precision@n* discussed in the next section).

- Hyperbolic GMM (H-GMM) Given the known labels of the train data, a GMM is estimated. The parameters of the GMM are obtained by applying one maximisation step given the train nodes (Section 4.1.2) using the following estimate:

$$w_{ik} = \frac{y_{i,k}}{\sum_{k=0}^{K} y_{i,k}}$$

with $y_{i,k} = 1$ if node i belongs to the community k and $y_{i,k} = 0$ otherwise. To predict the community $\hat{k}$ of a given node embedded as $x_i \in \mathbb{B}^m$, we use Bayes decision rule:

$$\hat{k} = \text{argmax}_k \mathbb{P}(k|x_i) = \text{argmax}_k w_{ik} f(x_i|\hat{\mu_k}, \hat{\sigma_k})$$

with $\hat{\mu_k}, \hat{\sigma_k}$ the estimated parameters of the k-th Gaussian distribution.

- Hyperbolic Logistic Rregression (H-LR) In order to further compare the baseline *ComE* with our proposed method developed in this paper, we propose similarly to Cavallari et al. 2017 to learn a classifier. For the baseline *ComE*, we use the usual Euclidean logistic regression. For its Riemannian version we rely on the hyperbolic logistic regression proposed in Ganea et al. 2018. From this paper, recall that for a given hyper-plane $H_{a,p}$ defined by a point $p \in \mathbb{B}^m$ and a tangent vector $a \in T_p\mathbb{B}^m$, the distance of a given $x \in \mathbb{B}^m$ to $H_{a,p}$ is:

$$d(x, H_{a,p}) = sinh^{-1}\left(\frac{2| < -p \oplus x, a_k > |}{(1 - || - p \oplus x||^2)||a_k||}\right) \tag{4.11}$$

Then, the probability of a node x to belong to the community k is modelled by

$$\mathbb{P}(z = k|x) = \sigma\big(sign(-p \oplus x)d(x, H_{a,p})\big) \tag{4.12}$$

with z the community latent variable.

4.4.1 Community Classification Results

In this section, experiments in the supervised framework are presented. We will evaluate the three ways to predict labels explained previously in section 2.1.4: H-B, a supervised version of K-Means based on the Riemannian barycenter, H-GMM, hyperbolic Gaussian mixture models, or H-LR, hyperbolic Logistic Regression (LR) based on geodesics. For all experiments we apply a 5-cross-validation process with 20% of the dataset for validation. The *Precision@1* is reported for mono-label datasets (i.e., each element belongs to a unique community), additionally *Precision@3* and *5* are reported for multi-label datasets (i.e., each element can belong to several communities). Table 4.3 reports the mean and standard deviation of the *Precision* over 5 runs.

H-B and H-GMM outperform H-SVM and H-LR when the number of communities is large (e.g., the *football* dataset). We empirically show that geodesics separators are not always suited for classification problems. Differences in results between H-SVM and H-LR are mainly due to the quality of the learned embeddings. Indeed, Cho et al. 2018 uses embeddings preserving first and second-order proximity only without taking into account community awareness.

Table 4.3 shows that the proposed embedding method often obtains better or similar performances for the different datasets especially in low dimensions. H-GMM is particularly

Table 4.3 – Results for the different classification methods. With : H-B the supervised hyperbolic K-Means based on hyperbolic barycenter; H-GMM the supervised hyperbolic Gaussian mixture model; H-LR the regression logistic in hyperbolic space; *ComE* Cavallari et al. 2017.

Dataset	Dim	H-B	H-GMM	H-LR	*ComE*
DBLP	2	86.6±1.6	**88.7**±1.1	82.9±8.0	75.3
	5	90.0±0.5	90.9±0.5	**91.2**±0.6	89.8
	10	90.2±0.5	90.7±0.4	**91.6**±0.5	90.1
Wikipedia	2	4.9/1.6/1.0	45.1/28.2/21.5	**47.2/28.7/21.3**	**47.2/28.5/21.1**
	5	5.1/1.7/1.1	45.1/28.6/21.3	**47.4/29.1/21.5**	**47.2/28.7/21.8**
	10	8.3/2.8/1.7	44.6/29.1/21.4	47.5/29.6/21.9	**48.1/29.6/22.1**
BlogCatalog	2	3.8/2.8/4.0	**17.5/12.6/10.8**	16.8/12.6/10.7	16.2/12.5/10.7
	5	13.3/5.9/5.5	25.1/15.8/12.6	24.8/16.0/12.6	**25.9/16.4/12.8**
	10	23.0/9.2/7.2	33.1/19.3/14.4	**35.5/20.2/15.0**	34.1/19.3/14.4
Flickr	2	5.9/2.5/1.7	**23.3/14.3/10.8**	18.7/11.2/8.4	20.3/12.3/9.3
	5	12.8/4.8/3.1	26.8/16.5/12.6	**28.3/16.4/12.2**	26.6/15.5/11.7
	10	14.8/5.4/3.5	26.3/16.3/12.7	**31.4/18.3/13.6**	31.1/17.9/13.3

advantageous for *DBLP* and *Flickr* with superior performances when using two dimensional embedding. Using Riemannian logistic regression outperforms the baseline for *Flickr* and *DBLP* in 5 dimensions and reaches similar performances for the remaining datasets, demonstrating the effectiveness of our method to learn community representations. Notice that for *DBLP*, results of *ComE* in 128 (92%) dimensions are comparable to our results in only 10 dimensions. However, for Wikipedia and BlogCatalog, *ComE* remains better in 128 dimensions reaching respectively 49% and 43% of *Precision@1*.

Convergence and sensibility to initialisation Convergence for node embedding is based on the number of epochs assumed sufficient when the value of the loss function is quasi-constant. The community detection algorithm is sensitive with respect to the initial parameters used for embedding and initial values of EM algorithm as well. Relying on previous works, for the embedding, we considered initialisation suggested in Nickel et al. 2017. For the EM algorithm, we perform a K-means initialisation to deduce initial means while variances are random.

Table 4.4 – Performances obtained by our method compared to Hyperbolic-SVMCho et al. 2018 (H-SVM) for small-scale datasets for 2-dimensional embedding. Results are the means for 5-folds cross-validation for 5 experiments.

Dataset	H-SVM	H-B	H-GMM	H-LR
Karate	86±3	**92**±11.	91±11.	87±15.
Polblogs	93±1	95±0.9	95±0.9	**96**±1.2
Books	73±4	83±8.3	**83**±7.5	82±7.4
Football	24±3	**80**±7.8	79±8.0	39±11.

Hyperbolic SVM Comparison To support the relevance of the proposed classifier approaches, we compare in Table 4.4 our results with *Hyperbolic* SVM Cho et al. 2018. The performances are reported for embedding of small datasets in two dimensions, using 5 folds cross-validation. In terms of hyper-parameters, the context size is set to 3 for *Football* and 2 for the other datasets. The reported results are directly taken from the H-SVM paper (where 5 folds cross-validation are used as well).

4.4.2 Discussion on Hyperbolic graph embeddings

In this section, we discuss and give insight regarding the reasons why a classification algorithm could be suitable or not for a dataset. Additional experiments and comparisons are provided to explain why some datasets obtain poor performances regardless of the used classification algorithm. We propose therefore to compare our method with the baseline of the Most Common Community (MCC, described next). This comparison will illustrate difficulties for classification algorithms to deal with GSD exhibiting imbalanced community distributions when embedded in low dimensions.

In particular, it is the case for *Wikipedia* and *Blogcatalog* datasets where some communities contain a single node. Finally, thanks to the visualization of the predictions of the different supervised algorithms, we justify the relevance of using Gaussian distributions in some situations for classification rather than classifiers based on geodesics such as the Logistic Regression.

Comparison with Most Common Community (MCC)

In this section, we comment on the results and discuss the reasons related to classification algorithms and datasets that led to low performances for two-dimensional representations. In particular, these situations produce low performances in the unsupervised setting which, however, achieve better scores in the supervised one. The main intuition explaining this performance gap is the tendency of supervised classification methods to annotate all nodes with the most common community in the dataset (i.e., the community that labels the highest number of nodes). To better illustrate this claim, we propose to visualize in Table 4.5 the Most Common Community baseline (MCC). The MCC baseline associates each node with the community withdrawn from the probability distribution of the known true labels and can be seen as a naive classification method. Formally, the probability vector of the known labels $\hat{y}$ is written :

$$\hat{y} = \frac{\sum_{i=1}^{N} y_i}{\sum_{i=0}^{N} \sum_{k=0}^{K} y_{ik}} \tag{4.13}$$

where $y_{ik} = 1$ if the true label of node v_i is k and 0 if not, and y_i the labeling vector of node v_i (i.e., the k-th component of y_i is 1 if v_i belongs to the community k).

Table 4.5 – Precisions at 1, 3, 5 for HLR (Hyperbolic Logistic Regression) for 2 and 10 dimensional embeddings, MCC-CV (mean of the most common community using 5-cross validation sets) and MCC-A (most common community for the entire dataset)

Dataset	HLR (2D)	HLR (10D)	MCC-CV	MCC-A
Karate	87	-	53.3	50.0
PoolBlog	96	-	52.0	51.9
Books	82	-	46.6	46.6
Football	39	-	2.6	11.3
DBLP	79	92	38.1	38.0
Wikipedia	47/29/21	47/30/22	47/28/20	47/28/20
BlogCatalog	17/13/11	35/20/15	16/12/10	16/12/10
Flickr	19/11/8	-	17/10/7	17/10/7

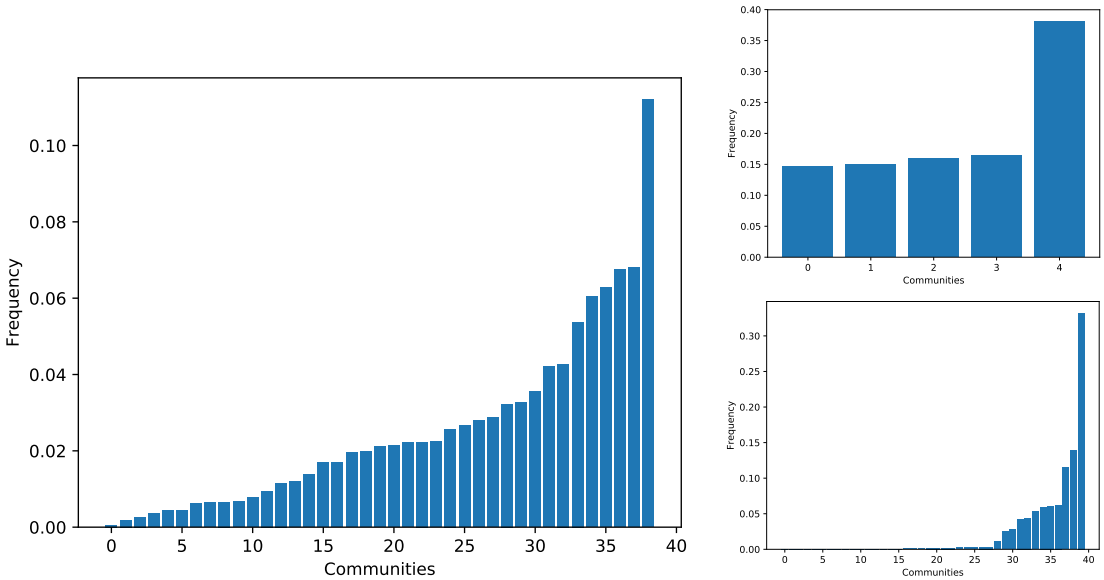
Table 4.5 empirically demonstrates that in 2 dimensions the HLR method is unable to efficiently capture the community structures of *Wikipedia* and *BlogCatalog* since it performs equally the same as MCC. Moreover, for *Wikipedia* the most common community labels nearly half of the graph nodes as shown in Figure 4.5. Taking a closer look at the same Figure, we notice how community distributions are largely unbalanced for *Wikipedia* and *BlogCatalog*. Furthermore, for *Wikipedia* some communities are represented by a single node, making this dataset particularly difficult for classifiers to capture the least represented communities. On the contrary, the fact that communities of *DBLP* are more uniformly distributed contributes in achieving better performances with HLR in only two dimensions.

4.4.2.1 Visualization :

In this part, visualizations of the learned embeddings and prediction of the different classification algorithms are shown. Then advantages and disadvantages of each classifier are discussed.

Figure 5 shows the learned embeddings for the graph datasets *Karate*, *PoolBlog*, *Books*, *Football* and *DBLP* where the color of each node corresponds to its real community. Notice how the embedding lead to clear separate clusters, allowing for classification algorithms to efficiently retrieve communities.

However, when visually comparing the results of *GMM* and *HLR* in figures 4.6 and 6 (refer to the annex), notice how *HLR* struggles to find geodesics separating accurately communities of *Football*. Indeed, the learned two-dimensional embeddings are non-separable using geodesics for *Football*. This difficulty is, however, better handled by *GMM*. Moreover, *HLR* performances on *Football* achieve $\approx 39\%$ while reaching $\approx 80\%$ with *GMM*, thus supporting the benefit of making use of classification with Gaussian distributions.



(a) BlogCatalog

(b) DBLP and Wikipedia

Figure 4.5 – The distributions $\hat{y}$ of communities for *DBLP*, *Wikipedia* and *BlogCatalog* dataset

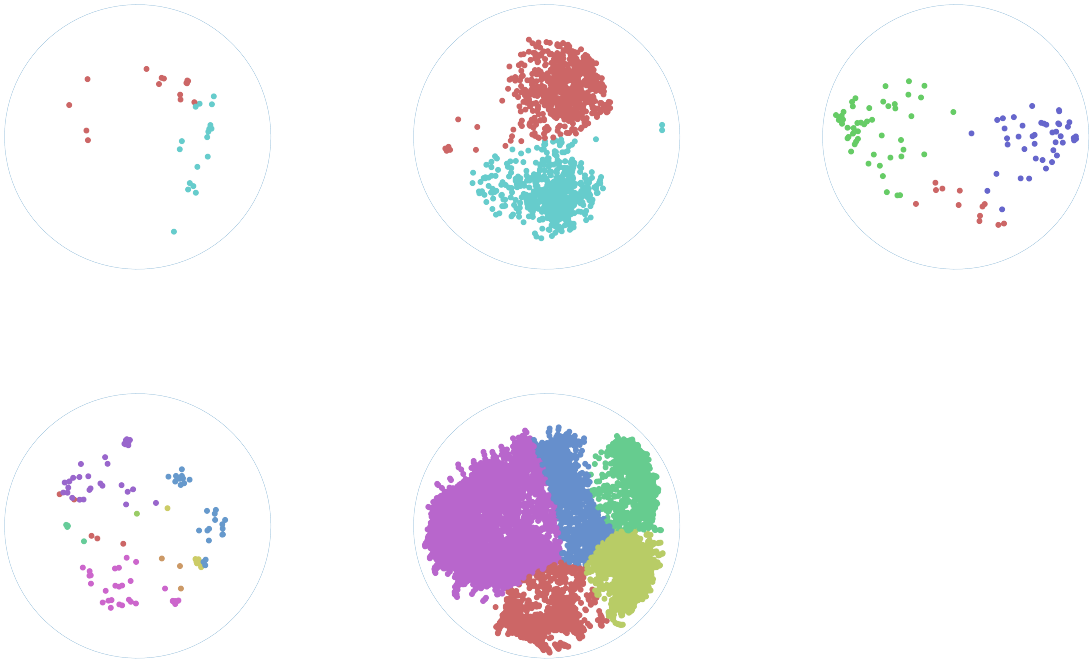
4.5 Hyperbolic embeddings and structured data

In this chapter, we presented a methodology that combines hyperbolic graph embeddings and clustering techniques together for applications to community detection and classification. To this end, we designed a framework to learn community-aware embeddings on the Poincaré Ball. To assess the relevance of a hyperbolic approach to learn communities, we experimented with both supervised and unsupervised applications of this framework. We used adaptations to the Riemannian setting of clustering algorithms K-Means, expectation-maximization and Logistic Regression. We proposed as well an analysis while visualizing the learned representations to explain why certain types of classification algorithms perform better than others in some situations.

Our proposed method was compared with state-of-art ComE (its counterpart in the Euclidean space). Our approach provided better or similar results when using the same number of dimensions. Today,this work is being reviewed at the IPM conference. A preliminary version of the paper is available on arxiv (Gerald et al. 2019b). This work allowed us to contribute to the Geomstats python library (Miolane et al. 2020a).

In upcoming work, we plan to adapt the presented approach to handle larger dimensions to learn communities on even larger graphs, held back today by difficulties for correctly scaling the normalization factor of Gaussian distributions on hyperbolic space.

Complementing previous work on hyperbolic embedding, we have empirically supported the effectiveness of these approaches on large real-structured data. Moreover, the topology of hyperbolic spaces is particularly relevant for hierarchical data, as demonstrated by the



(a) karate

Figure 4.6 – Hyperbolic embeddings colored according to regression logistic prediction for *karate*, *polblog*, *books*, *football* and *dblp* graphs

work of Nickel et al. 2017. Thus, the representation of hierarchically structured data such as labels in multi-label classification tasks is relevant within the hyperbolic space.

Moreover, we point out that hyperbolic embeddings can possibly be used in classification tasks, reaching similar or better performances than the Euclidean counterpart. However, in the proposed framework, embeddings are not conditioned by the labels vector but only using edge information on the contrary of embeddings based classification approaches. In addition we did not consider a feature space and its mapping function to a hyperbolic representation. We thus propose in the next chapter to address classification, firstly by considering embedding documents and labels in a same space and secondly in learning parametric function from a feature space describing examples to the hyperbolic space.

CHAPTER 5

HYPERBOLIC MULTI-LABEL CLASSIFICATION

Contents

5.1 Classification based continuous embeddings 92
5.1.1 Learning representation for classification 92
5.1.2 Hyperbolic embeddings for classification 93
5.2 A joint labels/examples hyperbolic embedding 96
5.2.1 Evaluation 99
5.2.2 Discussion 99
5.3 Examples based hyperbolic embedding 100
5.3.1 Prediction performances 103
5.3.2 Ensembling 103
5.3.3 Discussion 105
5.4 Discussion and perspective 106

In the past chapter, we adressed the detection and classification of structured data through hyperbolic representation. As multi-label classification based on representation learning often depends on the discovery of the latent structure, we strongly believe that hyperbolic embeddings instead of Euclidean embeddings could lead to better representation. Moreover, hyperbolic embeddings demonstrated to works better in low dimension than the Euclidean counterpart, thus, partially contributing to alleviate the prediction times and the storage.

Although previous classification task showed its relevance for graph embeddings, we still do not know how it can perform adressing the large-scale multi-label classification challenges.

In this chapter, we address the multi-label classification task using hyperbolic embeddings. Contrary to graph embeddings, multi-label classification involves additional information. In graph embeddings we only have edge information to embed nodes, in multi-label classification we have labels and documents information. To make an analogy with graph, labels can be considered as the nodes and edges as the association of labels occuring togethers in documents. For documents we have at our disposal the features describing a document. To map the documents to the hyperbolic space, we have to learn a projection. The latter involving learning a parametric function from the space of features of the documents to their hyperbolic representation (representation space).

To this end, likewise related works, we base our training scheme learning the mapping (function projecting features to the embedding space) function with gradient descent optimization. However, we would exploit together the information of labels and the features of documents. In the first proposal, we embed labels and documents in a hyperbolic space and learn their representation using distance of examples to their labels. To learn the projection from features space to hyperbolic space we propose different approximations.

If this method reaches satisfying performances for some corpus, it drastically failed on datasets with numerous underrepresented labels. Being difficult to get accurate label representation in this case, we propose different improvements such as directly integrate in the learning process example neighborhood (based on labels similarity). In most cases, labels representation is consistent and have interesting properties and retrieve hierarchy from these representations could endorse learning tree-like classifiers. We then discuss ongoing works whose final objective is to retrieve labels hierarchy or a label graph.

5.1 Classification based continuous embeddings

In this section we recall main approaches addressing large-scale multi-label classification through continuous embedding. We give deeper insight on how and why state-of-the-art methods work. We then present briefly the hyperbolic counterpart and discuss its advantages and drawback.

5.1.1 Learning representation for classification

Representation in classification got as the main objective finding a projection that make similarly labeled documents close to each other. To this end the objective mostly relies on maximizing a similarity metric between documents with similar topics. More formally, let be s_l a similarity on label vectors and s_e a similarity in the embedded space and $f : F \to F'$ a projection function (map function) in the target space. Typical objective is subsquently to design a constraint that preserve label vectors similarity in the embedding space:

$$\forall (\mathbf{x}_i, \mathbf{y}_i), (\mathbf{x}_j, \mathbf{y}_j), (\mathbf{x}_k, \mathbf{y}_k) \in \mathcal{D}, s_e(f(\mathbf{x}_i), f(\mathbf{x}_j)) \leq s_e(f(\mathbf{x}_i), f(\mathbf{x}_k))$$

$$\Longleftrightarrow$$

$$s_l(\mathbf{y}_i, \mathbf{y}_j) \leq s_l(\mathbf{y}_i, \mathbf{y}_k) \tag{5.1}$$

The constraint above ensure that K-NN decoding methods get the closest neighbors in terms of labels. If embeddings similarity is often designed using negative distance or scalar product, finding a relevant label vector similarity is the key point to guarantee performance competitiveness. Mostly state-of-the-art methods rely on a set of similar document label

vector $\mathcal{N}^{(i)}$ containing the M closest neigbhors according to a similarity. This similarity mainly relies on the number of common labels, typically the cosine distance.

Various objective functions has been designed to get similar labels and embedding similarity. For instance one can minimize the difference between representations and labels similarities (Kush Bhatia et al. 2015):

$$\min_f \mathbb{E}_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{D}} \sum_{j\in\mathcal{N}^{(i)}} (\mathbf{y}_i^t\mathbf{y}_j - f(\mathbf{x}_i)^t f(\mathbf{x}_j))^2$$

With $N^{(i)}$ set of examples close to the ith example in terms of label vector similarity. However, minimize $(y_i^t y_j - f(\mathbf{x}_i)^t f(\mathbf{x}_j))^2$ is equivalent to add the constraint $s_e(x_i, x_j) = s_l(y_i, y_j)$ to the constraint 5.1, adding thus an additional difficulty.

In order to handle this issue, one can only make closer examples with similar labels in ensuring dissimilar examples to be far from each other. The most common solution relies on triplet loss :

$$\min_f \mathbb{E}_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{D}} \sum_{\substack{j\in\mathcal{N}^{(i)}\\ k\notin\mathcal{N}^{(i)}}} (s_e(f(\mathbf{x}_i), f(\mathbf{x}_k)) - s_e(f(\mathbf{x}_i), f(\mathbf{x}_j)))$$

We often prefer when the number of labels is large, modeling the likelihood (Tagami 2017a) instead of the triplet loss, allowing to more efficiently push back negative examples :

$$\min_f \mathbb{E}_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{D}} \sum_{j\in\mathcal{N}^{(i)}} \frac{e^{s_e(f(\mathbf{x}_i),f(\mathbf{x}_j))}}{e^{s_e(f(\mathbf{x}_i),f(\mathbf{x}_j))} + \sum_{k\notin\mathcal{N}^{(i)}} e^{s_e(f(\mathbf{x}_i),f(\mathbf{x}_k))}} \tag{5.2}$$

However, considering all examples that do not belong to the neighbor set is intractable in terms of computation time, commonly we only sample few (about 5 to 10) negative examples. In addition to the representation learning, a main issue is the prediction time complexity when relying on K-NN algorithms for prediction. To tackle this issue split the input space using for instance K-Means (SLEEC from Kush Bhatia et al. 2015) or partition based classifiers (AnnexML from Tagami 2017a).

In this chapter we propose to adjust the main idea using instead of Euclidean embedding, a representation lying in the hyperbolic space.

5.1.2 Hyperbolic embeddings for classification

Paying attention to label organization in extreme classification corpus, we noticed that labels are often organized hierarchically. To convince the reader, in the corpus *wiki10* one should notice first that label distribution looks similar to an exponential distribution (see figure 5.1). In all the corpus, we have common labels (head labels) and rare labels

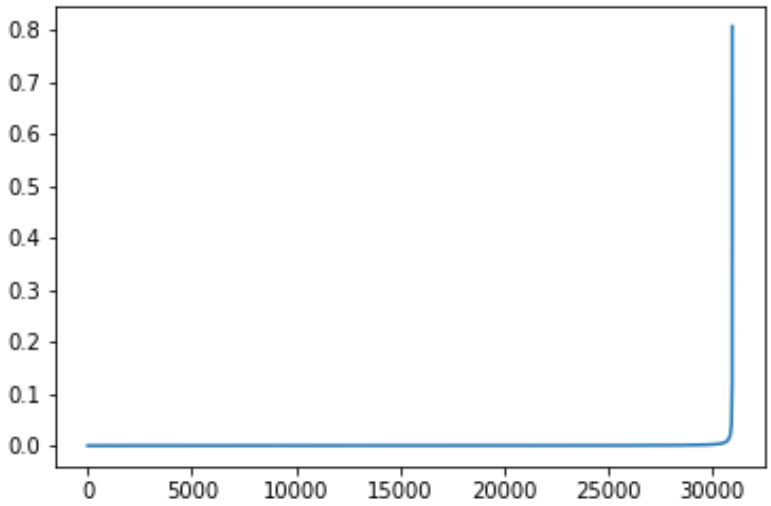


Figure 5.1 – Labels distribution on *wiki10*

(tail labels) which should correspond to the specificity of labels: head labels are general concepts and tails are specific concepts (we have a similar discussion in chapter 2 section 2.1.3). However, we cannot assert that labels describe a strict hierarchy.

Those similarities with hierarchical structures are thus making hyperbolic space a logic candidate to embed labels. We can design a framework embedding both labels and examples in the same space and predict labels according to closest labels of an example in this space. Which leads to optimize the following cost :

$$\min_f \mathbb{E}_{(\mathbf{x}_i,\mathbf{y}_i)\sim\mathcal{D}} \sum_{j\in\{l|y_{il}=1\}} \frac{e^{s(f(\mathbf{x}_i),\mathbf{r}_j)}}{e^{s(f(\mathbf{x}_i),\mathbf{r}_j)} + \sum_{k\in\{l|y_{il}=0\}} e^{s(f(\mathbf{x}_i),\mathbf{r}_k)}} \tag{5.3}$$

Where unlike equation 5.2 we make use of distance between representation of an example $f(x_i)$ and representation of a label r_j. Of course, learning a parametric function and optimizing representation of labels cannot be performed as in Euclidean space, we thus optimize through algorithms based on projecting embeddings on the manifold. This projection technique shares similarities with the one proposed in Tay et al. 2017, we will give further details in following sections.

In figure 5.2 we depict 2 dimensional labels embeddings on the *Wiki10* corpus considering the joint learning of labels and examples. One should first notice that indeed labels are organized by their frequency: more points are red colored more they represent an head labels, on the contrary more they are yellow more they represents tail labels. However, it remains difficult to associate documents embeddings to their labels. Indeed, if documents

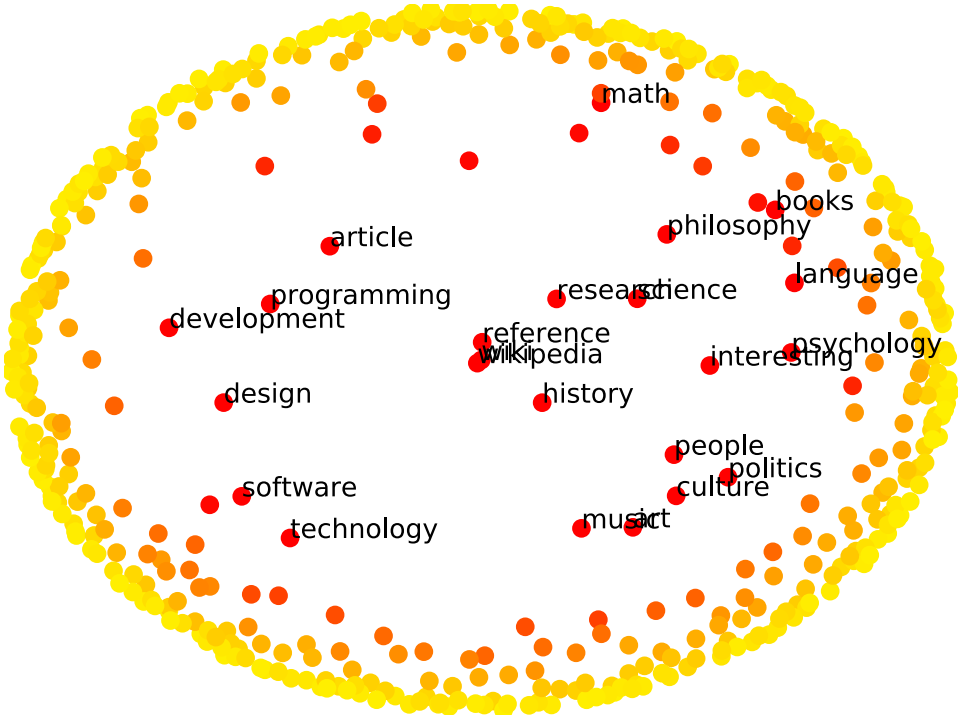


Figure 5.2 – Embedding wiki10 subset of labels into hyperbolic space

embeddings are close to the center, K-NN can not output tail labels, on the contrary documents on the boundary will be far from head labels. As the gradient will be stronger for labels close to the bound of the ball, documents will mostly be closer to the bound than to the origin. In section 5.2 we thus report results considering only K-NN predictor among examples, it show that indeed similarly annotated documents are embedded close to each others. However, a question remains: is it necessary to embed labels since prediction based on labels representation do not produce competitives performances ? Similarly to labels topology, it should exist an organization between documents, particularly on the corpora based on articles like Wikipedia ones. It is thus pertinent to use the classical learning scheme consisting of constraining examples to be close according to a criterion mainly based on a label vector similarity. This last classification approach is studied in section 5.3.

Using Hyperbolic instead of Euclidean space raised several difficulties, especially training parametric projection function such as linear ones. If learning embeddings is quite well defined making use of *Riemannian Gradient Descent*, learning parametric function to project documents remains difficult. Several works proposed solutions; such as in *Hyperbolic Neural Networks* (Mathieu et al. 2019a; Q. Liu et al. 2019; Chami et al. 2019).

Many representation learning algorithms rely on two steps learning, learning embeddings and then learning mappings. We propose to also study this last scheme, using thus the classical Riemannian optimization algorithm for learning embeddings and learning a linear approximation for the latter step. That allows us to evaluate embeddings and mapping

(learning the parametric function f) independently and thus better target the performance bottleneck.

As described below, in this section we are motivated by hyperbolic embeddings for learning representation that keep at most as possible the structure of the labels and therefore examples. We exploit two different tracks, one relying on learning together label/example representation, the second based on more classical representation learning schemes. In a final section we discuss and sumarize proposed approaches, we also extends the discussion to the labels embeddings.

5.2 A joint labels/examples hyperbolic embedding

As motivated in the previous section, large-scale classification corpus could rely on a label taxonomy, in most cases organized as a hierarchy. At the top of the hierarchy general concept are expressed such domain like "history", "geography", "mathematics" or "information technologies" on the contrary labels on the bottom of the hierarchy are more specific such as "pythagoras", "python" or "revolutions" (sampled on *wiki10* corpus). As a strict hierarchy does not exist, we rather prefer to speak of an acyclic graph.

One of the difficulties dealing with those annotation is the label redundancy : some labels have close or same meaning. For instance, we could retrieve labels "game" and "games" in the *wiki10* corpus and examples could be labeled with the one or the latter without any obvious reason. Thus, corpora are frequently noisy, learning representation of labels alleviate this issue by having labels with same meaning close to each other.

Hyperbolic representation learning related works showed its efficiency to embed structured data within limited representation size. However most of the works stated it on small corpora with a strict hierarchy and a clean annotation and labels (Nickel et al. 2017; Sa et al. 2018).

In this section we study the ability of hyperbolic embeddings to tackle large-scale classification without any prior on the hierarchical structure of data. To this end, we design a framework that aims learning labels and examples embedding in a same space. We thus expect label embeddings to draw the hierarchical structure of data. For documents we expect their representation being closer to the boundaries

In order to learn these representations jointly, we design a loss that aims to set closer the embedding of an example and the embedding of a label when a label annotate an example. For prediction, two choices can be considered: 1) retrieving neighbor labels of examples, and then annotate the examples with the n closest labels; 2) considering label vectors of closest documents representations and aggregate labels of closest documents (the documents of the training set are stored for the prediction).

Notations :

We mainly preserve previous notations, with $\mathcal{D} \subset F \times Y$ the dataset of tuple containing features vectors and label vectors. Contrary to previous proposition, we get here two types of embedding, label embeddings are denoted by the exponent l such that r_j^l is the label representation of the label j and the example representation denoted by the exponent e such that r_i^e it is the representation of the ith example. We also need a projection from the features space to the representation space $f : F \to F'$, F' being the embedded space. Let f be a linear function (unless otherwise specified) represented by the matrix $\Theta \in \mathcal{R}^{n \times m}$ with $n = dim(F)$ and m the dimension of the representation space. Moreover unless specified, we would consider $F' = \mathcal{B}^m(0,1)$ i.e. the m-dimensional centered ball within a radius of one. Lastly, we will also need for each example a negative sampling set as use in the previous section. We denote it by $\mathcal{N}_{(i)}^e$ when it is a set of examples and $\mathcal{N}_{(j)}^l$ for labels, we further give details on the construction of those sets.

Loss function :

In this part we propose a similar loss to equation 5.2, however, instead of processing distances between representation of examples, we use distance of the representation of an example to the representation of one of its labels. Let $\mathcal{N}_{(j)}^l = \{k | \mathbf{y}_{jk} = 1\}$ the label set such that $(\mathbf{x}_j, \mathbf{y}_j) \in \mathcal{D}$; the proposed loss for a document representation $r_j^e = f_\theta(x_j)$ and a label i is given by:

$$l(r_j^e, i) = -log \left(\frac{e^{-\alpha d_h(r_j^e, r_i^l)}}{\left(\sum_{k \notin \mathcal{N}_{(j)}^l} e^{-\alpha d_h(r_j^e, r_k^l)} \right) + e^{-\alpha d_h(r_j^e, r_i^l)}} \right)$$

With d_h the Poincaré hyperbolic distance defined with the equation 2.15.

The intuition is the following: each example must be closer to its labels than other labels. The loss function becomes intractable when the number of labels is high due to the set $\mathcal{N}_{(j)}^l$, common practice to leverage this issue consists of sampling negative examples. In practice, we sample for each pair of examples about 5 to 10 negative labels according to uniform distribution (due to the extremely large number of labels it is unlikely to sample a positive label). As frequent labels (labels occurring in many documents) are more frequently updated and must be closer to most examples, their representation will be closer to the origin of the ball (using Poincaré ball representation). At the opposite, rare labels should be close to the boundary of the ball.

Optimization

If optimization methods for Riemiannian manifolds are well known considering only embeddings, learning a parametric function f_θ from the features space to the hyperbolic one is still difficult. In order to take into account the slope of the space curvature, we base optimization on the approximate gradient descent algorithm presented in the chapter 2.1.2. We also need to ensure that the computed vector $f_\theta(\mathbf{x})$ remains in the ball, to do so we project the output of the linear function onto the ball by normalizing vectors having a norm greater than one :

$$\pi(x) = \begin{cases} \frac{\mathbf{x}}{\|\mathbf{x}\|+\varepsilon} & \text{if } \|\mathbf{x}\| \geq 1 \\ \mathbf{x} & \text{otherwise} \end{cases} \tag{5.4}$$

Considering at a step t of the algorithm the representation vector l_i^t of the i-th label, the update of the representation at $t+1$ is given by :

$$\mathbf{r}_i^{l,(t+1)} = \mathbf{r}_i^{l,t} - \eta \left(\frac{1 - \left\| \mathbf{r}_i^{l,t} \right\|_2^2}{2} \right)^2 \nabla_{\mathbf{r}_i^{l,t}} l(\mathbf{x}_j, i)$$

With η the learning rate. In a similar way, the update of the parameter θ^t of the mapping function f_θ is given by the following rule :

$$\theta^{t+1} = \theta^t - \eta \nabla_{\theta^t} \mathbf{r}_j^{e,t} \left(\frac{1 - \left\| \mathbf{r}_j^{e,t} \right\|_2^2}{2} \right)^2 \nabla_{\mathbf{r}_j^{e,t}} l(x_j, i)$$

with $\mathbf{r}_j^{e,t} = \pi(f_{\theta^t}(\mathbf{x}_j))$

Inference step

To annotate a new example $\mathbf{x}$, we embed the example to the representation space with $\pi(f_\theta(\mathbf{x}))$. Then an *approximate K Nearest Neighbor Search* according to the hyperbolic distance d is performed to retrieve the closest training examples. The number of occurrences of each label associated to embedding within the neighborhood of $\pi(f_\theta(\mathbf{x}))$ representation is aggregated and ranked accordingly. The most relevant labels correspond to the most occurring labels in the neighborhood, with the highest apparition frequency. We also experimentally noticed that performances did not decrease when considering a naive clustering while speeding up the inference (choose centroids among the representations of the training examples to construct the clusters).

This last approximate search allows largely decrease the inference time. In the remainder of this paper, we will use $\frac{N}{3000}$ random clusters by default.

5.2.1 Evaluation

Experimental settings :

All evaluated datasets are real-world corpora based on Wikipedia annotations (*Wiki10-31K, WikiLSHTC-325K*), web page annotations (*Delicious-200K*) or web market items/query annotations (*AmazonCat-13K, Amazon-670K*). The datasets have major differences: the Wikipedia and the web page description dataset have a hierarchical label organization while the two amazon datasets have a flatter label organization; statistics of those datasets are depicted in the section 2.1.2.

The table 5.1 shows the results of the proposed approach (*Hyp*) compared to the state-of-the-art results. Reported results come from the K. Bhatia et al. 2016 except for the state-of-the-art algorithm (*AnnexML*) for which the results are those reported in Tagami 2017a. Following their experimental settings, the predictions are made by an ensemble of models, averaging the vote of 15 different models. The performances differ widely with respect to the nature of the dataset: on *Wiki10* and *delicious-200K*, our method is very close to the best results; on the other datasets, there is a gap between the performances of our approach and the others. The main difference between the two sets of corpora is the hierarchical latent organization of the labels: the performances are better with our approach when such hierarchical organization is deep like for *Wiki10* or *Delicious*. At the contrary, results with more flat hierarchies like *Amazon* did not succeed in reaching competitive performances.

Effect of the embedding dimension Poincaré ball model is known for the ability to efficiently embed hierarchical elements in a low-dimensional space. We design an experiment to verify this hypothesis by training our model using different embedding sizes and by comparing it to the *AnnexML* algorithm without pre-partitioning (corresponding in fact to a Euclidean embedding). The table 5.2 shows the results for the *wiki10 corpus*. For low dimensions (5 or 10), our approach slightly outperforms the Euclidean embedding especially in the case of a single learner. With the growth of the number of dimensions, Euclidean embedding tends to outperform our approach.

The dimension of the embedding space is crucial in the case of extreme classification, as it is a key factor for the inference time: the k-nearest neighbor algorithm used for the prediction is linear with respect to the embedding size, thus the complexity of the inference step can be drastically improved if a very low embedding size is sufficient for good prediction performances.

5.2.2 Discussion

Reported performances of learning shows that hyperbolic embeddings can perform at least similarly to Euclidean methods. However, the reported experiments show that for large-scale datasets the method is often lower than other approaches. Then, we

Dataset		Hyp	SLEEC	Parabel	FastXML	PfastreXML	AnnexML
amazonCat	P@1	89.8%	90.5%	93.0%	93.1%	91.8%	93.6%
	P@3	73.5%	76.3%	79.16%	78.2%	78.0%	78.4%
	P@5	58.7%	61.5%	64.5%	63.4%	63.7%	63.3%
wiki10	P@1	85.6%	85.9%	84.3%	83.0%	83.6%	86.5%
	P@3	72.9%	73.0%	72.6%	67.5%	68.6%	74.3%
	P@5	62.4%	62.7%	63.4%	57.8%	59.1%	64.2%
Delicious-200K	P@1	46.5%	47.85%	47.0%	43.07%	41.72%	46.6%
	P@3	40.8%	42.21%	40.1%	38.66%	37.83%	40.8%
	P@5	37.8%	39.43%	36.6%	36.19%	35.58%	37.8%
WikiLSHTC-325K	P@1	43.2%	54.8%	65.0%	49.8%	56.0%	63.4%
	P@3	26.5%	33.4%	43.2%	33.1%	36.8%	40.7%
	P@5	19.4%	23.9%	32.1%	24.5%	27.1%	29.8%
Amazon-670K	P@1	31.1%	35.1%	44.9%	37.0%	39.5%	42.0%
	P@3	28.1%	31.3%	39.8%	33.3%	35.8%	36.7%
	P@5	26.0%	28.7%	36.0%	30.5%	33.1%	32.8%

Table 5.1 – Results of our approach (Hyp) compared to several state-of-the-art approaches. For *SLEEC* and *AnnexML*, the predictions are made by aggregating the label vote of 15 different models. *FastXML* and *PfastreXML* aggregate 50 trees prediction. *AnnexML* merging 1 to 3 classifiers results. All reported results come from the *Extreme Classification repository*, except for *AnnexML* for which the reported results are those of the original paper.

can extend the model to get better performances in several ways: 1) New optimization process have reported better performances in Riemiannian spaces with adaptative gradient descent methods; 2) In this works we use an embedding method that from feature space representation of points produces Poincaré points which set the method not formally hyperbolic; 3) Taking into account directly a similarity measure on examples as it is common in other methods should increase performances. We address those improvements in the following section.

5.3 Examples based hyperbolic embedding

In this section we propose a variation of the previous method by taking into account into the learning procedure example similarity. If labels/examples embedding previously proposed seems to work, it does not allow tackling state of the arts methods, particularly working only on *Wiki10* and *Delicious-200K* corpora. The first assumption: decoding using labels examples neighbor search did not work. We can thus unconsider learning label representation and only focus on learning representation of documents. Moreover, learning the projection function f_θ is difficult, since this one relies on approximation of

#Learners	# Dimensions	Hyperbolic	Euclidean
1	5	78.9/54.1/43.1	77.02/53.5/42.5
3	5	81.3/60.0/50.0	81.1/59.2/48.7
6	5	81.1/61.5/52.0	81.3/60.3/50.4
9	5	81.2/61.8/52.6	81.2/60.6/50.7
12	5	81.2/61.8/53.0	81.2/60.6/50.9
15	5	81.2/61.6/53.1	81.2/60.6/50.9
1	10	81.5/61.6/50.9	79.1/61.4, 50.4
3	10	82.8/66.5/56.0	83.0/66.2/55.0
6	10	82.6/67.4/57.5	83.2/67.3/56.9
9	10	82.6/67.6/57.7	83.1/67.5/57.4
12	10	82.5/67.4/57.7	83.1/67.4/57.6
15	10	82.6/67.6/58.0	83.1/67.5/57.5
1	25	82.9/67.7/57.1	82.5/67.8/57.5
3	25	84.8/70.1/59.7	84.8/71.0/60.5
6	25	84.6/71.0/60.3	85.6/71.7/61.5
9	25	84.5/71.0/60.5	85.5/77.9/61.7
12	25	84.5/71.1/60.7	85.8/71.9/61.7
15	25	84.6/71.3/60.7	85.8/72.0/67.9

Table 5.2 – Influence of the size of the representations and the number of aggregated models for the *Wiki10-31K* corpus for the precision @1,3,5. Hyperbolic refers to our approach, Euclidean refers to a Euclidean embedding similar to the AnnexML algorithm without pre-clustering.

the *Riemannian Gradient Descent* using the projection π. Dividing the learning process into two different steps, by first learning the representation and the second, learning the parametric function to project the docyments to their representation have many advantages. First, we can more easily state which of the two processes is struggling, secondly, we get more freedom in selecting the mapping function, such as using linear regression on embeddings.

In this section we report ongoing results obtained by this approach, proposing different losses to tackle the classification challenges.

Learning Representation

Example similarity loss. In order to ensure similarly annotated documents are close to each other in the representation space, we first need a metric computing a label vector similarity. An usual approach is to consider cosine label similarity between each labels

vector documents. In practice we consider the set $\mathcal{N}^e_{(j)}$ composed by the 10 closest neigbhors according to cosine similarity of labels :

$$\mathcal{N}^e_{(j)} = \underset{S \subset I, |S|=C, j \notin S}{\arg\max} \sum_{k \in S} \frac{\mathbf{y}_j . \mathbf{y}_k^t}{\|\mathbf{y}_j\| \, \|\mathbf{y}_k\|}$$

With C a hyper-parameter if not precised we consider $C = 10$. However, computing each of those sets became intractable for a large size vector y_i and a large number of examples. To alleviate the computing process we should first remark that example share some labels, named head labels (see section 2.1.2), those labels are thus not pertinent for finding closest neighbors according to cosine label distance. Thus we remove head labels from labels vectors using the *AnnexMl* proposed procedure, consisting in preserving only labels respecting the following condition :

$$\sum_{j=1}^{L} \frac{n_j(n_j - 1)}{2} \leq \alpha N, n_1 \leq n_2 \leq \ldots \leq n_L$$

With α a hyper parameter empirically chosen, n_i the number of occurrences of the label i in the training set.

Considering $F' \in \mathcal{B}^{N \times R}$ as the matrix of embedded documents, where $\mathbf{r^e_i}_i \in Z$ correspond to the Poincaré representation of ith document in $\mathcal{D}$, we defined the following loss function :

$$L_E = \mathbb{E}_{(\mathbf{x}_i, \mathbf{y}_i) \sim \mathcal{D}} \sum_{j \in \mathcal{N}^e_{(i)}} \frac{e^{-d_h(\mathbf{r}^e_i, \mathbf{r}^e_j)}}{e^{-d(\mathbf{r}^e_i, \mathbf{r}^e_j)} + \sum\limits_{k \sim U(1,2,\ldots,|\mathcal{D}_{train}|)}^{B} e^{-d_h(\mathbf{r}^e_i, \mathbf{r}^e_k)}}$$

We minimize this loss using Riemannian Approximate Gradient Descent (algorithm .3).

Locality loss. One of the main difficulties using negative sampling process is that there are few chances to be informative at a certain step. Uniformally sampling examples will lead at a certain point only considering already far examples when in reality we want to get further negative examples close to the representation. To do so we propose an additional loss that at each step compute a negative neighborhood based on the 50 closest neighbors. Let $\mathcal{N}^r_{(i)}$ being a set containing closest documents index according to their projection in the hyperbolic space. The objective is thus to move away documents in $\mathcal{N}^r_{(i)}$ that have disimilar annotation. More formally:

$$\mathcal{N}^r_{(i)} = \underset{S' \subset I, |S'|=50, j \notin S'}{\arg\min} \sum_{k \in S} d_h(r^e_i, r^e_j)$$

Then $\bar{\mathcal{N}}^e_{(j)}$ is the set of local negative examples

$$\bar{\mathcal{N}}^e_{(j)} = \underset{S \subset I, |S|=10, j \notin S}{\arg\min} \sum_{k \in \mathcal{N}^r_{(j)}} \frac{y_j . y_k^t}{\|y_j\| \, \|y_k\|}$$

Thus the overall loss function to minimize is given by:

$$L_L = \mathbb{E}_{(\mathbf{x}_i,\mathbf{y}_i)\sim\mathcal{D}} \sum_{j\in\mathcal{N}^e_{(i)}} \frac{e^{-d(\mathbf{z}_i,\mathbf{z}_j)}}{e^{-d(\mathbf{z}_i,\mathbf{z}_j)} + \sum_{k\sim\bar{\mathcal{N}}^e_{(j)}} e^{-d(\mathbf{z}_i,\mathbf{z}_k)}}$$

Learning embedding function :

To learn embedding function $f : F \to \mathcal{B}^n$ we use a linear function and update the weight using usual gradient descent. To avoid having points out of the manifold we thus project using the function π defined in equation 5.4. To learn the mapping we minimize the following cost :

$$L_f = \mathbb{E}_{(\mathbf{x}_i,\mathbf{y}_i)\sim\mathcal{D}} d(f(\mathbf{x}_i), \mathbf{z}_i)$$

Where d is a distance, typically l_2, the scalar product (maximized in this case) or in our work d_h the Poincaré ball hyperbolic distance.

5.3.1 Prediction performances

As an ongoing work, we present preliminary experiments on only three corpora. In the table 5.3 below, we reported precision scores using our methods and the different learning costs described previously. We observe clear improvement using the local loss, particularly on precision at 3 and 5 on wiki10. Those result thus shows that hyperbolic embeddings are suited to the extreme classification problems. However, we did not succeed in obtaining good performances for more complex corpora. Particularly using the locality based loss is difficult since finding neighbors of examples could be extremely time costly in regard to the number of documents in the training set.

Dataset	EE	EE+O	AnnexML	AnnexML (15)(Tagami 2017a)
Wiki10-31K	84.5/71.0/60.1	85.4/73.0/62.4	83.9/71.8/61.4	86.5/74.3/64.2
WikiLSHTC	37.7/21.1/15.3	-	-	63.4/40.7/29.8
Amazon670K	-	21.7/19.7/18.4	-	42.1/36.7/32.8

Table 5.3 – Precision scores at 1/3/5 for our method compared to AnnexML (State-of-the-Art XML representation). All the scores are given for one learner except for last column where Tagami 2017a results are reported for 15 learners. EE meaning example/example loss and O locality loss

5.3.2 Ensembling

In order to consolidate the process of prediction, representation methods (also valid for tree approaches) mostly use the ensembling approach. In our specific case, we ensemble

several models by summing the scores labels vector of each model, thus the final label vector for an example considering s_{it} the score vector for example i obtained by the model $t \in 1, 2, \ldots n$ is :

$$s_i = \sum_{t=1}^{n} s_{it}$$

#learners	P@	nDCG@	PsnDCG
Representation-size = 50			
1	84.7/72.6/62.1	84.6/75.4/67.4	12.0/12.3/12.7
3	86.4/73.6/63.0	86.4/76.6/68.5	11.9/12.3/12.7
6	86.4/73.8/63.0	86.4/76.7/68.5	11.8/12.3/12.6
9	86.4/73.9/63.4	86.4/76.9/68.8	11.8/12.3/12.7
Representation-size = 10			
1	82.6/68.2/57.3	82.6/71.5/63.1	11.0/11.1/11.2
3	84.9/70.1/59.5	84.9/73.5/65.3	10.9/11.1/11.4
6	85.1/70.8/60.2	85.1/74.1/65.9	10.8/11.2/11.4
9	85.3/71.0/60.5	85.3/74.3/66.1	10.8/11.2/11.5

Table 5.4 – Results obtained by ensembling multiple learners and different embeddings dimensionality. To evaluate we use different metrics, the precision, the nDCG and the PsnDCG described in the section 2.1.4

In the table 5.4 below, we show the results obtained with the presented method using at most 9 models, prediction being aggregate by the previous formula. First we should remark that performances do not necessarily increase using more learners, particularly within 50 dimensional embeddings, performances only increase up to 3 learners, this result is encouraging showing that the model does not need large ensembling to perform well.

Propensity score. One could also notice that the *PsnDCG* decreases using several learners. However it is intuitively explainable, since this measure relies on infrequent labels, aggregating examples will lead to increase the score of the most common ones. For this last metric, we approximitavelly perform as well as *SLEEC*(reaching 11.14/11.86/12.4 aggregating 15 models), but get much lower performances than the *PFastreXML* (19.02/18.34/18.43) tree approaches that directly optimize this metric. We should to compare fairly to *SLEEC* running it for only one learner.

Precision and Negative Cumulated gain. For the precision and negative cumulated gain, we reach performances that are similar to the state of the Art *AnnexML* and better than *SLEEC* using fewer learners.

Low dimensionality. However, we do not observe a real gain using lower-dimensional embeddings as hyperbolic embedding usually does. At the current state of this works it

not clear yet what is the main issues to get better embeddings within low dimensionality. However, we noticed during the learning step several differences between training and testing loss: for all experiments learning the mapping function will lead at a certain step to decrease performances on the validation (see figure 5.3); however, continuing the optimization will lead to better convergence and performances for the training set. To this end, we use early stopping mechanism, stopping optimization when best precision is reached on the validation set.

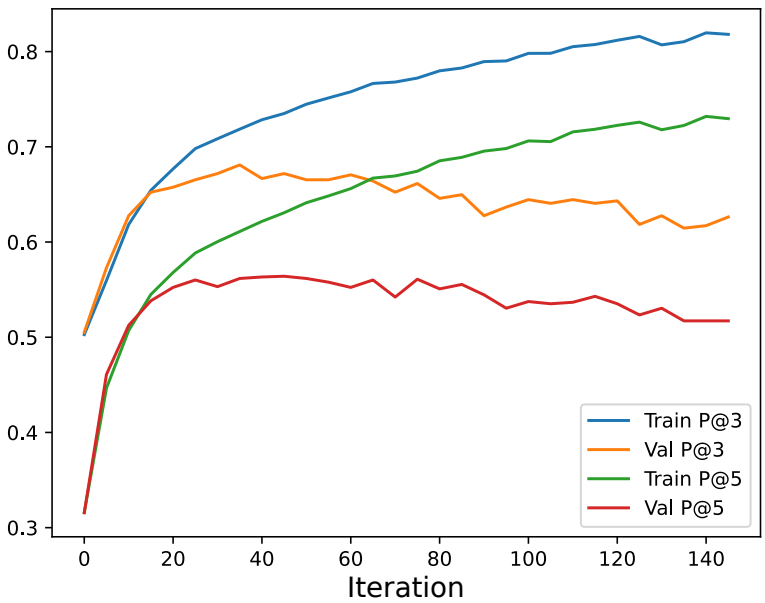


Figure 5.3 – Learning and overfitting for the mapping part. Validation decrease whereas the training performances still increase.

An interpretation of the problems is not the expressiveness of the space using low dimensional embedding but it could be due to overfitting effect. There are several solutions that could reduce this behavior, one could be to use different models for mapping and optimizing using hyperbolic parametric function. However, there is no at the best of our knowledge efficient methods for learning non-sequential parametric function from bag-of-word (or more generally fixed sparse features space). The second approach would rely on using regularization methods, however, using usual l_2 within the current mapping model is meaningless.

5.3.3 Discussion

In this section we proposed to consider the classical framework for multi-label classification involving example/example similarity for learning embeddings such as *SLEEC* or

AnnexML. At the current state of the work, we show that learning embeddings within hyperbolic space could efficiently work on specific corpus. However, learning a mapping function from *Bag-of-words* features space remains difficult and no well-defined methods are today designed to make it work efficiently in *hyperbolic* space.

5.4 Discussion and perspective

In this last chapter, we studied different approaches to achieve representation based multi-label classification. We modeled a joint labels and documents embeddings, validated at the CAP conference in 2019 (Gerald et al. 2019a). This first approach demonstrates that hyperbolic embedding based approach can address the classification task. The joint embedding first idea relies on labels/documents representation using K-NN predictor. This first decoding method did not allow competitive results, leading to consider the classical scheme of prediction using documents representation based K-NN instead. In a second direction, we improved the framework considering optimization of a ranking loss on documents and incorporate a loss relying on the embedding locality. This proposal produced state-of-the-art results on *Wiki10* corpus, but at the price of a very time costly training procedure. This last work is not achieved yet, and we are today considering many improvements.

We currently prospect to use the representations of labels in different classification schemes. Indeed, within hyperbolic space, the norm of elements is meaningful: labels embedded close to the origin tend to correspond to general concepts and on the contrary labels close to the bound to specific concepts. Thus, if labels are describing a taxonomy or a hierarchy we should expect to have tree depth represented by distance to the origin while keeping similar labels close to each other.

We are currently looking at method to retrieve hierarchy from the hyperbolic embedding space to eventually building hierarchical classifiers on the top of the hierarchy.

At the current state of our work, we began evaluating label based representation using *Mean-Rank* metric :

$$\frac{1}{|V|}\sum_{i=0}^{|V|}\frac{1}{|\{(v_i,v_j)\in E\}|}\sum_{(v_i,v_j)\in E}|\{v_k|d(\mathbf{r}_i,\mathbf{r}_k)\leq d(\mathbf{r}_i,\mathbf{r}_j),(v_i,v_k)\notin E\}|$$

With $G(V,E)$ the graph composed of labels sharing an edge if occurring in a same document. The current results are shown in the figure 5.5, showing that in low dimension hyperbolic embeddings outperform Euclidean embeddings.

Moreover, we believe that it should be possible to exploit the label structure to retrieve taxonomy and eventually to design the structure of hierarchical classifiers. Several applications need knowledge on the structure often represented with graphs. If the majority of corpora rely on a taxonomy among labels this information remains unknown, however, retrieving this structure should be possible using labels hyperbolic representation.

Space / Dim	2	5	10	20	50
Poincaré Ball (Hyperbolic)	3654	1924	1610	1448	1388
Euclidean	5520	3173	1709	920	448

Table 5.5 – Comparison of *Mean Rank* between hyperbolic and Euclidean Space on the *Wiki10* dataset

A remaining question is the ability of designing efficient classifier having as prior knowledge the label representation. If intuitively, performances in terms of mean rank and design of the space tend to show that the we obtained a better structure of labels using hyperbolic space, building a classifier on it still is difficult. Unpublished recent works have started to propose algorithms for reconstructing hierarchical structure beyond labels. With this structure we should be in capacity to build a hierarchical classifier having small classifier at each node (a node corresponding to a label). We thus tested several approaches to rebuild this structure, for instance choosing an arity n for a tree and selecting from a parent node the n closest labels further away from the root. Different roots can be designed such selecting the root as the origin of the ball or the centroids of labels representation. Similarly, we could also select parents/children using occurrences of each label, thus for a parent node, select the n closest label representations such that those labels occur less in the corpus.

At the current state of the work, none of the current approaches have given significant performances yet, however, current results on the labels structure encouraging us to look further for more effective approaches.

CHAPTER 6

CONCLUSION

Contents

6.1 Contributions and adressed challenges 109
6.2 Remaining challenges 110
6.3 Discussion on the advance of the domain 112

In this thesis the studied representation learning approaches to tackle *extreme classification* challenges. We mainly focus on two of these challenges the time complexity and the prediction accuracy. To this end we proposed new learning concepts with the binary and atoms networks, and studied hyperbolic embeddings to deal with the *extreme classification* task.

In this conclusion we first review the different contribution proposed. In a following section, we discuss of what challenges remains, and accordingly how can we improve the proposed approaches. We finally discuss of *extreme classification*'s trending approaches, how it evolved during those last four years and the future directions of the domain.

6.1 Contributions and adressed challenges

In this manuscript we studied the extreme classification through representation learning approaches, we proposed three main contributions addressing two different classification challenges. In the first chapter (chapter 3) we proposed a method based on learning binary representation of documents having as the main objective to lower the time complexity. Results on middle size mono-label corpora show the interest of the approach, performing better than *ECOC* based on K-NN decoding and being faster to decode than *Multi-layer* perceptron models. Results particularly show that learning binary codes or partition leads to a better tradeoff between accuracy and time complexity performances than selecting codes randomly. However, we encounter difficulties adapting the algorithm for multi-label settings. In order to take into account labels correlations, we introduced the *Atoms Network*. This last proposal evaluated on small datasets offered interesting performances but without tackling state-of-the-art methods. However, a similar recent framework named *Ground Testing Matrix* started to works efficiently.

Based on matrix testing decoding technic we strongly believe that the proposed approach could lead to better performances with some optimization improvements (see section 6.2).

On the other hand expressiveness of binary representation is not sufficient when the objective is to represent complex relation. Particularly, the labels of extreme multi-label classification corpora often include a hidden taxonomy such as graph or tree structure. Considering this prior, recent approaches based on *Riemannian Geometry* raised our interest as they proved their efficiency to embed taxonomy and more particularly graph structures. To render the ability of hyperbolic space to represent structured data we proposed a graph embedding method to address the community embedding tasks. For this purpose we introduced a hyperbolic transposition of *maximization expectation* algorithm to fit a hyperbolic *Gaussian Mixture* model. Finally, we proposed two tasks on learned embeddings: a community detection task and a classification task based on hyperbolic geometry clustering and hyperbolic geometry classifiers. Experiments demonstrate the efficiency of hyperbolic space to embed such data. Moreover, results show the approach is particularly efficient in low dimensional space with few exceptions. Besides, these works have allowed us to contribute to the *Geomstats* (Miolane et al. 2020b) information geometry python library in implementing the geometry of the Poincaré ball model and its associated clustering algorithms.

In the chapter 4, we proposed a framework to learn embeddings for extreme classification using hyperbolic space. For this last contribution, we first embedded labels and examples jointly where labels embeddings are used to structure the space. Secondly, similarly to usual approach we also considered cost over examples based on their label similarities to design a more accurate representation space which embeds only the examples. Presented results on large-scale corpora show their efficiency particularly on the corpus based on Wikipedia: *wiki10*.

6.2 Remaining challenges

Despite our studies and novel approaches proposed during this PhD, many challenges are remaining in order to tackle large-scale classification challenges. Indeed, many improvements of the proposed methods could be envisaged.

Binary embedding and imitation learning. The *DSNC* model suffer from a major drawback, the optimization process is an approximation of a gradient descent. If in the preliminaries experiments we designed optimization based on the *REINFORCE*, this latter strugles to converge efficiently. It is well known that policy gradient methods have a long exploration step. With the number of possible codes to explore, the convergence using directly *REINFORCE* based optimization is thus intractable. Netherveless, having a good initialization could make it works and converge faster. With the straigh-through estimator we potentially have a better initialization, thus one track to explore would be to use *STE* as initialization and to continue the learning using the policy gradient framework.

The Ground testing challenges. Parallel to the atoms network model, the ground testing framework have been proposed. Those two methods are pretty similar, they mainly differ in decoding process and construction of the atoms. If nor the proposed framework and ground testing approaches produce state-of-the art performances, it should be improved. A first simple approach consists of highly increase the number of atoms, since even with a large number of atoms decoding could be time efficient. A second should be to better build the atoms (or the ground testing matrix), for instance taking into account additional losses to explicitly preserve similar labels in atoms. Recently, unpublished approaches have been proposed considering this framework, however still learning selectors and atoms separately. The resulting ideas could be exploited to better build atoms matrix, such as clustering on the labels correlation matrix.

Hyperbolic embedding and hiearchical structure. The ability of hyperbolic representations to correctly represent graph structures makes them a good choice for extreme classification challenge. Despite good results in our approaches and in the state of the art, parametric methods for transforming documents from the feature space into their hyperbolic representations remain currently inefficient. Related work on hyperbolic neural network learning is today built on approximations which may lead to poor performances compared to their Euclidean counterpart. In our work, we have indeed used two processes : the first well defined is the learning of representation the second corresponding to the learning of an embedding function being approximated by considering the normalization of the embedded documents. On the latter, it would be important to evaluate several solutions, the first one considering the features space as a hyperbolic space (embeddings each feature) and thus calculating the representation by means. However this solution suffers from the fact that we do not have a closed form for the means within the Poincaré model.

In Ganea et al. 2018, authors proposed the counterpart of linear neural layer. This last proposal relies on the decomposition of the linear output in the direction and the values (this last one is normalized to ensure being in the ball). At the current state of our works, we tested this linear layer without reaching competitive performances. However, this last one takes as input a vector lying in the hyperbolic space where in our case documents features are not. Many solutions are imaginable to apply this layer: projecting the input feature vector to the hyperbolic space and apply the linear hyperbolic layer on it, computing the barycenter of documents words in the hyperbolic space or using Euclidean representation first and project it to the hyperbolic space using the exponential map (apply then linear hyperbolic layer). We started to experiment the first approach. The second approach as stated above is intractable relying on the exact expression of the barycenter, however, we could consider approximated barycenter calculation. The last approach has a main drawback, it can not render more information than one encoded by the *Euclidean* representation; thus, a solution is to consider larger Euclidean representation than hyperbolic one. We plan to experiment with these three approaches soon.

On the other hand sequential models are well defined by the *Hyperbolic Neural Networks* paper which opened a new field of research using sequential data of documents. Thus, exploring large-scale classification within sequential neural networks could improve performances of hyperbolic representation for classification (a large portion of *Extreme Classification* corpus could be transformed as sequential data using text order).

To sum up in this thesis we explored representation based methods to address the large-scale classification challenges. With the first approach, we proposed a novel method to fasten prediction thanks to binary documents representation. With the second method, we proposed to take into account structure of the labels and the documents to produce accurate representations using hyperbolic space. With those contributions we show that those approaches can turn effective.

6.3 Discussion on the advance of the domain

During the last four years, many improvements have been realized in the machine learning domain. Particularly, deep learning approaches changed the rules in machine learning: first, in image classification thanks to the *CNN* and later with the *Residual Networks* which alleviate the vanishing gradient issue for very deep neural networks. In text classification, attention redefined the paradigm with memory networks, later the self attention proposed in *transformer* networks greatly improved performances. Extreme classification challenges can make use of those new technologies considering sequential representation of text rather than *bag-of-word*. At the best of our knowledge only few use it today. However, recent works based on *BERT* (Chang et al. 2019b) or using pre-trained transformers (Chang et al. 2019a) have been proposed for extreme classification.

In a close future, we can reasonably expect important changes in *extreme classification* approaches, particularly by including deep learning progress on raw data. However, deep learning methods often rely on a specific data structure at the contrary of current approaches which rely on already processed features of data. The strengh of proposed approaches and state-of-the-art approaches is the generality of the framework, however specific deep learning methods could lead to perform better. For future works, it would be interresting to explore extreme classification considering more data driven deep neural architectures.

BIBLIOGRAPHY

Adcock, Aaron B., Blair D. Sullivan, and Michael W. Mahoney (2013). "Tree-Like Structure in Large Social and Information Networks". In: *2013 IEEE 13th International Conference on Data Mining (ICDM)* (cit. on p. 68).

Afsari, Bijan (2011). "Riemannian L^p Center of Mass: Existence, Uniqueness and Convexity." In: *Proceedings of the American Mathematical Society* (cit. on p. 70).

Agrawal, Rahul, Archit Gupta, Yashoteja Prabhu, and Manik Varma (2013). "Multi-Label Learning with Millions of Labels: Recommending Advertiser Bid Phrases for Web Pages". In: *Proceedings of the 22Nd International Conference on World Wide Web*. New York, NY, USA (cit. on p. 23).

Akbarnejad, Amirhossein and Mahdieh Soleymani Baghshah (2016). "An Efficient Large-Scale Semi-Supervised Multi-Label Classifier Capable of Handling Missing Labels". In: *arXiv preprint arXiv:1606.05725* (cit. on p. 27).

Allwein, Erin L, Robert E Schapire, and Yoram Singer (2000). "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers". In: *Journal of machine learning research* (cit. on p. 32).

Andoni, Alexandr, Piotr Indyk, Huy L. Nguyen, and Ilya P. Razenshteyn (2013). "Beyond Locality-Sensitive Hashing". In: *CoRR* (cit. on p. 45).

Andoni, Alexandr and Ilya P. Razenshteyn (2015). "Optimal Data-Dependent Hashing for Approximate Near Neighbors". In: *CoRR* (cit. on p. 45).

Arnaudon, M. and L. Miclo (2014). "Means in Complete Manifolds : Uniqueness and Approximation". In: *ESAIM Probability and statistics* (cit. on p. 70).

Arnaudon, M., L. Yang, and F. Barbaresco (2011). "Stochastic Algorithms for Computing P-Means of Probability Measures, Geometry of Radar Toeplitz Covariance Matrices and Applications to HR Doppler Processing". In: *International Radar Symposium (IRS)* (cit. on pp. 68, 70).

Arnaudon, Marc, Frédéric Barbaresco, and Le Yang (2013). "Riemannian Medians and Means With Applications to Radar Signal Processing". In: *Journal of Selected Topics in Signal Processing* (cit. on pp. 69, 73, 77).

Babbar, Rohit and Bernhard Shoelkopf (2016). "DiSMEC - Distributed Sparse Machines for Extreme Multi-Label Classification". In: *arXiv:1609.02521 [cs, stat]* (cit. on pp. 20, 30).

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). "Neural Machine Translation by Jointly Learning to Align and Translate". In: (cit. on p. 19).

Barachant, A., S. Bonnet, M. Congedo, and C. Jutten (2012). "Multiclass Brain–Computer Interface Classification by Riemannian Geometry". In: *IEEE Trans. Biomed. Eng.* (cit. on p. 68).

Bécigneul, Gary and Octavian-Eugen Ganea (2018). *Riemannian Adaptive Optimization Methods* (cit. on pp. 41, 78, 127).

Bengio, Samy, Jason Weston, and David Grangier (2010). "Label Embedding Trees for Large Multi-Class Tasks". In: *Advances in Neural Information Processing Systems 23*. Ed. by J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta. Curran Associates, Inc. (cit. on p. 22).

Bengio, Yoshua, Nicholas Léonard, and Aaron C. Courville (2013). "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation". In: *CoRR* (cit. on pp. 45, 51, 61, 65).

Bentley, Jon Louis (1975). "Multidimensional Binary Search Trees Used for Associative Searching". In: *Commun. ACM* (cit. on p. 45).

Berger, Adam (1999). "Error-Correcting Output Coding for Text Classification". In: *IJCAI-99: Workshop on Machine Learning for Information Filtering* (cit. on p. 44).

Bhatia, K., K. Dahiya, H. Jain, A. Mittal, Y. Prabhu, and M. Varma (2016). *The extreme classification repository: Multi-label datasets and code*. URL: http://manikvarma.org/downloads/XC/XMLRepository.html (cit. on pp. 14, 62, 99).

Bhatia, Kush, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain (2015). "Sparse Local Embeddings for Extreme Multi-Label Classification". In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc. (cit. on pp. 5, 28, 30, 35, 56, 93).

Bloom, Burton H. (1970). "Space/Time Trade-Offs in Hash Coding with Allowable Errors". In: *Commun. ACM* (cit. on p. 25).

Boguñá, M., F. Papadopoulos, and D. Krioukov (2010). "Sustaining the Internet with Hyperbolic Mapping". In: *Nature Communications* (cit. on p. 68).

Bonnabel, S. (2013). "Stochastic Gradient Descent on Riemannian Manifolds". In: *IEEE Trans. Autom. Control.* (cit. on pp. 70, 126–128).

Boser, Bernhard E., Isabelle M. Guyon, and Vladimir N. Vapnik (1992). "A Training Algorithm for Optimal Margin Classifiers". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. New York, NY, USA: ACM (cit. on p. 15).

Cavallari, Sandro, Vincent W. Zheng, HongYun Cai, Kevin Chen-Chuan Chang, and Erik Cambria (2017). "Learning Community Embedding with Community Detection and Node Embedding on Graphs". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017* (cit. on pp. 68, 69, 75, 84, 85).

Chamberlain, Benjamin Paul, James Clough, and Marc Peter Deisenroth (2017). "Neural Embeddings of Graphs in Hyperbolic Space". In: *13th International Workshop on Mining and Learning with Graphs* (cit. on p. 68).

Chami, Ines, Rex Ying, Christopher Ré, and Jure Leskovec (2019). *Hyperbolic Graph Convolutional Neural Networks* (cit. on pp. 40, 68, 95).

Chang, Wei-Cheng, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit Dhillon (2019a). *Taming Pretrained Transformers for Extreme Multi-label Text Classification*. arXiv: 1905.02331 [cs.LG] (cit. on p. 112).

Chang, Wei-Cheng, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit Dhillon (2019b). "X-BERT: eXtreme Multi-label Text Classification with using Bidirectional Encoder Representations from Transformers". In: *arXiv preprint arXiv:1905.02331* (cit. on p. 112).

Chen, Boli, Xin Huang, Lin Xiao, Zixin Cai, and Liping Jing (2019). "Hyperbolic Interaction Model For Hierarchical Multi-Label Classification". In: *CoRR* (cit. on pp. 29, 40).

Chen, Yao-nan and Hsuan-tien Lin (2012). "Feature-Aware Label Space Dimension Reduction for Multi-Label Classification". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc. (cit. on pp. 27, 30, 34).

Cho, Hyunghoon, Benjamin Demeo, Jian Peng, and Bonnie Berger (2018). "Large-Margin Classification in Hyperbolic Space". In: *CoRR* (cit. on pp. 69, 84–86).

Choromanska, Anna and John Langford (2014). "Logarithmic Time Online Multiclass Prediction". In: *arXiv:1406.1822 [cs]* (cit. on p. 22).

Cisse, Moustapha M, Nicolas Usunier, Thierry Artières, and Patrick Gallinari (2013). "Robust Bloom Filters for Large MultiLabel Classification Tasks". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Curran Associates, Inc. (cit. on p. 25).

Dasarathy, B. V. (1991). *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. Los Alamitos, CA: IEEE Computer Society Press (cit. on p. 34).

Fukushima, Kunihiko and Sei Miyake (1982). "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Visual Pattern Recognition". In: *Competition and Cooperation in Neural Nets*. Springer (cit. on p. 19).

Galar, Mikel, Alberto Fernández, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera (2013). "Dynamic Classifier Selection for One-vs-One Strategy: Avoiding Non-Competent Classifiers". In: *Pattern Recogn.* (cit. on p. 52).

Ganea, O.-E., G. Becigneul, and T. Hofmann (2018). "Hyperbolic Neural Networks". In: (cit. on pp. 29, 40, 69, 76, 84, 111).

Gerald, Thomas and Nicolas Baskiotis (2019a). "Joint Label/Example Hyperbolic Representation for Extreme Classification". In: URL: https://www.irit.fr/pfia2019/en/cap-2/cap-2019-accepted-papers/ (cit. on p. 106).

Gerald, Thomas, Nicolas Baskiotis, and Ludovic Denoyer (2017). "Binary stochastic representations for large multi-class classification". In: pp. 155–165 (cit. on p. 65).

Gerald, Thomas, Hadi Zaatiti, Hatem Hajri, Nicolas Baskiotis, and Olivier Schwander (2019b). "From Node Embedding To Community Embedding : A Hyperbolic Approach". In: arXiv: 1907.01662 [cs.LG] (cit. on p. 88).

Gionis, Aristides, Piotr Indyk, and Rajeev Motwani (1999). "Similarity Search in High Dimensions via Hashing". In: (cit. on p. 44).

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative Adversarial Networks". In: (cit. on p. 19).

Gromov, M. (1987). "Hyperbolic Groups". In: *Essays in Group Theory*. New York, NY: Springer New York (cit. on p. 68).

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778 (cit. on p. 52).

Hsu, Daniel J., Sham M. Kakade, John Langford, and Tong Zhang (2009). "Multi-Label Prediction via Compressed Sensing". In: *CoRR* (cit. on p. 26).

Indyk, Piotr and Rajeev Motwani (1998). "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality". In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. New York, NY, USA: ACM (cit. on p. 44).

Jain, Himanshu, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma (2019). "Slice: Scalable Linear Extreme Classifiers Trained on 100 Million Labels for Related Searches". In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (cit. on p. 30).

Jain, Himanshu, Yashoteja Prabhu, and Manik Varma (2016). "Extreme Multi-Label Loss Functions for Recommendation, Tagging, Ranking & Other Missing Label Applications". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*. San Francisco, California, USA: ACM Press (cit. on pp. 13, 24, 30).

Jalan, Ankit and Purushottam Kar (2019). "Accelerating Extreme Classification via Adaptive Feature Agglomeration". In: *arXiv preprint arXiv:1905.11769* (cit. on p. 30).

Jernite, Yacine, Anna Choromanska, and David Sontag (2016). "Simultaneous Learning of Trees and Representations for Extreme Classification and Density Estimation". In: *arXiv preprint arXiv:1610.04658* (cit. on p. 22).

Jo, Jaemin, Jinwook Seo, and Jean-Daniel Fekete (2017). "A Progressive K-d Tree for Approximate k-Nearest Neighbors". In: (cit. on p. 45).

Kim, Yoon (2014). "Convolutional Neural Networks for Sentence Classification". In: *CoRR* (cit. on p. 19).

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: *CoRR* (cit. on p. 52).

Kingma, Diederik P. and Max Welling (2013). "Auto-Encoding Variational Bayes". In: (cit. on p. 19).

Kong, Eun Bae and Thomas G Dietterich (1995). "Error-Correcting Output Coding Corrects Bias and Variance". In: *Machine Learning Proceedings 1995*. Elsevier (cit. on p. 20).

Krioukov, Dmitri, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá (2010). "Hyperbolic Geometry of Complex Networks". In: *Physical Review E* (cit. on p. 68).

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems* (cit. on p. 19).

Lai, Siwei, Liheng Xu, Kang Liu, and Jun Zhao (2015). "Recurrent Convolutional Neural Networks for Text Classification". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI Press (cit. on p. 19).

LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1989). "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Comput.* (cit. on p. 19).

Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* (cit. on p. 19).

Lin, Frank and William W. Cohen (2010). "Power Iteration Clustering". In: *ICML* (cit. on p. 68).

Liu, Jingzhou, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang (2017). "Deep Learning for Extreme Multi-Label Text Classification". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '17*. Shinjuku, Tokyo, Japan: ACM Press (cit. on pp. 19, 29).

Liu, Pengfei, Xipeng Qiu, and Xuanjing Huang (2016). "Recurrent Neural Network for Text Classification with Multi-Task Learning". In: *CoRR* (cit. on p. 19).

Liu, Qi, Maximilian Nickel, and Douwe Kiela (2019). *Hyperbolic Graph Neural Networks* (cit. on pp. 40, 68, 95).

Mathieu, Emile, Charline Le Lan, Chris J. Maddison, Ryota Tomioka, and Yee Whye Teh (2019a). *Continuous Hierarchical Representations with Poincaré Variational Auto-Encoders* (cit. on pp. 40, 95).

Mathieu, Emile, Charline Le Lan, Chris J. Maddison, Ryota Tomioka, and Yee Whye Teh (2019b). "Hierarchical Representations with Poincaré Variational Auto-Encoders". In: *CoRR* (cit. on pp. 70, 71).

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 26 (NIPS)*. Curran Associates, Inc. (cit. on p. 75).

Miller, George A (1995). "WordNet: A Lexical Database for English". In: *Communications of the ACM* (cit. on p. 67).

Miolane, Nina, Alice Le Brigant, Johan Mathe, Benjamin Hou, Nicolas Guigui, Yann Thanwerdas, Stefan Heyder, Olivier Peltre, Niklas Koep, Hadi Zaatiti, Hatem Hajri, Yann Cabanes, Thomas Gerald, Paul Chauchat, Christian Shewmake, Bernhard Kainz, Claire Donnat, Susan Holmes, and Xavier Pennec (2020a). "Geomstats: A Python Package for Riemannian Geometry in Machine Learning". In: arXiv: 2004.04667 [cs.LG] (cit. on p. 88).

Miolane, Nina, Alice Le Brigant, Johan Mathe, Benjamin Hou, Nicolas Guigui, Yann Thanwerdas, Stefan Heyder, Olivier Peltre, Niklas Koep, Hadi Zaatiti, Hatem Hajri, Yann Cabanes, Thomas Gerald, Paul Chauchat, Christian Shewmake, Bernhard Kainz, Claire Donnat, Susan Holmes, and Xavier Pennec (2020b). *Geomstats: A Python Package for Riemannian Geometry in Machine Learning* (cit. on p. 110).

Nickel, Maximillian and Douwe Kiela (2017). "Poincaré Embeddings for Learning Hierarchical Representations". In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. (cit. on pp. 29, 40, 68, 69, 74, 75, 85, 89, 96, 126–128).

Norouzi, M., A. Punjani, and D. J. Fleet (2012). "Fast Search in Hamming Space with Multi-Index Hashing". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. Providence, RI: IEEE (cit. on pp. 34, 45).

Norouzi, Mohammad, Ali Punjani, and David J. Fleet (2013). "Fast Exact Search in Hamming Space with Multi-Index Hashing". In: *arXiv:1307.2982 [cs]* (cit. on pp. 34, 45, 49, 50).

Ovinnikov, Ivan (2019). "Poincaré Wasserstein Autoencoder". In: *arXiv preprint arXiv:1901.01427* (cit. on p. 70).

Partalas, Ioannis, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artières, George Paliouras, Éric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Gallinari (2015). "LSHTC: A Benchmark for Large-Scale Text Classification". In: *CoRR* (cit. on p. 19).

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (cit. on p. 74).

Pennec, Xavier (2006). "Intrinsic Statistics on Riemannian Manifolds: Basic Tools for Geometric Measurements". In: *Journal of Mathematical Imaging and Vision* (cit. on p. 70).

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 (cit. on pp. 29, 75, 79).

Prabhu, Yashoteja, Anil Kag, Shilpa Gopinath, Kunal Dahiya, Shrutendra Harsola, Rahul Agrawal, and Manik Varma (2018a). "Extreme Multi-Label Learning with Label Features for Warm-Start Tagging, Ranking & Recommendation". In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. New York, NY, USA: ACM (cit. on pp. 25, 30).

Prabhu, Yashoteja, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma (2018b). "Parabel: Partitioned Label Trees for Extreme Classification with Application to Dynamic Search Advertising". In: *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*. Lyon, France: ACM Press (cit. on p. 30).

Prabhu, Yashoteja and Manik Varma (2014). "FastXML: A Fast, Accurate and Stable Tree-Classifier for Extreme Multi-Label Learning". In: ACM Press (cit. on pp. 24, 30).

Puget, Raphael, Nicolas Baskiotis, and Patrick Gallinari (2015). "Sequential Dynamic Classification for Large Scale Multiclass Problems". In: *Extreme Classification Workshop at ICML*. Lille, France (cit. on p. 21).

Quinlan, J Ross (2014). *C4. 5: Programs for Machine Learning*. Elsevier (cit. on p. 22).

Reddi, Sashank J., Satyen Kale, Felix Yu, Dan Holtmann-Rice, Jiecao Chen, and Sanjiv Kumar (2018). "Stochastic Negative Mining for Learning with Large Output Spaces". In: *arXiv:1810.07076 [cs, stat]* (cit. on p. 29).

Rokach, Lior and Oded Z Maimon (2008). *Data Mining with Decision Trees: Theory and Applications*. World scientific (cit. on p. 22).

Rosenblatt, Frank (1958). "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain." In: *Psychological review* (cit. on p. 15).

Rouvière, François (2016). *Initiation à La Géométrie de Riemann*. Ed. by Calvage et Mounet (cit. on p. 37).

Sa, Christopher De, Albert Gu, Christopher Ré, and Frederic Sala (2018). "Representation Tradeoffs for Hyperbolic Embeddings". In: *CoRR* (cit. on pp. 38–40, 68, 96).

Said, Salem, Lionel Bombrun, Yannick Berthoumieu, and Jonathan H. Manton (2017). "Riemannian Gaussian Distributions on the Space of Symmetric Positive Definite Matrices". In: *IEEE Trans. Information Theory* (cit. on p. 68).

Said, Salem, Hatem Hajri, Lionel Bombrun, and Baba C. Vemuri (2018). "Gaussian Distributions on Riemannian Symmetric Spaces: Statistical Learning With Structured Covariance Matrices". In: *IEEE Trans. Information Theory* (cit. on pp. 69, 70, 72).

Schapire, Robert E. and Yoram Singer (2000). "BoosTexter: A Boosting-Based System for Text Categorization". In: *Machine Learning* (cit. on p. 17).

Siblini, Wissam, Pascale Kuntz, and Frank Meyer (2018). "CRAFTML, an Efficient Clustering-Based Random Forest for Extreme Multi-Label Learning". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Stockholmsmässan, Stockholm Sweden: PMLR (cit. on pp. 24, 30).

Siblini, Wissam, Pascale Kuntz, and Frank Meyer (2019). "A review on dimensionality reduction for multi-label learning". In: IEEE Trans. on Knowledge and Data Engineering (cit. on p. 27).

Skovgaard, Lene Theil (1984). "A Riemannian Geometry of the Multivariate Normal Model". In: *Scandinavian journal of statistics* (cit. on p. 70).

Spielmat, D. A. (1996). "Spectral Partitioning Works: Planar Graphs and Finite Element Meshes". In: *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society (cit. on p. 68).

Sprechmann, Pablo, Roee Litman, Tal Ben Yakar, Alexander M Bronstein, and Guillermo Sapiro (2013). "Supervised Sparse Analysis and Synthesis Operators". In: *Advances in Neural Information Processing Systems* (cit. on p. 35).

Tagami, Yukihiro (2017a). "AnnexML: Approximate Nearest Neighbor Search for Extreme Multi-Label Classification". In: ACM Press (cit. on pp. 5, 12, 28–30, 36, 93, 99, 103).

Tagami, Yukihiro (2017b). "Learning Extreme Multi-Label Tree-Classifier via Nearest Neighbor Graph Partitioning". In: ACM Press (cit. on p. 24).

Tai, Farbound and Hsuan-Tien Lin (2012). "Multilabel Classification with Principal Label Space Transformation". In: *Neural Computation* (cit. on pp. 27, 30, 34).

Tay, Yi, Anh Tuan Luu, and Siu Cheung Hui (2017). "Enabling Efficient Question Answer Retrieval via Hyperbolic Neural Networks". In: *CoRR* (cit. on pp. 40, 94).

Tifrea, Alexandru, Gary Bécigneul, and Octavian-Eugen Ganea (2018). "Poincaré GloVe: Hyperbolic Word Embeddings". In: *CoRR* (cit. on p. 40).

Tu, Cunchao, Hao Wang, Xiangkai Zeng, Zhiyuan Liu, and Maosong Sun (2016). "Community-Enhanced Network Representation Learning for Network Analysis". In: *CoRR* (cit. on p. 68).

Ubaru, Shashanka and Arya Mazumdar (2017). "Multilabel Classification with Group Testing and Codes". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. International Convention Centre, Sydney, Australia: PMLR (cit. on pp. 21, 33, 57).

Ungar, Abraham Albert (2008). "A Gyrovector Space Approach to Hyperbolic Geometry". In: *Synthesis Lectures on Mathematics and Statistics* (cit. on p. 129).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). *Attention Is All You Need* (cit. on p. 19).

Vulić, Ivan, Daniela Gerz, Douwe Kiela, Felix Hill, and Anna Korhonen (2016). *HyperLex: A Large-Scale Evaluation of Graded Lexical Entailment* (cit. on p. 67).

Wang, Daixin, Peng Cui, and Wenwu Zhu (2016). "Structural Deep Network Embedding". In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM (cit. on p. 68).

Wang, Yining, Liwei Wang, Yuanzhi Li, Di He, Wei Chen, and Tie-Yan Liu (2013). "A Theoretical Analysis of NDCG Ranking Measures". In: *Proceedings of the 26th Annual Conference on Learning Theory (COLT 2013)* (cit. on p. 24).

Wei, Yunchao, Wei Xia, Junshi Huang, Bingbing Ni, Jian Dong, Yao Zhao, and Shuicheng Yan (2014). "CNN: Single-Label to Multi-Label". In: *CoRR* (cit. on p. 19).

Weston, Jason, Samy Bengio, and Nicolas Usunier (2011a). "WSABIE: Scaling Up To Large Vocabulary Image Annotation". In: (cit. on p. 23).

Weston, Jason, Ameesh Makadia, and Hector Yee (2011b). "Label Partitioning For Sublinear Ranking". In: (cit. on p. 23).

Williams, Ronald J (1992). "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Machine learning* (cit. on pp. 47, 51).

Wilson, Benjamin and Matthias Leimeister (2018). *Gradient Descent in Hyperbolic Space* (cit. on p. 127).

Yen, Ian E H, Xiangru Huang, Kai Zhong, Pradeep Ravikumar, and Inderjit S Dhillon (2016). "PD-Sparse : A Primal and Dual Sparse Approach to Extreme Multiclass and Multilabel Classification". In: (cit. on p. 20).

Yen, Ian E.H., Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit Dhillon, and Eric Xing (2017). "PPDsparse: A Parallel Primal-Dual Sparse Method for Extreme Classification". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge*

Discovery and Data Mining - KDD '17. Halifax, NS, Canada: ACM Press (cit. on pp. 20, 30).

Yu, Hsiang-Fu, Prateek Jain, Purushottam Kar, and Inderjit S. Dhillon (2013). "Large-Scale Multi-Label Learning with Missing Labels". In: *arXiv:1307.5101 [cs]* (cit. on pp. 27, 30).

Zhang, Chongsheng, Changchang Liu, Xiangliang Zhang, and George Almpanidis (2017). "An up-to-date comparison of state-of-the-art classification algorithms". In: *Expert Systems with Applications* 82, pp. 128–150 (cit. on p. 18).

Zhang, Min-Ling and Zhi-Hua Zhou (2005). "A K-Nearest Neighbor Based Algorithm for Multi-Label Classification". In: *2005 IEEE International Conference on Granular Computing* (cit. on p. 34).

Zhang, Min-Ling and Zhi-Hua Zhou (2007). "ML-KNN: A Lazy Learning Approach to Multi-Label Learning". In: *Pattern recognition* (cit. on p. 34).

Zhang, Wenjie, Junchi Yan, Xiangfeng Wang, and Hongyuan Zha (2017). "Deep Extreme Multi-Label Learning". In: *arXiv:1704.03718 [cs]* (cit. on p. 29).

Zheng, Vincent W., Sandro Cavallari, HongYun Cai, Kevin Chen-Chuan Chang, and Erik Cambria (2016). "From Node Embedding To Community Embedding". In: *CoRR* (cit. on p. 68).

Zhong, Guoqiang and Mohamed Cheriet (2013). "Adaptive Error-Correcting Output Codes". In: (cit. on p. 44).

Zhong, Guoqiang, Kaizhu Huang, and Cheng-Lin Liu (2010). "Learning ECOC and Dichotomizers Jointly from Data". In: *Neural Information Processing. Theory and Algorithms: 17th International Conference, ICONIP 2010, Sydney, Australia, November 22-25, 2010, Proceedings, Part I*. Ed. by Kok Wai Wong, B. Sumudu U. Mendis, and Abdesselam Bouzerdoum. Berlin, Heidelberg: Springer Berlin Heidelberg (cit. on p. 44).

Zhu, Xiaojin and Zoubin Ghahramani (2002). *Learning from Labeled and Unlabeled Data with Label Propagation*. Tech. rep. (cit. on p. 68).

Zou, Hui and Trevor Hastie (2005). "Regularization and Variable Selection via the Elastic Net". In: *Journal of the Royal Statistical Society, Series B* (cit. on p. 20).

Appendices

1 Decoding Algorithm DSNC

In this section we depict the hashing based algorithm (algorithm .1) described in the chapter 3.

Algorithm .1 Decoding Through Hashing

```
procedure STORAGE(W)
    L = []
    for i ∈ [0, 1, ..., 2^c − 1] do
        v ← (i//2^{c−1} mod 2, i//2^{c−2} mod 2, ..., i//2^0 mod 2)
        k ← arg max(W v^t)
        L ← L + [k]
    end for
end procedure
procedure PREDICTION(x, e, L)
    r ← e(x)
    i ← r_{c−1} × 2^{c−1} + r_{c−2} × 2^{c−2} + ··· + r_0 × 2^0 return L[i]
end procedure
```

2 Riemannian Manifold and Hyperbolic space

In this part of the appendix we provide details on Riemannian manifolds, with deeper insight about hyperbolic space.

2.1 Riemannian Manifold

2.1.1 Tangent Space

An important concept when we are working with *Riemannian* manifold is the tangent space concept. Tangent space corresponds to the space generated by a tangent vector to the manifold considering a *base* point. The tangent space at a point $x \in M$ is noted $T_x\mathcal{M}$ for tangent space of manifold $\mathcal{M}$ in x. In *Riemannian* geometry the operation mapping a point to the tangent space on a point x is named the logarithmic map operator denoted Log_x. Inversely the operator mapping a vector v on $T_x\mathcal{M}$ is named exponential map denoted Exp_x. Those operators are an important concept allowing moving a point in manifold, by applying transformation in the tangent space and then mapping tangent vector to the considered manifold.

In the specific case of *Poincaré Ball* model we first have to define the *Moëbus* addition :

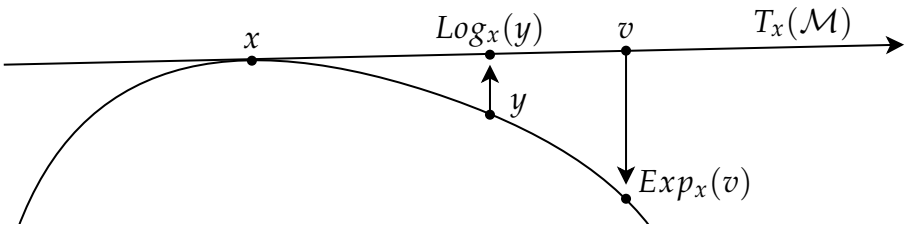


Figure 1 – Illustrating the principle of tangent space with the two operators exponential and logarithmic map.

$$\begin{aligned} \oplus : \quad & \mathcal{B}^n \times \mathcal{B}^n \to \mathcal{B}^n \\ & (x, y) \mapsto x \oplus y \end{aligned} \tag{1}$$

With for $x, y \in \mathcal{B}^n$:

$$x \oplus y = \frac{(1 + 2(x.y)\,\|y\|^2)x + (1 - \|x\|^2)y}{1 + 2(x.y) + \|x\|^2\,\|y\|^2} \tag{2}$$

Then the logarithmic map $Log_x : \mathcal{B}^n \to T_x\mathcal{M}$ for a point $y \in \mathcal{B}^n$ is given by :

$$Log_x(y) = (1 - \|x\|^2)tanh^{-1}(\|(-x \oplus y)\|)\frac{-x \oplus y}{\|-x \oplus y\|} \tag{3}$$

And exponential map $Exp_x : T_x\mathcal{M} \to \mathcal{B}^n$ for $v \in T_x\mathcal{M}$:

$$Exp_x(v) = x \oplus \left(tanh(\frac{\|v\|}{1 - \|x\|^2})\frac{v}{\|v\|} \right) \tag{4}$$

In the figure 1 we pictured the action of both logarithmic and exponential map on a manifold $\mathcal{M}$.

2.1.2 Optimization

Optimization methods in Riemannian manifold rely on the tangent space concept, initially proposed in *"Stochastic Gradient Descent on Riemannian Manifolds"* Bonnabel 2013 (Riemannian Stochastic Gradient Descent (RSGD)). Computing gradient descent or ascent step on the tangent space at a point is first done by computing the gradient of an error function on the tangent space at the same point to finally remap it to the considered manifold using the exponential map operator. More recently, Nickel et al. 2017 proposed hyperbolic embedding method on the Poincare Ball model based on a different optimization process based not on an exact method but on approximation of the exponential map applied on a rescaled gradient. However, many drawbacks can be addressed such that

update can lead to have a point outside of the manifold (corrected by using a projection that normalizes point to have a l_2 norm lower than one), or not only specific to this method the computation time of a step as it involves much more calculation than Euclidean gradient descent. To reduce calculation time, learning Poincaré representation can be performed on one of hyperbolic homeomorphism such as the hyperboloid model Wilson et al. 2018. However, all those approaches do not take into account previously applied gradient (previous step applied on the same weight) while recent Euclidean counterparts rely on adaptive methods leading to whether faster retrieve local minima or reach better convergence. The lack of adaptive methods for Riemannian manifolds has recently been addressed in the *"Riemannian Adaptive Optimization Methods"* (Bécigneul et al. 2018) paper proposing the Riemannian counterpart of two leading Euclidean methods *Adam* and *AMSGrad*.

Algorithm .2 Exact Poincare Ball RSGD algorithm

1: **procedure** STEP(x_t, y_t, f, η)
2: $\quad g_f \leftarrow \nabla_{d_{\mathcal{B}}(x_t,y_t)} f(d_{\mathcal{B}}(x_t, y_t))$
3: $\quad g_{d_{\mathcal{B}}} \leftarrow -\frac{log_{x_t}(y_t)}{d_{\mathcal{B}}(x_t,y_t)}$
4: $\quad \nabla_{x_t}^{T_{x_t}\mathcal{B}} f(d_{\mathcal{B}}(x_t, y_t)) \leftarrow g_f g_{d_{\mathcal{B}}}$
5: $\quad x_{t+1} \leftarrow exp_x(\eta \nabla_{x_t}^{T\mathcal{B}})$
6: **end procedure**

Algorithm .3 Approximate Poincare Ball PRSGD algorithm

1: **procedure** STEP(x_t, y_t, f, η)
2: $\quad g_e \leftarrow \nabla_{x_t} f(d_{\mathcal{B}}(x_t, y_t)$
3: $\quad r \leftarrow \frac{(1-\|x^t\|^2)^2}{4})$
4: $\quad x_{t+1} \leftarrow proj(x_t - \eta r g_e)$
5: **end procedure**

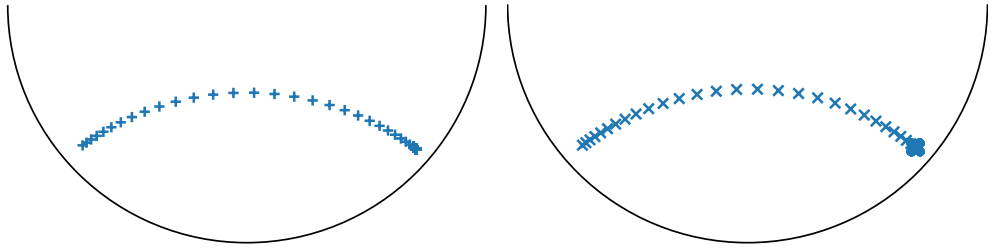
The algorithms .3 (Riemannian Stochastic Gradient Descent) and .2 (Projected Riemannian Stochastic Gradient Descent) refer respectively to the methods introduced in Bonnabel 2013 and Nickel et al. 2017. For the projected method, we refer to $proj$ as the projection defined by :

$$proj(x) = \begin{cases} \frac{x}{\|x\|} - \epsilon, & \text{if } \|x\| \geq 1 \\ x & \text{otherwise} \end{cases}$$

the figure 2 depict two proposed gradient descent approaches minimizing squared distance between the point $x = \begin{pmatrix} -0.8 \\ 0.4 \end{pmatrix}$ and $y = \begin{pmatrix} -0.5 \\ 0.5 \end{pmatrix}$. If the latter method (PRSGD) suffer from precision, its main advantage relies on the computation speed in avoiding the exponential map computation. If we should think that original gradient methods will rely on better convergence, experiments shown in adaptive methods paper Wilson et al. 2018 empirically demonstrate that in particular case the projected gradient descent may lead to better optimum.

Parallel Transport.

Definition .1 (Parallel Transport). Let consider $X_\gamma : \mathcal{I} \to \mathbb{R}^m$ a C^∞ class application, that for each $t \in \mathcal{I}$ associate a tangent vector to the manifold $\mathcal{M}$ at $\gamma(t)$ (a geodesic)



(a) Gradient Descent using exact exponential map

(b) Gradient Descent using retraction (Nickel et al. 2017)

Figure 2 – Compareason of Riemannian gradient descent algorithm proposed in Bonnabel 2013 for figure 2a and Nickel et al. 2017 for figure 2b. In both figure we minimize the squared hyperbolic distance between two points.

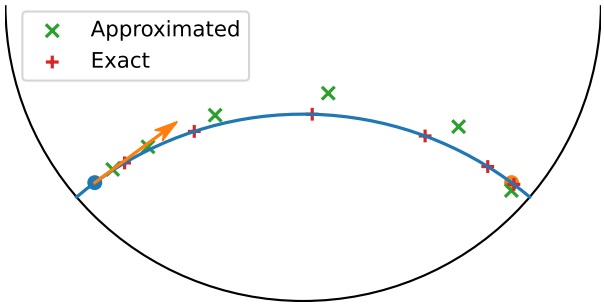


Figure 3 – Visualization of exact and approximated RSGD

$v = X_\gamma(t) \in T_{\gamma(t)}\mathcal{M}$ (X_γ is named a vector field along geodesic γ). Thus considering $X_\gamma(t_0) = v_0$, if $\frac{\nabla X_\gamma(t)}{dt} = 0 \ \forall t \in \mathcal{I}$ (covariant derivative or orthogonal projection on the tangent space at the point $\gamma(t)$ of the derivative), then $X_\gamma(t)$ is called the parallel transport of v_0 along γ.

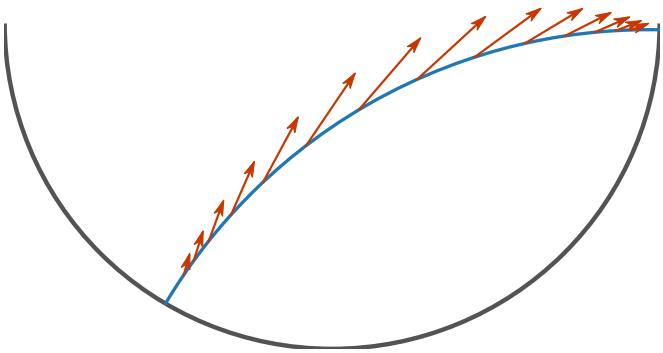


Figure 4 – Visualization of parallel transport of a vector (red) along a geodesic (blue) on the Poincaré disk.

Intuitively, imagine an object goes along a geodesic in $(\mathcal{M}, g_m)$, parallel transport can be thought as a vector having same direction and norm to the point a view of the object moving along the geodesic. For instance, in the hyperboloid model $\mathbb{R}^3$ for extrinsic point of view ($\mathbb{R}^3$ coordinate system) transporting a vector v along a geodesic will lead to a similar vector at all point of the geodesics. In the Poincaré ball model (see figure 4) due to the distortion of the space, the transported vector from an extrinsic point of view will change in norms and direction but for the intrinsic point of view (local observator) still represent the same vector (length according to the distance and direction from the geodesic are preserved).

Optimization within adaptative methods. For recent adaptative methods the key point to make it work rely on parallel transport, i.e. transport a vector from a point x to another y. In Euclidean adaptive methods, the update often relies on the previous gradient such that the applied vector update is a combination of current gradient g_t and previous gradient combination m_{t-1}, i.e. update at a step t is given by $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$. However, Euclidean space as no curvature such that it is not necessary to adapt the vector to the current weight. Parallel transport translates vectors along geodesics, in the case of Poincare Ball transporting vector along a geodesic will modify the direction of the vector. $\Phi_{x\to y}(v)$ denote the transport of a vector v from x to y (based on the unique geodesic going through x and y). From the connection between gyrovector and hyperbolic space proposed by Ungar 2008, parallel transport can be written :

$$\Phi_{x\to y}(v) = \texttt{gyration}(y, -x, v)\frac{\lambda_x}{\lambda_y} \tag{5}$$

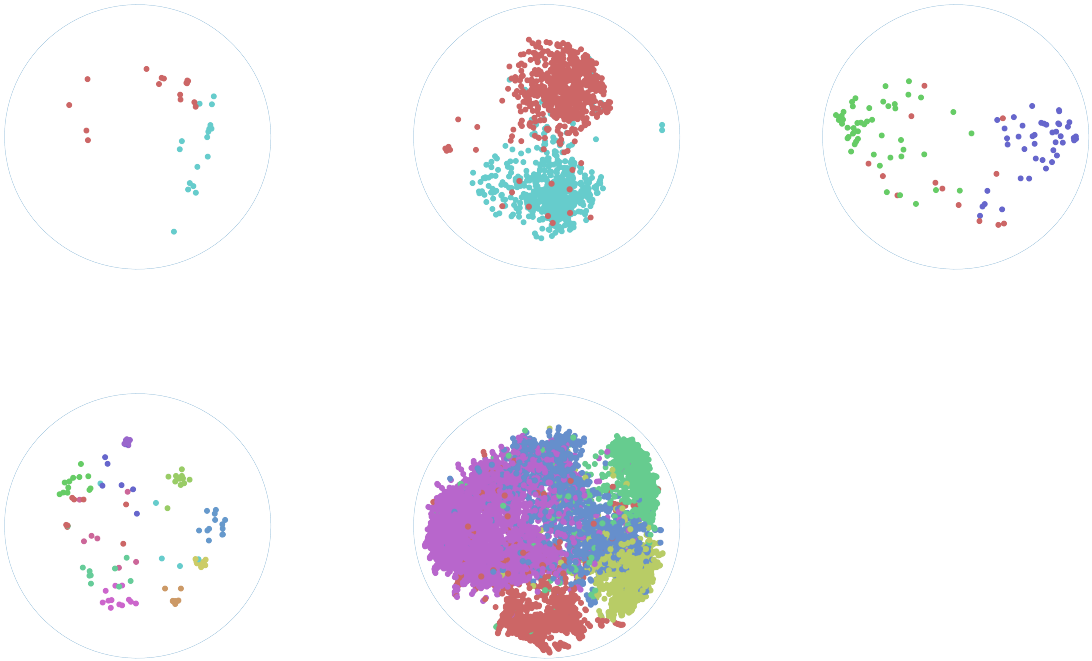
With λ_x the conformal factor in our case $\lambda_x = \frac{2}{1-\|x\|^2}$, and the operator gyration given in Ungar 2008 by the equation 1.27.

Algorithm .4 Poincare Ball RAMSGrad

procedure STEP$(x_t, y_t, f, \eta, \beta_1, \beta_2, m_t^x, v_t^x, \tau_t^x)$
  $g_f \leftarrow \nabla_{d_{\mathcal{B}}(x_t,y_t)} f(d_{\mathcal{B}}(x_t, y_t))$
  $g_{d_{\mathcal{B}}} \leftarrow -\frac{log_{x_t}(y_t)}{d_{\mathcal{B}}(x_t,y_t)}$
  $\nabla_{x_t}^{T_{x_t}\mathcal{B}} f(d_{\mathcal{B}}(x_t, y_t)) \leftarrow g_f g_{d_{\mathcal{B}}}$
  $m_{t+1}^x \leftarrow \beta_1 \tau_t^x + (1-\beta_1)\nabla_{x_t}^{T_{x_t}\mathcal{B}} f(d_{\mathcal{B}}(x_t, y_t))$
  $v_{t+1}^x \leftarrow \beta_2 v_t^x + (1-\beta_2)\left\|\nabla_{x_t}^{T_{x_t}\mathcal{B}} f(d_{\mathcal{B}}(x_t, y_t))\right\|^2$
  $x_{t+1} \leftarrow exp_{x_t}(-\eta \frac{m_{t+1}^x}{\sqrt{v_{t+1}^x}})$
  $\tau_{t+1} \leftarrow \Phi_{x_t\to x_{t+1}}(m_{t+1}^x)$
end procedure

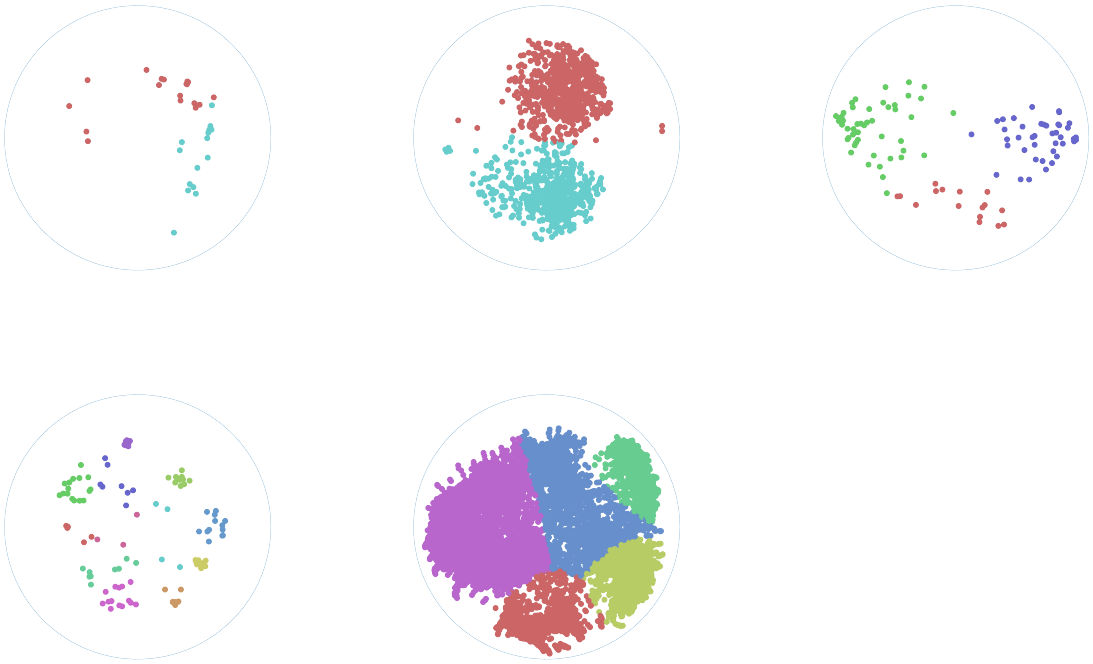
3 Community embedding additional visualization

In this part of the appendix, we report the additional community embeddings classification visualization. In the figure 5 we report 2 dimensional hyperbolic embeddings with color referring to the real community of classes. In the figure 6 we report 2 dimensional hyperbolic embeddings with color referring to the prediction according to gaussians of the mixtures. The figure 7 show embeddings nodes colored by the hyperbolic logistic regression (geodesics base separator). One should notice that in 2 dimensional embeddings *GMM* behave better than using geodesics separator.



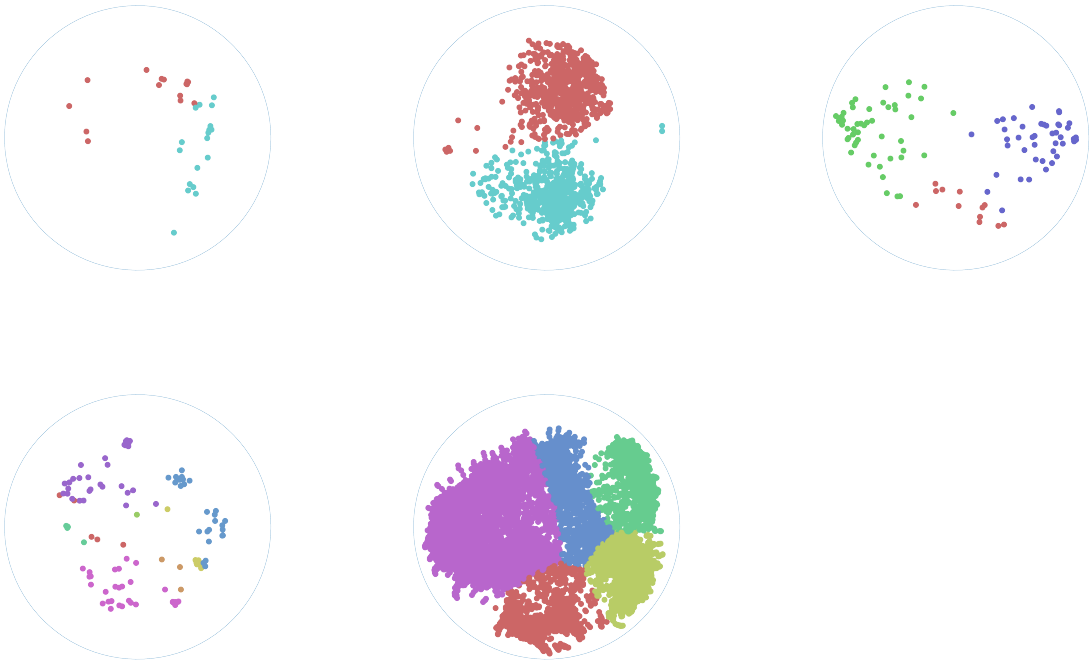
(a) karate

Figure 5 – Hyperbolic embeddings colored according to ground truth for *karate*, *polblog*, *books*, *football* and *dblp* graphs



(a) karate

Figure 6 – Hyperbolic embeddings colored according to gmm prediction for *karate*, *polblog*, *books*, *football* and *dblp* graphs



(a) karate

Figure 7 – Hyperbolic embeddings colored according to regression logistic prediction for *karate*, *polblog*, *books*, *football* and *dblp* graphs