UNIVERSITÀ
DEGLI STUDI
FIRENZE

PhD Program in Smart Computing
Dipartimento di Ingegneria dell'Informazione (DINFO)

Multi-Multi-Instance Learning Networks

Alessandro Tibo

Dissertation presented in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Smart Computing

PhD Program in Smart Computing
University of Florence, University of Pisa, University of Siena

Multi-Multi-Instance Learning Networks

Alessandro Tibo

Advisor:

Prof. Paolo Frasconi

Head of the PhD Program:

Prof. Paolo Frasconi

Evaluation Committee:
Prof. Mathias Niepert, *NEC Labs Heidelberg*
Prof. Andrea Passerini, *University of Trento*

XXXI ciclo — October 2019

To 瑾

Acknowledgments

I would like to express my gratitude to my advisor Prof. Paolo Frasconi, who helped and motived me along my Ph.D study. I would also like to thank you my Supervisory Committee, Prof. Manfred Jaeger and Prof. Marco Gori, who also helped and reviewed my work. I would like to give special thanks to the University of Aalborg that hosted me as visitor. Finally I would like to acknowledge Regione Toscana which gave me the chance to attend my Ph.D study.

Abstract

We introduce an extension of the multi-instance learning problem where examples are organized as nested bags of instances (e.g., a document could be represented as a bag of sentences, which in turn are bags of words). This framework can be useful in various scenarios, such as text and image classification, but also supervised learning over graphs. As a further advantage, multi-multi instance learning enables a particular way of interpreting predictions and the decision function. Our approach is based on a special neural network layer, called bag-layer, whose units aggregate bags of inputs of arbitrary size. We prove theoretically that the associated class of functions contains all Boolean functions over sets of sets of instances and we provide empirical evidence that functions of this kind can be actually learned on semi-synthetic datasets. We finally present experiments on text classification and on citation graphs and social graph data, showing that our model obtains competitive results with respect to other approaches such as convolutional networks on graphs.

Contents

Chapter 1

Introduction

The growth of computer science allowed, among other things, to make available to everyone knowledge and discoveries that before only a few people had access to, and therefore they remained almost unknown. This fact has definitely contributed to accelerating the progress of what was being discovered by students, researchers, and all those people interested in the subject who studied, deepened and further developed such discoveries, in any part of the world. Therefore, in a few years, it took place a real revolution which has affected all fields of science, knowledge and humanity. Today, from the smallest and most remote country of any continent to the most populous metropolis, data, news, knowledge, new discoveries and information are available to anyone who is interested in. Huge amounts of data of any kind are created and widespread every day. This allows, and at the same time requires, that technology also evolves. The evolution of technology is also aimed at creating algorithms able to analyze and extract information from the data that allows us to increase the knowledge itself.

In this respect, an interesting and exciting sector, which has already reached a great stage of evolution but it still has a potential expansion of research, is represented by *artificial intelligence*. This field will certainly represent a big step forward, I would say a real epochal leap, for man and for the organization of life on earth. In the last few years, thanks to the large amount of data which are created every day, and the development of technology, great progress has been made in this area. Machine Learning (a branch of artificial intelligence) is the reference of the research which this thesis is based on. Currently, machine learning is able to solve a number of problems including the classification of images (e.g. dogs vs cats), face recognition (given an image to recognize a person), the analysis of the opinion expressed by a review (positive or negative). From all these examples we understand that the data are treated as individual objects.

Our first challenge is to adapt machine learning models to handle data that are linked to each other through a well-defined part-of relationship, for example, im-

ages can be viewed as bags of pixels, molecules as bags of atoms. Although in the literature there have been proposed several approaches to manage this kind of data, we will provide a generalization of the methods so far defined, by considering the data organized in deeper levels of nesting and by relaxing the underlying assumptions. For example, images can be viewed as bags of patches and patches can be viewed as bags of pixels; molecules can be viewed as bags of atoms and atoms can be viewed as bags of neutrons, protons, and electrons.

One of the biggest limitations, in the considered state-of-the-art machine learning models, i.e. neural networks, is that they can not provide an explanation which justifies the decision: for example, given a trained neural network on radiograph images, the model is "only" able to say whether the patients are sick or not. As second challenge, we will show in this thesis how to learn an interpretable representation of our model for human beings.

1.1 Contribution and Organization

We present in this thesis the multi-multi instance learning (MMIL) problem, a generalization of multi-instance learning (MIL) framework. In MMIL framework, data are organized as labeled nested bags of instances. Only top-level bag labels are observed, while lower level bag and instance labels are latent. The MMIL framework can be useful in various scenarios, where problems are naturally described as bags-of-bags, such as graph classification, image classification, and text classification. Data, organized as nested bags of instances, can be learnt with special neural networks layers, called bag-layers. We propose a solution of the MMIL problem based on neural network in which we used a special layer called *bag-layer*. Bag-layers aggregate internal representations of instances (or bag of instances) and can be intermixed with other common neural network layers. We will also show and prove some theoretical results concerning the expressiveness of our model.

Furthermore, we show that MMIL leads to a particular form of interpretability for which we propose a framework, based on rules extraction, able to explain the MMIL models. Indeed, by leveraging the specific structure of our model, we are able to explain the prediction for a particular data point, and to describe the structure of the decision function in terms of symbolic rules. This latter property is a major contribution, as we provide a global interpretation of MMIL model rather than a local and domain-specific interpretation approaches, used in many related works (for more details see Chapter 4).

The thesis is organized as follows: we start in Chapter 2 with a survey of related works concerning multi-instance learning (MIL), the state-of-the-art works related to neural networks for graphs, and some related works about the interpretability

of machine learning models. Then, we introduce our main contributions which are threefold:

- in Chapter 3 we introduce the multi-multi instance learning (MMIL) model;
- in Chapter 4 we propose a framework for interpreting multi-multi-instance learning networks;
- in Chapter 5 we show how to use MMIL model for machine learning graph tasks.

Finally in Chapter 6, we report all the outcomes of the thesis.

Although this thesis reports the research on MMIL, other works, not directly related to MMIL, were studied during my Ph.D. study. The reader can find all the other research materials in Appendix E, in which we report the list of all the submitted and published papers.

Chapter 2

Background

This chapter provides a comprehensive survey of some related works to the thesis. First, we will start with multi-instance learning, introducing the problem, the relative notation, and some techniques for solving multi-instance learning problems. Then, we give a brief survey of the state-of-the-art related works on neural networks capable to learn graphs. Finally, we briefly list the current state-of-the-art interpretable approaches for machine learning models.

2.1 Multi-instance learning

In the standard multi-instance learning (MIL) setting, data consists of labelled bags of instances. Supervision is provided only for the bags, and the instance labels are not provided. A real application of MIL may consist of medical images for which only patient diagnoses are available instead of local annotations (within images) provided by experts. Furthermore, several problems can be naturally formulated in the MIL setting. For example, in the drug activity prediction problem (Dietterich et al., 1997), the goal is to predict whether a molecule induces a given effect by one (or more) of its conformations. Observing the effect of individual conformations is unfeasible, while treating molecules as bags of conformations (MIL formulation) is viable.

In the following, $\mathcal{X}$ denotes the *instance space* (it can be any set), $\mathcal{Y}$ the *bag label space* for the observed labels of example bags, and $\mathcal{Y}^I$ the *instance label space* for the unobserved (latent) instance labels. For any set $\mathcal{A}$, $\mathcal{M}(\mathcal{A})$ denotes the set of all multisets of $\mathcal{A}$. An example in MIL is a pair $(S, y) \in \mathcal{M}(\mathcal{X}) \times \mathcal{Y}$, which we interpret as the observed part of an instance-labelled example $(S^I, y) \in \mathcal{M}(\mathcal{X} \times \mathcal{Y}^I) \times \mathcal{Y}$. $S = \{x_1, \ldots, x_{|S|}\}$ is thus a multiset of instances, and

$$S^l = \{(x_1, y_1), \ldots, (x_{|S|}, y_{|S|})\}$$

a multiset of labeled instances.

Examples are drawn from a fixed and unknown distribution $p(S^I, y)$. Furthermore, it is typically assumed that the label of an example is conditionally independent of the individual instances given their labels, i.e. $p(y|(x_1, y_1), \ldots, (x_{|S|}, y_{|S|})) = p(y|y_1, \ldots, y_{|S|})$. In the classic setting, introduced in (Dietterich, 2000) and used in several subsequent works (Maron and Lozano-Pérez, 1998; Wang and Zucker, 2000; Andrews et al., 2003), the focus is on binary classification ($\mathcal{Y}^I = \mathcal{Y} = \{0, 1\}$) and it is postulated that $y = \mathbb{1}\left\{0 < \sum_j y_j\right\}$, (i.e., an example is positive iff at least one of its instances is positive). We refer to this setup, as *standard MIL problems*. More complex assumptions are possible and thoroughly reviewed in (Foulds and Frank, 2010). Supervised learning in this setting can be formulated in two ways: (1) learn a function $F : \mathscr{M}(\mathcal{X}) \mapsto \mathcal{Y}$ that classifies whole examples, or (2) learn a function $f : \mathcal{X} \mapsto \mathcal{Y}^I$ that classifies instances and then use some aggregation function defined on the multiset of predicted instance labels to obtain the example label.

Axis-parallel Hyper-rectangle

Dietterich et al. (1997) were the first to propose an algorithm, called axis-parallel hyper-rectangle (APR), for solving *standard MIL problems*. The key idea of APR is to find an axis-parallel hyper-rectangle in the instance space $\mathcal{X}$ which at the same time maximizes the inclusion of instances belonging to positive bags, and minimizes the inclusion of instances belonging to negative bags. Dietterich (2000) proposed three approaches for finding such hyper-rectangle:

- a *standard* algorithm which finds the smallest APR that bounds all the instances belonging to positive bags;
- an *outside-in* algorithm which first constructs the smallest APR that bounds all the instances belonging to positive bags, and then shrinks the APR in order to exclude false positives;
- an *inside-out* algorithm that, starting from a random seed point, grows the smallest APR which covers at least one instance belonging to a positive bag and no instance belonging to negative bags.

Note that all algorithms belong to the formulation (1). This approach intrinsically assumes that it exists an hyper-rectangle which correctly includes all the positive bags and excludes all the negative bags. In real application this assumption can easily fail, and in this respect several approaches that tried to find a collection of hyper-rectangles, hyper-spheres, or hyper-ellipses were proposed in (Maron and Lozano-Pérez, 1998; Xiao et al., 2017; Zhang and Goldman, 2002; Tax and Duin, 2008).

Diverse Density

Diverse Density (DD) was proposed in (Maron and Lozano-Pérez, 1998) as a general framework for solving *standard MIL problems*. The main idea of DD approach is to find a point in the instance space that is close to at least one instance from each positive bag, and is far from instances in negative bags. The optimal point x_d^* is obtained by maximizing the diversity density which measures how many positive bags are close to x_d^*, and how many negative bags are far from x_d^*. We briefly show a probabilist approach for deriving the Diverse Density. Given a multi-instance learning dataset $\mathcal{D} = \{(S_i, y_i) \in \mathcal{M}(\mathcal{X}) \times \{0,1\}\}_{i=1}^N$, by convention we denote with $\mathcal{D}^+$ the set of positive bags and with $\mathcal{D}^-$ the set of negatives bags. The diversity density point $x_d \in \mathcal{X}$ is defined as $x_d^* = \arg\max_{x_d} p(x_d | S \in \mathcal{D})$, where p is the probability of x_d to be the diverse density point given the bags in the dataset. By using the Bayes rule and by assuming uniform prior over x_d location, this is equivalent to $x_d^* = \arg\max_{x_d} p(S \in \mathcal{D} | x_d)$. By making the additional assumption that the bags are conditionally independent given x_d we can write

$$x_d^* = \arg\max_{x_d} \prod_{S^+ \in \mathcal{D}^+} p(S^+ | x_d) \prod_{S^- \in \mathcal{D}^-} p(S^- | x_d). \tag{2.1}$$

Using the Bayes rule once more (and again assuming a uniform prior over concept location), Equation 2.1 equivalent to

$$x_d^* = \arg\max_{x_d} \prod_{S^+ \in \mathcal{D}^+} p(x_d | S^+) \prod_{S^- \in \mathcal{D}^-} p(x_d | S^-). \tag{2.2}$$

Equation 2.2 represents the definition of diverse density. Finally

$$p(S^+ | x_d) = 1 - \prod_{x \in S^+} p(x | x_d),\ p(S^- | x_d) = 1 - \prod_{x \in S^-} p(x | x_d).$$

A possibility for estimating $p(x|x_d)$ is a gaussian-like distribution as, $p(x|x_d) = e^{-\|x - x_d\|^2}$. Intuitively, if one of the instances in a positive bag is close to x_d, $p(x|x_d)$ is high. Likewise, if each positive bag contains an instance close to x_d and no negative bags close to x_d, then x_d will have high diverse density. The above optimization problem is solved by using gradient descent. Usually the search is repeated using the instances from each positive bag as starting points. Several extension of DD have been proposed (Yang and Lozano-Perez, 2000; Zhang and Goldman, 2002), in which different assumptions were made on the relations between instance and bag labels. Note that with this approach is always possible to retrieve the latent labels attached with the instances as well as the labels attached with the bags by using the estimated distribution probabilities. As a major drawback of this approach, we can observe that the distribution $p(x|x_d)$ has to be guessed.

Citation kNN

The popular k Nearest Neighbor (k-NN) approach can be also adapted for solving *standard MIL problems* by defining a proper distance between bags. Wang and Zucker (2000) proposed the minimum Hausdorff distance as distance metric

$$D(S_1, S_2) = \min_{x \in S_1} \min_{z \in S_2} \|x - z\|.$$

By using D, it is always possible to label the unlabelled bags, even though it could happen that the majority of the labels of K nearest neighbors do not correspond to the true label of unlabelled bags. The main reason lies on the underlying prediction generation scheme of kNN, *majority voting* which can be easily fooled by the false positive instances in positive bags. The algorithm can be regularized by considering not only the nearest bags of a bag S, but also the bags that count S as their neighbors. Note that contrarily to the DD approach (see Section 2.1), kNN methods are unable to predict the instances labels.

MI-SVM and mi-SVM

Andrews et al. (2003) proposed two generalizations of SVM (Vapnik, 1963) for solving *standard MIL problems*. As for the case of SVM by convention, we denote positive labels with 1 and negative labels with -1. Given a MIL dataset $\mathcal{D} = \{(S_i, y_i) \in \mathcal{M}(\mathcal{X}) \times \{-1, 1\}\}_{i=1}^{N}$, the two approaches, called *mi-SVM* and *MI-SVM* are defined as follows:

$$\begin{aligned} \textit{mi-SVM} \quad \min_{\{y_{ij}\}} \min_{w,b,\xi} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i,j}\xi_{ij} \\ \text{subject to} \quad & y_{ij}(w^T x_{ij} + b) \geq 1 - \xi_i \\ & \xi_{ij} \geq 0 \\ & y_{ij} \in \{-1, 1\} \\ & \sum_j \frac{y_{ij}+1}{2} \geq 1 \; \forall i \; s.t. \; y_i = 1 \\ & y_{ij} = -1 \; \forall i \; s.t. \; y_i = -1 \end{aligned} \tag{2.3}$$

$$\begin{aligned} \textit{MI-SVM} \quad \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & \forall i \; : \; y_i \max_j (w^T x_{ij} + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{2.4}$$

Note that with x_{ij} we denote the jth instance in the bag S_i, while with y_{ij} we denote the label associated with x_{ij}. *mi-SVM* explicitly treats the instance labels y_{ij} as unobserved hidden variables subject to constraints defined by their bag labels y_i. On the other hand, *MI-SVM* maximizes the bag margin, which is defined by the margin of the "most positive" instance in case of positive bags, or the margin of the "least negative" instance in case of negative bags. In the *mi-SVM* formulation, the margin of each instance in a positive bag matters, although the instance labels can be assigned arbitrarily in order to to maximize the margin. On the other hand, in the *MI-SVM* formulation only one instance for positive bag matters, since that it will determine the margin of the bag. Both *mi-SVM* and *MI-SVM* formulations leads to a mixed integer programming problem in which the optimal labelling and the optimal hyperplane has to be found. Although this problem cannot be solved efficiently even for small size datasets, in Andrews et al. (2003) is proposed an heuristic approach for finding a sub-optimal solution.

Multi-instance Neural Networks

As for the SVM case, also a generalization of neural networks for multi-instance learning problems were proposed by Ramon and De Raedt (2000). Given a neural network, f parametrized by θ, and a MIL dataset

$$\mathcal{D} = \{(S_i, y_i) \in \mathcal{M}(\mathcal{X}) \times \{0,1\}\}_{i=1}^{N},$$

each instance $x_{ij} \in S_i$ is first processed by a replica of a neural network f with parameters θ. In this way, a bag of output values $\{f(x_{1i};\theta), \ldots, f(x_{|S_i|i};\theta)\}$ is computed for each bag of instances. These values are then aggregated by a smooth version of the max function:

$$F(S_i) = \frac{1}{M} \log \left(\sum_j e^{Mf(x_{ij};\theta)} \right)$$

where M is a constant controlling the sharpness of the aggregation (the exact maximum is computed when $M \to \infty$).

2.2 Neural Networks for Graphs

Graphs are data structure, which can be found in several application contexts: social networks, molecules, citation networks, and many more real data can be modelled as graphs. Several machine learning tasks can be defined on graphs. For example we may want to recommend new friends to user in a social dataset, classify the role of a molecule, or predict the class of a paper given its citing and cited papers.

As we will see in Chapter 5, the proposed framework, which this thesis is based on, is also suitable for supervised learning over graphs, i.e., tasks such as graph

classification, node classification, and edge prediction. In this chapter we will summarize the state-of-art methods capable of learning on graphs.

Weisfeiler-Lehman Algorithm

Although the Weisfeiler-Lehman (WL) algorithm (Weisfeiler and Lehman, 1968) may not appear so related with the neural networks, we will see in the next Sections that all the presented neural network approaches for graphs are inspired by WL.

Given a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges, and $L = \{h_1^0, \ldots, h_{|V|}^0\}$ is a set of labels (node coloring) associated with $v_1, \ldots, v_{|V|}$, $v_i \in \mathcal{V}$, the Weisfeiler-Lehman (Algorithm 1) produces an unique assignment of node labels for G. In the Algorithm 1, $\mathcal{N}_i$ denotes the neighbors of v_i, and $\text{hash}(\cdot)$ is an hash function. The key idea of the algorithm is to augment the node labels by the sorted set of node labels of neighbouring nodes, and compress these augmented labels into new, short labels. By applying the Algorithm 1 in turn on two graphs G_1 and G_2, we are able to tell if G_1 and G_2 are isomorphic. If the sets of the node coloring are not identical for G_1 and G_2, it means that G_1 and G_2 are not isomorphic. Contrarily, if the sets of the node coloring are identical after n iterations, it means that either G_1 and G_2 are isomorphic, or the algorithm has not been able to determine that they are not isomorphic.

Algorithm 1: WL algorithm (Weisfeiler & Lehmann, 1968)

Input: A graph $G = (\mathcal{V}, \mathcal{E})$, initial node coloring $(h_1^0, \ldots, h_{|V|}^0)$
Output: Final node coloring $(h_1^T, \ldots, h_{|V|}^T)$
$t \leftarrow 0$
while *stable node coloring is reached* **do**
  for $v_i \in \mathcal{V}$ **do**
    $h_i^{(t+1)} \leftarrow \text{hash}(\sum_{j \in \mathcal{N}_i} h_j^t)$
  end
end

The graph neural network model

Scarselli et al. (2009) proposed a neural network model, called graph neural network (GNN) model, that extends existing neural network methods for processing the data represented in the graph domain. This approach is suitable for both node and graph classification tasks. We will introduce more notation for better explaining this approach. Let $G = (\mathcal{V}, \mathcal{E})$ be a graph, L and X be a set of labels and a set of attributes,

respectively, associated with the nodes of G. With $\mathcal{N}_i$ we denote the neighborhood for a node $v_i \in \mathcal{V}$. In the GNN framework two networks $x_i = f(l_i, x_{\mathcal{N}_i}, l_{\mathcal{N}_i}; w_f)$, $o = g(f; w_g)$, with parameters w_f and w_g are trained, where $x_i \in X$ represents the attributes associated with v_i, $x_{\mathcal{N}_i}$ are the attributes associated with neighborhood of x_i, $l_i \in L$ is the label associated with v_i, $l_{\mathcal{N}_i}$ are the labels associated with $x_{\mathcal{N}_i}$, and $o \in \mathbb{R}^m$ is a generic output value. f aims to learn a representation for each node $v \in \mathcal{V}$, and a g aims to learn how to map a representation of each node into some output values. For example g may learn how to classify nodes for a node classification task. According to the Banach Theorem if f is a contraction, i.e. it exists $0 \leq \mu < 1$ such that $\forall\ x_i, x_j \in X$, $\|f(l_i, x_{\mathcal{N}_i}, l_{\mathcal{N}_i}) - f(l_j, x_{\mathcal{N}_j} - l_{\mathcal{N}_j})\| \leq \mu \|x_i - x_j\|$, then $x_i = f(l_i, x_{\mathcal{N}_i}, l_{\mathcal{N}_i}; w_f)$ has an unique solution. In practices the Jacobi iterative method for solving non-linear equations is used for all $x \in X$

$$x_i^{(t+1)} = f(l_i, x_{\mathcal{N}_i}^{(t)}, l_{\mathcal{N}_i}; w_f) \tag{2.5}$$

$$o_i = g(x_i^{(t)}; w_g). \tag{2.6}$$

Note the relation between Equation 2.5 and Algorithm 1 is immediate. Indeed it is sufficient to replace the hash$(\cdot)$ with f, and the node coloring with the attributes X and the label L. Note that Equation 2.5 is more general as it can handle continuous data, rather than only integer number, since that f is a neural network and $x \in X$ are continuous attributes in general. f and g are trained according to the back propagation through time algorithm. Furthermore by adding a constraint on the norm of the gradient of f, the contraction propriety of f is guaranteed for any structure of f, in terms of layers and activation functions.

Deep graph kernels

Deep graph kernels (DGK) (Yanardag and Vishwanathan, 2015) learn latent representations of sub-structures for graphs. This approach has been used by the authors for graph classification tasks. DGKs are defined by recursively decompose a graph into subgraphs. Many techniques can be used for decomposing a graph:

- *R-decomposition strategy (Haussler, 1999)*. A graph G is decomposed according to a relation $R(G, a)$ which holds true if a is "part" of G and forms $R^{-1}(G) = \{a : R(G, a)\}$, the bag of all parts of G[1];
- *Graphets*. A graph is decomposed into all the unique sub-graphs of size k;
- *Weisfeiler-Lehman kernel* see Section 2.2 ;

[1]It is worthwhile to mention that this strategy is adapted for many graph kernels in which given two graph G and G', a kernel $k(G, G')$ is expressed in terms of substructured kernels k_p, i.e. $k(G, G') = \sum_{s,s'} k_p(G, G')$, $s \in R^{-1}(G), s' \in R^{-1}(G')$.

- *Shortest-Path kernel*. A graphs is decomposed into shortest-paths.

Then, vectors embeddings are trained on the sub-structures with CBOW (Mikolov et al., 2013a) or Skip-gram (Mikolov et al., 2013b) models. Each graph-kernel feature is the normalized vector of frequencies where each entry of the vector represents the occurrences of each sub-structure in the graph. Finally, the Euclidean space or some other domain-specific Reproducing Kernel Hilbert Space (RKHS) is used to define the dot product between the vectors of frequencies.

Patchy-SAN

Patchy-SAN (Niepert et al., 2016) is another suitable approach for graphs classification. The idea is to construct features from each graph, which are suitable for learning convolutional neural networks (CNNs). Given a set of graphs, a receptive field size k, a number of receptive fields w, and a stride s, Patchy-SAN (Select-Assemble-Normalize) applies the following steps to each graph:

1. a fixed-length sequence of nodes is selected from each graph. For each graph G this step identifies a sequence of nodes for which receptive fields are created. First, the vertices of the input graph are sorted with respect to a given graph labelling (e.g. WL algorithm in Section 1). Then, the resulting node sequence is traversed using a given stride s and for each visited node, a receptive field is constructed (steps 2 and 3), until exactly w receptive fields have been created. The stride s determines the distance, relative to the selected node sequence, between two consecutive nodes for which a receptive field is created;

2. a fixed-size neighborhood is assembled for each node in the selected sequence. The nodes of the neighborhood are the candidates for the receptive field. By performing a breadth-first search on each node of the selected sequence, the nodes are collected into a set N until at least k nodes are in N, or until there are no more neighbors to add;

3. the extracted neighborhood graph are ordered according to a normalization algorithm. The normalization imposes an order on the nodes of the neighborhood graph so as to map from the unordered graph space to a vector space with a linear order. The basic idea is to leverage graph labelling procedures that assigns nodes of two different graphs to a similar relative position in the respective adjacency matrices if and only if their structural roles within the graphs are similar. We remand the reader to see (Niepert et al., 2016) for further details on this step;

4. learn neighborhood representations with convolutional neural networks from the resulting sequence of patches.

Graph Convolutional Network (GCN)

GCN (Kipf and Welling, 2017) proposed a method based on an efficient variant of convolutional neural networks which operate directly on graphs. Given a *undirected* graph $G = (\mathcal{V}, \mathcal{E})$, a set $X = \{x_1, \ldots, x_{|V|},\ x_i \in \mathbb{R}^m\}$ of nodes attributes, a GCN convolutional layer is defined as follows

$$h^{(t+1)} = \sigma(\hat{A}h^{(t)}W^{(t)}), \tag{2.7}$$

where $W^{(t)}$ are trainable weights, $h^{(t)}$ is a tensor of size $|\mathcal{V}| \times m$ ($h_i^{(0)} = x_i\ \forall i$), which represents the nodes, σ is an activation function e.g. sigmoid, ReLU, tanh, and $\hat{A}$ is the normalized adjacency matrix. $\hat{A}$ is defined as follows

$$\hat{A} = \tilde{D}^{-\frac{1}{2}}(A + I_{|V|})\tilde{D}^{-\frac{1}{2}}, \tag{2.8}$$

where A is adjacency matrix, I the identity matrix, and $\tilde{D}_{ii} = \sum_j (A + I_{|V|})_{ij}$. Note that I forces the self-loops for all the nodes in the graph. Furthermore the assumption of G to be undirected is crucial, as this guarantied that D is invertible. Again the relation with Algorithm 1 is immediate. Indeed it is sufficient to replace the hash$(\cdot)$ function with Equation 2.7.

GraphSAGE

GraphSAGE (Hamilton et al., 2017) learns embeddings for nodes of graphs by also using an algorithm inspired on the Weisfeiler-Lehman isomorphism test. The embeddings are generated by sampling and aggregating features from local neighborhoods of nodes. The initial representation h_v^0 for each node v corresponds to the attribute vector associated with v. For a fixed number K of times, a neural network $f(v; W_1, W_2) = (f_2 \circ f_1)(v)$, where f_1, f_2 are the layers, learns iteratively first the k-th neighborhood $\mathcal{N}(v)$ representation as

$$h_{\mathcal{N}(v)}^k = f_1(\{h_u^{k-1}; \forall\ u \in \mathcal{N}(v)\}; W_1), \tag{2.9}$$

and then the node representation as

$$h_v^k = f_2(\text{CONCAT}(h_v^{k-1}, h_{\mathcal{N}(v)}^k); W_2). \tag{2.10}$$

CONCAT represent the concatenation between the two vectors h_v^{k-1}, $h_{\mathcal{N}(v)}^k$. f_1 represents a special case of neural networks in which a set of representations of its input is aggregated by using max, $mean$, or $LSTM$. Once again the connection with Algorithm 1 is immediate. Indeed it is sufficient to replace the hash$(\cdot)$ function with Equations 2.9, and 2.10.

2.3 Interpretability

Neural networks have achieved state-of-the-art results in several important tasks such as image classification (Krizhevsky et al., 2012), object detection (Ren et al., 2015) and text classification (Conneau et al., 2017). In spite of their ability to learn powerful representations from data, leading to high prediction accuracy, neural network decisions are largely opaque, making it difficult to explain and understand the prediction associated with a new data point. In many applications such as medical context, providing an explanation of the decision of the model is crucial. In detection of cancer cells for example is very important to have the reasoning of the decision in order to make a decision valuable.

Decision Trees

Decision trees (Breiman, 2017) are machine learning models commonly used to solve classification problems. Although decision trees suffer of several drawbacks such as poor accuracy and instability, i.e. a small change in the data can lead to a large change in the structure of the optimal decision tree, they have the advantage to be easy to understand and to interpret. Indeed decision tree can be visualized and can be represented as a set of "if-then" rules. This property leads decision trees to be still used in all the situations where decisions must be made effectively and reliably, e.g. medical diagnosing.

The problem of learning the optimal decision tree is NP-complete (Laurent and Rivest, 1976). Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as greedy algorithms where locally optimal decisions are made at each node, and hence cannot guarantee to return the globally optimal decision tree.

Several algorithms were proposed to learn decision trees, such as ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993), C5.0 (an improved version of C4.5), and CART (Breiman, 2017). In this thesis we will briefly show the idea behind CART.

Let us consider the dataset $\mathcal{D} = \{(\boldsymbol{x_i}, y_i), \boldsymbol{x_i} \in \mathbb{R}^n, y_i \in \{1, \dots, K\}\}_{i=1}^N$. A decision tree partitions the instance space recursively into regions which contain examples with the same labels. Let $Q = \mathcal{D}$ be the initial set. For each possible split $s = (j, t)$, where $j \in \{1, \dots, n\}$ corresponds to an entry of the instance vector $\boldsymbol{x_{ij}}$, and t is a threshold, the set Q is partitioned into two subsets, $Q_l(s) = \{(\boldsymbol{x_i}, y_i); \boldsymbol{x_{ij}} \leq t\}$, $Q_r(s) = \{(\boldsymbol{x_i}, y_i); \boldsymbol{x_{ij}} > t\}$. The split s is chosen in order to minimize the average of the Gini impurity function H over $Q_l(s)$ and $Q_r(s)$

$$s^* = \arg\min_s \frac{|Q_l(s)|}{|Q|} H(Q_l(s)) + \frac{|Q_r(s)|}{|Q|} H(Q_r(s)).$$

If p_k is the fraction of the instances labeled with class k in the set Q, $H(Q)$ is defined as

$$H(Q) = 1 - \sum_{k=1}^{K} p_k^2.$$

The partitioning strategy is applied recursively on $Q_l(s^*)$ and $Q_r(s^*)$ until the maximum depth is reached, the minimum allowable number of instances within the leaf is reached or there is only one instance within the leaf.

Layer-wise Relevance Propagation (LRP)

Lapuschkin et al. (2016); Samek et al. (2016) formalize a framework for a specific domain scenario: image classification. Let us assume to have a trained neural network $f : \mathbb{R}^m \mapsto \mathbb{R}$, on dataset $\mathcal{D} = \{(x_i, y_i)\ x_i \in \mathbb{R}^m, y_i \in \mathbb{R}\}_{i=1}^N$. In this context x_i represents an image, and each of its entry represents a pixel. Each pixel of each image, x_{ij}, is associated with a *relevance score* R_{ij}, such that $\sum_j R_{ij} = f(x_i)$. If f consist of L layers, the relevance score it is defines as follows:

$$R_{ij}^{(\ell+1)} = \left(\alpha \frac{(x_{ij} w_{ij}^{(\ell)})^+}{(\sum_i x_{ij} w_{ij}^{(\ell)})^+} - \beta \frac{(x_{ij} w_{ij}^{(\ell)})^-}{(\sum_i x_{ij} w_{ij}^{(\ell)})^-} \right) R_{ij}^{(\ell)},\ \ell = 1, \ldots, L, \tag{2.11}$$

where $()^+$, and $()^-$ denote the positive and the negative contributions, respectively. $w_{ij}^{(l)}$ is the weight that connects the neuron i to neuron j at the layer l. If $\alpha - \beta = 1$ the layer-wise conservation property holds for each layer, i.e. $\sum_j R_j^{(\ell+1)} = \sum_j R_j^{(\ell)}$. Under this assumptions, it follows

$$f(x_i) = \sum_j R_{ij}^L. \tag{2.12}$$

For learning the interpretable model in this specific domain is then sufficient to train a linear model for each x_i. Note that this approach explains single instances (images) and it does not provide a global explanation for the whole neural network.

Local Interpretable Model-agnostic Explanations (LIME)

Ribeiro et al. (2016) provides explanations for individual predictions as a solution to the "trusting a prediction" by approximating a machine learning model with an interpretable model. The authors assume that instances are given in a representation which is understandable to humans, regardless of the actual features used by the model. For example for text classification an interpretable representation may be the binary vector indicating the presence or absence of a word.

An "interpretable" model is defined as a model that can be readily presented to the user with visual or textual artefacts (linear models, decision trees, or falling rule lists), which locally approximates the original machine learning model.

Given a machine learning model f, an interpretable model g is trained for each instance. For each instance x_i, a set of instances X_i is generated around x_i by dropping out randomly some nonzero entries from x_i. Given a similarity measure D, e.g. scalar product, gaussian kernel, cosine distance, g is trained by minimizing

$$\min_{x_v \in X_i} D(x_i, x_v)(f(x_v) - g(x_v))^2.$$

Similarly to LRP, also for this framework an "interpretable" model is trained for each instance, i.e. explanations are given instance by instance.

Interpreting multi-instance learning models

Some of the models, seen in Chapter 3, provide by definition a natural interpretation of themselves. In particular, all the methods which label the instances, i.e. *Diverse Density*, *mi-SVM*, and *multi-instance neural networks*, are immediately interpretable. The only rule which globally explains the model is intrinsically defined by the MIL assumption, i.e. a bag is positive iff at least one of its instances is positive. Moreover by inspecting the instance labels, we can immediately understand which instance is responsible for the class of the bag.

For better clarifying the benefits of MIL interpretation let us consider a real example in which we have access to a biological dataset consisting of images of tissues. Each image is associate with a binary label: 1 if the tissue is unhealthy, and 0 otherwise. Let also assume, that a tissue is unhealthy if it contains at least a sick cell. In this respect, it is reasonable to treat this problem as MIL task in which each image can be decomposed into a set of patches. By training a MIL model which label the instances, we are able to interpret each image. Indeed for each image we can show which patches are responsible for the image labels.

Chapter 3

Multi-multi instance learning

We introduce an extension of the multi-instance learning problem where examples are organized as nested bags of instances (e.g., a document could be represented as a bag of sentences, which in turn are bags of words). This framework can be useful in various scenarios, such as text and image classification. Our approach is based on a special neural network layer, called bag-layer, whose units aggregate bags of inputs of arbitrary size. We prove theoretically that the associated class of functions contains all Boolean functions over sets of sets of instances and we provide empirical evidence that functions of this kind can be actually learned on semi-synthetic datasets.[1]

Relational learning takes several different forms ranging from purely symbolic (logical) representations, to a wide collection of statistical approaches (De Raedt et al., 2008) based on tools such as probabilistic graphical models (Jaeger, 1997; De Raedt et al., 2008; Richardson and Domingos, 2006; Getoor and Taskar, 2007), kernel machines (Landwehr et al., 2010), and neural networks (Frasconi et al., 1998; Scarselli et al., 2009; Niepert et al., 2016).

Multi-instance learning (MIL) is perhaps the simplest form of relational learning where data consists of labeled bags of instances. Introduced in (Dietterich et al., 1997), MIL has attracted the attention of several researchers during the last two decades and has been successfully applied to problems such as image and scene classification (Maron and Ratan, 1998; Zha et al., 2008; Zhou et al., 2012), image annotation (Yang et al., 2006), image retrieval (Yang and Lozano-Perez, 2000; Rahmani et al., 2005), Web mining (Zhou et al., 2005), text categorization (Zhou et al., 2012) and diagnostic medical imaging (Hou et al., 2015; Yan et al., 2016). In classic MIL, labels are binary and bags are positive iff they contain at least one positive

[1] Part of the content of this chapter has been published as "A network architecture for multi-multi-instance learning" in *Joint European Conference on Machine Learning and Knowledge Discovery in Database*, Skopje, 2017 (Tibo et al., 2017).

instance (existential semantics). For example, a visual scene with animals could be labeled as positive iff it contains at least one tiger. Various families of algorithms have been proposed for MIL, including axis parallel rectangles (Dietterich et al., 1997), diverse density (Maron and Lozano-Pérez, 1998), nearest neighbors (Wang and Zucker, 2000), neural networks (Ramon and De Raedt, 2000), and variants of support vector machines (Andrews et al., 2003).

In this chapter, we extend the MIL setting (see Section 2.1) by considering examples consisting of labeled nested bags of instances. Labels are observed for top-level bags, while instances and lower level bags have associated latent labels. For example, a potential offside situation in a soccer match can be represented by a bag of images showing the scene from different camera perspectives. Each image, in turn, can be interpreted as a bag of players with latent labels for their team membership and/or position on the field. We call this setting multi-multi-instance learning (MMIL), referring specifically to the case of bags-of-bags[2]. In our framework, we also relax the classic MIL assumption of binary instance labels, allowing categorical labels lying in a generic alphabet. This is important since MMIL with binary labels under the existential semantics would reduce to classic MIL after flattening the bag-of-bags.

We propose a solution to the MMIL problem based on neural networks with a special layer called *bag-layer*. Unlike previous neural network approaches to MIL learning (Ramon and De Raedt, 2000), where predicted instance labels are aggregated by (a soft version of) the maximum operator, bag-layers aggregate internal representations of instances (or bags of instances) and can be naturally intermixed with other layers commonly used in deep learning. Bag-layers can be in fact interpreted as a generalization of convolutional layers followed by pooling, as commonly used in deep learning.

The MMIL framework can be immediately applied to solve problems where examples are naturally described as bags-of-bags. For example, a text document can be described as a bag of sentences, where in turn each sentence is a bag of words. The range of possible applications of the framework is however larger. In fact, every structured data object can be recursively decomposed into parts, a strategy that has been widely applied in the context of graph kernels (see e.g., (Haussler, 1999; Gärtner et al., 2004; Passerini et al., 2006; Shervashidze et al., 2009; Costa and De Grave, 2010; Orsini et al., 2015)). Hence, MMIL is also applicable to supervised graph classification. Experiments on bibliographical and social network datasets confirm the practical viability of MMIL for these forms of relational learning (see Chapter 5).

This chapter is organized as follows. In Section 3.1 we formally introduce the MMIL setting. In Section 3.2 we formalize bag layers and the resulting neural net-

[2] the generalization to deeper levels of nesting is straightforward but not explicitly formalized in the chapter for the sake of simplicity.

work architecture for MMIL. In Section 3.3 we discuss some related works. extracting rules from trained networks of bag-layers. In Section 3.4 we report experimental results on a semi-synthentic dataset.

3.1 Multi-multi-instance learning framework

We show in this Section an extension of the standard multi-instance learning framework seen in Section 2.1: the multi-multi instance learning framework (MMIL). In the MMIL setting, we call the elements of $\mathcal{M}(\mathcal{M}(\mathcal{X}))$ and $\mathcal{M}(\mathcal{X})$ *top-bags* and *sub-bags*, respectively.

Now postulating unobserved labels for both the instances and the sub-bags, we interpret examples (X, y) as the observed part of fully labeled data points $(X^l, y) \in \mathcal{M}(\mathcal{M}(\mathcal{X} \times \mathcal{Y}^I) \times \mathcal{Y}^S) \times \mathcal{Y}$, where $\mathcal{Y}^S$ is the space of sub-bag labels. Fully labeled data points are drawn from a distribution $p(X^l, y)$.

As in MIL, we make some conditional independence assumptions. Specifically, we assume that instance and sub-bag labels only depend on properties of the respective instances or sub-bags, and not on other elements in the nested multiset structure X^l (thus excluding models for contagion or homophily, where, e.g., a specific label for an instance could become more likely, if many other instances contained in the same sub-bag also have that label). Furthermore, we assume that labels of sub-bags and top-bags only depend on the labels of their constituent elements. Thus, for $y \in \mathcal{Y}^S$, and a bag of labeled instances $S^l = \{(x_1, y_1), \ldots, (x_{|S|}, y_{|S|})\}$ we have:

$$p(y|S^l) = p(y|y_1, \ldots, y_{|S|}). \tag{3.1}$$

Similarly for the probability distribution of top-bag labels given the constituent labeled sub-bags.

Example 3.1.1. *In this example we consider bags-of-bags of handwritten digits (as in the MNIST dataset). Each instance (a digit) has attached its own latent class label in $\{0, \ldots, 9\}$ whereas sub-bag (latent) and top-bag labels (observed) are binary. In particular, a sub-bag is positive iff it contains an instance of class 7 and does not contain an instance of class 3. A top-bag is positive iff it contains at least one positive sub-bag. Figure 3.1 shows a positive and a negative example.*

Example 3.1.2. *A top-bag can consist of a set of images showing a potential offside situation in soccer from different camera perspectives. The label of the bag corresponds to the referee decision $\mathcal{Y} \in \{\text{offside}, \text{not offside}\}$. Each individual image can either settle the offside question one way or another, or be inconclusive. Thus, there are (latent) image labels $\mathcal{Y}^S \in \{\text{offside}, \text{not offside}, \text{inconclusive}\}$. Since no offside should be called when in doubt, the top-bag is labeled as 'not offside' if and only if it either contains at least one image labeled*

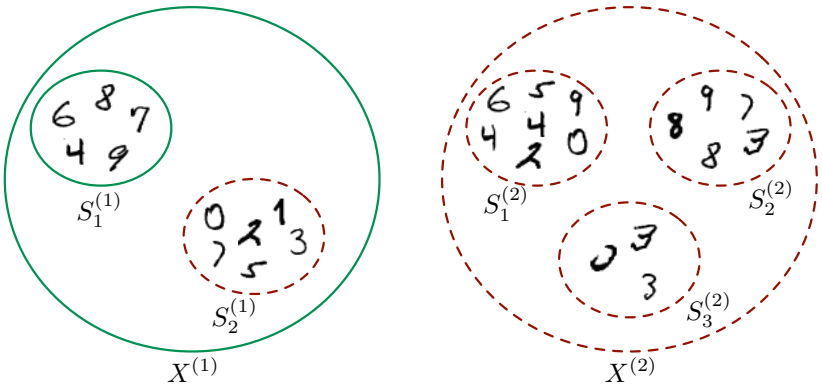


Figure 3.1: A positive (left) and a negative (right) top-bag for Example 3.1.1. Solid green lines represent positive bags while dashed red lines represent negative bags.

'not offside', or all the images are labeled 'inconclusive'. Images, in turn, can be seen as bags of player instances that have a label $\mathcal{Y}^I \in \{behind, in\ front, inconclusive\}$ *according to their relative position with respect to the potentially offside player of the other team. An image then is labeled 'offside' if all the players in the image are labeled 'behind'; it is labeled 'not offside' if it contains at least one player labeled 'in front', and is labeled 'inconclusive' if it only contains players labeled 'inconclusive' or 'behind'.*

Example 3.1.3. *In text categorization, the bag-of-word representation is often used to feed documents to classifiers. Each instance in this case consists of the indicator vector of words in the document (or a weighted variant such as TF-IDF). The MIL approach has been applied in some cases (Andrews et al., 2003) where instances consist of chunks of consecutive words and each instance is an indicator vector. A bag-of-bags representation could instead describe a document as a bag of sentences, and each sentence as a bag of word vectors (constructed for example using Word2vec or GloVe).*

3.2 A network architecture for MMIL

Bag layers

We model the conditional distribution $p(y|X)$ with a neural network architecture that handles bags-of-bags of variable sizes by aggregating intermediate internal representations. For this purpose, we introduce a new layer called *bag-layer*. A bag-layer takes as input a bag of m-dimensional vectors $\{\phi_1, \ldots, \phi_n\}$, and first computes k-dimensional representations

$$\rho_i = \alpha\left(w\phi_i + b\right) \tag{3.2}$$

using a weight matrix $w \in \mathbb{R}^{k \times m}$, a bias vector $b \in \mathbb{R}^k$, and an activation function α (such as ReLU, tanh, or linear). The bag layer then computes its output as:

$$g(\{\phi_1, \ldots, \phi_n\}; w, b) = \underset{i=1}{\overset{n}{\Xi}} \rho_i \tag{3.3}$$

where Ξ is element-wise aggregation operator (such as max or average). Both w and b are tunable parameters. Note that Equation 3.3 works with bags of arbitrary cardinality. A bag-layer is illustrated in Figure 3.2.

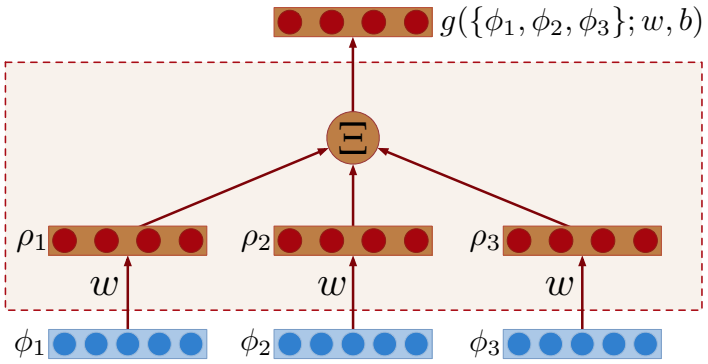


Figure 3.2: A bag-layer receiving a bag of cardinality $n = 3$. In this example $k = 4$ and $m = 5$.

Networks with a single bag-layer can process bags of instances (as in the standard MIL setting). To solve the MMIL problem, two bag-layers are required. The bottom bag-layer aggregates over internal representations of instances; the top bag-layer aggregates over internal representations of sub-bags, yielding a representation for the entire top-bag. In this case, the representation of each sub-bag $S_j = \{x_{j,1}, \ldots, x_{j,|S_j|}\}$ would be obtained as

$$\phi_j = g(x_{j,1}, \ldots, x_{j,|S_j|}; w_s, b_s) \;\; j = 1, \ldots, |X| \tag{3.4}$$

and the representation of a top-bag $X = \{S_1, \ldots, S_{|X|}\}$ would be obtained as

$$\phi = g(\phi_1, \ldots, \phi_{|X|}; w_t, b_t) \tag{3.5}$$

where (w_s, b_s) and (w_t, b_t) denote the parameters used to construct sub-bag and top-bag representations. Furthermore, different aggregation functions can be also evaluated in parallel.

Note that nothing prevents us from intermixing bag-layers with standard neural network layers, thereby forming networks of arbitrary depth. In this case, each $x_{j,\ell}$ in Eq. (3.4) would be simply replaced by the last layer activation of a deep network taking $x_{j,\ell}$ as input. Denoting by θ_s the parameters of such network and by $N_s(x; \theta_s)$ its last layer activation when fed with instance x, Eq. (3.4) becomes

$$\phi_j = g(N_s(x_{j,1}; \theta_s), \ldots, N(x_{j,|S_j|}; \theta_s); w_s, b_s) \;\; j = 1, \ldots, |X|. \tag{3.6}$$

Similarly, we may use a network N_t, with parameters θ_t, to transform sub-bag representations. As a result, Eq. (3.5) becomes

$$\phi = g(N_t(\phi_1;\theta_t),\ldots,N_t(\phi_{|X|};\theta_t);w_t,b_t). \tag{3.7}$$

Of course the top-bag representation can be itself further processed by other layers. An example of the overall architecture is shown in Figure 3.3, while the detailed Python implementation is provided in Appendix, A.

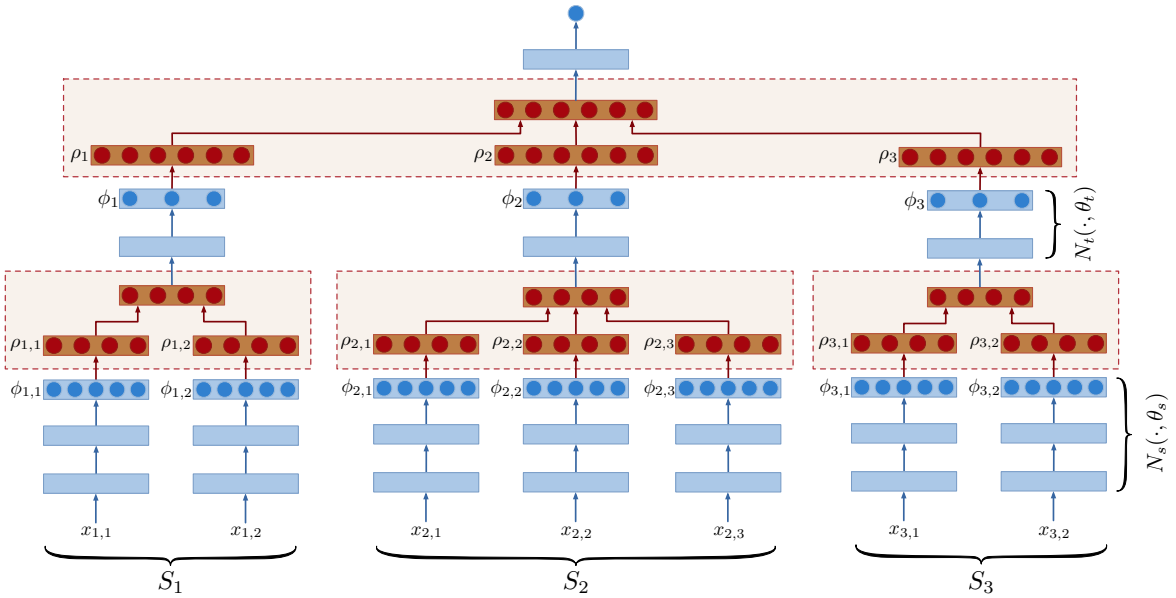


Figure 3.3: Network for multi-multi instance learning applied to the bag-of-bags $\{\{x_{1,1},x_{1,2}\},\{x_{2,1},x_{2,2},x_{2,2}\},\{x_{3,1},x_{3,2}\}\}$. Bag-layers are depicted in red with dashed borders. Blue boxes are standard (e.g., dense) neural network layers. Note that parameters θ_s in each of the seven bottom vertical columns are shared, as well parameters θ_t in the middle three columns.

Expressiveness of networks of bag-layers

We focus here on a deterministic (noiseless) version of the MMIL setting described in Section 3.1 where labels are deterministically assigned and no form of counting is involved. We show that under these assumptions, the architecture of Section 3.2 has enough expressivity to represent the solution to the MMIL problem. Our approach relies on classic universal interpolation results for neural networks (Hornik et al., 1989). Note that existing results hold for vector data, and this section shows that they can be leveraged to bag-of-bag data when using the architecture of Section 3.2.

Definition 3.2.1. We say that data is generated under the deterministic MMIL setting if the following conditions hold true:

1. instance labels are generated by an unknown function $\hat{f} : \mathcal{X} \mapsto \mathcal{Y}^I$, i.e., $y_{j,\ell} = \hat{f}(x_{j,\ell})$, for $j = 1, \ldots, |X|$, $\ell = 1, \ldots |S_j|$;

2. sub-bag labels are generated by an unknown function $\hat{g} : \mathcal{M}(\mathcal{Y}^I) \mapsto \mathcal{Y}^S$, i.e., $y_j = \hat{g}(\{y_{j,1}, \ldots, y_{j,|S_j|}\})$;

3. the top-bag label is generated by an unknown function $\hat{h} : \mathcal{M}(\mathcal{Y}^S) \mapsto \{0, 1\}$, i.e., $y = \hat{h}(\{y_1, \ldots, y_{|X|}\})$.

When data is generated in the deterministic setting, then the label of a bag-of-bags

$$X = \{\{x_{1,1}, \ldots, x_{1,S_1}\}, \ldots, \{x_{|X|,1}, \ldots, x_{|X|,|S_{|X|}|}\}\},$$

would be produced as

$$y = \hat{h}\left(\left\{\hat{g}\left(\{\hat{f}(x_{1,1}), \ldots, \hat{f}(x_{1,|S_1|})\}\right), \ldots, \hat{g}\left(\{\hat{f}(x_{|X|,1}), \ldots, \hat{f}(x_{|X|,|S_{|X|}|})\}\right)\right\}\right)$$

Note that the classic MIL formulation (Maron and Lozano-Pérez, 1998) is recovered when examples are sub-bags, $\mathcal{Y}^I = \{0, 1\}$, and $\hat{g}(\{y_1, \ldots, y_{|S|}\}) = \mathbb{1}\left\{0 < \sum_{\ell=1}^{|S|} y_\ell\right\}$. Other generalized MIL formulations (Foulds and Frank, 2010; Scott et al., 2005; Weidmann et al., 2003) can be similarly captured in this deterministic setting.

For a multiset s let $\text{set}(s)$ denote the set of elements occurring in s. E.g. $\text{set}(\{0, 0, 1\}) = \{0, 1\}$.

Definition 3.2.2. We say that data is generated under the non-counting deterministic MMIL setting if, in addition to the conditions of Definition 3.2.1, both $\hat{g}(s)$ and $\hat{h}(s)$ only depend on $set(s)$.

The following result indicates that a network containing a bag-layer with max aggregation is sufficient to compute the functions that label both sub-bags and top-bags.

Lemma 3.2.1. *Let $C = \{c_1, \ldots, c_M\}, D = \{d_1, \ldots, d_L\}$ be sets of labels, and let $\hat{g} : \mathcal{M}(C) \mapsto D$ be a labeling function for which $\hat{g}(s) = \hat{g}(s')$ whenever $set(s) = set(s')$. Then there exist a network with one bag-layer that computes $\hat{g}$.*

Proof. We construct a network N where first a bag-layer maps the multiset input s to a bit-vector representation of $set(s)$, on top of which we can then compute $\hat{g}(s)$ using a standard architecture for Boolean functions.

In detail, N is constructed as follows: the input $s = \{y_1, \ldots, y_{|s|}\}$ is encoded by $|s|$ M-dimensional vectors ϕ_i containing the one-hot representations of the y_i. We construct a bag-layer with $k = m = M$, w is the $M \times M$ identity matrix, b is zero,

α is the identity function, and Ξ is *max*. The output of the bag-layer then is an M-dimensional vector ψ whose i'th component is the indicator function $\mathbb{1}\{c_i \in s\}$.

For each $j \in \{1, \ldots, L\}$ we can write the indicator function $\mathbb{1}\{\hat{g}(set(s)) = d_j\}$ as a Boolean function of the indicator functions $\mathbb{1}\{c_i \in s\}$. Using standard universal approximation results (see, e.g., (Hornik et al., 1989), Theorem 2.5) we can construct a network that on input ψ computes $\mathbb{1}\{\hat{g}(set(s)) = d_j\}$. L such networks in parallel then produce an L-dimensional output vector containing the one-hot representation of $\hat{g}(s)$. □

Theorem 3.2.2. *Given a dataset of examples generated under the non-counting deterministic MMIL setting, there exist a network with two bag-layers that can correctly label all examples in the dataset.*

Proof. We first note that the universal interpolation result of (Hornik et al., 1989) can be applied to a network taking as input an instance x which appears in any data example, and generating the desired label $\hat{f}(x)$. We then use Lemma 1 twice, first to form a network that computes the sub-bag labeling function $\hat{g}$, and then to form a network that computes the top-bag labeling function $\hat{h}$. □

3.3 Related Works

Multi-instance neural networks

Ramon and De Raedt (2000) proposed a neural network solution to MIL where each instance x_j in a bag $X = \{x_1, \ldots, x_{|X|}\}$ is first processed by a replica of a neural network f with weights w. In this way, a bag of output values $\{f(x_1; w), \ldots, f(x_{|X|}; w)\}$ computed for each bag of instances. These values are then aggregated by a smooth version of the max function:

$$F(X) = \frac{1}{M} \log \left(\sum_j e^{Mf(x_j;w)} \right)$$

where M is a constant controlling the sharpness of the aggregation (the exact maximum is computed when $M \to \infty$). Recall that a single bag-layer (as defined in Section 3.2) can used to solve the MIL problem. Still, a major difference compared to the work of (Ramon and De Raedt, 2000) is that bag-layers perform aggregation at the *representation* level rather than at the output level. In this way, more layers can be added on the top of the aggregated representation, allowing for more expressiveness. In the classic MIL setting (where a bag is positive iff at least one instance is positive) this additional expressiveness is not required. However, it allows us to solve slightly more complicated MIL problems. For example, suppose each instance has a latent variable $y_j \in 0, 1, 2$, and suppose that a bag is positive iff it contains at

least one instance with label 0 and no instance with label 2. In this case, a bag-layer with two units can distinguish positive and negative bags, provided that instance representations can separate instances belonging to the classes 0, 1 and 2. The network proposed in (Ramon and De Raedt, 2000) would not be able to separate positive from negative bags. Indeed, as proved in Section 3.2, networks with bag-layers can represent any Boolean function over sets of instances.

Convolutional neural networks

Convolutional neural networks (CNN) (Fukushima, 1980; LeCun et al., 1989) are the state-of-the-art method for image classification (see, e.g., (Szegedy et al., 2016)). It is easy to see that the representation computed by one convolutional layer followed by max-pooling can be emulated with one bag-layer by just creating bags of adjacent image patches. The representation size k corresponds to the number of convolutional filters. The major difference is that a convolutional layer outputs spatially ordered vectors of size k, whereas a bag-layer outputs a set of vectors (without any ordering). This difference may become significant when two or more layers are sequentially stacked. Figure 3.4 illustrates the relationship between a convolu-

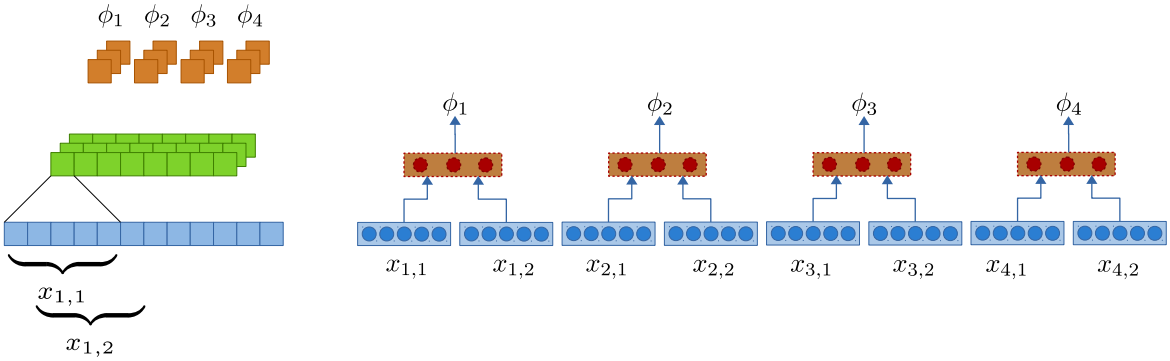


Figure 3.4: One convolutional layer with subsampling (left) and the corresponding bag-layer (right). Note that the convolutional layer outputs $[\phi_1, \phi_2, \phi_3, \phi_4]$ whereas the bag-layer outputs $\{\phi_1, \phi_2, \phi_3, \phi_4\}$.

tional layer and a bag-layer, for simplicity assuming a one-dimensional signal (i.e., a sequence). When applied to signals, a bag-layer essentially correspond to a disordered convolutional layer and its output needs further aggregation before it can be fed into a classifier. The simplest option would be to stack one additional bag-layer before the classification layer. Interestingly, a network of this kind would be able to detect the presence of a short subsequence regardless of its position within the whole sequence, achieving invariance to arbitrarily large translations

We finally note that it is possible to emulate a CNN with two layers by properly defining the structure of bags-of-bags. For example, a second layer with filter size 3

on the top of the CNN shown in Figure 3.4 could be emulated with two bag-layers fed by the bag-of-bags

$$\{\{\{x_{1,1}, x_{1,2}\}, \{x_{2,1}, x_{2,2}\}, \{x_{3,1}, x_{3,2}\}\}, \{\{x_{2,1}, x_{2,2}\}, \{x_{3,1}, x_{3,2}\}, \{x_{4,1}, x_{4,2}\}\}\}.$$

A bag-layer, however, is not limited to pooling adjacent elements in a feature map. One could for example segment the image first (e.g., using a hierarchical strategy (Arbeláez et al., 2011)) and then create bags-of-bags by following the segmented regions.

Nested SRL Models

In Statistical Relational Learning (SRL) a great number of approaches have been proposed for constructing probabilistic models for relational data. Relational data has an inherent bag-of-bag structure: each object o in a relational domain can be interpreted as a bag whose elements are all the other objects linked to o via a specific relation. These linked objects, in turn, also are bags containing the objects linked via some relation. A key component of SRL models are the tools employed for aggregating (or combining) information from the bag of linked objects. In many types of SRL models, such an aggregation only is defined for a single level. However, a few proposals have included models for nested combination (Jaeger, 1997; Natarajan and Van der Ven, 2018). Like most SRL approaches, these models employ concepts from first-order predicate logic for syntax and semantics, and (Jaeger, 1997) contains an expressivity result similar in spirit to the one we present in the following section 3.2.

A key difference between SRL models with nested combination constructs and our MMIL network models is that the former build models based on rules for conditional dependencies which are expressed in first-order logic and typically only contain a very small number of numerical parameters (such as a single parameter quantifying a noisy-or combination function for modelling multiple causal influences). MMIL network models, in contrast, make use of the high-dimensional parameter spaces of (deep) neural network architectures. Roughly speaking, MMIL network models combine the flexibility of SRL models to recursively aggregate over sets of arbitrary cardinalities with the power derived from high-dimensional parameterisations of neural networks.

3.4 Experiments

We evaluated our model on two experimental setups:

1. we constructed a multi-multi instance learning semi-synthetic dataset from MNIST, in which digits were organized in bags-of-bags of arbitrary cardinality.

This setup follows the Example 3.1.1 shown in Section 3.1. The aim of this experiment is to show the ability of the network to learn functions that have generated the data according to Theorem 3.2.2 in Section 3.2;

2. we constructed semi-synthetic dataset from MNIST, placing digits randomly into a background images of black pixels. The aim of this experiment is to show the disordered convolution property discussed in Section 3.3;

MNIST

Results of Section 3.2 show that networks with bag layers can represent any labelling function in the non-counting deterministic MMIL setting. We show here that these networks trained by gradient descent can actually learn such functions from MMIL data.

In this section we show two results concerning the ability of the MMIL networks of learning the rules that generated the data. By considering digit images extracted from MNIST dataset as instances, we constructed two different setups: a "logic scenario" and an "algebraic scenario". For both the scenarios, we constructed two MMIL datasets, $\mathcal{D}_l$ and $\mathcal{D}_a$, for the logic and the algebraic scenarios, respectively. Each top-bag X, in both $\mathcal{D}_l$ and $\mathcal{D}_a$, is a bag-of-bags of images. For each sub-bag S_j, we denote by $Y_j = \{y_{j,1} \dots, y_{j,|S_j|}\}$ the set of instance labels in sub-bag S_j, where $y_{j,\ell}$ are derived from the MNIST dataset labels.

Logic Scenario The setup is similar to Example 3.1.1 in Section 3.1, but the classification rule is more complicated. We labeled each sub-bag according to the following rules:

- sub-bag S_j has label 0 iff one of the following conditions is satisfied: $\{1,3,5,7\} \subset Y_j$, $\{2,1,3\} \subset Y_j$, $\{3,2,7,9\} \subset Y_j$;
- sub-bag S_j has label 1 iff S_j has not label 0 and one of the following conditions is satisfied: $\{8,9\} \subset Y_j$, $\{4,5,6\} \subset Y_j$, $\{7,2,1\} \subset Y_j$;
- sub-bag S_j has label 2 iff S_j has not label 0 and S_j has not label 1.

Finally each top-bag is positive if and only if it contains at least a sub-bag of class 1. Observe that data generated according to those rules satisfied the conditions of Definitions 3.2.1 and 3.2.2. Using these rules, we generated a balanced training set of $5,000$ top-bags and a balanced test set of $5,000$ top-bags. Sub-bag and top-bag cardinalities were uniformly sampled in the interval $[2,6]$. Instances in the training set were randomly sampled with replacement from the 60,000 MNIST training images, while instances in the test set were randomly sampled with replacement from the 10,000 MNIST test images. The model we used for training $\mathcal{D}_l$ has the following

structure: two stacked convolutional layers (the first with 32 channels, while the second with 64 channels) with kernel size 5×5, batch normalization, max pooling 2×2 and ReLU activations. Following the convolutional layers, the MMIL network has a dense layer with 1,024 units with ReLU activation and dropout with probability 0.5, two stacked bag-layers with 100 units, ReLU activations, and max as a aggregation function, and finally a dense layer with 1 unit with sigmoid activation.

The model was trained by minimizing the L2-regularized binary cross-entropy loss (with L2 penalty $5 \cdot 10^{-4}$). We stress the fact that instance and sub-bag labels were not used to form the training objective. We ran 200 epochs of the Adam optimizer (Kingma and Ba, 2014) with learning rate 0.001. The model reached an accuracy equals to 96.44% on the test set which confirms that the network is able to recover the latent logic function that was used in the data generation process with a reasonably high accuracy.

Algebraic Scenario For the MMIL dataset $\mathcal{D}_a$, we labelled each sub-bag S_j with the product of the instance labels, $y_{j,\ell}$, which belongs to it. Each top-bag is then labelled with the summation of the labels associated to the sub-bags contained in it. For example, let us consider a top-bag $X = \{S_1, S_2, S_3\}$ in which each sub-bag S_1 and S_2 contain 3 instances, and S_3 contains 2 instances. The labels associated with the instances are $\{1, 3, 5\}$, $\{7, 2, 3\}$, and $\{8, 0\}$ for S_1, S_2, and S_3, respectively. Hence the labels attached to the sub-bags are 15, 42 and 0. Finally, the label attached to X is 57. By using these rules, we generated in turn training and test sets containing 500k top-bags. Sub-bag and top-bag cardinalities were uniformly sampled in the interval $[1, 3]$. Instances in the training set and in the test set were randomly sampled with replacement from the 60,000 MNIST training images and the 10,000 MNIST test images, respectively.

The model we used for training $\mathcal{D}_a$ has the following structure: two stacked convolutional layers (the first with 6 channels, while the second with 16 channels) with kernel size 5×5, batch normalization, max pooling 2×2 and ReLU activations. Following the convolutional layers, the MMIL network has a two stacked bag-layers with $1,000$ units, ReLU activations and summation as aggregation function, a dense layer with $1,000$ units and ReLU activation, and finally a dense layer with 12 units with sigmoid activation. We represented the top-bag labels as base 2 numbers. Since that the top-bag labels range from 0 to 2,187, the number of bits for representing the top-bag labels is, hence, 12.

In this experiment, we also trained a standard convolutional network, i.e. without bag-layers, which has the same structure as the MMIL network (with the exception of the bag-layers). Starting from the MMIL dataset $\mathcal{D}_a$, we constructed a compatible dataset for the standard convolutional network by concatenating all the digits within a top-bag into one image. MNIST dataset consists of 28×28 pixels

images of digits. The final image, of size 28×252, contains at most 9 MNIST digits, in which the first 3 blocks of 28×28 pixels are reserved for the first sub-bag, the second 3 blocks of 28×28 pixels are reserved for the second sub-bag, and the last 3 blocks of 28×28 pixels are reserved for the last sub-bag. Whereas the cardinalities of sub-bags and instances are lower than 3, we used an empty image (in place of a MNIST image), i.e. with all the pixels equal to 0. Figure 3.5 depicts an example of how a top-bag $X \in \mathcal{D}_a$ is transformed into a compatible input image for the standard convolutional network.

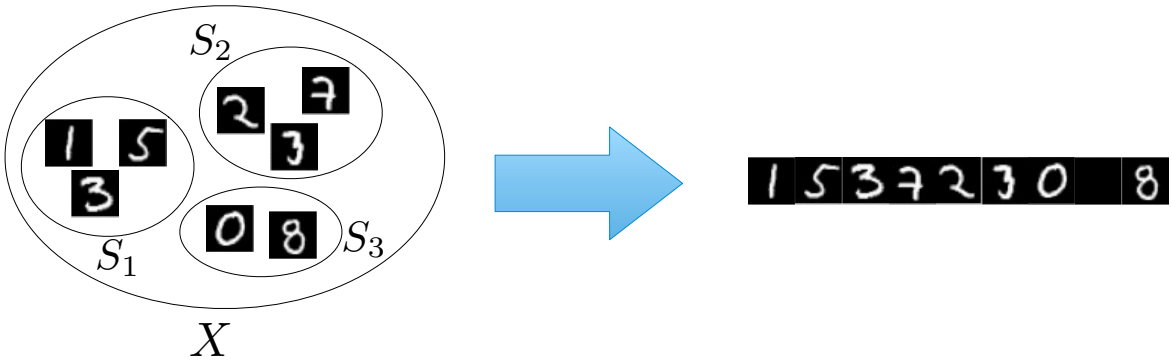


Figure 3.5: A top-bag X is transformed into a CNN input data.

Both the models were trained by minimizing the binary cross-entropy loss. We ran 200 epochs of the Adam optimizer (Kingma and Ba, 2014) with learning rate 0.001. The MMIL model reached an accuracy equals to 91.23%, while the standard convolutional network model reached an accuracy equals to 79.52%. These results confirm that our model is able to exploit the unordering property of the bag-layers described in Section 3.3. While for the MMIL network two sub-bags containing for example $\{1, 2, 3\}$ and $\{3, 2, 1\}$ are equivalent, for the standard convolutional network those sets differ. We will deepen this bag-layer property in the next experiment.

As a last experiment, we compared our model against DeepProbLog (Manhaeve et al., 2018) in which the authors proposed a simpler experiment on MNIST where the cardinalities of the sub-bags and instances are fixed to 2 and 1, respectively. In this respect we constructed a MIL (multi-instance learning) dataset in which bags contain exactly two digits. Hence, we trained an MIL network which has the same structure of the CNN network used in (Manhaeve et al., 2018) (with the exception for the bag-layer). We obtained an accuracy on the test set equals to 98.52% which is comparable with the accuracy ($\approx$ 98%) obtained in (Manhaeve et al., 2018). We would like to point out that in (Manhaeve et al., 2018) the summation function which labels a bag is part of the background knowledge, while for our approach the label function is assumed to be latent.

Multi-multi instance learning as Disordered Pooling

As discussed in Section 3.3, bag-layers can be used as disordered convolutional layers. The aim of the following experiment is to show this property on a simple scenario, and to compare results with a standard convolutional approach. We first constructed a dataset MNIST^{EXT} by placing the 28×28 pixels MNIST digit images on a black background of size $w \times w$ (we generated datasets for several values of w, i.e. $w \in \{105, 135, 165, 195\}$). MNIST^{EXT} images are labelled with the digit they contain. The goal is to classify digits regardless of their location within the images. Our purpose here is to show how the location-invariance of the bag-layer function can be exploited to obtain robust classification results for raw image data without centering or cropping preprocessing. As MNIST, the MNIST^{EXT} dataset, is split in training and test sets containing respectively 60,000 images and 10,000 images. Digits of training and test images are placed in the top and bottom half of the background, respectively. On the same datasets we trained a MMIL network and a standard convolutional network.

Concerning the MMIL network we constructed MMIL data as follows: each top-bag X represents an image in MNIST^{EXT}. For each we extracted *macro-patches* of size 15×15 and consecutive macro-patches are overlapped by 5 pixels. Each macro-patch represents a sub-bag $S_j \in X$. For each macro-patch we extracted *micro-patches* of size 5×5 and consecutive micro-patches are overlapped by 3 pixels. Each micro-patch represents an instance $x_{j,\ell} \in S_j$. Note that the choice of the micro-patches, macro-patches and w sizes is in general arbitrary. Here we chose those specific values for avoiding, as far as possibile, the zero padding for adapting the sizes and then waste of computational resources. Furthermore choosing w fixed for each experiment is crucial in order to compare the MMIL network with the convolutional neural network. Indeed while our approach can handle bags of different size the convolutional model requires that input images have fixed size. The structure of the MMIL network we used in this experiment consists of two stacked bag-layers (of size 100, and 200 respectively), with max aggregation functions followed by ReLU activation, a dense layer with 1,024 units with ReLU activation, a Dropout layer with probability 0.5, and a dense layer with Softmax activation with 10 units. The model was trained by minimizing the categorical cross-entropy loss. We ran 20 epochs of the Adam optimizer with learning rate 0.0001.

Concerning the convolutional network, we used a model composed of 4 convolutional blocks, a dense layer of 1024 units with ReLU activation, a Dropout layer with probability 0.5 and a Softmax Layer of 10 units. The first convolutional block contains a convolutional layer with a kernel of size 7×7 and 32 channels while the other blocks contain convolutional layers of size 5×5 and 64 channels. Each convolutional layer is followed by a ReLU activation, MaxPooling of size 2×2 and a Dropout layer with probability 0.5. The model was trained by minimizing the cat-

egorical cross-entropy loss. We ran 20 epochs of the Adam optimizer with learning rate 0.001.

In Figure 3.6 we report the accuracy of MMIL network and CNNs as a function of w. We note that the accuracy of the CNN decreases as w grows large, while accuracy the MMIL network remains stable. Those results confirm that MMIL network is able to learn location-invariant features.

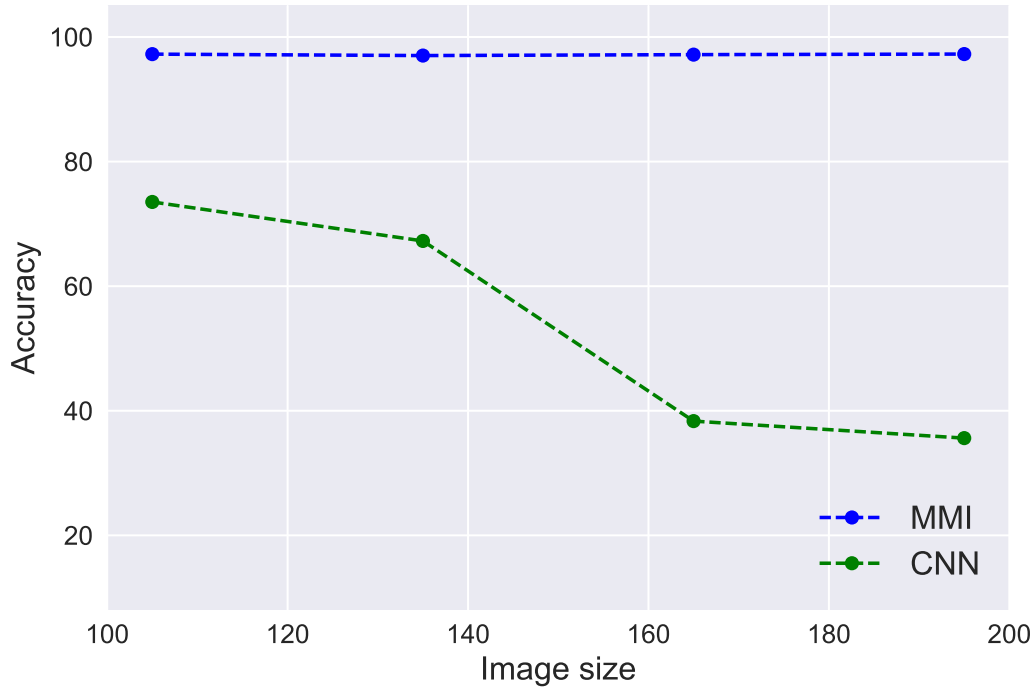


Figure 3.6: Accuracies of MMIL network (blue) and convolutional neural network (green) in function of window size w.

Chapter 4

Interpreting MMIL networks

> *As a further advantage, multi-multi instance learning enables a particular way of interpreting predictions and the decision function. We propose a framework, based on rule extractions, able to explain our model. We present empirical results on semi-synthetic data showing that such class of functions can be actually learned from data. We also present experiments on text classification.*[1]

Recently, the question of interpretability has become particularly prominent in the neural network context. Lapuschkin et al. (2016); Samek et al. (2016) explain predictions of a classifier f for each instance $\boldsymbol{x} \in \mathbb{R}^n$, by attributing scores to each entry of $\boldsymbol{x}$. A positive $R_i > 0$ or negative $R_i < 0$ score is then assigned to $\boldsymbol{x_i}$, depending whether $\boldsymbol{x_i}$ contributes for predicting the target or not. Ribeiro et al. (2016) also provided explanations for individual predictions as a solution to the "trusting a prediction" by approximating a machine learning model with an interpretable model. The authors assumed that instances are given in a representation which is understandable to humans, regardless of the actual features used by the model. For example for text classification an interpretable representation may be the binary vector indicating the presence or absence of a word. An "interpretable" model is defined as a model that can be readily presented to the user with visual or textual artefacts (linear models, decision trees, or falling rule lists), which locally approximates the original machine learning model.

Multi-multi instance learning enables a particular way of interpreting the models by reconstructing instance and sub-bag latent variables. This allows to explain the prediction for a particular data point, and to describe the structure of the decision function in terms of symbolic rules. Suppose we could recover the latent labels associated with instances or inner bags. These labels would provide useful additional

[1] Part of the content of this chapter has been submitted as "Learning and Interpreting Multi-Multi-Instance Learning Networks" in *Journal of Machine Learning Research*, 2018. `http://arxiv.org/abs/1810.11514`

information about the data since we could group instances (or inner bags) that share the same latent label and attach some semantics to these groups by inspection. For example, in the case of textual data, grouping words or sentences with the same latent label effectively discovers *topics* and the decision of a MMIL text document classifier can be interpreted in terms of the discovered topics. In practice, even if we cannot recover the true latent labels, we may still derive *pseudo-labels* from patterns of hidden units activations in the bag-layers. The major differences between all those methods and our interpretation framework, described in Section 4.1, is that with the latter we are able to provide a global interpretation for the whole MMIL network, as well as to explain individual example.

4.1 Interpreting networks of Bag-layers

Interpreting the predictions in the supervised learning setting amounts to provide a human understandable explanation of the prediction. Transparent techniques such as rules or trees retain much of the symbolic structure of the data and are well suited in this respect. On the contrary, predictions produced by methods based on numerical representations are often opaque, i.e., difficult to explain to humans. In particular, representations in neural networks are highly distributed, making it hard to disentangle a clear semantic interpretation of any specific hidden unit. Although many works exist that attempt to interpret neural networks, they mostly focus on specific application domains such as vision (Lapuschkin et al., 2016; Samek et al., 2016).

The MMIL settings offers some advantages in this respect. Indeed, if instance or sub-bag labels were observed, they would provide more information about bag-of-bags than mere predictions. Latent variables are indeed associated with each individual "part" of the top-bag, as opposite to the prediction which is associated with the whole. To clarify our vision, MIL approaches like mi-SVM and MI-SVM in (Andrews et al., 2003) are not equally interpretable: the former is *more* interpretable than the latter since it also provides individual instance labels rather than simply providing a prediction about the whole bag. These standard MIL approaches make two assumptions: first all labels are binary, second the relationship between the instance labels and the bag label is predefined to be the existential quantifier. In our case we relax these assumptions by allowing labels in a categorical alphabet and by allowing more complex mappings between bags of instance labels and sub-bag labels. Our approach may also provide a richer explanation due to the nested structure of the data as bags-of-bags. We follow the standard MIL approaches in that we also assume a deterministic mapping from component to bag labels, i.e., we assume the data can be modelled in the deterministic MMIL setting according to Definition 3.2.1.

The idea we propose in the following is based on four steps. First, we employ clustering at the level of instance and sub-bag representations to construct *pseudo-labels* as surrogates for hypothesized actual latent labels. Pseudo-labels obtained in this way are abstract symbols without any specific semantics. Hence, in the second step we provide semantic interpretations of the pseudo-labels for human inspection. Third, we apply a transparent learner to extract a human-readable representation of the mappings between pseudo-labels at the different levels of a bag-of-bags structure. Finally, we explain predictions for individual top-bag examples by exhibiting the relevant components and their pseudo-labels which determine the predicted top-bag label. We stress the fact that our interpreting framework provides a global interpretation for an MMIL model as well as local explanations for individual predictions.

We now describe each of these steps in detail. As before, for ease of exposition we assume in the following a two-level bag-of-bags structure. The method directly applies also to other nesting depths.

Clustering and pseudo-label construction. Given labeled top-bag data $\{(X^{(i)}, y^{(i)}),\ i = 1, \ldots, m\}$ and a trained MMIL network we consider the multi-sets of sub-bag and instance representations computed by the bag layers:

$$\mathcal{S}^S = \{\rho_j^{(i)} \mid i = 1, \ldots, m, j = 1, \ldots, n^{(i)}\}.$$

$$\mathcal{S}^I = \{\rho_{j,k}^{(i)} \mid i = 1, \ldots, m, j = 1, \ldots, n^{(i)}, k = 1, \ldots, l_j^{(i)}\}$$

where the $\rho_j^{(i)}$ and $\rho_{j,k}^{(i)}$ are the representations according to (3.2).

Given the number of clusters k^S and k^I we run a clustering procedure on $\mathcal{S}^S$ and on $\mathcal{S}^I$ (separately), obtaining clusters $\{\mathcal{C}_\ell^S, \ell = 1, \ldots k^S\}$ and $\{\mathcal{C}_\ell^I, \ell = 1, \ldots k^I\}$. We finally associate each sub-bag and each instance with the cluster index of their representation, and use them as pseudo-labels $\hat{y}_j^{(i)} \in \hat{\mathcal{Y}}^S := \{v_1, \ldots, v_{k^S}\}$ and $\hat{y}_{j,k}^{(i)} \in \hat{\mathcal{Y}}^I := \{u_1, \ldots, u_{k^I}\}$.

Interpreting pseudo-labels. Clusters can be directly inspected in the attempt to attach some meaning to pseudo-labels. For example in the case of textual data, a human could inspect word clusters, similarly to what has been suggested in the area of topic modelling (Blei et al., 2003; Griffiths and Steyvers, 2004).

To facilitate inspection, we propose an approach to characterize clusters in terms of their most characteristic elements. To this end, we define a ranking of the elements in each cluster according to a score function based on intra-cluster distances. Consider a sub-bag $S_j^{(i)}$ whose bag-layer representation $\rho_j^{(i)}$ belongs to cluster $\mathcal{C}_\ell^S$. We define the score

$$s(\rho_j^{(i)}) = \min_{p=1,\ldots,k^S, p\neq\ell} \|\rho_j^{(i)} - \mu_p\|, \tag{4.1}$$

where μ_p is the centroid of the pth cluster. Thus, a point $\rho_j^{(i)}$ obtains a high score when it is well separated from the means of all the clusters it does not belong to. The procedure for ranking instances is analogous. We use the cluster elements with maximal score to illustrate and interpret the semantic nature of a cluster. Note that this is different from the more common approach of interpreting clusters by way of their centroids.

In some cases the cluster elements may be equipped with some true, latent label. In such cases we can alternatively characterize pseudo-labels in terms of their correspondence with these actual labels. An example of this will be seen in Section 4.2 below.

Learning interpretable rules. We next describe how we construct interpretable functions that approximate the actual (potentially noisy) relationships between pseudo-labels in the MMIL network.

Let us denote a bag of pseudo-labels as $\{\hat{y}_l : c_l \mid l = 1, \ldots, |\hat{\mathcal{Y}}|\}$, where c_l is the multiplicity of label $\hat{y}_l$. An attribute-value representation of the bag can be immediately obtained in the form of a label frequency vector $(f_{c_1}, \ldots, f_{c_{|\hat{\mathcal{Y}}|}})$, where $f_{c_l} = c_l / \sum_{p=1}^{|\hat{\mathcal{Y}}|} c_p$ is the frequency of the label in the bag. Alternatively, we can also use a 0/1-valued label occurrence vector $(o_{c_1}, \ldots, o_{c_{|\hat{\mathcal{Y}}|}})$ with $o_{c_l} = \mathbb{1}\{c_l > 0\}$. Jointly with an output label y, this attribute-value representation provides one supervised example for a propositional learner such as a decision tree.

In the two level MMIL case, we learn in this way functions $\hat{g}, \hat{h}$ mapping multisets of instance pseudo-labels to sub-bag pseudo-labels, and multisets of sub-bag pseudo-labels to top-bag labels, respectively (cf. Definition 3.2.1). In the second case, our target labels are the predicted labels of the original MMIL network, not the actual labels of the training examples. Thus, the objective is to construct rules that best explain the MMIL model, not the rules that provide the highest accuracy themselves.

The instance-level clustering $\{\mathcal{C}_\ell^I, \ell = 1, \ldots k^I\}$ defines a labeling function $\hat{f} : \mathcal{X} \mapsto \hat{\mathcal{Y}}^I$ by associating any (test) instance with the index of its nearest centroid. Taken together, the three functions $\hat{f}, \hat{g}, \hat{h}$ provide a complete classification model for a top-bag based on the input features of its instances. We refer to the accuracy of this model with regard to the predictions of the original MMIL model as its *fidelity*.

We use fidelity on a validation set as the criterion to select the cardinalities for $\hat{\mathcal{Y}}^S$ and $\hat{\mathcal{Y}}^I$ by performing a grid search over k^S, k^I value combinations.

Explaining individual classifications. The classification provided by $\hat{f}, \hat{g}, \hat{h}$ for an input top-bag X will often rely only on small subsets of sub-bags and instances contained in X (cf. the classic multi-instance setting, where a positive classification can rely only on a single positive instance). We can therefore explain classifications for

individual examples by exhibiting the critical substructures of X that support the prediction. The details of this step are typically quite domain specific, and we will illustrate one version of it in the experimental section.

4.2 Experiments

1. we constructed a multi-multi instance semi-synthetic dataset from MNIST, in which digits were organized in bags-of-bags of varying cardinality. This setup follows the example 3.1.1 shown in Section 3.1. The aim of this experiment is to show the ability of the network to learn functions that satisfy the assumptions of Theorem 3.2.2 in Section 3.2. Furthermore we interpreted the network by using the approach described in Section 4.1;

2. we decomposed a sentiment-analysis text dataset into MMIL data and MIL data. The goal is to show the differences between the interpretation of the two models.

MNIST

The data is structured exactly as in Example 3.1.1. We formed a balanced training set and validation set of 4,000 and 1,000 top-bags respectively, using MNIST digits. Both sub-bag and top-bag cardinalities were uniformly sampled in $[2, 6]$. Instances were sampled with replacement from the MNIST training set (60,000 digits). A test set of 5,000 top-bags was similarly constructed but instances were sampled from the MNIST test set (10,000 digits). Details on the network architecture and the training procedure are reported in Appendix B in Table B.1. We stress the fact that instance and sub-bag labels were not used for training. The learned network achieved an accuracy on the test set of 98.42%, confirming that the network is able to recover the latent logic function that was used in the data generation process with a high accuracy.

We show next how the general approach of Section 4.1 for constructing interpretable rules recovers the latent labels and logical rules used in the data generating process. Note that this experiments is similar to the one reported in Section 3.4, but with simpler rules, in order to provide the reader easier results to follow for what concerns the interpretation. Pseudo-labels and rules are learnt with the procedure described in Section 4.1. Clustering was performed with K-Means, and decision trees were used as propositional learners. As described in Section 4.1, we determined the number of instance and sub-bag pseudo-labels by maximizing the fidelity of the interpretable model on the validation data via grid search, and in this way found $k^I = 6$, and $k^S = 2$, respectively. Full results of the grid search are depicted as a heat-map in Appendix B (Figure B.1).

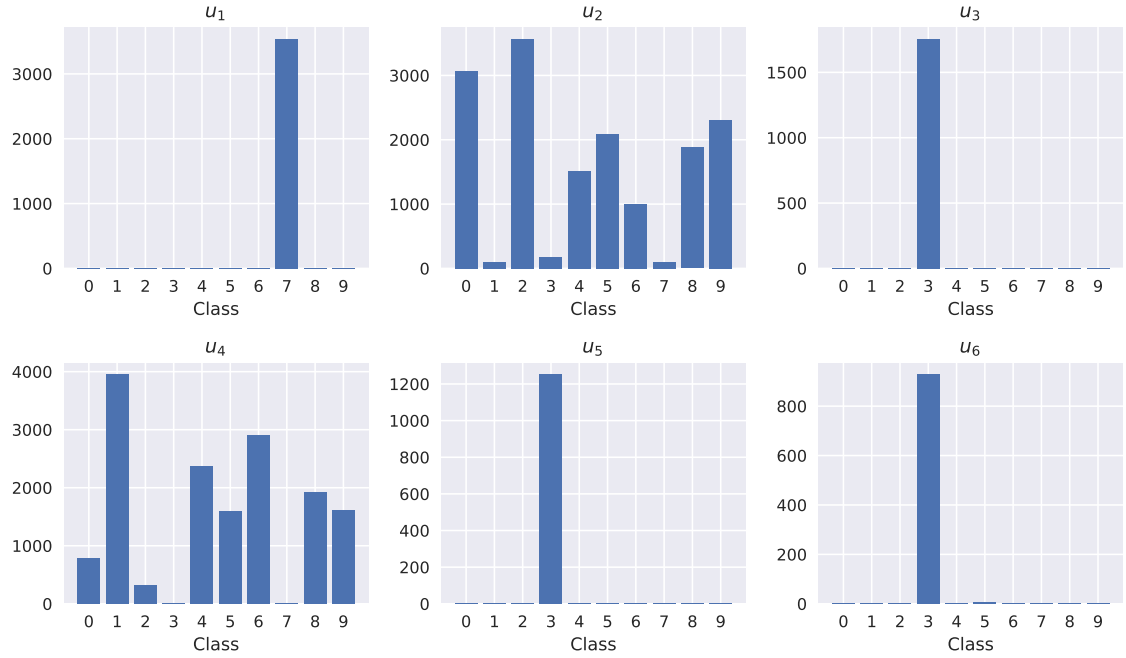


Figure 4.1: Correspondence between pseudo-labels u_i and actual digit class labels

We can interpret the instance pseudo labels by analysing their correspondence with the actual digit labels. It is then immediate to recognize that pseudo-label u_1 corresponds to the digit 7, u_3, u_5, and u_6 all correspond to digit 3, and u_2 and u_4 correspond to digits other than 7 and 3. All correspondences are shown by histograms in Figure 4.1. From a decision tree trained to predict pseudo labels of sub-bags S from instance pseudo label occurrence vectors $(o_{u_1}, \ldots, o_{u_6})$ we then extract the following rules defining the function $\hat{g}$:

$$\begin{array}{ll} 1 & \hat{g} = v_1 \leftarrow o_{u_1}{=}1, o_{u_3}{=}0, o_{u_5}{=}0, o_{u_6}{=}0. \\ 2 & \hat{g} = v_2 \leftarrow o_{u_1}{=}0. \\ 3 & \hat{g} = v_2 \leftarrow o_{u_3}{=}1. \\ 4 & \hat{g} = v_2 \leftarrow o_{u_5}{=}1. \\ 5 & \hat{g} = v_2 \leftarrow o_{u_6}{=}1. \end{array} \tag{4.2}$$

Based on the already established interpretation of the instance pseudo-labels u_1, u_3, u_5, u_6 we thus find that the sub-bag pseudo-label v_1 gets attached to the sub-bags that contain a seven and not a thre, i.e., it corresponds to the latent 'positive' label for sub-bags.

Similarly, we extracted the following rule that predict the class label of a top-bag X based on the occurrence vector (o_{v_1}, o_{v_2}) of sub-bag pseudo-labels.

$$\begin{array}{ll} 1 & \hat{h} = \text{positive} \leftarrow o_{v_1}{=}1 \\ 2 & \hat{h} = \text{negative} \leftarrow o_{v_1}{=}0 \end{array} \tag{4.3}$$

Hence, in this example, the true rules behind the data generation process were perfectly recovered. Note that perfect recovery does not necessarily imply perfect accuracy of the resulting rule-based classification model $\hat{f}, \hat{g}, \hat{h}$, since the initial instance pseudo labels $\hat{f}(x)$ do not correspond with the digit labels with 100% accuracy. Nonetheless, in this experiment the classification accuracy of the interpretable rule model on the test set was 98.18%, only 0.24% less than the accuracy of the original model, which it approximated with a fidelity of 99.16%.

IMDB

In this section we apply our approach to a real-world dataset for sentiment analysis. The main objective of this experiment is to demonstrate the feasibility of our model interpretation framework on real-world data, and to explore the trade-offs between an MMIL and MIL approach. We use the IMDB (Maas et al., 2011) dataset, which is a standard benchmark movie review dataset for binary sentiment classification. We remark that this IMDB dataset differs from the IMDB graph datasets described in Section 5.1. IMDB consists of 25,000 training reviews, 25,000 test reviews and 50,000 unlabelled reviews. Positive and negative labels are balanced within the training and test sets. Text data exhibits a natural bags-of-bags structure by viewing a text as a bag of sentences, and each sentence as a bag of words. Moreover, for the IMDB data it is reasonable to associate with each sentence a (latent) sentiment label (positive/negative, or maybe something more nuanced), and to assume that the overall sentiment of the review is a (noisy) function of the sentiments of its sentences. Similarly, sentence sentiments can be explained by latent sentiment labels of the words it contains.

A MMIL dataset was constructed from the reviews, where then each review (top-bag) is a bag of sentences. However, instead of modeling each sentence (sub-bag) as a bag of words, we represented sentences as bags of trigrams in order to take into account possible negations, e.g. "not very good", "not so bad". Figure 4.2 depicts an example of the decomposition of a two sentence review X into MMIL data. Each word is represented with Glove word vectors (Pennington et al., 2014) of size 100, trained on the dataset. The concatenation of its three Glove word vectors then is the feature vector we use to represent a trigram. We here use Glove word vectors for a more pertinent comparison of our model with the state-of-the-art (Miyato et al., 2017). Nothing prevents us from using a one-hot representation even for this scenario. In order to compare MMIL against multi-instance (MIL) we also constructed a multi-instance dataset in which a review is simply represented as a bag of trigrams.

We trained two neural networks for MMIL and MIL data respectively, which have the following structure:

- **MMIL network**: a Conv1D layer with 300 filters, ReLU activations and ker-

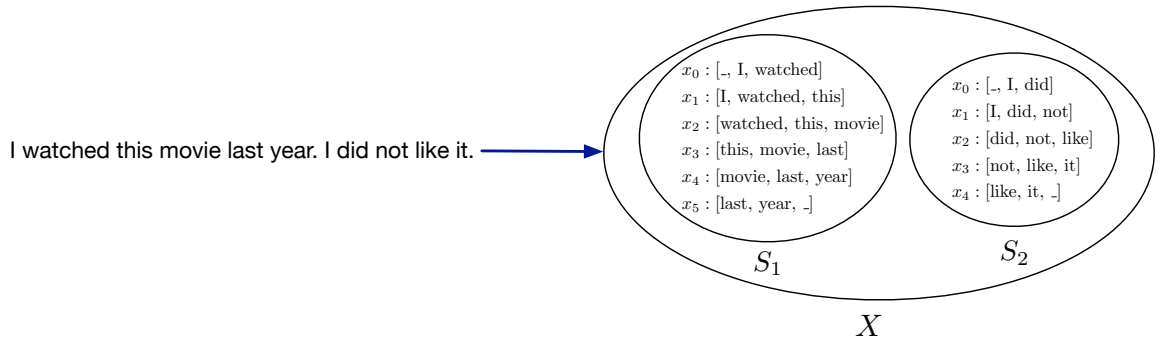


Figure 4.2: A review transformed into MMIL data. The word "_" represents the padding.

nel size of 100, two stacked bag-layers (with ReLU activations) with 500 units each (250 max-aggregation, 250 mean-aggregation) and an output layer with sigmoid activation;

- **MIL network**: a Conv1D layer with 300 filters, ReLU activations and kernel size of 100, one bag-layers (with ReLU activations) with 500 units (250 max-aggregation, 250 mean-aggregation) and an output layer with sigmoid activation;

The models were trained by minimizing the binary cross-entropy loss. We ran 20 epochs of the Adam optimizer with learning rate 0.001, on mini-batches of size 128. We used also virtual adversarial training (Miyato et al., 2017) for regularizing the network and exploiting the unlabelled reviews during the training phase. Although our model does not outperform the state-of-the-art (94.04%, Miyato et al. (2017)), we obtained a final accuracy of 92.26% for the MMIL network and 91.73% for the MIL network. Those results show that the MMIL representation here leads to a slightly higher accuracy than the MIL representation.

When accuracy is not the only concern, our models have the advantage that we can distill them into interpretable sets of rules following our general strategy. As in Section 4.2, we learnt pseudo-labels and rules for both the MMIL model and MIL model. Using 2,500 reviews as a validation set, we obtained in the MMIL case 4 and 5 pseudo-labels for sub-bags and instances, respectively, and in the MIL case 6 pseudo labels for instances. Full grid search results on the validation set are reported in Appendix C (Figure C.1).

We now focus on the interpretation of pseudo-labels, following the approach described in Section 4.1 and using intra-cluster distances (Eq. 4.1) to compute scores. In Tables 4.1 and 4.2 we report the top-scoring sentences and trigrams, respectively, sorted by decreasing score. It can be seen that sentences labeled by v_1 or v_4 express negative judgments, sentences labeled by v_2 are either descriptive, neutral or ambiguous, while sentences labeled by v_3 express a positive judgment. Similarly, we

see that trigrams labeled by u_1 express positive judgments while trigrams labeled by u_2 or u_4 express negative judgments. Columns printed in grey correspond to pseudo-labels that do not actually appear in the extracted rules (see below), and they do not generally correspond to a clearly identifiable sentiment. Percentages in parenthesis in the headers of these tables refer to fraction of sentences or trigrams associated with each pseudo-label (the total number of sentences in the dataset is approximately 250 thousand while the total number of trigrams is approximately 4.5 million). A similar analysis was performed in the MIL setting (results in Table 4.3).

MMIL rules. Using a decision tree learner taking pseudo-label frequency vectors $(f_{u_1}, \ldots, f_{u_5})$ as inputs, we obtained the rules reported in Table 4.4 for mapping a bag S of instance pseudo-labels to a sub-bag pseudo-label. Even though these rules are somewhat more difficult to parse than the ones we obtained in Section 4.2, they still express relatively simple relationships between the triplet and sentence pseudo-labels. Especially the single sentence pseudo-label v_3 that corresponds to a clearly positive sentiment has a very succinct explanation given by the rule of line 6. Rules related to sentence pseudo-label v_2 are printed in grey. Since v_2 is not used by any of the rules shown in Table 4.5 that map sub-bag (sentence) pseudo-labels to the top-bag (review) class labels, the rules for v_2 will never be required to explain a particular classification.

MIL rules. For the MIL model, rules map a bag S, described by its instance pseudo-label frequency vector $(f_{u_1}, \ldots, f_{u_5})$, to the bag class label. They are reported in Table 4.6. Note that only two out of the six instance pseudo-labels are actually used in these rules.

By classifying IMDB using the rules and pseudo-labels, we achieved an accuracy of 87.49% on the test set for the MMIL case and 86.37% for the MIL case. Fidelities for MMIL and MIL cases were 90.40% and 88.10%, respectively. We thus see that the somewhat higher complexity of the rule-based explanation of the MMIL model also corresponds to a somewhat higher preservation of accuracy. As we demonstrate by the following example, the multi-level explanations derived from MMIL models can also lead to more transparent explanations for individual predictions.

An example of prediction explanation. As an example we consider a positive test-set review for the movie Bloody Birthday, which was classified correctly by the MMIL rules and incorrectly by the MIL rules. Its full text is reported in Table 4.7. Classification in the MMIL setting was positive due to applicability of rule 2 in Table 4.5. This rule only is based on sentences with pseudo-labels v_1 and v_3, and therefore sentences assigned any other pseudo-labels do not actively contribute to this classification. These irrelevant sentences are dimmed in the printed text (Table 4.7,

v_1 (11.37%)	v_2 (41.32%)	v_3 (15.80%)	v_4 (31.51%)
overrated poorly written badly acted	I highly recommend you to NOT waste your time on this movie as I have	I loved this movie and I give it an 8/ 10	It's not a total waste
It is badly written badly directed badly scored badly filmed	This movie is poorly done but that is what makes it great	Overall I give this movie an 8/ 10	horrible god awful
This movie was poorly acted poorly filmed poorly written and overall horribly executed	Although most reviews say that it isn't that bad i think that if you are a true disney fan you shouldn't waste your time with...	final rating for These Girls is an 8/ 10	Awful awful awful
Poorly acted poorly written and poorly directed	I've always liked Madsen and his character was a bit predictable but this movie was definitely a waste of time both to watch and make...	overall because of all these factors this film deserves an 8/ 10 and stands as my favourite of all the batman films	junk forget it don't waste your time etc etc
This was poorly written poorly acted and just overall boring	If you want me to be sincere The Slumber Party Massacre Part 1 is the best one and all the others are a waste of...	for me Cold Mountain is an 8/ 10	Just plain god awful

Table 4.1: Interpreting sentence (sub-bag) pseudo-labels in the MMIL setting.

u_1 (5.73%)	u_2 (8.68%)	u_3 (28.86%)	u_4 (2.82%)	u_5 (53.91%)
_ 8/ 10	trash 2 out	had read online	it's pretty poorly	give this a
an 8/ 10	to 2 out	had read user	save this poorly	like this a
for 8/ 10	_ 2 out	on IMDb reading	for this poorly	film is 7
HBK 8/ 10	a 2 out	I've read innumerable	just so poorly	it an 11
Score 8/ 10	3/5 2 out	who read IMDb	is so poorly	the movie an
to 8/ 10	2002 2 out	to read IMDb	were so poorly	this movie an
verdict 8/ 10	garbage 2 out	had read the	was so poorly	40 somethings an
Obscura 8/ 10	Cavern 2 out	I've read the	movie amazingly poorly	of 5 8
Rating 8/ 10	Overall 2 out	movie read the	written poorly directed	gave it a
it 8/ 10	rating 2 out	Having read the	was poorly directed	give it a
fans 8/ 10	film 2 out	to read the	is very poorly	rating it a
Hero 8/ 10	it 2 out	I read the	It's very poorly	rated it a
except 8/ 10	score 2 out	film reviews and	was very poorly	scored it a
Tracks 8/ 10	Grade 2 out	will read scathing	a very poorly	giving it a
vote 8/ 10	Just 2 out	_ After reading	very very poorly	voting it a
as 8/ 10	as 2 out	about 3 months	Poorly acted poorly	are reasons 1
strong 8/ 10	and 2 out	didn't read the	are just poorly	it a 8
rating 8/ 10	rated 2 out	even read the	shown how poorly	vote a 8
example 8/ 10	Rating 2 out	have read the	of how poorly	a Vol 1
... 8/ 10	conclusion 2 out	the other posted	watching this awful	this story an

Table 4.2: Interpreting trigram (instance) pseudo-labels in the MMIL setting.

u_1 (13.53%)	u_2 (41.53%)	u_3 (3.03%)	u_4 (5.47%)	u_5 (31.58%)	u_6 (4.85%)
production costs _	give it a	only 4/10 _	is time well-spent	... 4/10 ...	_ Recommended _
all costs _	gave it a	score 4/10 _	two weeks hairdressing	.. 1/10 for	Highly Recommended _
its costs _	rated it a	a 4/10 _	2 hours _	rate this a	Well Recommended _
ALL costs _	rating it a	_ 4/10 _	two hours _	gave this a	_ 7/10 _
possible costs _	scored it a	average 4/10 _	finest hours _	give this a	13 7/10 _
some costs _	giving it a	vote 4/10 _	off hours _	rated this a	rate 7/10 _
cut costs _	voting it a	Rating 4/10 _	few hours _	_ Not really	.. 7/10 _
rate this a	gave this a	.. 4/10 _	slow hours _	4/10 Not really	this 7/10 _
gave this a	give this a	is 4/10 _	three hours _	a 4/10 or	Score 7/10 _
rating this a	rate this a	this 4/10 _	final hours _	of 4/10 saying	solid 7/10 _
give this a	giving this a	of 4/10 _	early hours _	rate it a	a 7/10 _
and this an	gives this a	movie 4/10 _	six hours _	give it a	rating 7/10 _
give this an	like this a	verdict 4/10 _	48 hours _	gave it a	to 7/10 _
given this an	film merits a	gave 4/10 _	4 hours _	given it a	viewing 7/10 _
gave this an	Stupid Stupid Stupid	13 4/10 _	6 hours _	giving it a	it 7/10 _
rating this an	_ Stupid Stupid	disappointment 4/10 _	five hours _	scored it a	score 7/10 _
rate this an	award it a	at 4/10 _	nocturnal hours _	award it a	movie 7/10 _
all costs ...	given it a	rating 4/10 _	17 hours _	Cheesiness 0/10 Crappiness	is 7/10 _
all costs ..	makes it a	... 4/10 _	for hours _	without it a	drama 7/10 _
_ Avoid _	Give it a	rate 4/10 _	wasted hours _	deserves 4/10 from	Recommended 7/10 _

Table 4.3: Interpreting trigram (instance) pseudo-labels in the MIL setting.

1	$\hat{g} = v_1 \leftarrow f_{u_1} \leq 6.03, f_{u_2} > 10.04, f_{u_4} \in (2.77, 12.59]$.
2	$\hat{g} = v_1 \leftarrow f_{u_1} \leq 16.90, f_{u_4} > 12.59$.
3	$\hat{g} = v_2 \leftarrow f_{u_1} \leq 8.43, f_{u_2} \leq 8.88, f_{u_4} \leq 2.77$.
4	$\hat{g} = v_2 \leftarrow f_{u_1} > 3.20, f_{u_2} \in (8.88, 20.39], f_{u_4} \leq 2.77$.
5	$\hat{g} = v_2 \leftarrow f_{u_1} > 6.03, f_{u_2} \leq 6.03, f_{u_4} \in (2.77, 12.59]$.
6	$\hat{g} = v_3 \leftarrow f_{u_1} > 8.43, f_{u_2} \leq 8.88, f_{u_4} \leq 2.77$.
7	$\hat{g} = v_4 \leftarrow f_{u_1} \leq 3.20, f_{u_2} > 8.88, f_{u_4} \leq 2.77$.
8	$\hat{g} = v_4 \leftarrow f_{u_1} > 3.20, f_{u_2} > 20.39, f_{u_4} \leq 2.77$.
9	$\hat{g} = v_4 \leftarrow f_{u_1} \leq 6.03, f_{u_2} \leq 10.04, f_{u_4} \in (2.77, 12.59]$.
10	$\hat{g} = v_4 \leftarrow f_{u_1} > 6.03, f_{u_2} > 6.03, f_{u_4} \in (2.77, 12.59]$.
11	$\hat{g} = v_4 \leftarrow f_{u_1} > 16.90, f_{u_4} > 12.59$.

Table 4.4: Rules extracted from the MMIL network for mapping instance pseudo-labels into a sub-bag pseudo-label. Numbers express percentages, i.e. the literal $f_{u_1} \leq 6.03$ means that the frequency of instance pseudo-label u_1 is less than 6.03% in the sub-bag. For readability, rules are written as definite clauses with a Prolog-like syntax where $\leftarrow$ is the implication and conjuncted literals are joined by a comma.

1	$\hat{h} = \text{positive} \leftarrow f_{v_1} \leq 4.04, f_{v_3} \leq 12.63, f_{v_4} \leq 39.17$.
2	$\hat{h} = \text{positive} \leftarrow f_{v_1} \leq 12.97, f_{v_3} > 12.63$.
3	$\hat{h} = \text{positive} \leftarrow f_{v_1} > 12.97, f_{v_3} > 25.66$.
4	$\hat{h} = \text{negative} \leftarrow f_{v_1} \leq 4.04, f_{v_3} \leq 12.63, f_{v_4} > 39.17$.
5	$\hat{h} = \text{negative} \leftarrow f_{v_1} > 4.04, f_{v_3} \leq 12.63$.
6	$\hat{h} = \text{negative} \leftarrow f_{v_1} > 12.97, f_{v_3} \in (12.63, 25.66]$.

Table 4.5: Rules mapping sentence pseudo-labels into review sentiment labels. See the caption of Table 4.4 for details on the syntax.

1	$\hat{h} = \text{positive} \leftarrow f_{u_3} \leq 1.11, f_{u_6} \leq 3.42$.
2	$\hat{h} = \text{positive} \leftarrow f_{u_3} \leq 2.21, f_{u_6} > 3.42$.
3	$\hat{h} = \text{positive} \leftarrow f_{u_3} \in (2.21, 5.81], f_{u_6} > 6.30$.
4	$\hat{h} = \text{negative} \leftarrow f_{u_3} \in (1.11, 2.21], f_{u_6} \leq 3.42$.
5	$\hat{h} = \text{negative} \leftarrow f_{u_3} > 2.21, f_{u_6} \leq 6.30$.
6	$\hat{h} = \text{negative} \leftarrow f_{u_3} > 5.81, f_{u_6} > 6.30$.

Table 4.6: Classification rules for the MIL model. See the caption of Table 4.4 for details on the syntax.

top part). A first, high-level explanation of the prediction is thus obtained by simply using the sentences that are active for the classification as a short summary of the most pertinent parts of the review.

This sentence-level explanation can be refined by also explaining the pseudo-labels for the individual sentences. For example, sentence "Bloody Birthday a . . . " was assigned pseudo-label v_1 using rule 2 of Table 4.4. This rule is based on frequencies of the trigram pseudo-labels u_1 and u_4. Occurrences of trigrams with these labels are highlighted in boldface and superscripted with the trigram pseudo-label index in the text, thus exhibiting the sub-structures in the sentence that are pertinent for the classification. Similarly, the other three relevant sentences were all assigned label v_3 because of rule 6 in Table 4.4, which is based on the pseudo-labels u_1, u_2, u_4. These formal, logical explanations for the classifications are complemented by the semantic insight into the pseudo-labels provided by Tables 4.2 and 4.1.

The review was classified as negative in the MIL setting. The applicable rule here was rule 5 in Table 4.6 which involved triplet pseudo-labels u_3, and u_6. The relevant triplets are highlighted in boldface in the lower part of Table 4.7.

A second example of prediction explanation is reported in Appendix C.

4.3 Empirical optimization analysis

Large (and hard to interpret) bag-layers are necessary to allow gradient descent to find a good solution to the optimization problem. For example, in Section 4.2, the multi-multi instance learning network, used in the MNIST "logic scenario" experiment, had two bag-layers with 100 units each. For that specific problem the minimum number of units for the first bag-layer is 2: 1 unit for recognizing a 7 and 1 unit for recognizing a 3. On the other hand the minimum number of units for the second bag-layer is 1, activated, for example, if it exists a sub-bag with at least a 7 and no 3. Unfortunately if we choose the minimum number of units (or even a "too small number"), we empirically noticed that the gradient descent will find a poor solution for the optimization problem. Contrarily if we train a model for which the number of units is oversized, and then we use some compression techniques, e.g. (Buciluǎ et al., 2006), we are able to obtain a model in which the number of the units for the bag-layers tends to the minimum required, and with low loss in terms of accuracy.

Although at the moment we are unable to formally prove what previously claimed, we will provide empirical evidences for the *IMDB* experiment (see Section 4.2). For this purpose we propose a two-steps strategy: first, we train a MMIL network, M, with a large number of nodes per layer, on the IMDB dataset (decomposed as MMIL dataset), and second, we train another MMIL network, M_c to mimic M. M_c has the same structure as M, but less nodes per layer than M. We will refer to the second step as the *compression* step. The compression step is related to (Buciluǎ et al., 2006)

Story about three eclipse (maybe even Indigo, ha) children beginning their love for murder. Oh, and the people who are "hot" on their trail.
[v_1] Bloody **Birthday, a pretty mediocre title**[4] for the **film, was a nice lil**[1] surprise. I was in no way expecting a film that dealt with blood-thirsty psychopath kids.
[v_3] And I may say it's also **one of the best flicks**[1] I've seen with kids as the villains. By the end of the movie I seriously wanted these kids to die in horrible fashion.
[v_3] **It's a really solid 80s**[1] horror flick, but how these kids are getting away with all this mayhem and murder is just something that **you can't not**[2] think about. Even the slightest bit of investigation would easily uncover these lil sh!ts as the murderers. But there seems to be only a couple police in town, well by the end, only one, and he seemed like a dimwit, so I suppose they could have gotten away with it. Haha, yeah, and I'm a Chinese jet-pilot.
Nevertheless, this movie delivered some evilass kids who were more than entertaining, a lot of premarital sex and a decent amount of boobage. No kiddin! If you're put off by the less than stellar title, dash it from your mind and give this flick a shot. [v_3] It's **a very recommendable and underrated 80s**[1] horror flick.

Story about three eclipse (maybe even Indigo, ha) children beginning their love for murder. Oh, and the people who are "hot" on their trail.
Bloody **Birthday, a pretty mediocre title**[3] for the film, was a nice lil surprise. I was in no way expecting a film that dealt with blood-thirsty psychopath kids. And I may say it's also **one of the best flicks**[6] I've seen with kids as the villains. By the end of the movie I seriously wanted these kids **to die in horrible fashion**[3].
It's a really solid[6] 80s horror flick, but how these kids are getting away with all this mayhem and murder is just something that you can't not think about. Even the slightest bit of investigation would easily uncover these lil sh!ts as the murderers. But there seems to be only a couple police in town, well by the end, only one, and he seemed like a dimwit, so I suppose they could have gotten away with it. Haha, yeah, and I'm a Chinese jet-pilot.
Nevertheless, this movie delivered some evilass kids who were more than entertaining, a lot of premarital sex and a decent amount of boobage. No kiddin! If you're put off by **the less than**[6] stellar title, dash it from your mind and give this flick a shot. It's **a very recommendable and underrated 80s**[6] horror flick.

Table 4.7: A sample positive review. Top: MMIL labeling. Bottom: MIL labeling.

with the difference that we preserve the overall structure of the network rather than train a shallow network to mimic a deep network. For further information about compression, the reader is referred to (Buciluǎ et al., 2006). Given a top-bag X we denote with $M(X^i)$, $M_c(X^i)$ the output of M and M_c, respectively. We trained $M_c(X^i)$ in order to minimize the mean squared error loss

$$\mathcal{L}(M(X), M_c(X)) = \sum_X \|M_c(X^i W_c, b_c) - M(X)\|^2, \tag{4.4}$$

where W_c and b_c represent the weights and the biases of M_c, respectively. Note that the parameters of M are kept frozen during the training of M_c.

The experimental setup for *IMDB* remained exactly the same as that described in the respective section, with the exceptions of the number of units. Starting from the trained MMIL network, M, (with 92.26% of accuracy) we trained 2 different MMIL compressed models for several different number of nodes:

- A compressed MMIL model, M_{c_1}, using both supervised and unsupervised data;
- A compressed MMIL model, M_{c_2}, using only supervised data.

Furthermore we trained also a new MMIL model, M_n, (on IMDB, decomposed as MMIL dataset) for several different number of nodes using only supervised data.

M_{c_1} and M_{c_2} were trained by minimizing the mean squared error loss (see Equation 4.4) on the output of M. Contrarily, M_n was trained by minimizing the binary cross entropy on the true labels. For all the models we ran 20 epochs of the Adam optimizer for 10 times with learning rate 0.001, clipping the norm to 1, on mini-batches of size 200.

Results are depicted in Figure 4.3. For all the models we chose several values for the filters of the 1D Convolution layer, the units of the first bag-layer, and the units of the second bag-layer. We draw the following conclusions:

1. M_{c_1} outperforms M_n (on average);
2. M_{c_2} outperforms M_{c_1} (on average), i.e. exploiting unsupersived data for mimicking M seemed to help the optimizer to reach a better solution.

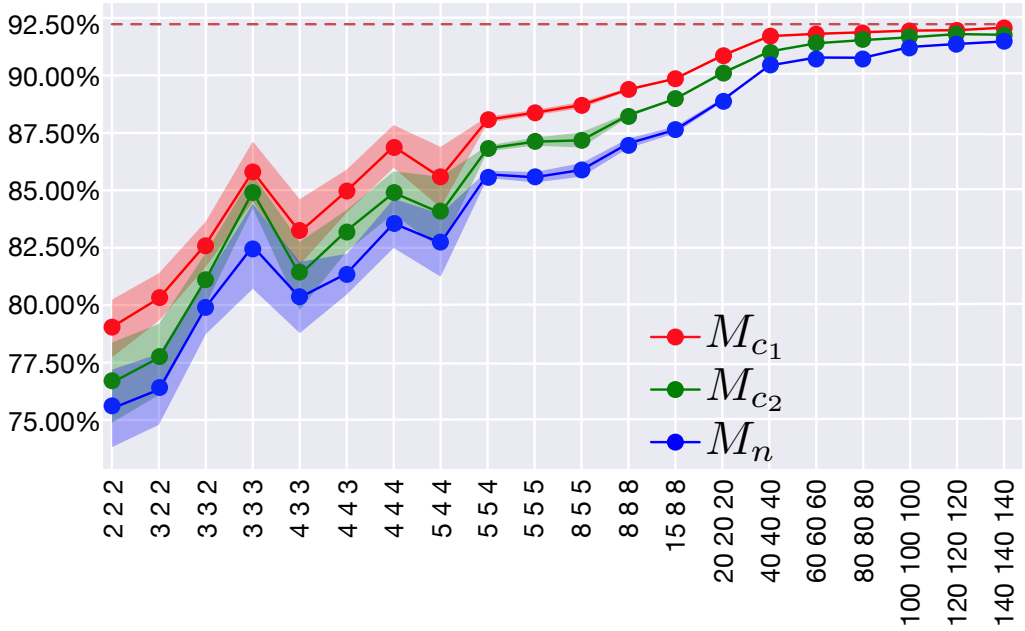


Figure 4.3: On x-axis is reported the size of the model (filters of Conv1D layer, units of first bag-layer, units of second bag-layer). On y-axis is reported the accuracy on the test set. The dashed line represents the accuracy of M.

Chapter 5

MMIL for graph learning

The multi-multi instance learning framework can be also useful on supervised learning over graphs. We present in this Chapter experiments on citation graphs and social graph data, showing that our model obtains competitive results with respect to other approaches such as convolutional networks on graphs.[1]

The convolutional approach has been also recently employed for learning with graph data. The idea is to reinterpret the convolution operator as a message passing algorithm on a graph where each node is a signal sample (e.g., a pixel) and edges connect a sample to all samples covered by the filter when centered around its position (including a self-loop). The major difference between graphs and signals is that no obvious ordering can be defined on neighbors. This message passing strategy over graphs was originally proposed in (Gori et al., 2005; Scarselli et al., 2009) and reused with variants in several later works. Kipf and Welling (2017) for example, propose to address the ordering issue by sharing the same weights for each neighbor (keeping them distinct from the self-loop weight). They show that message-passing is closely related to the 1-dimensional Weisfeiler-Lehman (WL) method for isomorphism testing (one convolutional layer corresponding to one iteration of the WL-test) and can be also motivated in terms of spectral convolutions on graphs. On a side note, similar message-passing strategies were used before in the context of graph kernels (Shervashidze et al., 2011; Neumann et al., 2012). Niepert et al. (2016) proposed ordering via a "normalization" procedure that extends the classic canonicalization problem in graph isomorphism. Hamilton et al. (2017) propose an extension of the approach in (Kipf and Welling, 2017) where representations of the neighbors are aggregated by a general differentiable function that can be as simple as an average or as complex as a recurrent neural network. Additional related works

[1] Part of the content of this chapter has been submitted as "Learning and Interpreting Multi-Multi-Instance Learning Networks" in *Journal of Machine Learning Research*, 2018. `http://arxiv.org/abs/1810.11514`

include (Duvenaud et al., 2015), where CNNs are applied to molecular fingerprint vectors, and (Atwood and Towsley, 2016) where a diffusion process across general graph structures generalizes the CNN strategy of scanning a regular grid of pixels.

The MMIL perspective can also be used to derive algorithms suitable for supervised learning over graphs, i.e., tasks such as graph classification, node classification, and edge prediction. In all these cases, one first need to construct a representation for the object of interest (a whole graph, a node, a pair of nodes) and then apply a classifier. A suitable representation can be obtained in our framework by first forming a bag-of-bags associated with the object of interest (a graph, a node, or an edge) and then feeding it to a network with bag-layers. In order to construct bags-of-bags, we follow the classic R-decomposition strategy introduced by Haussler (1999). In the present context, it simply requires us to introduce a relation $R(A, a)$ which holds true if a is a "part" of A and to form $R^{-1}(A) = \{a : R(A, a)\}$, the bag of all parts of A. Parts can in turn be decomposed in a similar fashion, yielding bags-of-bags. In the following, we focus on undirected graphs $G = (V, E)$ where V is the set of nodes and $E = \{\{u, v\} : u, v \in V\}$ is the set of edges. We also assume that a labelling function $x : V \mapsto \mathcal{X}$ attaches attributes to vertices. Variants with directed graphs or labeled edges are straightforward and omitted here in the interest of brevity.

Graph classification A simple solution is to define the part-of relation $R(G, g)$ between graphs to hold true iff g is a subgraph of G and to introduce a second part-of relation $S(g, v)$ that holds true iff v is a node in g. The bag-of-bags associated with G is then constructed as $X = \{\mu(x, S^{-1}(g)) : g \in R^{-1}(G)\}$ where $\mu(f, A)$ maps all elements of A through function f. In general, considering all subgraphs is not practical but suitable feasible choices for R can be derived borrowing approaches already introduced in the graph kernel literature, for example decomposing G into cycles and trees (Horváh et al., 2004), or into neighbors or neighbor pairs (Costa and De Grave, 2010) (some of these choices may require three levels of bag nesting, e.g., for grouping cycles and trees separately).

Node classification In some domains, the node labelling function itself is bag-valued. For example in a citation network, $x(v)$ could be the bag of words in the abstract of the paper associated with node v. A bag-of-bags in this case may be formed by considering a paper v together all papers in its neighborhood $N(v)$ (i.e., its cites and citations): $X(v) = \{x(u), u \in \{v\} \cup N(v)\}$. A slightly more rich description with three layers of nesting could be used to set apart a node and its neighborhood: $X(v) = \{\{x(v)\}, \{x(u), u \in N(v)\}\}$.

5.1 Experiments

We tested our model on two different graph tasks:

1. a node classification task in which we focused on real citation network datasets where data can be naturally decomposed into bags-of-bags (MMIL data) or bags (MIL data). The goal is to understand whether MMIL and MIL decompositions are reasonable representations for citation networks and whether MMIL representation is more suitable than MIL representation. Finally we compared our approach with the state-of-art architectures, and we interpreted our results;

2. a graph classification task in which we tested our model on real social network graphs, where the data can be easily decomposed into bags-of-bags. We compared our approach against the state-of-art architectures.

Citation Datasets

In the following experiments, we apply MMIL to graph learning according to the general strategy described above. We also present a (generalized) MIL approach to graph learning in which latent instance labels need not be binary, and need not be related to the bag label according to the conventional MIL rule. We considered three citation datasets from (Sen et al., 2008): Citeseer, Cora, and PubMed. Finally, the MMIL network trained on PubMed will be mapped into an interpretable model using the procedure described in Section 4.1.

We view the datasets as graphs where nodes represent papers described by titles and abstracts, and edges are citation links. We treat the citation links as undirected edges, in order to have a setup as close as possible to earlier works, (Kipf and Welling, 2017; Hamilton et al., 2017). The goal is to classify papers according to their subject area.

We collected the years of publication for all the papers of each dataset, and for each dataset determined two thresholds $y_1 < y_2$, so that papers with publication year $y \leq y_1$ amount to approximately 40% of the data and are used as the training set, papers with publication year $y_1 < y \leq y_2$ formed a validation set of about 20%, and papers with publication year $y > y_2$ are the test set of 40% of the data. Table 5.1 reports the statistics for each dataset. More details on the temporal distributions in the three datasets are given in Appendix D (Figure D.1).

MMIL data was constructed from citation networks in which a top-bag X corresponds to a paper represented as the bag of nodes S_j containing the paper itself and all its neighbors. The nodes $S_j \in X$ are further decomposed as (sub-) bags of the words contained in the text (i.e. title and abstract) attached to the node. An

Dataset	# Classes	# Nodes	# Edges	# Training	# Validation	# Test
Citeseer	6	3,327	4,732	1,560 ($y \leq$ '99)	779 ('99 $< y \leq$ '00)	988 ($y >$ '00)
Cora	7	2,708	5,429	1,040 ($y \leq$ '94)	447 ('94 $< y \leq$ '95)	1,221 ($y >$ '95)
PubMed	3	19,717	44,338	8,289 ($y \leq$ '97)	3,087 ('97 $< y \leq$ '01)	8,341 ($y >$ '01)

Table 5.1: Structure of the citation graphs. With y we denote the year of publication. Citeseer classes are 6 among Agents, Artificial Intelligence (AI), Database (DB), Human-computer Interaction (HCI), Information Retrieval (IR), Machine Learning (ML). Cora classes are 7 among Case Based, Genetic Algorithms, Neural Networks, Probabilist Methods, Reinforcement Learning, Rule Learning, Theory. PubMed classes are 3 among Diabetes Mellitus Experimental (DME), Diabetes Mellitus Type 1 (DMT1), Diabetes Mellitus Type 2 (DMT2).

instance $x_{j,l} \in S_j$ is a word. Similarly, MIL data was constructed in which each paper is simply represented as the bag of all words appearing in the text of the paper or its neighbors. Figure 5.1 shows an example of MMIL and MIL decompositions starting from a node and its neighborhood of a citation graph. Words are encoded as one-hot vectors, in order to evaluate the capability of our model to learn relevant intermediate representations of bags from scratch.

We used an MMIL network model with two stacked bag-layers with ReLU activations with 250 units. The MIL model has one bag-layer with ReLU activations with 250 units. For both MMIL and MIL we proposed two versions which differ only for the aggregation functions for the bag-layers: one version uses max, the other uses mean aggregation. All models were trained by minimizing the softmax cross-entropy loss. We ran 100 epochs of the Adam optimizer with learning rate 0.001 and we early stopped the training according to the loss on the validation set.

As baselines, we considered näive Bayes and logistic regression. For these two models we reduced the task to a standard classification problem in which papers are represented by bag of words feature vectors (only for the words associated with the papers themselves, not considering citation neighbors). We also compared our models against GCN (Kipf and Welling, 2017) and GraphSAGE (Hamilton et al., 2017), which are briefly described in Section 3.3. GCN represents nodes as bags of words, while GraphSAGE exploits the sentence embedding approach described by (Arora et al., 2017). For comparison reasons and given that bag of words represent the most challenging and standalone approach which does not rely in any embedding representation of words, we encoded the nodes as bag of words for both GCN and GraphSAGE. As GraphSAGE allows to use both max and mean as aggregation

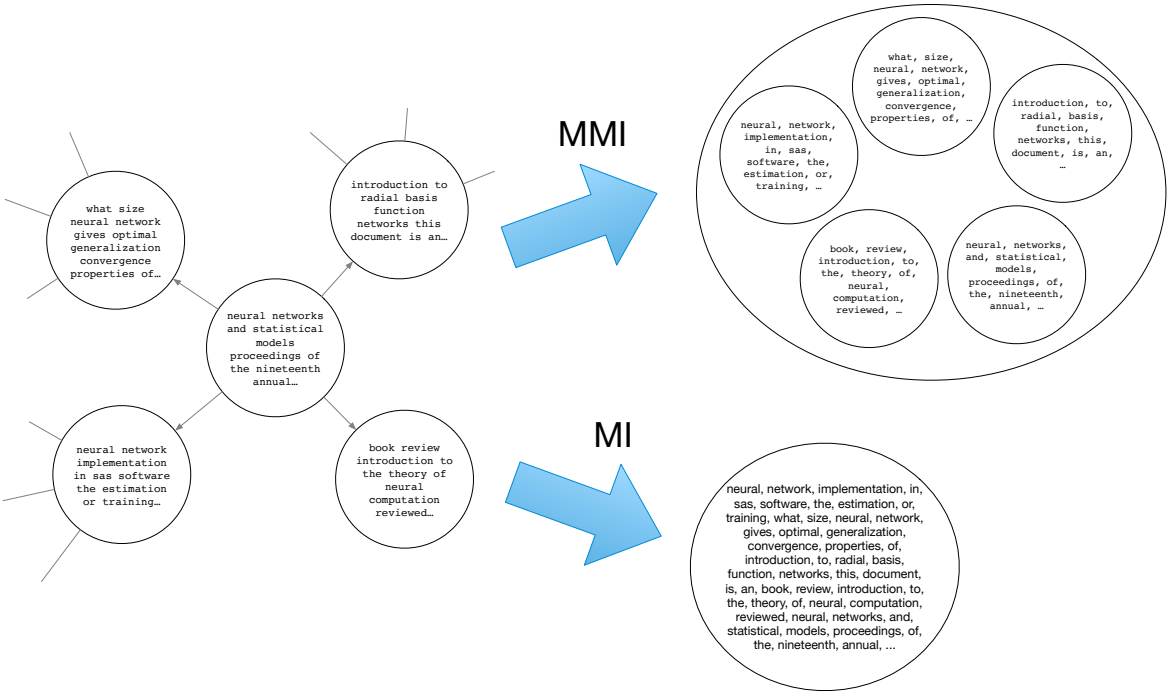


Figure 5.1: Given node an its neighborhood of a citation graph (left picture) we decomposed it as MMIL data (upper right picture) and MIL data (bottom right picture).

functions, we compared our models against both versions.

Results in Table 5.2 report the accuracy for all models. The MMIL networks outperform the other methods. MIL networks show a similar performance to GCN and GraphSage on Cora and Citeseer, and are close to MMIL on PubMed. It is noteworthy that the quite generic MMIL framework which here only is instantiated for graph data as a special case outperforms the methods that are specifically designed for graphs.

Model	Cora	Citeseer	PubMed
Naive Bayes (Bernoulli)	71.34%	63.77%	75.47%
Logistic Regression	74.94%	64.37%	73.67%
GCN (Kipf and Welling, 2017)	82.23%	66.50%	78.66%
GraphSage (Hamilton et al., 2017) MeanPool	80.18%	66.19%	75.59%
GraphSage (Hamilton et al., 2017) MaxPool	80.43%	67.61%	76.60%
MI-Mean	79.93%	62.96%	81.15%
MI-Max	81.08%	67.41%	80.22%
MMI-Mean	82.80%	**70.75%**	**81.27%**
MMI-Max	**84.03%**	69.64%	80.65%

Table 5.2: Accuracies on the test sets. Best results highlighted in bold.

Our general interpretability approach can also be applied to the MIL and MMIL models for the citation graphs, similarly to the MNIST and IMDB experiments in Section 4.2. We show here interpretability results for the PubMed citation dataset from in Section 5.1. Our interpretability study is for the MMI-Mean and MI-Mean models.

We first learn and interpret pseudo-labels. The optimal number of pseudo-labels for the MMIL model turned to be 3 ($v_1, \ldots, v_3$) and 5 ($u_1, \ldots, u_5$) for sub-bags and instances, respectively. On the other hand, the optimal number of pseudo-labels for the MIL model turned to be 3 ($u_1, \ldots, u_3$) for the instances. Figure D.2 in Appendix D depicts an heat-map which shows the fidelities on the validation set in function of the number of pseudo-labels for both instances and sub-bags for the MMIL model.

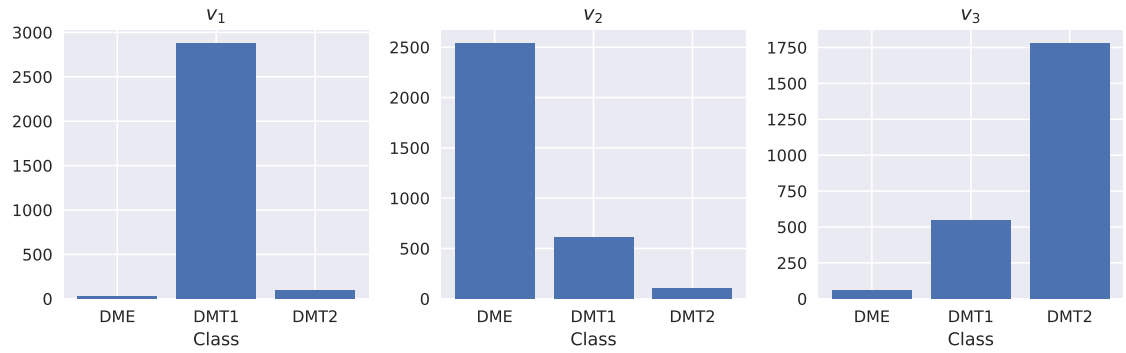


Figure 5.2: Correspondence between pseudo-labels and actual paper class labels.

Sub-bags in the MMIL decomposition also are papers, and therefore also are labeled with the actual class label. The number of inferred sub-bag pseudo-labels matches the number of actual classes, and Figure 5.2 shows that there is a clear correspondence between pseudo-labels and actual labels. Pseudo-labels for the Instances (words) are interpreted using the same approach as in Section 4.2. The result is shown in Table 5.7 and allows us to quite clearly recognize certain "topics" that correspond to the pseudo-labels. A similar interpretation for the instance (word) pseudo-labels in MIL case is given in Table 5.6.

Next we present the rules learned by a decision tree learner based on pseudo-label frequency feature vectors. For the MMIL model, Table 5.3 reports the rules mapping a bag of instance pseudo-labels to a sub-bag pseudo-label. Note that u_1 is not used in these rules. Similarly, Table 5.4 reports the rules mapping a bag of sub-bag pseudo-labels to a top-bag label. For the MIL model, Table 5.5 reports the rules mapping a bag of instance pseudo-labels into the corresponding top-bag label.

By classifying PubMed using the rules and pseudo-labels, we achieved an accuracy on the test set equals to 76.88% for the MMIL case and 79.25% for the MIL case. Fidelities for MMIL and MIL cases were 84.75% and 87.99%, respectively. Both of the

1	$\hat{g} = v_1 \leftarrow f_{u_2} \leq 44.53, f_{u_3} \leq 22.70, f_{u_5} \leq 6.47.$
2	$\hat{g} = v_1 \leftarrow f_{u_2} \leq 44.53, f_{u_3} \leq 31.31, f_{u_5} > 6.47.$
3	$\hat{g} = v_1 \leftarrow f_{u_2} > 44.53, f_{u_4} \leq 14.22, f_{u_5} > 14.83.$
4	$\hat{g} = v_2 \leftarrow f_{u_2} > 44.53, f_{u_3} \leq 22.60, f_{u_4} > 14.22.$
5	$\hat{g} = v_2 \leftarrow f_{u_2} > 44.53, f_{u_4} \leq 14.22, f_{u_5} \leq 14.83.$
6	$\hat{g} = v_3 \leftarrow f_{u_2} \leq 44.53, f_{u_3} > 22.70, f_{u_5} \leq 6.47.$
7	$\hat{g} = v_3 \leftarrow f_{u_2} \leq 44.53, f_{u_3} > 31.31, f_{u_5} > 6.47.$
8	$\hat{g} = v_3 \leftarrow f_{u_2} > 44.53, f_{u_3} > 22.60, f_{u_4} > 14.22.$

Table 5.3: Rules extracted from the MMIL network for mapping instance pseudo-labels into a sub-bag pseudo-label. See the caption of Table 4.4 for details on the syntax.

1	$\hat{h} = DME \leftarrow f_{v_1} \leq 8.51, f_{v_2} > 63.96.$
2	$\hat{h} = DMT1 \leftarrow f_{v_1} \in (8.51, 20.26], f_{v_2} > 63.96.$
3	$\hat{h} = DMT1 \leftarrow f_{v_1} > 20.26, f_{v_3} \leq 55.49.$
4	$\hat{h} = DMT2 \leftarrow f_{v_1} \leq 20.26, f_{v_2} \leq 63.96.$
5	$\hat{h} = DMT2 \leftarrow f_{v_1} > 20.26, f_{v_3} > 55.49.$

Table 5.4: Rules mapping sub-bag pseudo-labels into top-bag labels. See the caption of Table 4.4 for details on the syntax.

1	$\hat{h} = DME \leftarrow f_{u_1} > 49.80.$
2	$\hat{h} = DMT1 \leftarrow f_{u_1} \leq 49.80, f_{u_2} \leq 30.60, f_{u_3} \leq 30.65.$
3	$\hat{h} = DMT1 \leftarrow f_{u_1} \leq 49.80, f_{u_2} > 30.60, f_{u_3} \leq 35.66.$
4	$\hat{h} = DMT1 \leftarrow f_{u_1} \leq 49.80, f_{u_2} \leq 30.60, f_{u_3} > 30.65.$
5	$\hat{h} = DMT1 \leftarrow f_{u_1} \leq 49.80, f_{u_2} > 30.60, f_{u_3} > 35.66.$

Table 5.5: Classification rules for the MIL model. See the caption of Table 4.4 for details on the syntax.

results are still comparable and competitive with the methods described in Table 5.2. Thus, in this case the interpretable MIL model outperforms the interpretable MMIL model in terms of accuracy. However, for explaining individual classifications, the MMIL model can still have advantages due to the multi-level explanations it supports. Similarly as was done for text in Section 4.2, one can first explain the predicted label of a paper in terms of the citing/cited papers with relevant pseudo-labels, and then refine this explanation by tracing the pseudo-label assignments of papers to the pseudo-labels of their words. In the MIL model, on the other hand, one is limited to explanations at the word level.

u_1 - 48.33%	u_2 - 60.09%	u_3 - 41.92%
animals	children	non
induction	juvenile	subjects
induced	multiplex	patients
experimental	hla	indians
rats	childhood	fasting
rat	adolescents	obesity
dogs	conventional	pima
caused	girls	american
days	ascertainment	mexican
strains	autoimmune	indian
bl	dr	mody
experiment	infusion	oral
untreated	child	bmi
wk	siblings	obese
sz	intensive	men
restored	healthy	prevalence
sciatic	paediatric	resistance
experimentally	spk	tolerance
sprague	boys	mutations
partially	sharing	igt

Table 5.6: MIL case. Each column represents words belonging to the associated pseudo-labels. The percentage next to the pseudo-label names refers to the number of of words associated with the pseudo-labels ($\approx$ 15k). The words are ranked by the intra-cluster distance in descending order.

u_1 - 21.28%	u_2 - 28.76%	u_3 - 27.25%	u_4 - 12.84%	u_5 - 9.87%
normalization	animals	non	subjects	children
greatly	experimental	indians	patients	multiplex
susceptibility	induced	pima	patient	ascertainment
lymphocytes	induction	obesity	individuals	conventional
pregnant	rats	oral	type	juvenile
always	dogs	fasting	analysis	girls
organ	made	mexican	sample	night
destruction	rat	obese	cascade	childhood
tx	strains	medication	otsuka	pittsburgh
contraction	bl	bmi	forearm	adolescents
antibodies	caused	mody	gdr	infusion
sequential	wk	indian	reported	denmark
tract	counteracted	tolerance	mmol	intensified
decarboxylase	partially	look	age	child
recipients	rabbits	index	gox	beef
livers	days	agents	dependent	sharing
mt	conscious	resistance	isoforms	knowing
cyclosporin	sciatic	maturity	meals	paediatric
lv	tubules	gk	score	unawareness
laboratories	myo	ii	affinities	pubert

Table 5.7: MMIL case. Each column represents words belonging to the associated pseudo-labels. The percentage next to the pseudo-label names refers to the number of of words associated with the pseudo-labels ($\approx$ 15k). Pseudo-label u_1 is colored in grey since it is not used for constructing the rules. The words are ranked by the intra-cluster distance in descending order.

Social Network Datasets

We finally test our model on a slightly different type of prediction problems for graph data, where the task is graph classification, rather than node classification as in the previous section. For this we use the following six publicly available datasets first proposed by Yanardag and Vishwanathan (2015).

- COLLAB is a dataset where each graph represent the ego-network of a researcher, and the task is to determine the field of study of the researcher, which is one of *High Energy Physics*, *Condensed Matter Physics*, or *Astro Physics*.
- IMDB-BINARY, IMDB-MULTI are datasets derived from IMDB. First, genre-specific collaboration networks are constructed where nodes represent actors / actresses who are connected by an edge if they have appeared together in a movie of a given genre. Collaboration networks are generated for the genres *Action* and *Romance* for IMDB-BINARY and *Comedy*, *Romance*, and *Sci-Fi* for IMDB-MULTI. The data then consists of the ego-graphs for all actors/actresses in all genre networks, and the task is to identify the genre from which an ego-graph has been extracted.
- REDDIT-BINARY, REDDIT-MULTI5K, REDDIT-MULTI12K are datasets where each graph is derived from a discussion thread from Reddit. In those graphs each vertex represent a distinct user and two users are connected by an edge if one of them has responded to a post of the other in that discussion. The task in REDDIT-BINARY is to discriminate between threads originating from a discussion-based subreddit (TrollXChromosomes, atheism) or from a question / answers-based subreddit (IAmA, AskReddit).

  The task in REDDIT-MULTI5K and REDDIT-MULTI12K is a multiclass classification problem where each graph is labeled with the subreddit where it has originated (*worldnews, videos, AdviceAnimals, aww, mildlyinteresting* for REDDIT-MULTI5K and *AskReddit, AdviceAnimals, atheism, aww, IAmA, mildlyinteresting, Showerthoughts, videos, todayilearned, worldnews, TrollXChromosomes* for REDDIT-MULTI12K).

We transformed each dataset into MMIL data by treating each graph as a top-bag X. Each node of the graph with its neighborhood, is a sub-bag $S_j \in X$, while each node $x_{j,l} \in S_j$ is an instance.

In these six datasets no features are attached to the nodes. We therefore defined a node feature vector based on the degrees $deg(x_{j,l})$ of the nodes as follows: let deg^* be the maximum degree of any node. For $i = 1, \ldots, deg^*$ we then define

$$x^i_{j,l} = \begin{cases} \frac{1}{\sqrt{deg(x_{j,l})}} & if\ i < deg(x_{j,l}) \\ 0 & otherwise, \end{cases} \tag{5.1}$$

By using this representation the scalar product of two node feature vectors will be high if the nodes have similar degrees, and it will be low for nodes with very different degrees.

The MMIL networks have the same structure for all the datasets: a dense layer with 500 nodes and ReLU activation, two stacked bag-layers with 500 units (250 max units and 250 mean units), and a dense layer with dim_{out} nodes and linear activation, where dim_{out} is 3 for COLLAB, 2 for IMDB-Binary, and 3 for IMDB-MULTI, 2 for REDDIT-BINARY, 5 for REDDIT-MULTI5K, and 11 for REDDIT-MULTI12K. We performed a 10 times 10 fold cross-validation, training the MMIL networks by minimizing the binary cross-entropy loss (for REDDIT-BINARY and IMDB-BINARY) and the softmax cross-entropy loss (for COLLAB, IMDB-MULTI, REDDIT-5K, REDDIT-12K). We ran 100 epochs of the Adam optimizer with learning rate 0.001 on mini-batches of size 20.

We compared our method against DGK (Yanardag and Vishwanathan, 2015), Patchy-SAN (Niepert et al., 2016), and SAEN (Orsini et al., 2018).

Dataset	DGK	Patchy-SAN	SAEN	MMIL
COLLAB	73.09 ± 0.25	72.60 ± 2.15	78.50 ± 0.69	$\mathbf{79.46 \pm 0.31}$
IMDB-BINARY	66.96 ± 0.56	71.00 ± 2.29	71.59 ± 1.20	$\mathbf{72.62 \pm 1.04}$
IMDB-MULTI	44.55 ± 0.52	45.23 ± 2.84	48.53 ± 0.76	$\mathbf{49.42 \pm 0.68}$
REDDIT-BINARY	78.04 ± 0.39	86.30 ± 1.58	$\mathbf{87.22 \pm 0.80}$	86.54 ± 0.64
REDDIT-MULTI5K	41.27 ± 0.18	49.10 ± 0.70	$\mathbf{53.63 \pm 0.51}$	53.42 ± 0.67
REDDIT-MULTI12K	32.22 ± 0.10	41.32 ± 0.42	$\mathbf{47.27 \pm 0.42}$	45.25 ± 0.48

Table 5.8: Accuracies with standard deviations in graph classification. Best results are highlighted in bold.

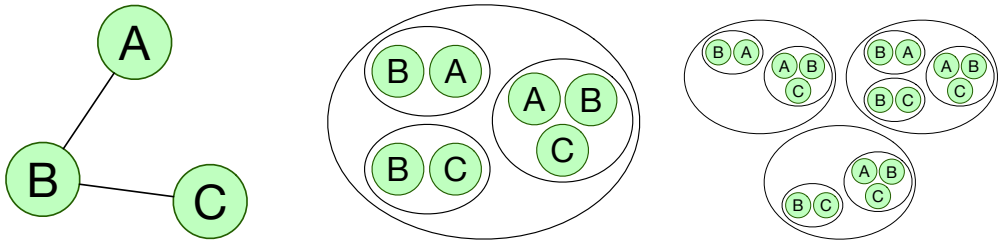
Results in Table 5.8 show that MMIL networks and SAEN perform comparably, with some advantages of these two methods over Patch-SAN, and more pronounced advantages over DGK.

5.2 Relation with Neural Network for Graphs

The MMIL models and the related graph decomposition used in the citation network experiments have similarities with GCN (Kipf and Welling, 2017) and GraphSAGE (Hamilton et al., 2017). It can be proved (see Kipf and Welling (2017) for further details) that the GCN aggregation function is equivalent up to a constant to

$$h_v^{(t+1)} = MEAN(\{h_v^{(t)}\} \cup \{h_u^{(t)}\ \forall u \in N(v)\})\ t = 1, \ldots, K,$$

where v represents a node, $N(v)$ its neighborhood, $h_v^{(t)}$ the feature vector associated to the node v at the time step t, and K represents the maximum number of iterations. MMIL can actually simulate GCN models by decomposing the graph in the appropriate manner and by using K bag-layers. For example, a neural network with one bag-layer with *mean* as aggregation function is, hence, equivalent to the GCN aggregation function for $K = 1$. Figure 5.3 depicts two examples of the appropriate decompositions of a graph in order to simulate GCN for $K = 1$ and $K = 2$.



(a) Graph (b) Dataset for $K = 1$ (c) Dataset for $K = 2$

Figure 5.3: Example of graph (left) decompositions for simulating GCN for $K = 1$ (middle) and $K = 2$ (right).

Compared to GraphSAGE our approach has another difference. By recalling the equation 2.10 in Chapter 2, i.e. $h_v^k = f_2(\text{CONCAT}(h_v^{k-1}, h_{\mathcal{N}(v)}^k); W_2)$, it is immediate to notice that we can not directly use our approach to simulate GraphSAGE. Indeed in the GraphSAGE aggregation function there is an asymmetry due to the concatenation operator which allows, by construction, to treat the node and its neighborhood separately. With our approach this is not immediately possibile, even though by using a description of the dataset with three layer of nesting (and three bag-layers in the models) we would obtain a scenario in which a node and its neighborhood are set apart, i.e.

$$X(v) = \{\{x(v)\}, \{x(u), u \in \mathcal{N}(v)\}\},$$

where $x(v)$ is the feature vector associated with the node v, and $\mathcal{N}(v)$ is the neighborhood of v. However this represenation does not allow to distinguish between the node and its neighbors as they are treated symmetrically. To overcome this problem a possible solution is briefly explained in Section 6.1.

Chapter 6

Conclusions

We have introduced the MMIL framework for handling data organized in nested bags. The MMIL setting allows for a natural hierarchical organization of data, where components at different levels of the hierarchy are unconstrained in their cardinality. We have identified several learning problems that can be naturally expressed as MMIL problems. For instance, image, text or graph classification are promising application areas, because here the examples can be objects of varying structure and size, for which a bag-of-bag data representation is quite suitable, and can provide a natural alternative to graph kernels or convolutional network for graphs. The fact that bags do not impose an order on their elements directly leads to useful spatial invariance properties when applied to image data. Furthermore we proposed new way of thinking in terms of interpretability. Although some MIL models can be easily interpreted by exploiting the learnt instance labels and the assumed rule, MMIL networks can be interpreted in a finer level: by removing the common assumptions of the standard MIL, we are more flexible and we can first associate labels to instances and sub-bags and then combine them in order to extract new rules. Finally, we proposed a different perspective to see convolutions on graphs. In most of the neural network for graphs approaches convolutions can be interpreted as message passing schema, while in our approach we provided a decomposition schema.

We proposed a neural network architecture involving the new construct of bag-layers for learning in the MMIL setting. Theoretical results show the expressivity of this type of model. In the empirical results we have shown that learning MMIL models from data is feasible, and the theoretical capabilities of MMIL networks can be exploited in practice, e.g., to learn accurate models for noiseless data, or location invariant models for image classification. Furthermore MMIL networks can be applied in a wide spectrum of scenarios, such as text, image, and graphs. For this latter we showed that MMIL is competitive with the state-of-the-art models on node and graph classification tasks, and, in many cases, MMIL models outperform the others.

In this thesis, we have focused on the setting where whole bags-of-bags are to

be classified. In conventional MIL learning, it is also possible to define a task where individual instances are to be classified. Such a task is however less clearly defined in our setup since we do not assume to know the label spaces at the instance and sub-bag level, nor the functional relationship between the labels at the different levels.

6.1 Future works

A possible direction for future work would be to extend the architecture proposed in Chapter 3 for addressing graph problems in a more general fashion. In Section 5.2 of Chapter 5 we presented the differences between our framework and some state-of-the-art models for graph learning. In particular we showed that GraphSAGE (Hamilton et al., 2017) allows to treat nodes and their neighbors separately. With our model this is not directly feasible and a possible way to overcome this limitation (and hence an extension of our framework) would be to combine a MIL and MMIL models by concatenating the bag-layer of the MIL and second bag-layer of the MMIL model. The bag-layer of the MIL model would output a representation for the node while the second bag-layer of the MMIL model would output a representation for its neighborhood.

Appendix A

Bag-layer implementation

In this Chapter we report the Python 3.6 implementation of the bag-layer (see Section 3.2). The code depends only on TensorFlow[1] (version 1.8 or above).

Bag-Layer implementation in Python

```
import tensorflow as tf

class BagLayer:
    def __init__(self, input_dim, output_dim, name, aggregation,\
                 activation_fn=tf.nn.relu):
        """Bag-Layer contructor

        Parameters:
            input_dim : int
                size of the previous layer. In case there
                this is the first layer, 'input_dim' corresponds
                to the input size.
            output_dim : int
                number of bag-layer units.
            name : str
                name of the tensors.
            aggregation : str
                a string which represents the aggregation function
                among 'mean', 'sum', 'max'.
            activation_fn : callable
                the element-wise function to perform before
                the aggregation, e.g. lambda x:x,
                tf.sigmoid, tf.nn.relu. Default: tf.nn.relu.
        """

        if aggregation.lower() not in ['mean', 'max', 'sum']:
            raise ValueError("Wrong aggregation function.\
                              Please use one among: mean, max, sum")
        if aggregation == 'mean':
            self.aggregation_fn = tf.segment_mean
        if aggregation == 'max':
            self.aggregation_fn = tf.segment_max
        if aggregation == 'sum':
            self.aggregation_fn = tf.segment_sum
        self.name = name
```

[1] https://www.tensorflow.org

```
        self.W = tf.get_variable('{}-W'.format(self.name),\
                                 shape=(input_dim, output_dim),\
                                 initializer=tf.glorot_normal_initializer())
        self.b = tf.get_variable('{}-b'.format(self.name),\
                                 shape=(output_dim,),\
                                 initializer=tf.constant_initializer(0.))
        self.activation_fn = activation_fn

    def __call__(self, x, idx):
        """Applies the bag-layer aggregation function to
           a batch of sets.

        Parameters:
            x : 2darray (tf.float32)
                a 2d array which represents all the instance features
                of all the sets in the batch.
            idx : 1darray (tf.int32)
                a 1d array whose entries are
                the ids of the sets where the instances belong to.
                The length of idx must be equal to length of x.
                idx must be input in not decreasing order.
                For example idx = [0,0,1], means that
                the first two rows of x represent features
                belonging
                to the same set, and the last row of x
                represent feature belongin to another set.

        Returns :
            aggregations : ndarray (tf.float32)
        """

        activations  = self.activation_fn(tf.matmul(x, self.W) + self.b)
        aggregations = self.aggregation_fn(activations, idx)

        return aggregations
```

For better understanding how the Bag-Layer works let us consider a toy example. Let $X_1 = \{\{2,5\},\{1,3,5\}\}$ and $X_2 = \{\{2,4\}\}$ be two top-bags containing digits ranging from 1 to 5. In the following we consider a MMIL network, N, consisting of only two stacked bag-layers, $bl-1$ and $bl-2$. Each bag-layer has exactly 1 unit, their corresponding weights are equal to 1, and their corresponding biases are equal to 0. Finally, let us assume that the aggregation functions are the *max* and *sum* for $bl-1$ and $bl-2$, respectively. This simple network will return for each top-bag the sum of the max number containing in each sub-bag. In the example $N(X_1) = 10$, $N(X_2) = 4$. Figure A.1 depicts how the top-bags are processed through the MMIL network N step by step.

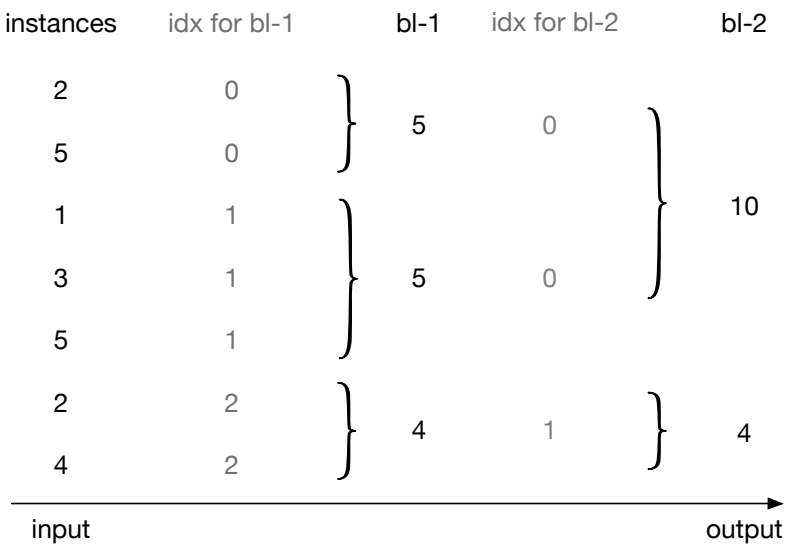


Figure A.1: A MMIL network consisting of two-stacked bag-layers, processes two top-bags.

Appendix B

Details for the Experiments on Semi-Synthetic Data (Section 4.2)

Layer	Parameters
Convolutional Layer	kernel size 5×5 with 32 channels
Batch Normalization	
ReLU	
Max Pooling	kernel size 2×2
Dropout	probability 0.5
Convolutional Layer	kernel size 5×5 with 64 channels
Batch Normalization	
ReLU	
Max Pooling	kernel size 2×2
Dropout	probability 0.5
Dense	1024 units
ReLU	
Dropout	probability 0.5
BagLayer (ReLU activation)	200 units
ReLU	
BagLayer (ReLU activation)	200 units
ReLU	
Dense	1 unit

Table B.1: Neural network structure for MMI MNIST dataset. The model was trained by minimizing the binary cross entropy loss. We ran 200 epochs of the Adam optimizer (Kingma and Ba, 2014) with learning rate 0.001 and mini-batch size of 20.

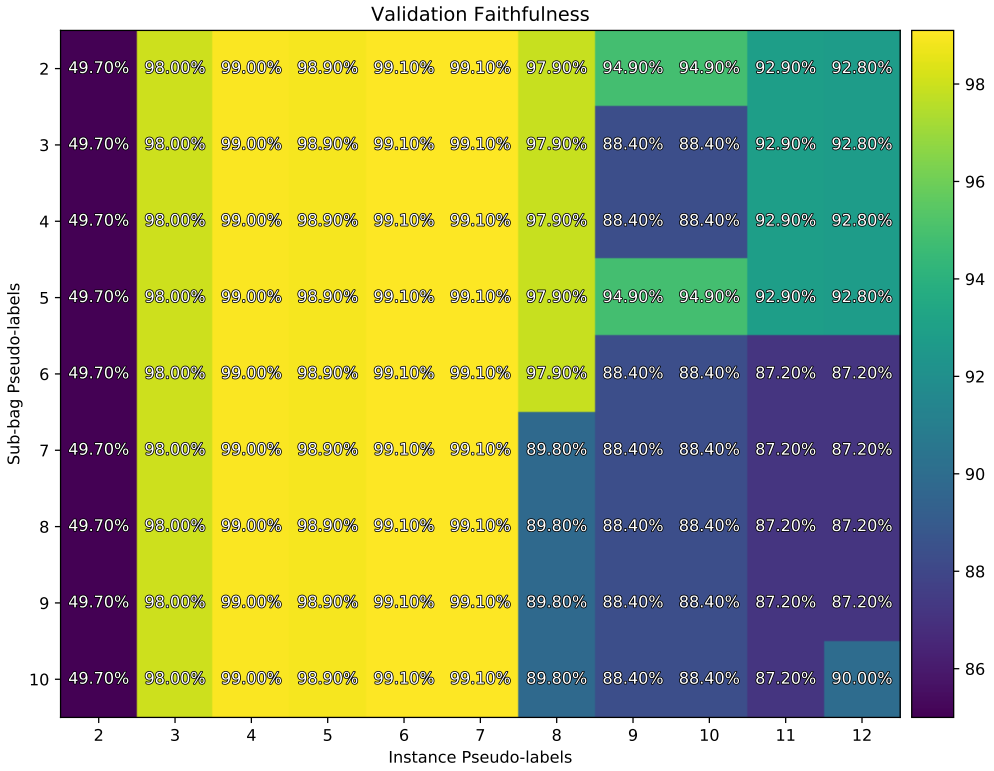


Figure B.1: Fidelities on the validation set for MNIST. We used 20% of the training set as validation set. On the x-axis are reported the number of instance pseudo-labels, while on the y-axis are reported the number of sub-bag pseduo-labels. Due to the fact there are several combinations which maximize the validation fidelity, we chose the one which has less pseudo-labels. Hence, the best fidelity on the validation set is obtained by choosing 6 pseudo-labels for instances and 2 pseudo-labels for sub-bags.

Appendix C

Details for the Experiments on Sentiment Analysis (Section 4.2)

We report here a second example of classification explanation. Here we are considering a positive review that was mis-classified as negative by the MMIL rules, and correctly classified by the MIL model. Following the same typesetting conventions as used in Table 4.7, the review and the labeling of the prediction-relevant parts are shown in Table C.1. In the MMIL case, classification was due to rule 6 in Table 4.5. The sentence "The storyline is ..." was assigned label v_1 by rule 1 in Table 4.4, whereas the sentence "The mental patients..." was assigned label v_3 by rule 6 in Table 4.4. The positive classification in the MIL case was due to rule 2 in Table 4.6, which is based on pseudo-labels u_3 and u_6.

Young, ambitious nurse Ms. Charlotee (Rosie Holotik) is sent to work at a mental asylum out in the middle of nowhere. During the course of 3 days, she encounters strange happenings, even a patient in her bedroom watching her, yet she still stays.
[v_3] The mental patients are all a little eye rolling (espically by the **Judge), but my favorite was**[1] the old crazy biddy (Rhea MacAdams).
[v_1] **The storyline is**[4] **okay at best**[2] **, and**[1] the acting is **surprisingly alright, but**[2] after awhile it's gets to be **a little much**[2]. But, still it's fun, quirky, strange, and original. xNote: The thing inside the basement is hardly horrifying, so the title is a little bananas.

Young, ambitious nurse Ms. Charlotee (Rosie Holotik) is sent to work at a mental asylum out in the middle of nowhere. During the course of 3 days, she encounters strange happenings, even a patient in her bedroom watching her, yet she still stays. The mental patients are all a little eye rolling (espically by the **Judge), but my favorite was**[6] the old crazy biddy (Rhea MacAdams).
The storyline is okay[3] **at best, and**[6] the **acting is surprisingly**[6] alright, but after awhile it's gets to be a little much. **But, still it's fun, quirky, strange**[6], and original. xNote: The thing inside the basement is hardly horrifying, so the title is a little bananas.

Table C.1: A sample positive review. Top: MMIL labeling. Bottom: MIL labeling.

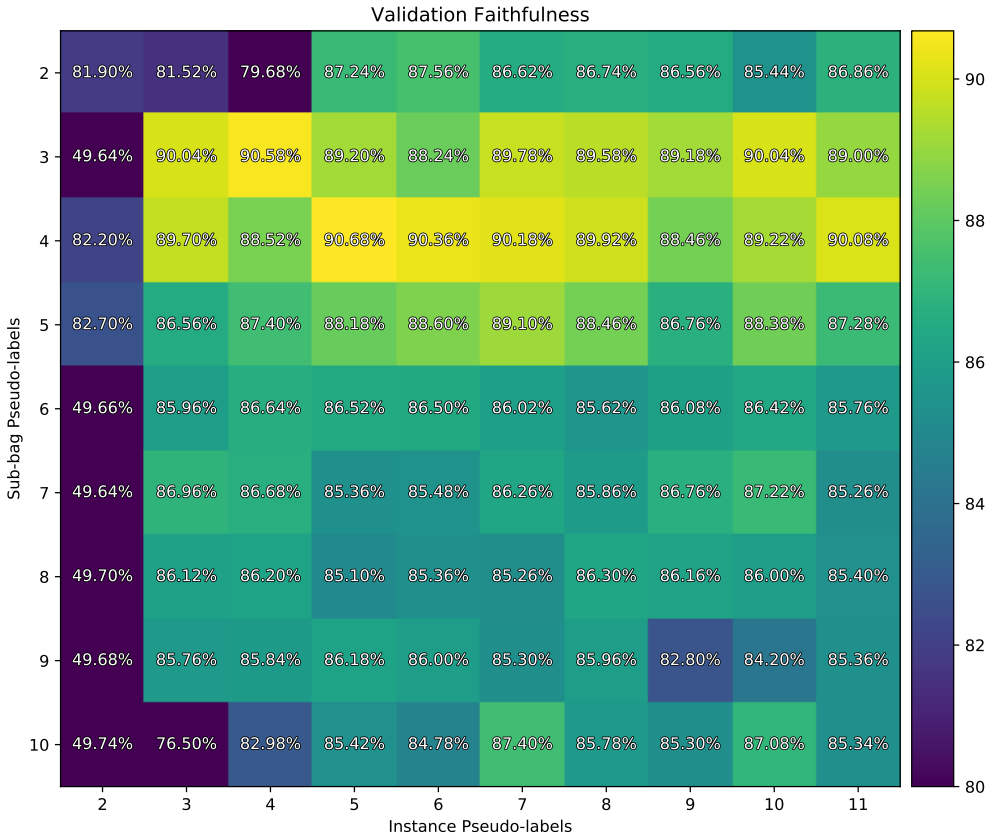


Figure C.1: Fidelities on the validation set for IMDB. On the x-axis are reported the number of instance pseudo-labels, while on the y-axis are reported the number of sub-bag pseduo-labels. The best fidelity on the validation set is obtained by choosing 10 pseudo-labels for instances and 3 pseudo-labels for sub-bags.

Appendix D

Details for the Experiments on Citation Datasets (Section 5.1)

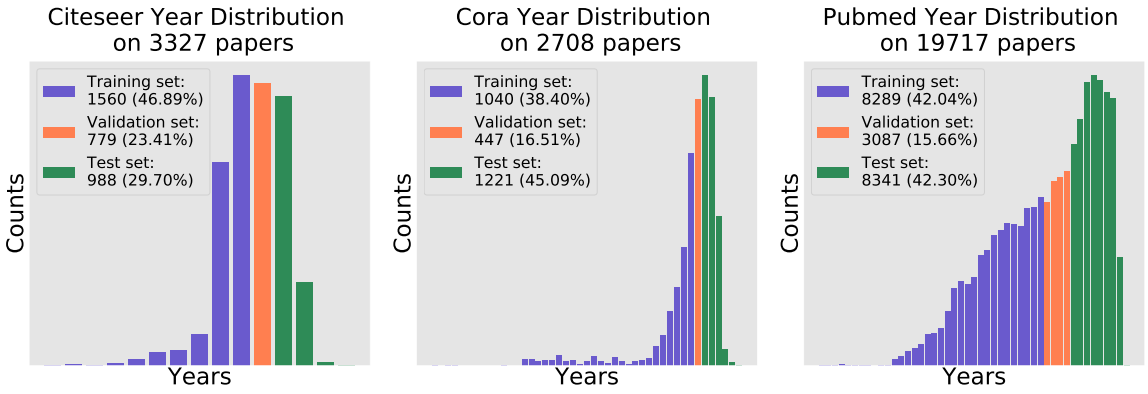


Figure D.1: Distribution of papers over years for Citaseer, Cora, and PubMed.

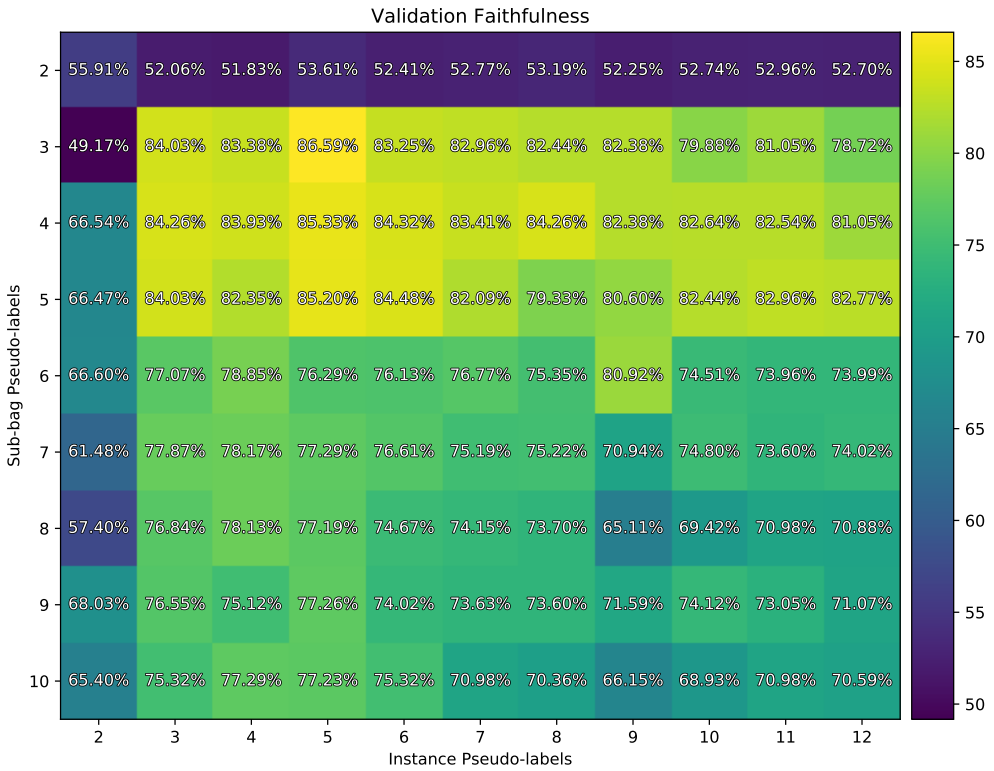


Figure D.2: Fidelities on the validation set for PubMed. On the x-axis are reported the number of instance pseudo-labels, while on the y-axis are reported the number of sub-bag pseduo-labels. The best fidelity on the validation set is obtained by choosing 5 pseudo-labels for instances and 3 pseudo-labels for sub-bags.

Appendix E

Publications

Journal papers

1. AP. Di Giovanna, **Alessandro Tibo**, L. Silvestri, MC. Müllenbroich, I. Costantini, ALA. Mascaro, L. Sacconi, P. Frasconi, FS. Pavone, "Whole-brain vasculature reconstruction at the single capillary level", *Scientific reports*, 2018. **Candidate's contributions**: designed and carried out the segmentation experiments.

Peer reviewed conference papers

1. **Alessandro Tibo**, P. Frasconi, M. Jaeger, "A network architecture for multi-multi-instance learning", *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages: 737–752, 2017. **Candidate's contributions**: designed algorithms, carried out theoretical analyses, conceived and planned the experiments, wrote the manuscript

2. AP. Di Giovanna, **Alessandro Tibo**, L. Silvestri, MC. Müllenbroich, I. Costantini, L. Sacconi, P. Frasconi, FS. Pavone, "Optimal staining and clearing protocol for whole mouse brain vasculature imaging with light-sheet microscopy", *European Conference on Biomedical Optics*, 2017. **Candidate's contributions**: designed and carried out the segmentation experiments.

Workshop papers

1. T. Raissi, **Alessandro Tibo**, P. Bientinesi, "Extended Pipeline for Content-Based Feature Engineering in Music Genre Recognition", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages: 2661–2665, 2018. **Candidate's contributions**: helped supervise the project, contributed to the final version of the manuscript

Papers under review

1. **Alessandro Tibo**, M. Jaeger, P. Frasconi "Learning and Interpreting Multi-Multi-Instance Learning Networks", *Journal of Machine Learning Research*, 2018.`http://arxiv.org/abs/1810.11514` **Candidate's contributions**: designed algorithms, carried out theoretical analyses, conceived and planned the experiments, wrote the manuscript

2. T. Borghuis, **Alessandro Tibo**, S. Conforti, L. Canciello, L. Brusci, P. Frasconi, "Off the Beaten Track: Using Deep Learning to Interpolate Between Music Genres", *IEEE MultiMedia*, 2018. `https://arxiv.org/pdf/1804.09808` **Candidate's contributions**: designed algorithms, conceived and planned the experiments

Bibliography

Andrews, S., Tsochantaridis, I., and Hofmann, T. (2003). Support vector machines for multiple-instance learning. In *Advances in neural information processing systems*, pages 577–584.

Arbeláez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour Detection and Hierarchical Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916. 00000.

Arora, S., Liang, Y., and Ma, T. (2017). A simple but tough-to-beat baseline for sentence embeddings. *International Conference on Learning Representations (ICLR)*.

Atwood, J. and Towsley, D. (2016). Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1993–2001.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Breiman, L. (2017). *Classification and regression trees*. Routledge.

Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM.

Conneau, A., Schwenk, H., Barrault, L., and Lecun, Y. (2017). Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1107–1116.

Costa, F. and De Grave, K. (2010). Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of the 26th International Conference on Machine Learning*, pages 255–262. Omnipress.

De Raedt, L., Demoen, B., Fierens, D., Gutmann, B., Janssens, G., Kimmig, A., Landwehr, N., Mantadelis, T., Meert, W., Rocha, R., et al. (2008). Towards digesting the alphabet-soup of statistical relational learning.

De Raedt, L., Frasconi, P., Kersting, K., and Muggleton, S., editors (2008). *Probabilistic inductive logic programming: theory and applications*, volume 4911 of *Lecture notes in computer science*. Springer, Berlin.

Dietterich, T. G. (2000). Ensemble Methods in Machine Learning. In *Multiple Classifier Systems*, number 1857 in Lecture Notes in Computer Science, pages 1–15. Springer Berlin Heidelberg. 02917.

Dietterich, T. G., Lathrop, R. H., and Lozano-Pérez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71.

Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional Networks on Graphs for Learning Molecular Fingerprints. *arXiv:1509.09292 [cs, stat]*. arXiv: 1509.09292.

Foulds, J. and Frank, E. (2010). A review of multi-instance learning assumptions. *The Knowledge Engineering Review*, 25(01):1. 00081.

Frasconi, P., Gori, M., and Sperduti, A. (1998). A general framework for adaptive processing of data structures. *IEEE Trans. on Neural Networks*, 9:768–786.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202. 01681.

Gärtner, T., Lloyd, J. W., and Flach, P. A. (2004). Kernels and distances for structured data. *Machine Learning*, 57(3):205–232.

Getoor, L. and Taskar, B., editors (2007). *Introduction to statistical relational learning*. MIT Press, Cambridge, Mass.

Gori, M., Monfardini, G., and Scarselli, F. (2005). A new model for learning in graph domains. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 2, pages 729–734. IEEE.

Griffiths, T. L. and Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, page 5228–5235.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034.

Haussler, D. (1999). Convolution kernels on discrete structures. Technical Report 646, Department of Computer Science, University of California at Santa Cruz. 01152.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366. 12001.

Horváh, T., Gärtner, T., and Wrobel, S. (2004). Cyclic pattern kernels for predictive graph mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 158–167. ACM.

Hou, L., Samaras, D., Kurc, T. M., Gao, Y., Davis, J. E., and Saltz, J. H. (2015). Efficient multiple instance convolutional neural networks for gigapixel resolution image classification. *arXiv preprint*.

Jaeger, M. (1997). Relational bayesian networks. In Geiger, D. and Shenoy, P. P., editors, *Proceedings of the 13th Conference of Uncertainty in Artificial Intelligence (UAI-13)*, pages 266–273, Providence, USA. Morgan Kaufmann.

Kingma, D. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*. 00204 arXiv: 1412.6980.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR)*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Landwehr, N., Passerini, A., De Raedt, L., and Frasconi, P. (2010). Fast learning of relational kernels. *Machine learning*, 78(3):305–342.

Lapuschkin, S., Binder, A., Montavon, G., Müller, K.-R., and Samek, W. (2016). The lrp toolbox for artificial neural networks. *The Journal of Machine Learning Research*, 17(1):3938–3942.

Laurent, H. and Rivest, R. L. (1976). Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1):15–17.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551. 01543.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Manhaeve, R., Dumančić, S., Kimmig, A., Demeester, T., and De Raedt, L. (2018). Deepproblog: Neural probabilistic logic programming. *arXiv preprint arXiv:1805.10872*.

Maron, O. and Lozano-Pérez, T. (1998). A framework for multiple-instance learning. In *Advances in neural information processing systems*, pages 570–576.

Maron, O. and Ratan, A. L. (1998). Multiple-instance learning for natural scene classification. In *ICML*, volume 98, pages 341–349.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Miyato, T., Dai, A. M., and Goodfellow, I. (2017). Adversarial training methods for semi-supervised text classification. *International Conference on Learning Representations (ICLR)*.

Natarajan, A. R. and Van der Ven, A. (2018). Machine-learning the configurational energy of multicomponent crystalline solids. *npj Computational Materials*, 4(1):56.

Neumann, M., Patricia, N., Garnett, R., and Kersting, K. (2012). Efficient graph kernels by randomization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 378–393. Springer.

Niepert, M., Ahmed, M., and Kutzkov, K. (2016). Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*.

Orsini, F., Baracchi, D., and Frasconi, P. (2018). Shift aggregate extract networks. *Frontiers in Robotics and AI*, 5:42.

Orsini, F., Frasconi, P., and De Raedt, L. (2015). Graph invariant kernels. In *Proceedings of the Twenty-fourth International Joint Conference on Artificial Intelligence*, pages 3756–3762.

Passerini, A., Frasconi, P., and Raedt, L. D. (2006). Kernels on prolog proof trees: Statistical learning in the ilp setting. *Journal of Machine Learning Research*, 7(Feb):307–342.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

Quinlan, R. (1993). 4.5: Programs for machine learning morgan kaufmann publishers inc. *San Francisco, USA*.

Rahmani, R., Goldman, S. A., Zhang, H., Krettek, J., and Fritts, J. E. (2005). Localized content based image retrieval. In *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pages 227–236. ACM.

Ramon, J. and De Raedt, L. (2000). Multi instance neural networks. In *Proceedings of the ICML-2000 workshop on attribute-value and relational learning*, pages 53–60. 00115.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine Learning*, 62:107–136.

Samek, W., Montavon, G., Binder, A., Lapuschkin, S., and Müller, K.-R. (2016). Interpreting the predictions of complex ml models by layer-wise relevance propagation. *arXiv preprint arXiv:1611.08191*.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80. 00073.

Scott, S., Zhang, J., and Brown, J. (2005). On generalized multiple-instance learning. *International Journal of Computational Intelligence and Applications*, 5(01):21–35. 00059.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*, 29(3):93. 00567.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. (2011). Weisfeiler-lehman graph kernels. *The Journal of Machine Learning Research*, 12:2539–2561.

Shervashidze, N., Vishwanathan, S. V. N., Petri, T., Mehlhorn, K., and Borgwardt, K. M. (2009). Efficient graphlet kernels for large graph comparison. In *AISTATS*, volume 5, pages 488–495. 00219.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *arXiv:1602.07261 [cs]*. 00127 arXiv: 1602.07261.

Tax, D. M. and Duin, R. P. (2008). Learning curves for the analysis of multiple instance classifiers. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 724–733. Springer.

Tibo, A., Frasconi, P., and Jaeger, M. (2017). A network architecture for multi-multi-instance learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 737–752. Springer.

Vapnik, V. (1963). Pattern recognition using generalized portrait method. *Automation and remote control*, 24:774–780.

Wang, J. and Zucker, J.-D. (2000). Solving multiple-instance problem: A lazy learning approach. In *Proceedings of the ICML-2000 workshop on attribute-value and relational learning*, pages 1119–1126. 00444.

Weidmann, N., Frank, E., and Pfahringer, B. (2003). A two-level learning method for generalized multi-instance problems. In *European Conference on Machine Learning*, pages 468–479. Springer. 00110.

Weisfeiler, B. and Lehman, A. (1968). A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16.

Xiao, Y., Liu, B., and Hao, Z. (2017). A sphere-description-based approach for multiple-instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 39(2):242–257.

Yan, Z., Zhan, Y., Peng, Z., Liao, S., Shinagawa, Y., Zhang, S., Metaxas, D. N., and Zhou, X. S. (2016). Multi-instance deep learning: Discover discriminative local anatomies for bodypart recognition. *IEEE transactions on medical imaging*, 35(5):1332–1343.

Yanardag, P. and Vishwanathan, S. (2015). Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374. ACM.

Yang, C., Dong, M., and Hua, J. (2006). Region-based image annotation using asymmetrical support vector machine-based multiple-instance learning. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2057–2063. IEEE.

Yang, C. and Lozano-Perez, T. (2000). Image database retrieval with multiple-instance learning techniques. In *Data Engineering, 2000. Proceedings. 16th International Conference on*, pages 233–243. IEEE.

Zha, Z.-J., Hua, X.-S., Mei, T., Wang, J., Qi, G.-J., and Wang, Z. (2008). Joint multi-label multi-instance learning for image classification. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

Zhang, Q. and Goldman, S. A. (2002). Em-dd: An improved multiple-instance learning technique. In *Advances in neural information processing systems*, pages 1073–1080.

Zhou, Z.-H., Jiang, K., and Li, M. (2005). Multi-instance learning based Web mining. *Applied Intelligence*, 22(2):135–147.

Zhou, Z.-H., Zhang, M.-L., Huang, S.-J., and Li, Y.-F. (2012). Multi-instance multi-label learning. *Artificial Intelligence*, 176(1):2291–2320. 00288.