Abstract

Multi-label classification is a generalization of binary classification where the task consists in predicting sets of labels. With the availability of ever larger datasets, the multi-label setting has become a natural one in many applications, and the interest in solving multi-label problems has grown significantly. As expected, deep learning approaches are now yielding state-of-the-art performance for this class of problems. Unfortunately, they usually do not take into account the often unknown but nevertheless rich relationships between labels. In this paper, we propose to make use of this underlying structure by learning to partition the labels into a Markov Blanket Chain and then applying a novel deep architecture that exploits the partition. Experiments on several popular and large multi-label datasets demonstrate that our approach not only yields significant improvements, but also helps to overcome trade-offs specific to the multi-label classification setting.

1. Introduction

Data sources, such as social media, news services, and crowd-sourced experiments, usually generate data where every instance can be assigned several labels (also called classes, categories, or tags). This turns modern classification problems into not only enormous scale but also multi-label in many practical settings. For example in text classification, a document naturally conveys different related topics (e.g., sports and tourism). In image categorization as well, a picture contains more than one object of interest. Furthermore, the labels usually carry rich semantic structures (Deng et al., 2009; Partalas et al., 2015).

A straightforward approach to Multi-Label Classification (MLC) consists in reducing the original problem to several independent single-label tasks (Tsoumakas & Katakis, 2007). This decomposition, called binary relevance, is appealing because of its simplicity and scalability. Unfortunately, it disregards the complex inter-dependencies that often exist between the labels. Consequently, it suffers from the lack of expressiveness and is prone to inconsistent predictions. On the other hand, the methods that exploit label dependencies can significantly improve the classification performance. However, most of the contributions hitherto suffer from the cost of increased training and prediction complexity and, hence, are limited in their applicability to cases with modest numbers of examples and categories.

The recent advancements in deep and representation learning have been successfully applied to address some of the issues of MLC (Nam et al., 2014). The proposed model in this prior study highlights the amenability of several tools, either developed or commonly used in deep learning, in the multi-label setting. In particular, it demonstrates that a simple multilayer perceptron with hidden rectified linear units (ReLU) (Glorot et al., 2011) trained with AdaGrad (Duchi et al., 2011) to jointly predict all the labels correctly can achieve state-of-the-art performance. The reason for this significant performance improvement in comparison to several existing approaches is threefold: Firstly, many real-world MLC problems exhibit Zipf’s law-like label frequency distribution, i.e., few categories are very popular while the majority is rare. Therefore, when AdaGrad is used, the learning rate becomes adapted to the data distribution by allowing larger gradient steps for the rarely occurring labels in comparison to more frequent ones. Secondly, hidden ReLUs encourage a sparse intermediate representation that is presumably more discriminative (Bengio et al., 2013b; Olshausen & Field, 1997). Moreover, dropout regularization (Hinton et al., 2012) prevents specialization of groups of hidden units into predicting only some particular classes. Finally, the shared intermediate representation across the tasks (labels) implicitly models the dependencies between them and provides sharing of the statistical strength. This procedure, also called multitasking (Caruana, 1997), is an important argument for learning deep architectures for MLC. It naturally exploits la-
bel dependencies which importance in the design of accurate multi-label classifiers has been developed into a \textit{opinio communis} (Read et al., 2011; Dembczyński et al., 2012; Cisse et al., 2013).

The categories in real world multi-label problems have a rich variety of relationships: Some labels may entail others (e.g., the presence of stars in a picture implies the relevance of night) or can be mutually exclusive (e.g., a realistic picture will likely not contain both trees and whales). There are also weaker types of relationships such as positive correlation (e.g., topics such as \textit{politics} and \textit{economics} tend to appear together). While such relationships can undoubtedly be exploited directly—some labels are predictable based on the co-occurrence with the others—the traditional deep neural networks incorporate label dependence information into the learning process only through multitasking (Nam et al., 2014; Bengio et al., 2013b). Indeed, all the recent architectures are designed to extract a hierarchy of representations from the input and to terminate with a single output layer that yields predictions. Even though they can be deep in the input space, they are flat in their output. One exception is the use of a hierarchical softmax (Jean et al., 2014) which organizes the set of labels as a tree. Unfortunately, unless the tree structure is known in advance, using a random tree has been shown to yield poor performance.

In this work, we propose a novel deep architecture for solving MLC tasks by directly leveraging label dependence information. Our proposal relies on a well-known observation in MLC (Deng et al., 2014): For a given example, it is often possible to infer the validity of several labels relying on the correctness information about a carefully chosen subset of categories. For instance, in scene classification, one does not need to observe an image before deciding the rightness of the label night once she knows that lights, stars or moon are present. In probabilistic terms, the labels \textit{light}, \textit{stars}, and \textit{moon} separate the label \textit{night} from the image; they are the Markov Blanket (MB) (Pearl, 2014). A partition of the set of tags into two subsets such that one subset is a Markov Blanket of the other, which we call a Markov Blanket Chained (MBC) partition, suggests a deep architecture composed of two main parts. The first part is akin to a traditional MLP and goes from the inputs to the labels of the Markov blanket. The second part goes from the Markov blanket to its complement set of categories, potentially with intermediate hidden layers. It uses the labels of the Markov Blanket as a high-level representation of the input data. Overall, the resulting Architecture is also Deep In the Output Space (ADIOS).

In practice, it is likely that an MBC partition does not strictly exist or cannot be precisely discovered since we only observe the relationships that are present in the data. However, it is often possible to learn a good approximate MB that contains enough information to predict the other labels accurately. Moreover, in the cases where the information provided by the estimated MB is insufficient, composing this high-level information with a representation directly extracted from the input yields improved performance.

The remainder of the paper is organized as follows: Section 2 summarizes the previous related work. Section 3 presents a simple approach to building an approximate Markov Blanket Chained partition of the set of labels. In Section 4, the details of the model and the learning algorithm are provided. Section 5 presents experimental results on several popular and large multi-label datasets validating the model and providing several insights.

2. Related Work

Label dependence exploitation is a blossoming line of research in multi-label classification (Read et al., 2011; Hariharan et al., 2010; Weston et al., 2013). It has recently attracted significant contributions that fall into four main groups. The first family of approaches, to which classifier chains belong (Read et al., 2011; Zaragoza et al., 2011), expands the initial representation of the data with additional features that represent the labels. The methods of the second group (Tai & Lin, 2012; Hsu et al., 2009) assume a low-rank label matrix and learn a low-dimensional embedding of the labels capturing interdependencies between them. The most representative works in this direction are Principle Label Space Transformation (Tai & Lin, 2012), Compressed Sensing (Hsu et al., 2009), and Bloom Filters (Cisse et al., 2013). The techniques of the third category, such as Label Partitioning (Weston et al., 2013), cluster the labels based on their correlations to predict consistent sets. Finally, techniques such as Laconic (Bengio et al., 2013a) use external knowledge about label co-occurrences to regularize the model. All these methods bear resemblances to those presented in this work in that they exploit dependencies between labels to improve labeling performance. However, they are neither trainable end-to-end nor do they enjoy the appealing properties of deep architectures.

A more related approach is the recently introduced HEX model (Deng et al., 2014) that connects a traditional deep neural network to the labels through a known graph of relationships (hierarchical and mutual exclusivity) extracted from the Knowledge Graph (Singhal, 2012). Therefore, the HEX model is also deep both in the input and output spaces. However, it has several significant differences compared to our proposal: Firstly, ADIOS does not use a set of relationships extracted from a knowledge base. Instead, the composition of the output layers is learned from the data.
3. Approximate MBC Partitioning

Let $\mathcal{L} = \{\ell_1, \ldots, \ell_m\}$ be a set of $m$ labels and $\mathcal{S} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ a training set sampled from the distribution $P$ defined on $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the space of instances and $\mathcal{Y} = 2^\mathcal{L}$ is the power set of $\mathcal{L}$. For every instance $x_i$, its corresponding label set is represented by an $m$-dimensional binary vector $y_i = (y_{i1}, \ldots, y_{im}) \in \{0, 1\}^m$ such that $y_{ij} = 1$ if class label $\ell_j$ is relevant to example $x_i$ and $y_{ij} = 0$ otherwise. We seek for a hypothesis $h : \mathcal{X} \rightarrow \mathcal{Y}$ learned from the training set $\mathcal{S}$, to predict the label set $y$ corresponding to unseen instances $x$ by exploiting the dependencies between the tags. An essential element in our proposal is the existence of a partition (or a good approximation thereof) of the set of labels $\mathcal{L}$ into an ordered set of disjoint subsets $(G_1, G_2)_{1 \leq p}$ such that every label $\ell \in G_1$ is independent of all the other labels given the labels in $G_{i-1}$. That is, the knowledge of $G_{i-1}$ renders unnecessary all the information about $\ell$ not contained in $G_{i-1}$. We call such a division a Markov Blanket Chained (MBC) partition. It generalizes the simple case where $X$ is a Markov Blanket of $Y$ (Nam et al., 2014).

An MBC may not exist in practice, sensu stricto. However, it is realistic to assume the existence of a good approximate partition of this kind. Indeed, in some application like bioinformatics, one may want to predict from proteins a heterogeneous set of labels consisting of {gene functions, symptoms, diseases}⁴. In this situation, it is common to predict symptoms from gene functions or diseases from symptoms. Here, we do not rely on an existing resource providing explicit dependencies between the labels. Instead, we restrict ourselves to a good partition of the set of labels into two disjoint subsets $G_1$ and $G_2$ ($G_1 \cup G_2 = \mathcal{L}$ and $G_1 \cap G_2 = \emptyset$) such that the labels in $G_1$ are predictive of those in $G_2$. To that end, we use the information gain as a criterion and formulate the problem as follows:

$$\arg\max_{G_1 \subseteq \mathcal{L}, |G_1| \leq k} I(G_2; G_1) = H(G_2) - H(G_2 | G_1)$$

where $H(\cdot)$ is the entropy. Since $(G_1, G_2)$ is a partition of $\mathcal{L}$, if $G_1$ is a Markov Blanket of $G_2$, then $I(G_2; G_1)$ is submodular and monotone because categories in $G_1$ are independent given those in $G_2$ (Krause & Guestrin, 2012). Therefore, the greedy algorithm would give a near optimal solution ($(1 - 1/e)$ approximation). However, if we start with $(G_2 = \mathcal{L}, G_1 = \emptyset)$, the independence assumption that guarantees the submodularity will clearly not hold. We use a backward procedure instead. We start with $(G_1 = \mathcal{L}, G_2 = \emptyset)$ and update $G_2$ at each step by adding to it the label $\ell_i \in G_1$ such that $H(G_2 \cup \{\ell_i\}) - H(G_2 \cup \{\ell_i\} | G_1 - \{\ell_i\})$ is maximal. The rationale behind this approach is that if $G_1$ is a Markov Blanket of $G_2$ and $\ell_i \in G_1$ has its Markov Blanket in $G_1 - \{\ell_i\}$, then $G_1 - \{\ell_i\}$ is a Markov Blanket of $G_2 \cup \{\ell_i\}$ as proved in Theorem 3 of (Koller & Sahami, 1995). The computation of the entropies at each step is exponential in the number of labels. To make this feasible, we use two approximations: (1) we only consider the combinations of labels represented in the training set. (2) Since $H(G_2 | G_1)$ decomposes into a sum of conditional entropies $H(\ell_i | G_1)$ with $\ell_i \in G_2$ (thanks to the independence), we restrict the conditioning set to the $k$ most correlated labels of $\ell_i$ in $G_1$. Moreover, one could also add the features $X$ in the conditioning set. However, this increases the complexity and does not improve the results. The procedure is summarized in Algorithm 1 where $\text{Corr}(\ell_i, \ell_k)$ represents the correlation between the labels $\ell_i$ and $\ell_k$. This algorithm can further be accelerated by caching intermediate results as in the lazy-greedy method for maximizing submodular functions (Minoux, 1978).

We show in the experiments that the partition found by this method serves our purpose of building an architecture where $G_1$ is predictive of $G_2$. Besides, the partitions obtained by Algorithm 1 are stable as they do not vary much with respect to the size of the data used to construct them. For example, on Delicious dataset (983 labels), we observe only a small difference between the partition obtained by using 30% of the training data compared to the one obtained using the entire dataset. Table 1 shows examples of labels from $G_1$ and $G_2$ after partitioning Delicious data.

---

⁴http://geneontology.org/
4. ADIOS Model

In this section, we present the ADIOS model that exploits an (approximate) MBC partition of the labels to learn a neural network that is deep both in the input and the output spaces. The first step towards this new model is the choice of a loss function suitable to the task of MLC. Recently, (Nam et al., 2014) has demonstrated that the Cross-Entropy (CE) should be preferred over other conventional loss functions in MLC such as the Pairwise Ranking loss when using neural networks for multilabel classification. Indeed, the CE has several desirable properties. It scales better with the number of labels, and it is easier to optimize (thanks to a better landscape). Moreover, it is consistent with both the Hamming Loss (HL) and the ranking loss. In the particular case of the ADIOS model, CE is particularly interesting as it decomposes over the subsets of the MBC partition, hence naturally suggesting the architecture of the model and its corresponding learning algorithm.

Proposition 1 Let $D = \{(x, y)\}_{1 \leq i \leq m}$ sampled from the joint distribution $(X \times Y)$, $((X, G_1), G_2)$ a Markov Blanket Chained partition of this distribution, and $G_1(y_k)$ denotes the projection of a set of labels $y_k$ to $G_1$. The CE loss of a hypothesis $h$ from a hypothesis class $H$ is given by:

$$
\mathcal{L}_{CE}^H(D) = \sum_{i=1}^m \mathcal{L}_{CE}^G(G_1(y_k), x_i) + \mathcal{L}_{CE}^G(G_2(y_k), (x_i, G_1(y_k)))
$$

Two main observations can be drawn from this proposition (see supplementary for the proof). First, it takes CE as used in (Nam et al., 2014) as a special case when $G_1 = \mathcal{L}$ and $G_2 = \emptyset$. Second, the loss function has a hierarchical structure. It has two main parts taking respectively as input, the original feature vector $x$, and the features $x_i$ combined with the label vector restricted to the subset $G_1$, $G_1(y)$. Thus, predicting the categories $G_2(y)$ for a given test instance $x$ would typically require $G_1(y)$ which is not known a priori. Therefore, it is necessary to rely on an estimation of $G_1(y)$ to predict $G_2(y)$. This implies the following question: why wouldn’t one ignore the dependence and directly use $x$ to predict $G_2(y)$? The reason is, if $G_1 \cup X$ is a Markov Blanket of $G_2$, $(G_1(y), x)$ is a richer representation that can be more discriminative when predicting labels in $G_2$. In particular, we show next that a neural network parameterization of the hypothesis $h$ combined with our proposed learning algorithm results in a model that can more accurately predict $G_2(y)$ based on an estimation of $G_1(y)$ and $X$ whenever $G_1$ carries enough information.

4.1. ADIOS DNN parameterization

We parameterize ADIOS as a deep neural network (DNN) building on the conclusions from the recent application of these models in the context of MLC. The set of parameters in the vanilla DNNs is the weight matrices transferring a given instance from the input layer to the output, potentially through several hidden layers. In contrast, ADIOS model has two distinct sets of parameters in agreement with the loss function. The first set of parameters, $\Theta_1$, operates in the input space and controls the prediction of $G_1(y)$. It consists of the weight matrices transferring the input $x$ to the set of labels $G_1$. The second set of parameters, $\Theta_2$, controls the prediction of $G_2(y)$. It contains the weight matrices going from the layer $G_1$ to $G_2$, potentially with additional matrices transferring information from the input to $G_2$. The final architecture has at least one input and two output layers as depicted in Figure 1b. Figure 1b corresponds to the following two cases: $G_1$ (left) or $G_1 \cup X$ (right) is a Markov Blanket of $G_2$. The latter, which combines the information from the labels and the original features, is the one we consider for the rest of the paper.
4.2. ADIOS Learning

For a fixed architecture according to a partition \((G_1, G_2)\) of the labels, learning the ADIOS model consists in minimizing the CE loss function defined in Proposition 1. For the neural network parameterization described previously, this loss function decomposes into two main components to which we add regularization terms for the sets of parameters \(\Theta = \{\Theta_1, \Theta_2\}\). For a given \(z = (x, y)\) we have:

\[
L^{CE}_\Theta(z) = L^{CE}_{\Theta_1}(z) + L^{CE}_{\Theta_2}(z) + \Omega(\Theta)
\]

(3)

where \(L^{CE}_{\Theta_1}\) and \(L^{CE}_{\Theta_2}\) are the CE losses calculated on the predictions of \(G_1(y)\) and \(G_2(y)\), respectively.

As mentioned previously, \(G_1(y)\) is usually not available at the test time. Hence, using \(G_1(y)\) as input to fit the parameters \(\Theta_2\) would cause a misfit between the train and test distributions. Therefore, during training, we resort to an estimate of \(G_1(y)\) output by the first part of the network. Various regularization procedures such as dropout (Srivastava et al., 2014) or batch normalization (Ioffe & Szegedy, 2015) can be applied to hidden layers either to prevent overfitting or to accelerate training. However, we found beneficial to add to each output layer a \(L_1\) penalty on the vector of its activations. This regularization constraints the network to produce sparse label vectors hence matching the sparsity of the output space in most MLC problems. Indeed for a given example, only a few labels are likely to be active at every output layer.

All the parameters of the network are jointly optimized using stochastic gradient descent. This naturally implements a sort of curriculum learning where we first learn to predict the subset of labels \(G_1\), then build on top of this knowledge and learn to predict \(G_2\) by composing the previous predictions with a high-level representation of the inputs \(x\) given by hidden layers (see Figure 1b-left). Joint optimization also provides sharing of statistical strength through multitasking. In fact, due to the depth in the output space, there are two levels of multitasking: between labels of the same group and across different groups of tags. As we show in the experiments, this results in improved performance compared to the standard MLP with a flat output layer. In fact, this multi-layer approach in the output space is akin to using recurrent networks to predict sequences of tokens, where the joint probability of the output sequence is factorized into the product of the probabilities of each token given the previous ones and the input.

5. Experiments

We conducted many experiments to verify the following claims: (1) the approximate Markov blankets, constructed as described in Section 3, are often better predictors of the rest of the labels than the original features, (2) label-based (explicit) high-level representations lead to better discrimination than hidden implicit features, and (3) ADIOS achieves the state-of-the-art results on popular and large datasets. Finally, we provide a few useful insights about the behavior of the models, deep in the output space.

5.1. Datasets

We used three readily available datasets that are popular in the multi-label community\(^2\): Delicious (text), MediaMill (video) and NUS-WIDE (images). Additionally, we preprocessed and used two other datasets of moderate and large size: image data from SUN2012 (Xiao et al., 2010) and text data from BioASQ competition of 2015\(^3\).

The SUN2012 image data was augmented to 100,000 examples using small random shifts and rotations. Using the Caffe library (Jia et al., 2014), each image was passed through a version of GoogleNet (Szegedy et al., 2015) pre-trained on ImageNet and the last layer composed of 1024 features were used as input to the ADIOS model. The labels were constructed from the object tags provided in the data.

The BioASQ data consisted of abstracts of medical articles and corresponding tags assigned by experts. We selected the 5000 most frequent tags and applied doc2vec (Le & Mikolov, 2014) to the corresponding abstracts using the gensim library\(^4\), producing 500-dimensional feature vectors per abstract, which were used as input to ADIOS. We further selected only the abstracts that were assigned at least four tags. This produced a multi-label dataset with approximately one million examples.

All the datasets were randomly split into a fixed training (60%), testing (20%) and validation sets (20%). The hyperparameters of all the models (the baselines and our proposed ADIOS model) were tuned using the validation set. Statistics of all the datasets are summarized in Table 2.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>Inputs</th>
<th>Labels</th>
<th>avg. labels per example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delicious</td>
<td>16,105</td>
<td>500</td>
<td>983</td>
<td>19.0</td>
</tr>
<tr>
<td>MediaMill</td>
<td>43,907</td>
<td>120</td>
<td>101</td>
<td>4.5</td>
</tr>
<tr>
<td>SUN2012</td>
<td>100,000</td>
<td>1024</td>
<td>2304</td>
<td>9.8</td>
</tr>
<tr>
<td>NUS-WIDE</td>
<td>269,648</td>
<td>500</td>
<td>81</td>
<td>2.4</td>
</tr>
<tr>
<td>BioASQ</td>
<td>1,181,338</td>
<td>500</td>
<td>4587</td>
<td>4.7</td>
</tr>
</tbody>
</table>

\(^2\)http://mulan.sourceforge.net\(^3\)http://bioasq.org\(^4\)http://radimrehurek.com/gensim
5.2. Experimental Setup

Baselines. In our experiments, we compared ADIOS to the following baselines:

- **Binary Relevance (BR)** (also called one-vs-all) is a set of $m$ independent logistic regression classifiers.
- **Principle Label Space Transformation (PLST)** (Tai & Lin, 2012) learns a low-dimensional representation of the labels by performing an SVD on the label matrix. Therefore, it assumes a low-rank label matrix and captures correlations between the labels.
- **Multi-Label Prediction via Compressed Sensing (MLCS)** (Hsu et al., 2009). This approach also assumes a low-rank label matrix. However, it embeds the labels in a lower dimensional space using random projections and relies on sparse recovery algorithms such as Orthogonal Matching Pursuit (OMP) or Lasso to recover the original labels from the predicted low-dimensional representation.
- **FastXML** (Prabhu & Varma, 2014) is an ensemble method that learns a set of hierarchies over the feature space to optimize the nDCG ranking metric.
- **Multi-Layer Perceptron** (MLP) trained to minimize the cross-entropy (CE) loss as in (Nam et al., 2014).

Hyper-parameters selection. For all the compared models (ADIOS and baselines), we used the validation set to select the hyper-parameters. For neural network models, we used hidden layers with 1024 ReLU units and trained the models using Adagrad with 20 to 50% dropout and batch normalization. Additionally, we used $L_2$ regularization on the weight matrices and $L1$ activity regularization on the output layers when it improved performance (typical values ranged between $10^{-4}$ to $10^{-3}$ for all the datasets).

For PLST and MLCS, we used linear regression with $L_2$ regularization to predict the low-dimensional embedding of the labels. Moreover, we fixed, the dimensionality of the embedding space to 500 on all datasets except on Delicious and NUS-WIDE where we used 50. Larger embedding dimensions did not increase performance. For MLCS, we used Lasso to recover the original labels. For FastXML, we fixed the number of trees per ensemble to 50.

Our model had a similar configuration to MLP: one hidden layer with 1024 ReLU units between inputs and $G_1$. We then added another 512-dimensional ReLU hidden layer between the hidden layer before $G_1$ and $G_2$ as well as direct connections between $G_1$ and $G_2$, as illustrated in the third diagram in Figure 1. Both hidden layers were regularized with 20-50% dropout and batch normalization. Additionally, activity regularization (with $10^{-4}$ to $10^{-3}$ coefficients) was imposed on the output layers. ADIOS models were implemented as an extension of the Keras machine learning library\(^3\).

Metrics. One of the crucial differences between the traditional multi-class (or binary) and the multi-label settings is the evaluation metrics of interest. While performance on the former task is well summarized by two principal measures, precision and recall (and their combinations, such as $F_1$), performance on the latter can be assessed in a variety of ways (Tsoumakas & Katakis, 2007). The main metrics used for comparison are Macro-F1 and Micro-F1, which are computed as follows:

- **Macro-F1** is a category-based evaluation measure. It weighs all categories equally and hence is more sensitive to the ability of the classifier to detect rare classes. Denoting the precision and recall of category $c$ by $P_c$ and $R_c$, respectively, Macro-F1 is given by:

$$P_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c}, \quad R_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c}$$

$$\text{Macro-F1} = \frac{1}{|C|} \sum_{c \in C} 2P_c \cdot R_c$$

where $\text{TP}_c$, $\text{FP}_c$ and $\text{FN}_c$ denote respectively the true-positives, false-positives, and false-negatives for the class label $c \in C$.

- **Micro-F1** is an instance-based evaluation measure and therefore weights higher categories that have higher fraction in the test set:

$$P = \frac{\sum_{c \in C} \text{TP}_c}{\sum_{c \in C} \text{TP}_c + \text{FP}_c}, \quad R = \frac{\sum_{c \in C} \text{TP}_c}{\sum_{c \in C} \text{TP}_c + \text{FN}_c}$$

$$\text{Micro-F1} = \frac{P \cdot R}{P + R}$$

- **Precision@K**: It is the fraction of correct predictions among the first K predicted labels. It measures how well an algorithm performs as a ranking model.

Prediction. Both the baselines and ADIOS, once trained, return scores for each label that need to be turned into specific predictions via thresholding. In the multilabel setting, mere 0.5 threshold is usually suboptimal regarding the metrics of interest. Therefore, we employ adaptive thresholding (Nam et al., 2014): For a given trained model (a baseline or ADIOS), for each sample in the training data, we find a threshold that maximizes $F_1$ score when applied to the model’s predictions. Next, we regress features of the training data to these optimal thresholds. Finally, when the model is tested on a new example, we obtain both

\(^3\)Keras library: [http://keras.io](http://keras.io). Our code is available at [https://github.com/alshedivat/adios](https://github.com/alshedivat/adios).
the scores (returned by the model) and the adaptive label-specific thresholds (predicted using regression)\(^6\).

### 5.3. Results

**Predictiveness of Markov blankets.** We start by validating our claim that given a good approximate MBC partition of the labels into \((G_1, G_2)\), the labels in \(G_1\) can, in fact, be better predictors of the labels in \(G_2\) than the original features. To that end, we first do an MBC partition of the set of labels into two equal subsets \(G_1\) and \(G_2\). Then, we compare an MLP trained on the original features with an MLP trained on the labels from \(G_1\) (i.e., using them as input vectors) to predict the labels in \(G_2\). In this task, we performed experiments on three of the datasets and measured macro \(F_1\) and \(P@5\) to capture the global performance and goodness of ranking.

The results are presented in Table 3. We notice that the first two datasets, Delicious, and MediaMill, Markov blankets turn out to be far more predictive than the original features regarding both global and ranking metrics.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>(X) size</th>
<th>(G_1) size</th>
<th>(maF1) on (G_2)</th>
<th>(P@5) on (G_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delicious</td>
<td>500</td>
<td>500</td>
<td>21.54</td>
<td>38.70</td>
</tr>
<tr>
<td>MediaMill</td>
<td>120</td>
<td>50</td>
<td>11.49</td>
<td>22.57</td>
</tr>
<tr>
<td>NUS-WIDE</td>
<td>500</td>
<td>40</td>
<td>11.93</td>
<td>10.64</td>
</tr>
</tbody>
</table>

\(G_1\) and \(G_2\) are better at predicting rare items in \(G_2\). At the same time, features turn out to be outperforming \(G_1\) representations concerning \(P@5\) metric, which means that they are better associated with the dominant (highly frequent) labels.

Interestingly, we observe quite an opposite behavior of predictiveness of features and \(G_1\) on the NUS-WIDE data. Features marginally outperform \(G_1\) labels in macro \(F_1\), but achieve twice lower \(P@5\) score. Such behavior is potentially the consequence of the low ratio of the number of labels to the number of features in this dataset. Therefore, the labels in \(G_1\) can capture only strong correlations while the features possess higher discriminative power over all the labels.

Nevertheless, we see that Markov blankets indeed have additional discriminative power on one or the other metric. Therefore, a clever combination of the information from features and \(G_1\) labels can help to overcome performance trade-offs. We further show that joint training of the proposed ADIOS model with the additional hidden layer can use this information and achieve state-of-the-art results regarding both global and ranking metrics.

**Predictiveness of Markov blankets.** We start by validating our claim that given a good approximate MBC partition of the labels into \((G_1, G_2)\), the labels in \(G_1\) can, in fact, be better predictors of the labels in \(G_2\) than the original features. To that end, we first do an MBC partition of the set of labels into two equal subsets \(G_1\) and \(G_2\). Then, we compare an MLP trained on the original features with an MLP trained on the labels from \(G_1\) (i.e., using them as input vectors) to predict the labels in \(G_2\). In this task, we performed experiments on three of the datasets and measured macro \(F_1\) and \(P@5\) to capture the global performance and goodness of ranking.

The results are presented in Table 3. We notice that the first two datasets, Delicious, and MediaMill, Markov blankets turn out to be far more predictive than the original features regarding both global and ranking metrics.

The results are presented in Table 3. We notice that the first two datasets, Delicious, and MediaMill, Markov blankets turn out to be far more predictive than the original features regarding both global and ranking metrics.

The main result. We now compare ADIOS to the previously described baselines on five datasets. To further vali-
ADIOS: Architectures Deep In Output Space

date the importance of learning a partition and the effectiveness of Algorithm 1, we also demonstrate the performance of an ADIOS architecture for which the partitions $(G_1, G_2)$ are created at random. We call this approach ADIOS$_{RND}$ as opposed to the model using a learned partition which we denote ADIOS$_{MBC}$. On each dataset, ADIOS$_{RND}$ has the same architecture as the best performing ADIOS$_{MBC}$ model. All the results are shown in Table 4. This table is complemented by Table 5 that contains the average ranks of every approach over the datasets on each of the five performance measures we use. The average ranks are used to decide the significance of the of the results by using a Bonferroni-Dunn test as suggested by Demšar (2006).

First, ADIOS$_{MBC}$ consistently outperforms all the compared methods on all the datasets across all the performance measures we consider in this study. Indeed, it achieves good results both on ranking metrics and classification ones while most of the other baselines are more specialized to one metric. For example, FastXML is particularly optimized for ranking (it is trained by minimizing nDCG); as a result, it achieves good $P@K$ while having low $F_1$-measures globally.

As suggested by the results of Table 3, combining the soft predictions of $G_1$ with high level hidden features to predict labels in $G_2$ boosts performance even when the original features are supposedly more predictive of labels in $G_2$. This is typically the case for the NUS-WIDE dataset where such combination leads to improved Macro $F_1$ score. Note that to achieve such improvements; it was important to learn a proper approximate MBC partition of the labels rather than randomly assign them to $G_1$ and $G_2$ layers. Indeed, on the same NUS-WIDE dataset, for example, ADIOS$_{RND}$ has substantially lower classification scores (Micro and Macro F-measures) than MLP despite using the same architecture as ADIOS$_{MBC}$.

In real-world multi-label applications, it is often the case that only a handful of target labels is of particular interest. In such case, one may use the rest of the labels as auxiliary to form an approximate Markov blanket for the target labels. We compared ADIOS$_{MBC}$ and MLP in such scenario on Delicious and MediaMill data by varying the size of $G_1$ layer and testing both models on $G_2$ labels only. The left two charts of Figure 2 demonstrate the superiority of ADIOS.

Also, as depicted in the right two plots of Figure 2, ADIOS$_{MBC}$ has a better behavior than MLP in the small sample setting. In particular on MediaMill, when we consider only 40% of the training data, the Macro $F_1$ of ADIOS is almost twice as good as MLP. This opens several perspectives regarding a better use of supervision to boost classification performance when data is scarce.

6. Perspectives

As available datasets become bigger and richer in terms of their annotations, multi-label classification becomes an important class of problems to handle with modern machine learning approaches. The advent of deep architectures has mostly focused on extracting complex representations from input data. In this paper, we presented ADIOS, an approach to better learn problems with underlying unknown structured label space. We have shown that using a factor representation of labels, determined by a learned Markov Blanket can result in improved performance with reduced complexity. While we have experimented with only 2 layers of labels, some problems can potentially benefit from an even deeper factorization. The approach could also be useful for zero-shot learning when one wants to predict $G_2$ in situations where only $G_1$ is observed during training. It also paves the way to new methods for transfer learning when the information for $G_1$ is available, but the actual task of interest is $G_2$.

Acknowledgements

M.C thanks Nicolas Usunier and Florent Perronnin for their insightful comments on the initial version of this
manuscript. Part of this work was done while M.C. and M.A. were at KAUST. We also thank LIP6-UPMC for access to its computational resources.

References


