Large-scale Multi-label Text Classification - Revisiting Neural Networks

Jinseok Nam, Jungi Kim, Iryna Gurevych and Johannes Fürnkranz
Department of Computer Science
Technische Universität Darmstadt
{nam@kdsl,kim@ukp,gurevych@ukp,juffi@ke}.informatik.tu-darmstadt.de

Abstract

Large-scale datasets with multi-labels are becoming readily available, and the demand for large-scale multi-label classification algorithm is also increasing. In this work, we investigate limitations of a neural network (NN) architecture that aims at minimizing pairwise ranking error, and propose to utilize a rather simple NN approach in large-scale multi-label text classification tasks with recently proposed learning techniques. Additionally, we present a simple threshold predictor in order to make real-valued NN outputs binary. Our experimental results show that the simple NN models equipped with recent advanced techniques such as rectified linear units, dropout, and adagrad performs as well as or even outperforms the previous state-of-the-art approaches on six large-scale textual datasets with diverse characteristics.

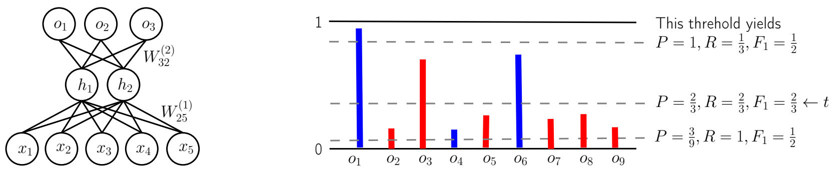
1 Introduction

As the amount of textual data on the web and digital libraries is increasing fast, the need for augmenting unstructured data with metadata is also increasing. Systematically maintaining a high quality digital library requires extracting a variety types of information from unstructured text, from trivial information such as title and author, to non-trivial information such as descriptive keywords and categories. Time- and cost-wise, manually extracting these underlying information from the fast-growing document collection is impractical.

Multi-label classification is an automatic approach for addressing such a problem, in particular assigning a set of categories from an established classification system to a given text. As large-scale datasets with multi-labels also become more readily available over the years, the demand for large-scale multi-label classification algorithm is also increasing. In the literature, one can find a number of multi-label classification approaches for a variety of tasks in different domains such as bioinformatics [1], music [33], and text [8]. By nature, multi-label classification is a harder task than single-label classification, because an instance may associate with multiple labels. Consequently, previous multi-label methods are more complex in comparison to single-label classifiers. One of the common drawbacks of the previous multi-label classification methods is that they usually do not scale up with the number of training examples and the number of distinct labels. Therefore, it is not easy, or sometimes infeasible, to adopt (the state-of-the-art) approaches to large-scale real-world dataset.

Recently, Deep neural network approaches have achieved promising performances in a variety of applications such as image processing [17, 16], speech [12], as well as many NLP tasks [30, 3]. As a result, neural network (NN) approaches are gaining great attention in the machine learning and the natural language processing community.

In this work, we propose to utilize a simple, but equipped with most recent and advanced techniques, single hidden layer NN approach in large-scale multi-label text classification tasks. The motivation


o_1
o_2
o_3
$W^{(2)}_{32}$
h_1
h_2
$W^{(1)}_{25}$
x_1
x_2
x_3
x_4
x_5
1
0
This threhold yields
$P = 1, R = \frac{1}{3}, F_1 = \frac{1}{2}$
$P = \frac{2}{3}, R = \frac{2}{3}, F_1 = \frac{2}{3} \leftarrow t$
$P = \frac{3}{9}, R = 1, F_1 = \frac{1}{2}$
o_1 o_2 o_3 o_4 o_5 o_6 o_7 o_8 o_9

(a) A neural network (b) Threshold decision

Figure 1: (a) is a diagram of a neural network that have a single hidden layer of two units. x and y are a unit in the input layer and the output layer, respectively. (b) shows how threshold for a training example is estimated based on prediction output $\mathbf{o}$ which results from trained neural network. Consider nine labels (or output units) and that o_1, o_4 and o_6 are relevant labels and the rest are irrelevant. Once training a neural network is done, we get a probability score for each label. The figure shows three exemplary threshold candidates (dashed lines), of which the middle one is the best choice that gives the highest F1 score. See Section 3 for more details.

behind exploiting such a simple approach is two folds; the first obvious one is that simple network configuration allows better scalability of the model and is more suitable for large-scale tasks. Secondly, as it has been shown in the literature [14], popular feature representation schemes for textual data such as variants of tf-idf term weighting already incorporate a certain degree of higher dimensional features, and we speculate that even a single-layer NN model can work well with text data. This paper provides an empirical evidence to support that a simple NN model equipped with recent advanced techniques for training NN performs as well as or even outperforms the previous state-of-the-art approaches on large-scale datasets with diverse characteristics.

2 Neural Networks for Multi-label Classification

Multi-label classification with large-scale textual data requires an efficient approach while preserving the similar discriminative power of multiple binary SVMs.

In this paper, we propose a neural network-based multi-label classification framework that is composed of a single hidden layer and operates with recent developments in neural network and optimization techniques that allow the model to converge into good regions of the error surface in a few steps of parameter updates. Our approach consists of two modules (Figure 1): a neural network that produces label scores and a label predictor that converts label scores into binary using threshold technique (in Section 3).

2.1 Preliminaries

Let us assume that we would like to make a prediction on L labels from D dimensional input features. Consider the neural network model with a single hidden layer in which F hidden units are defined and input units $\mathbf{x} \in \mathbb{R}^{D\times 1}$ are connected to hidden units $\mathbf{h} \in \mathbb{R}^{F\times 1}$ with weights $\mathbf{W}^{(1)} \in \mathbb{R}^{F\times D}$ and biases $\mathbf{b}^{(1)} \in \mathbb{R}^{F\times 1}$. The hidden units are connected to output units $\mathbf{o} \in \mathbb{R}^{L\times 1}$ through weights $\mathbf{W}^{(2)} \in \mathbb{R}^{L\times F}$ and biases $\mathbf{b}^{(2)} \in \mathbb{R}^{L\times 1}$. The network, then, can be written in a matrix-vector form, and we can construct a feed-forward networks $f_\Theta : \mathbf{x} \to \mathbf{o}$ as a composite of non-linear functions.

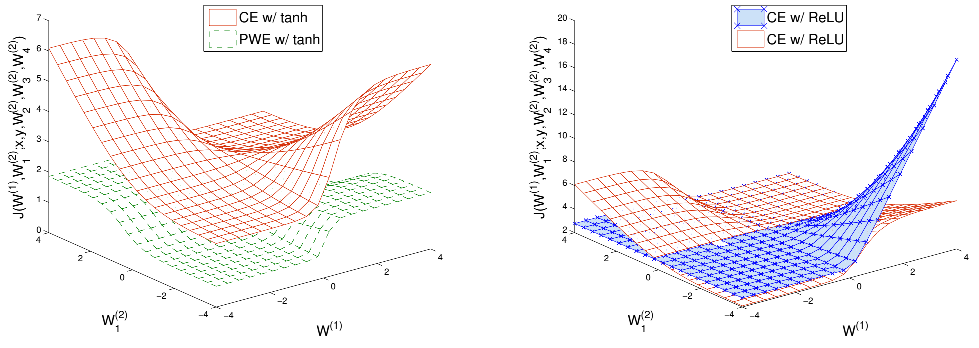
$$f_\Theta(\mathbf{x}) = f_o\left(\mathbf{W}^{(2)} f_h\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)}\right) \tag{1}$$

where $\Theta = \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}\}$, and f_o and f_h are activation functions in the output layer and the hidden layer, respectively.

Our aim is to find a parameter vector Θ that minimizes a cost function $J(\Theta; \mathbf{x}, \mathbf{y})$. The cost function measures discrepancy between predictions of the network and given targets $\mathbf{y}$. One of the commonly used cost functions for classification tasks is cross entropy error function (CE)

$$J(\Theta; \mathbf{x}, \mathbf{y}) = -\sum_l (y_l \log o_l) + (1 - y_l)\log(1 - o_l)) \tag{2}$$

where $o_l = f_\Theta(\mathbf{x})_{[l]}$ is the prediction for label l.



(a) Comparison of CE and PWE

(b) Comparison of tanh and ReLU

Figure 2: In (a), the upper curve (red, solid line) is for CE, and the bottom curve (green, dashed line) is for PWE. Both curves have similar shapes, but the curve for PWE has plateaus in which gradient descent can be very slow. In (b), two curves of CE are drawn. The x marked curve (blue) is with ReLU on the hidden layer and the other (red) is with tanh. CE with ReLU shows a very steep slope compared to CE with tanh. Such a slope can accelerate convergence speed in learning parameters using gradient descent.

As no closed-form solution exists to minimize the cost function, we use a gradient-based optimization method.

$$\Theta^{(\tau+1)} = \Theta^{(\tau)} - \eta \nabla_{\Theta^{(\tau)}} J(\Theta^{(\tau)}; \mathbf{x}, \mathbf{y}) \tag{3}$$

The parameter Θ is updated by adding a small step of negative gradients of the cost function $J(\Theta^{(\tau)}; \mathbf{x}, \mathbf{y})$ with respect to the parameter Θ at step τ. The parameter η, called the learning rate, determines the step size of updates.

2.2 Difficulty of Ranking Loss Minimization

[37] proposed a neural network framework, namely BP-MLL, which minimizes errors induced by incorrectly ordered pairs of labels, in order to exploit dependencies among labels. Instead of the cross entropy error (Equation 2) which does not take label relationship explicitly, BP-MLL introduces the loss function, we will call this as Pairwise error function (PWE) in this paper, as follows:

$$J(\Theta; \mathbf{x}, \mathbf{y}) = \frac{1}{|\mathbf{y}||\bar{\mathbf{y}}|} \sum_{(p,n)\in \mathbf{y}\times\bar{\mathbf{y}}} \exp(-(o_p - o_n)) \tag{4}$$

where p and n are positive and negative label index associated with training example $\mathbf{x}$. $\bar{\mathbf{y}}$ represents a set of negative labels and $|\cdot|$ stands for the cardinality. Thus BP-MLL is supposed to perform better than the very standard NN in multi-label problems because it takes label correlations into consideration. However, we have found that BP-MLL does not perform as expected in our preliminary experiments, particularly, on datasets in textual domain.

Plateaus To get an idea of how differently both objective functions behave as a function of parameters to be optimized, let us draw graphs containing cost function values. Consider two layers neural networks consisting of $W^{(1)} \in \mathbb{R}$ for the first layer, $\mathbf{W}^{(2)} \in \mathbb{R}^{4\times 1}$ for the second, output layer. Since we are interested in function values with respect to two parameters $W^{(1)}$ and $W_1^{(2)}$ out of 5 parameters, $\mathbf{W}_{\{2,3,4\}}^{(2)}$ set to fixed value c. In this paper we use $c = 0$. Note that it has been pointed out that the slope of the cost function as a function of the parameters plays an important role in learning parameters of neural networks [19, 10]. Figure 2 shows different shapes of the functions and slope steepness. We conjecture that these properties might explain why learning neural networks with ReLUs in the hidden layer converges faster than the other configurations, and BP-MLL performs poorly in most cases in our experiments.

Computational Expenses Another reason that NNs, specifically, BP-MLL models, have not shown good performance is that computational cost for the networks is getting more expensive as the number of labels grows. The error term $\delta_l^{(2)}$ for label l which are propagated to the hidden layer is given by the equation below.

$$\delta_l^{(2)} = \begin{cases} -\frac{1}{|\mathbf{y}||\bar{\mathbf{y}}|}\sum_{n\in\bar{\mathbf{y}}} \exp(-(o_l - o_n))f_o'(z_l^{(2)}), & \text{if } l \in \mathbf{y} \\ \frac{1}{|\mathbf{y}||\bar{\mathbf{y}}|}\sum_{p\in\mathbf{y}} \exp(-(o_p - o_l))f_o'(z_l^{(2)}), & \text{if } l \in \bar{\mathbf{y}} \end{cases} \quad (5)$$

where $z_l^{(2)}$ is the input to the output activation function for label l.

Whereas the computation of $\delta_l^{(2)} = -y_l/o_l + (1-y_l)/(1-o_l)f_o'(z_l^{(2)})$ for the CE can be performed efficiently, obtaining error terms $\delta_l^{(2)}$ for the PWE requires L times loop, that is, $O(L)$ times more expensive than one in ordinary NN utilizing the cross entropy error function. It also shows that BP-MLL has poorer scaling properties w.r.t. the number of unique labels. Also, the mediocre predictive power of BP-MLL, as shown in experimental results in Section 4, do not seem to justify its computational complexity and slow convergence speed.

2.3 Recent Advances in Neural Networks

In recent neural network literature, a number of techniques were proposed to overcome the difficulty of learning neural networks efficiently.

Activation Functions A new activation unit, which is called Rectified Linear Units (ReLU$(x) = \max(0, x)$), on the hidden layer yields better generalization performance [24, 11, 35]. The ReLU disables negative activation so that the number of parameters to be learned decreases during the training. This sparsity characteristic makes ReLUs advantageous over the traditional activation units such as *sigmoid* and *tanh* in terms of the generalization performance.

Learning Rate Adaptation Stochastic Gradient Descent (SGD) is a simple but an effective tool for minimizing the objective functions of NNs (Equation 3). When SGD is considered as an optimization tool, one of the problems is choice of the learning rate. A common approach is to estimate the learning rate which gives lower training errors on subsamples of training examples [18] and then decrease it over time. Furhermore, to accelerate learning speed of SGD, one can utilize momentum [28].

Instead of the fixed or scheduled learning rate, an adaptive learning rate method, namely Adagrad, was proposed [5]. The method determines the learning rate by keeping previous gradients to compute the learning rate for each dimension of parameters $\eta_{i,\tau} = \eta_0/\sqrt{\sum_{t=1}^{\tau} \Delta_{i,t}^2}$ where i stands for an index of each dimension of parameters and η_0 is the initial learning rate and shared by all parameters. For multi-label learning, it is often the case that while a few of labels occur frequently, the others do rarely, so that the rare ones need to be updated with larger steps in the direction of the gradient. If we use Adagrad, since the gradient of the parameter for the frequent labels will get smaller as the updates proceed, the learning rates for the frequent decrease. However, ones for the rare remain relatively large.

Regularization In principle, as a network gets larger, i.e. it has more hidden layers and hidden units, its expressive power also increases. If one is given large training examples and wishes to build a NN-based prediction system, training a larger networks will result in better performance than using a smaller one. The problem when training such a large network is that the model is more prone to getting stuck in local minima due to the huge number of parameters to learn.

Dropout [13] is a technique that makes use of a huge parameter space for improving generalization performance while preventing the trained models from getting stuck in local minima. The key idea of Dropout is to decouple hidden units that activate the same output together, by dropping out hidden units' activations randomly.

3 Thresholding

Once training of the neural network is finished, NNs can yield a probability distribution $p(\mathbf{o}|\mathbf{x})$ for a given document $\mathbf{x}$. The probability distribution can be used to rank labels, but not as the classification of labels. For transforming the ranked list of labels into a set of binary predictions, we train a multi-label threshold predictor from training data.

For each document $\mathbf{x}_m$, labels are sorted by the probabilities in descending order where l is a label index. Ideally, if NNs successfully learn a mapping function f_Θ, all correct (positive) labels will be placed on top of the sorted list and there should be large margin between the set of positive labels and the set of negative labels. Using F_1 score as a reference measure, we calculate classification performances at every pair of successive positive labels and choose a threshold value t_m that produces the best performance (Figure 1b).

Afterwards, we can train a multi-label thresholding predictor $\hat{\mathbf{t}} = T(\mathbf{x};\theta)$ to learn $\mathbf{t}$ as target values from input pattern $\mathbf{x}$. We use linear regression with $\ell 2$-regularization to learn θ

$$J(\theta) = \frac{1}{2M}\sum_{m=1}^{M}\left(T(\mathbf{x}_m;\theta) - t_i\right)^2 + \frac{\lambda}{2}\|\theta\|_2^2 \tag{6}$$

where $T(\mathbf{x}_m;\theta) = \theta^T\mathbf{x}_m$ and λ is a parameter which controls magnitude of $\ell 2$ penalty.

At test time, we easily can predict a binary output $\hat{y}_{kl}$ for label l of a test document $\mathbf{x}_k$ given label probabilities $\mathbf{o}_k$.

$$f(\hat{y}_{kl}|\mathbf{o}_k,\mathbf{x}_k) = \begin{cases} 1 & \text{if } o_{kl} > T(\mathbf{x}_k;\theta) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

4 Experiment

4.1 Evaluation Measures

Multi-label classifiers can be evaluated in two groups of measures: bipartition and ranking. Bipartition measures operate on classification results, i.e. a set of labels assigned by classifiers to each document, while ranking measures operate on the ranked list of labels. In order to evaluate the quality of a ranked list, we consider several ranking measures: Rank loss, One-Error, Coverage and Mean Average Precision (MAP) (See [29] for details). For bipartition measures, Precision, Recall, and F_1 score are conventional methods to evaluate effectiveness of information retrieval systems. There are two ways of computing such performance measures: Micro-averaged measures and Macro-averaged measures[1] (See [22] for details).

4.2 Baseline

We compared our proposed framework with binary relevance (BR) approach that uses Liblinear as a base learner and with BP-MLL which minimize ranking loss in a neural network architecture. For Liblinear, penalty parameter C was optimized in the range of $[10^{-3},\ldots,10^{3}]$ based on either average of micro- and macro-average F_1 or rankloss on validation set. The optimal number of hidden units of BP-MLL was selected among 20, 50, 100 and 500.

4.3 Dataset Description

We demonstrate our proposed approach on six datasets in text domain (Table 1).

[1]Note that scores computed by micro-averaged measures might be much higher than that by macro-averaged measures if there are many rarely-occurring labels for which the classification system does not perform well. This is because the macro-averaged measures consider each label equally, whereas the micro-averaged measures are dominated by the accuracy of frequent labels.

Table 1: Data statistics. Label cardinality is the average number of labels assigned to a document.

Dataset	# Documents	Vocabulary Size	# Unique Labels	Label Cardinality
Reuters-21578	10789	18637	90	1.13
RCV1-v2	804414	47236	103	3.24
EUR-Lex	19348	5000	3993	5.31
German Education Index	316061	20000	1000	7.16
Delicious	16105	500	983	19.02
Bookmarks	87856	2150	208	2.03

Reuters-21578 Reuters-21578 is a collection of Reuters news-wire. We use an ApteMod version [34] of the Reuters-21578 benchmark collection in which the number of training set and test set is 7,770 and 3,019 respectively. The collection consists of 90 unique labels appearing in one training and test document at least. The distribution of labels is highly skewed; the most frequent label appears about 40% in the training set, but the rare one appear in only one document.

RCV1-v2 This dataset [20] consists of over 800,000 documents, which are manually annotated from three code sets such as topics, industries, and regions. We used the pre-processed version of the dataset that is represented in tf-idf in which each document is cosine normalized, and considered only the Topic codes as target labels. The original dataset consists of 23,149 train and 781,265 test documents; we switched test and train sets to better train the models.

EUR-Lex The EUR-Lex dataset[2] is a collection of documents about European Union law. The collection contains different types of documents and a large number of labels (EUROVOC descriptors). We use 10-fold cross validation dataset[3] for a direct comparison with published results.

German Education Index The German Education Index[4] is a database of links to more than 800,000 scientific articles with metadata, e.g. title, authorship, language of an article and index terms. Most of them are in German and about 100,000 articles are in English. We consider a subset of the dataset consisting of 300,000 documents which have abstract as well as the metadata. Each document has multiple index terms which are carefully hand-labelled by human experts with respect to the content of articles. Unlike the three datasets described above, German Education Index are in raw data format. We processed plain text by removing stopwords and stemming each token. Originally a total number of distinct labels were very large, because a human annotator sometimes assigns unique key terms as labels which are important to explain the document but too specific to be used for other documents. To avoid the computational bottleneck from a large number of labels, we chose the 1,000 most common labels out of about 50,000. For experiments, we randomly split the dataset into 90% for training and 10% for test.

Delicious This dataset consists of textual information of web pages and labels extracted from a social bookmarking service *delicious*[5]. Labels are collaboratively tagged by multiple users. A document is represented with binary bag-of-word features, and on average has a relatively higher number of labels in comparison to other datasets.

Bookmarks Like delicious dataset, bookmarks dataset is also a collection of textual data in a web page and related tags assigned by users of a social bookmark system *bibsonomy*[6], and each instance is represented in a binary vector as well.

Please note that except for Delicious and Bookmarks, document representation of the rest is *tf-idf* with cosine normalization such that length of the document vector is 1 in order to account for the different document lengths.

[2]http://www.ke.tu-darmstadt.de/resources/eurlex

[3]In each fold, some of the labels in the test set does not appear in the training set. We ignore such labels when applying *bipartition* measures because they cannot be trained in a learning time.

[4]http://www.dipf.de/en/portals/portals-educational-information/german-education-index

[5]http://delicious.com

[6]http://www.bibsonomy.org/

Table 2: Evaluation on ranking and bipartition measures. In this table, AdaGrad and Dropout are abbreviated A and D following NNs as a subscript. Likewise, B and R followed by BR are to represent Bipartition and Ranking in thresholding criteria (See text for details). Note that we do not report performance scores for BPMLL on EUR-Lex because it has a lot of unique labels where training is highly expensive.

Eval. measures	Ranking				Bipartition					
	rankloss	oneError	Coverage	MAP	miP	miR	miF	maP	maR	maF
Reuters-21578										
NN_{A}	**0.0032**	0.0649	**0.7072**	0.9529	0.8896	**0.8713**	**0.8803**	**0.6433**	**0.4972**	**0.5410**
$\mathrm{NN}_{\mathrm{AD}}$	0.0037	0.0785	0.7850	0.9433	0.8874	0.8312	0.8584	0.6270	0.4541	0.5041
BPMLL	0.0062	0.0898	1.1037	0.9331	0.8399	0.8352	0.8376	0.5797	0.4331	0.4747
BR_{B}	0.0040	**0.0613**	0.8092	**0.9550**	**0.9300**	0.8096	0.8656	0.6050	0.3806	0.4455
BR_{R}	0.0040	**0.0613**	0.8092	**0.9550**	0.8982	0.8603	0.8789	0.6396	0.4744	0.5213
RCV1-v2										
NN_{A}	**0.0040**	0.0229	3.1639	0.9476	0.7854	**0.9176**	0.8464	0.6413	**0.7353**	0.6775
$\mathrm{NN}_{\mathrm{AD}}$	0.0041	**0.0216**	**3.1594**	**0.9480**	0.8607	0.8763	**0.8684**	0.7350	0.7056	**0.7118**
BPMLL	0.0067	0.0401	3.9473	0.9287	0.7181	0.9155	0.8049	0.5538	0.7381	0.6274
BR_{B}	0.0061	0.0301	3.8073	0.9375	**0.8857**	0.8232	0.8533	**0.7654**	0.6342	0.6842
BR_{R}	0.0051	0.0287	3.4998	0.9420	0.8156	0.8822	0.8476	0.6961	0.7112	0.6923
EUR-Lex										
NN_{A}	0.0195	0.2016	310.6202	0.5975	0.6346	0.4722	0.5415	0.3847	0.3115	0.3256
$\mathrm{NN}_{\mathrm{AD}}$	**0.0164**	**0.1681**	**269.4534**	**0.6433**	**0.7124**	0.4823	**0.5752**	**0.4470**	0.3427	0.3687
BPMLL	-	-	-	-	-	-	-	-	-	-
BR_{B}	0.0642	0.1918	976.2550	0.6114	0.6124	0.4945	0.5471	0.4260	**0.3643**	**0.3752**
BR_{R}	0.0204	0.2088	334.6172	0.5922	0.0329	**0.5134**	0.0619	0.2323	0.3063	0.2331
German Education Index										
NN_{A}	**0.0339**	**0.2927**	**136.1217**	**0.4899**	0.3787	**0.5010**	**0.4313**	0.3614	**0.3664**	**0.3477**
$\mathrm{NN}_{\mathrm{AD}}$	0.0370	0.2947	144.3832	0.4807	0.5061	0.3694	0.4271	**0.4794**	0.2661	0.3248
BPMLL	0.0457	0.5284	164.8777	0.3334	0.1874	0.5265	0.2764	0.1844	0.3635	0.2265
BR_{B}	0.0572	0.3052	221.0968	0.4533	**0.5141**	0.2318	0.3195	0.3913	0.1716	0.2319
BR_{R}	0.0434	0.3021	176.6349	0.4755	0.4421	0.3997	0.4199	0.4361	0.2706	0.3097
Delicious										
NN_{A}	0.0900	0.3309	**409.3301**	0.3928	**0.3902**	0.3829	0.3865	0.2428	0.1564	0.1665
$\mathrm{NN}_{\mathrm{AD}}$	**0.0884**	**0.3137**	410.7036	**0.4112**	0.3714	0.4333	**0.3999**	**0.3208**	0.1878	**0.2019**
BPMLL	0.0980	0.5482	447.6885	0.2914	0.2233	**0.5459**	0.3170	0.1324	**0.2268**	0.1509
BR_{B}	0.1184	0.4355	496.7444	0.3371	0.1752	0.2692	0.2123	0.0749	0.1336	0.0901
BR_{R}	0.1184	0.4358	496.8180	0.3371	0.2559	0.3561	0.2978	0.1000	0.1485	0.1152
Bookmarks										
NN_{A}	0.0834	0.5356	27.3684	0.4902	0.3435	0.3678	0.3553	0.2748	0.2806	0.2688
$\mathrm{NN}_{\mathrm{AD}}$	**0.0774**	**0.5062**	**25.2356**	**0.5191**	**0.4695**	0.3210	**0.3818**	**0.3740**	0.2403	**0.2802**
BPMLL	0.0823	0.6091	26.8440	0.4434	0.1070	**0.5642**	0.1799	0.1100	**0.4651**	0.1698
BR_{B}	0.0913	0.5318	29.6537	0.4868	0.2821	0.2546	0.2676	0.1950	0.1880	0.1877
BR_{R}	0.0895	0.5305	28.7233	0.4889	0.2525	0.4049	0.3110	0.2259	0.3126	0.2569

4.4 Experimental Result

We evaluate our proposed models and other baseline systems on datasets with varying statistics and characteristics, as described above (Table 2). We found that NNs outperform the baselines in most cases in terms of both ranking measures and bipartition measures.

4.4.1 Model configurations

$\mathrm{NN}_{\mathrm{AdaGrad}}$ stands for the single hidden layer neural networks which have ReLUs for its hidden layer and which are trained with SGD where each parameter of the neural networks has their own learning rate using AdaGrad. $\mathrm{NN}_{\mathrm{AdaGrad+Dropout}}$ additionally accounts Dropout based on the same settings as $\mathrm{NN}_{\mathrm{AdaGrad}}$. We used 1000 units in the hidden layer over all datasets. Additionally, $\ell 1$ regularization on hidden activations and $\ell 2$ regularization on weights were used.

BR approach using linear SVMs is an extension of (single labeled) multi-class classifiers. Positive and negative predictions are determined by hyperplanes from the learned parameter of base learners. These hyperplanes depend on the regularization parameter C. Lower C allows more training errors, but the higher one imposes more penalty on training errors. $\mathrm{BR}_{\mathrm{Bipartition}}$ refers to linear SVMs where C is optimized with bipartition measures on the validation dataset. BR models whose C are optimized on ranking measures are indicated as $\mathrm{BR}_{\mathrm{Ranking}}$.

In addition, we apply the same thresholding technique which we utilize in our NN approach (Section 3) on a ranked list produced by BR models ($\mathrm{BR}_{\mathrm{Ranking}}$).

4.4.2 Comparison of Algorithms on Ranking measures

The single layer NNs, i.e. $\mathrm{NNs}_{\mathrm{AdaGrad}}$ and $\mathrm{NNs}_{\mathrm{AdaGrad+Dropout}}$, clearly outperform BPMLL and BR on all datasets except on Reuters-21578. We observe that $\mathrm{NNs}_{\mathrm{AdaGrad+Dropout}}$ works better than non-Dropout NNs on EUR-Lex, Delicious and Bookmarks, otherwise non-Dropout NNs show better performance than Dropout NNs.

The performance difference between the two models from one dataset to another can be interpreted with the characteristics of the datasets. For example, EUR-Lex dataset has interesting characteristics such that the half of the labels in the training set do not appear in the test set, and the label space follows power-law distribution. In addition, the number of training examples are relatively small compared to the number of unique labels, which is almost 4,000. These properties make NN-based algorithms get stuck in poor local minima where generalization error may increase.

While BPMLL focuses on minimizing pairwise ranking errors (Equation 4), thereby capturing label dependency, $\mathrm{NN}_{\mathrm{AdaGrad}}$, which is simpler in terms of the objective function, works much better not only on rank loss but also on other ranking measures. BR approaches show acceptable performance on ranking measures even though label dependency was ignored during the training phase. We observe that changing penalty parameter C in Liblinear affects performance on ranking measures as expected. Generally, Liblinear with lower C as a base learner for BR gives better performance on ranking measures than one with higher C which adds more penalty on training error. If we choose C with respect to the bipartition performance, larger C is selected and $\mathrm{BR}_{\mathrm{Bipartition}}$ perform relatively poorer than $\mathrm{BR}_{\mathrm{Ranking}}$.

4.4.3 Comparison of Algorithms on Bipartition measures

For the real-world applications, it is also very important to assign a small number of labels for a document. As BR approaches train binary classifiers per label, it is straightforward to decide which label is associated with a given document at testing time. On the other hand, NN-based algorithms yield a good quality ranked list per example so that the additional thresholding step separates a ranked list of labels into groups of relevant and irrelevant labels.

$\mathrm{NN}_{\mathrm{AdaGrad}}$ and $\mathrm{NN}_{\mathrm{AdaGrad+Dropout}}$ perform as good as or better than other methods on bipartition measures as on ranking measures.

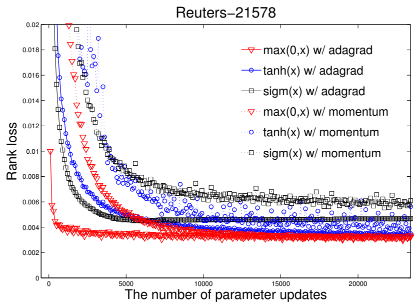
Interestingly, $\mathrm{BR}_{\mathrm{Ranking}}$ outperforms $\mathrm{BR}_{\mathrm{Bipartition}}$ in terms of micro-average F_1 and macro-average F_1. One exceptional case is EUR-Lex dataset. We conjecture that selection of C to get better performance on ranking measures allows for many irrelevant labels to be placed higher in the ranked list of labels. Specifically, one error and MAP of $\mathrm{BR}_{\mathrm{Ranking}}$ is lower than $\mathrm{BR}_{\mathrm{Bipartition}}$ because both measures considers the position of irrelevant labels in the ranked list. Furthermore, it lowers micro-average precision and results in extremely low micro-average F_1.

5 Discussion

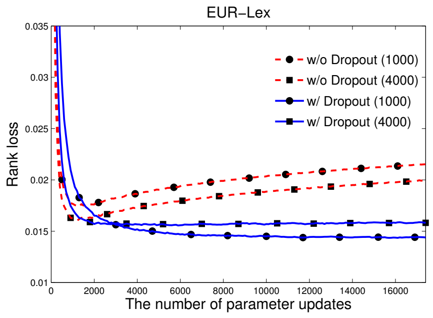
5.1 Effect of ReLUs with AdaGrad

Figure 3a shows that AdaGrad makes NNs with ReLUs at the hidden layer converge faster into a better weight space. This observation is consistent with the work by [35] in speech domain. Note that, in our preliminary experiments, we did not achieve outstanding performance from deep neural networks unlike their work where adding more hidden layers gives better performance.

One major advantage of combining recently proposed learning components such as ReLUs and AdaGrad is that learning parameters of NNs can be done quickly. Assuming that we are given a large number of features and labels in our problems of interest. As it is obviously painful to compute gradients and update parameters on such a huge search space, updating less number of parameter would be greatly helpful in large-scale applications.



(a) Type of activation units and SGD



(b) Improvement of generalization performance usnign Dropout

Figure 3: (a): Ranking loss on the test set of Reuters-21578. Combination of ReLUs and AdaGrad converges much faster than other combinations of activation units and gradient update schemes and better local minima as well. (b): Applying dropout prevents overfitting effectively

5.2 Effect of Training Neural Networks with Dropout

Dropout can address overfitting problems in neural networks with large capacity by stochastically turning off arbitrary hidden units during forward pass, then errors are propagated through only activated hidden units, that is,turned-off hidden units do not learn from the errors. This process makes it possible for each hidden unit to be activated independently from other hidden units. We observe that NNs overfit severely on EUR-Lex, due to the properties stated in Section 4.4.2. Likewise, NNs overfit on Bookmarks and Delicious but not as severely as on EUR-Lex. Figure 3b shows how Dropout affects rank loss of $\text{NN}_{\text{AdaGrad}}$ on the test examples of EUR-Lex. Without Dropout, rank loss increases gradually after few hundreds parameter updates. With Dropout, however, the score does not increase.

6 Related Work

6.1 Multilabel Classification Methods

A survey by [31] categorizes multilabel classification methods into two groups: problem transformation approaches and adaptation approaches. A problem transformation approach converts the multilabel classification task into another task, usually a set of simpler tasks, which is then tackled with an existing algorithm. An adaptation approach refers to a remodeled algorithm that directly addresses the multilabel classification task.

Problem transformation Much research effort for classification tasks in machine learning mainly focuses on single-label classification tasks. Problem transformation approaches transform a multilabel problem into a set of single label problems, and solve them using an extensively-studied single label classification algorithms, such as Support Vector Machines (SVMs) [2].

The most straightfoward transformation approach is binary relevance (BR); it constructs L binary classifiers, which are trained on L labels independently, and each classifier makes predictions for each label. However, labels often occur together, that is, the presence of a specific label may suppress or exhibit the probability of presence of other labels.

To address such a limitation of BR, pairwise decomposition (PW) and label powerset (LP) approaches consider label dependencies during the transformation by either generating pairwise subproblems [23] or the powerset of possible label combinations [32].

Memory-based learning Decision boundaries between labels can be learned by exploiting the the label space based on smoothness priors in the feature space. A well-known approach that utilizes smoothness priors is k-nearest neighbors (kNN). [36] extend kNN by maximizing posterior distribution over each label using maximum a posteriori (MAP) (MLkNN). However, MLkNN fails to consider the feasibility of computing MAP estimates on large training examples and dependencies between labels.

Modeling the Label Dependency Instead of training L independent classifiers in which label correlations are ignored, several approaches model the label dependency directly in a single learning algorithm. [6] presented a large-margin classifier RankSVMs that minimizes ranking loss by penalizing incorrectly ordered pairs of labels. [37] introduced a framework that learns ranking errors in neural networks via backpropagation (BP-MLL). While RankSVMs and BP-MLL achieve good performances, the high complexity of computing ranking loss and correcting errors hinder the two approaches from scaling to large-scale data.

Generative models also have been used to exploit the label dependencies in multilabel classification. [26] adopted Latent Dirichlet Allocation (LDA) to estimate one-to-one correspondence between topics and labels as well as word and topic distributions of multilabel documents. [27] also utilize LDA to capture the underlying structures of multilabel documents to provide additional components that account for dependencies between labels.

6.2 State-of-the-art multi-label classifiers and their limitation

The most prominent learning method for multi-label text classification is to use strong binary classifiers such as SVMs in the one-vs-all BR approach [27, 34]. It is well known that characteristics of high-dimensional and sparse data, i.e. text, make decision problems linearly separable [14], and this characteristic well suits the strengths of SVM classifiers.

Although SVMs using an SMO style convex optimization solver [25] find support vectors efficiently and show powerful performance on textual data, they are still very inefficient in dealing with large-scale data because the computational complexity is quadratic in terms of the number of training instances.[7] Unlike benchmark datasets, real-world text collections consist of a large number of training examples represented in a high dimensional space with a large amount of labels.

To handle such datasets, researchers have derived an efficient *linear* SVMs [15, 7] that can handle large-scale problems in *linear* training time. Linear solvers scale linearly in the order of M and show good performance on standard benchmarks. However, their performances decrease as the number of labels grows and the label frequency distribution gets skewed [21, 27]. In such cases, it is also intractable to employ methods that minimize ranking errors among labels [6, 37] or that learn joint probability distributions of labels [9, 4].

7 Conclusion

This paper presents a multi-label classification framework based on a neural network and a simple threshold label predictor. We have explored why BP-MLL as a multi-label text classifier does not perform well. Our experimental results showed the proposed framework is an effective method for the multi-label text classification task. Also, we have conducted extensive analysis to characterize the effectiveness of combining ReLUs with AdaGrad for fast convergence rate and utilizing Dropout to prevent overfitting which results in better generalization.

References

[1] Wei Bi and James T Kwok. Multi-label classification on tree-and dag-structured hierarchies. In *ICML*, 2011.

[2] Christopher J C Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[3] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 2008.

[4] Krzysztof Dembczy. Bayes Optimal Multilabel Classification via Probabilistic Classifier Chains. In *ICML*, pages 279–286, 2010.

[7] $O(M^2L)$ in the worst case, where M is the number of training examples

[5] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2010.

[6] A Elisseeff and J Weston. A kernel method for multi-labelled classification. In *NIPS*, 2001.

[7] Fan, Chang, Hsieh, Wang, and Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 2008.

[8] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *MLJ*, 73(2):133–153, 2008.

[9] Nadia Ghamrawi and Andrew McCallum. Collective multi-label classification. In *CIKM*, 2005.

[10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier networks. In *AISTATS*, 2011.

[12] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

[13] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[14] Thorsten Joachims. Text Categorization with Support Vector Machines : Learning with Many Relevant Features. In *ECML*, 1998.

[15] Thorsten Joachims. Training linear SVMs in linear time. In *SIGKDD*, 2006.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[17] Quoc Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg Corrado, Jeff Dean, and Andrew Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012.

[18] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: tricks of the trade*, pages 9–48. 2012.

[19] Esther Levin and Michael Fleisher. Accelerated learning in layered neural networks. *Complex systems*, 2:625–640, 1988.

[20] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *JMLR*, 5:361–397, 2004.

[21] Tie-Yan Liu, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. Support vector machines classification with a very large-scale taxonomy. *ACM SIGKDD Explorations Newsletter*, 2005.

[22] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*, volume 1. Cambridge University Press, 2008.

[23] Eneldo Loza Mencía, Sang-Hyeun Park, and Johannes Fürnkranz. Efficient voting prediction for pairwise multilabel classification. *Neurocomputing*, 73(7-9):1164–1176, 2010.

[24] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[25] John C Platt. Advances in kernel methods. chapter Fast train, pages 185–208. MIT Press, 1999.

[26] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D Manning. Labeled LDA : A supervised topic model for credit attribution in multi-labeled corpora. In *EMNLP*, 2009.

[27] Timothy N Rubin, America Chambers, Padhraic Smyth, and Mark Steyvers. Statistical Topic Models for Multi-Label Document Classification. *MLJ*, 88(1-2):157–208, 2011.

[28] David E Rumelhart, Geoffrey E Hintont, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[29] Robert E Schapire and Yoram Singer. BoosTexter : A Boosting-based System for Text Categorization. *MLJ*, 39(2):135–168, 2000. ISSN 08856125.

[30] Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. In *ACL*. 2013.

[31] G Tsoumakas and I Katakis. Multi Label Classification: An Overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

[32] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Random k-Labelsets for Multilabel Classification. 23(7):1079–1089, 2011.

[33] Konstantinos Trohidis Grigorios Tsoumakas, George Kalliris, and Ioannis Vlahavas. Multi-label classification of music into emotions. In *ISMIR*, 2008.

[34] Y Yang and S Gopal. Multilabel classification with meta-level features in a learning-to-rank framework. *MLJ*, pages 1–22, 2011.

[35] MD Zeiler, M Ranzato, R Monga, M Mao, K Yang, QV Le, P Nguyen, A Senior, V Vanhoucke, J Dean, et al. On rectified linear units for speech processing. In *ICASSP*, 2013.

[36] Min-Ling Zhang and Zhi-Hua Zhou. ML-KNN: A lazy learning approach to multi-label learning. *PR*, 40 (7):2038–2048, 2007.

[37] Min-Ling Zhang Min-Ling Zhang and Zhi-Hua Zhou Zhi-Hua Zhou. Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization. *IEEE TKDE*, 18(10):1338–1351, 2006.