Neural Network Methods for Multilabel Learning

Abhishek Kumar[1], Aditya Menon[2,*], Charles Elkan[3]
1 Google Inc., Mountain View, CA, USA. Email: abhishekkr@google.com
2 NICTA, Canberra, Australia. Email: akmenon@ucsd.edu
3 Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA, USA. Email: elkan@ucsd.edu
∗ **Email: akmenon@ucsd.edu**

Abstract

Multilabel learning is the extension of standard binary classifier learning where for a given instance, the goal is to predict a label vector that consists of multiple binary tags. Recent years have seen a panoply of different approaches to this problem, with varying degrees of simplicity, generality, and accuracy. In this paper, we propose the use of feedforward neural networks for this problem, and study their advantages over existing multilabel learning methods. The salient properties of this approach are (i) strong classification performance due to modeling of nonlinear latent structure, (ii) test-time inference that is linear in the number of tags, and (iii) relative interpretability. Compared to previous neural network methods for multilabel learning, we explain several design decisions that lead to a notable decrease in training time and increase in accuracy. Experiments show that our neural network approach outperforms existing multilabel learning methods on standard benchmark datasets and on a new dataset that we introduce here.

Introduction

In the classical supervised learning task of binary classifier training, the goal is to learn a model that, given an instance x, returns a binary prediction $y \in \{\pm 1\}$. The value y is considered to be the label of the example x, denoting whether or not it possesses some characteristic. For example, x may be an image represented by its pixel values, and y may denote whether or not the image contains a face. Multilabel learning is an extension of binary classification where the goal is to return *multiple* binary predictions, or equivalently a vector $y \in \{\pm 1\}^K$. The label y measures multiple characteristics of the example x, each of which we call a tag. For example, x may represent an image as before, and y can denote whether the faces of K specific people appear in the image.

In this paper, we are interested in neural network models for multilabel learning. We study two neural network architectures for multilabel learning that use a hidden layer. These approaches have several appealing properties. First, they learn nonlinear structure, with hidden units that capture tag correlations. Second, making predictions with a neural network is linear in the number of tags, whereas with several existing methods, it is exponential in the number of tags. Third, no tag ordering needs to be considered

when training, unlike several existing methods. Fourth, the learned hidden units may be analyzed to interpret correlations amongst features and tags. We demonstrate empirically that these neural models notably outperform major previous approaches for multilabel learning.

In recent years, a panoply of different multilabel learning approaches have been proposed, with varying degrees of simplicity, generality, and accuracy. In most cases, these methods have been compared to the simple baseline of *binary relevance*, where we train K independent binary classifiers, one for each tag. Despite being arguably the simplest possible approach to multilabel learning, binary relevance turns out to have strong empirical performance. In fact, recent work has shown that binary relevance is optimal for certain popular multilabel loss functions [1]. This stresses the importance of using it as a baseline to compare against any more complicated model. Our aim in this paper is to similarly propose that a neural network model be used as a baseline for any more complicated multilabel learning model: we will argue analytically and show empirically that such a model has numerous desirable properties for a multilabel learning approach, and systematically achieves higher accuracy than more complicated methods.

Notation

We assume that we work with instances from $\mathcal{X} \subseteq \mathbb{R}^D$, and that each instance has an associated label vector from $\mathcal{Y} = \{\pm 1\}^K$, consisting of a set of K tag values. We use y_k to denote the kth tag in the label y. For a matrix $A \in \mathbb{R}^{m \times n}$, $A^{(j)} \in \mathbb{R}^m$ denotes its jth column vector. The Frobenius norm of A is

$$||A||_F = \sqrt{\sum_{i,j} A_{ij}^2}.$$

For an integer K, we define $[K] = \{1, 2, \ldots, K\}$, and $\sigma(\cdot)$ is the sigmoid function

$$\sigma(z) = \frac{1}{1 + \exp(-z)}.$$

Materials and Methods

Ethics Statement

This research involved only public datasets, and no human or animal subjects. No IRB (institutional review board) or other approvals were needed.

Background

We now define the multilabel learning problem formally, and describe major existing methods for the problem.

The multilabel learning problem

Given a set of m training samples $\mathcal{T} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$ where $(x^{(i)}, y^{(i)}) \in \mathcal{X} \times \mathcal{Y}$, the goal of a multilabel training algorithm is to learn a classifier $f : \mathcal{X} \to \mathcal{Y}$. There are several popular loss functions used to measure the performance of a given classifier. The analogue of the zero-one loss in binary classification is the *subset zero-one loss*, which measures whether or not the predicted label vector perfectly matches the true label:

$$\ell_{0/1}(y, \hat{y}) = [y \neq \hat{y}].$$

One can relax this to minimize the number of individual tags that are incorrectly predicted, which results in the *Hamming loss*

$$\ell_h(y, \hat{y}) = \frac{1}{K} \sum_{i=1}^{K} [y_i \neq \hat{y}_i].$$

Other common loss functions for multilabel learning are presented in Table 2 and discussed below. Note that depending on the performance measure of interest, the form of the optimal classifier may vary [1].

Existing multilabel learning approaches

A straightforward solution to the multilabel learning problem is to decompose it into K separate binary classifier learning problems, one for each tag y_k. This method is known as *binary relevance*. Formally, binary relevance assumes that

$$p(y|x) = \prod_{k=1}^{K} p(y_k|x)$$

and one learns a model for each $p(y_k|x)$. That is, one assumes that all relevant information is contained in the marginal distribution for each tag. While this assumption may seem naïve, it in fact results in an optimal classifier for certain popular multilabel loss functions, such as the Hamming loss [1]. However, in practical situations where training data is limited, and for loss functions that take into account consistency of the entire tag sequence, it is intuitively beneficial to exploit correlations between tags to make better predictions. This intuition has motivated numerous multilabel learning methods, including kernel dependency estimation [2], nearest neighbor schemes [3], joint feature-tag subspace learning [4], and more; see [5–7] for detailed surveys.

Recently, [8] proposed a decomposition method called the classifier chain that is able to exploit tag correlations. As with binary relevance, the idea is to train K models, one for each tag, but in the model for tag k, we use as input features not only the data point x but also predicted values $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{k-1}$ of the $(k-1)$ tags previously modeled. The

method thus attempts to use relevant information about the previous tags to help predict the next tag.

The probabilistic classifier chain (PCC) method [1] extends the classifier chain approach in a probabilistic framework. A PCC estimates the conditional distribution $p(y|x)$ using the chain rule of probabilities:

$$p(y|x) = p(y_{\pi(1)}|x) \prod_{k=2}^{K} p(y_{\pi(k)}|x, y_{\pi(1)}, \ldots, y_{\pi(k-1)}) \tag{1}$$

where $\pi : [K] \to [K]$ is a fixed permutation of tags and $[K] = \{1, \ldots, K\}$. The PCC approach reduces learning a multilabel classifier to learning K independent probabilistic binary classifiers. These base classifiers may be, for example, logistic regression models with a feature representation ϕ and parameters W:

$$p(y_{\pi(k)} = 1|x, y_{\pi(1),\ldots,\pi(k-1)}; W) = \sigma(W^{\pi(k)} \cdot \phi(x, y_{\pi(1)}, \ldots, y_{\pi(k-1)})) \tag{2}$$

where $k \in \{2, \ldots, K\}$. Note that π must be fixed to define the ordering of the chain, with the simplest choice being the identity permutation.

Neural network methods for multilabel learning

We begin by considering binary relevance using logistic regression from a neural network viewpoint. We then describe our proposed neural network models that use hidden layers, discussing training and inference with these models. Having proposed these models, we then contrast them to other methods for multilabel learning, including previous approaches based on neural networks.

Binary relevance as a neural network

Recall that binary relevance is the multilabel learning approach where for each tag $k \in [K]$, one learns a separate model for $p(y_k|x)$. A common choice for $p(y_k|x)$ is (linear) logistic regression, which for the identity tag ordering results in the overall probability model

$$\begin{aligned} p(y|x; W) &= \prod_{k=1}^{K} p(y_k|x; W^{(k)}) \\ p(y_k = 1|x; W^{(k)}) &= \sigma(W^{(k)} \cdot x) \end{aligned} \tag{3}$$

where $W^{(k)} \in \mathbb{R}^D$ is a weight vector for the kth tag. As discussed earlier, binary relevance results in an optimal classifier for loss functions such as Hamming loss. However, for losses such as subset 0-1 loss, it is potentially suboptimal. This is intuitively clear, because the model does not attempt to exploit correlations or shared structure amongst the tags. This

motivates the question of how one can extend the binary relevance method to suit these different losses.

The approach of this paper is based on neural networks. We observe that binary relevance with logistic regression can be thought of as a neural network model with an input layer of D nodes, corresponding to the features x_d, and an output layer with K nodes, corresponding to the tags y_k. There are no hidden units, and connections from every input to output unit, as shown in Figure 1. The parameter vector $W^{(k)} \in \mathbb{R}^D$ in Equation 3 is the vector of weights entering the kth output node, which corresponds to the kth tag.

Given this interpretation, it is natural to study the effects of adding a hidden layer to the neural network. Neural networks with hidden layers have been used in a variety of domains, ranging over the years from clustering [9] to binary classification [10] to collaborative filtering [11] to text mining [12], for example. They have also been applied to multilabel learning problems [13, 14]; we discuss those approaches in detail below.

Basic Neural Network for Multi Label Learning (BN-MLL)

Building on the binary relevance model, we modify the network of Figure 1 by constructing a hidden layer comprising H hidden units $z_1, \ldots, z_H$, each of which has a real-valued unit value. We connect every input to every hidden unit, and every hidden unit to every output. Each such connection has a separate weight. The network is shown in Figure 2. Formally, the resulting model is

$$p(y|x;\alpha,\beta) = \prod_{k=1}^{K} p(y_k|x;\alpha^{(k)},\beta)$$

$$p(y_k = 1|x;\alpha^{(k)},\beta) = f\left(\alpha^{(k)} \cdot z\right) \tag{4}$$

$$z_h = g\left(\beta^{(h)} \cdot x\right) \tag{5}$$

where $\alpha^{(k)} \in \mathbb{R}^{H+1}$ is the vector of weights weight from each hidden unit to the kth output unit, $\beta^{(h)} \in \mathbb{R}^d$ is the vector of weights from each input unit to the hth hidden unit, and $f : \mathbb{R} \to [0, 1]$ and $g : \mathbb{R} \to \mathbb{R}$ are link functions to be specified. We call this model BN-MLL, an acronym for Basic Neural Network for Multi Label Learning.

The network is parameterized by $\alpha \in \mathbb{R}^{K\times(H+1)}$ and $\beta \in \mathbb{R}^{H\times(D+1)}$, with $x_0 = z_0 = 1$ to allow for intercept terms. The model parameters are learned by minimizing the regularized log-likelihood

$$\mathcal{L}(\alpha,\beta) = \frac{1}{m}\sum_{i,k} -\log p(y_k^{(i)}|x^{(i)};\alpha,\beta) + \frac{\lambda}{2}(||\alpha||_F^2 + ||\beta||_F^2) \tag{6}$$

where $\lambda \geq 0$ is the strength of regularization. When $\lambda = 0$, the optimal solution is the model that minimizes the KL divergence to the true conditional probability distribution

$p(y|x)$ [15]. The objective can also be interpreted as regularized log-loss. Log-loss belongs to the family of proper loss functions [16,17], which are those losses whose optimal solution is the model which minimizes an appropriate Bregman divergence to the true conditional probability distribution $p(y|x)$ [18].

Skip Layer Neural Network for Multi Label Learning (SLN-MLL)

The BN-MLL model can be extended by adding direct connections from the input nodes to the output nodes, as shown in Figure 3. The resulting model is

$$\begin{aligned} p(y|x;\alpha,\beta,\gamma) &= \prod_{k=1}^{K} p(y_k|x;\alpha^{(k)},\beta,\gamma^{(k)}) \\ p(y_k=1|x;\alpha^{(k)},\beta,\gamma^{(k)}) &= f\left(\alpha^{(k)}\cdot z+\gamma^{(k)}\cdot x\right) \\ z_h &= g\left(\beta^{(h)}\cdot x\right). \end{aligned}$$

Compared to the previous model, there are additional weights $\gamma \in \mathbb{R}^{K\times D}$. This network structure is sometimes called a feedforward network with a skip layer [19,20], so we call this model SLN-MLL, an acronym for Skip Layer Neural Network for Multi Label Learning. Intuitively, the weights in γ capture linear structure in the tag probabilities, while the hidden layer models nonlinear aspects of the relationship between examples and tags.

Another way to interpret the SLN-MLL model is that it augments BN-MLL with a binary relevance model, by equating $W^{(k)} \in \mathbb{R}^D$ in Equation 3 with $\gamma^{(k)} \in \mathbb{R}^D$. It has been observed that binary relevance has good empirical performance in terms of several metrics compared to other methods [8]. By making it an explicit component of the model, we hope to allow the SLN-MLL model to perform strictly better than binary relevance. The experimental results below show that this improvement indeed happens.

Inference in neural network models

Given some learned parameters θ, at test time the inference problem is to predict a label vector for a given instance $x \in \mathcal{X}$. For subset zero-one loss, the optimal prediction is

$$\hat{y}(x) = \text{argmax}_{y\in\{\pm1\}^K}\ p(y|x;\theta),$$

that is, to determine the label vector with highest predicted probability. In the neural network models, the computation of tag probability $p(y_i|x)$ is independent of the computations for other tags given weights on the hidden units. Therefore, to perform inference, we directly compute the estimated probability for each tag based on the learned weights of the hidden units. In BN-MLL for example, this is

$$\hat{p}(y_k=1|x;\hat{\alpha},\hat{\beta}) = f\left(\hat{\alpha}^{(k)}\cdot g\left(\hat{\beta}^{(h)}\cdot x\right)\right)$$

where $\hat{\alpha}, \hat{\beta}$ are parameter values learned from the training data. If desired, the predictions for each tag are thresholded to give binary outputs, and are combined to give an overall predicted label vector. Thus, inference requires $O(D \cdot H + K \cdot H)$ time, which is linear in the number of tags.

Training the neural network models

To train the neural network models, we apply standard backpropagation to compute gradients for all parameters. We use L-BFGS [21], a general purpose quasi-Newton optimizer that has been found to work slightly better than stochastic gradient descent for neural networks with a single hidden layer [22]. An additional advantage of L-BFGS over stochastic gradient descent is that the former does not require tuning a learning rate, which saves time during cross-validation.

We follow several recommendations from [23] to maximize the performance of the neural networks. In particular, we first normalize all inputs to have zero mean and unit variance. We initialize every parameter θ_{ba}, connecting node a to b in the network, as a random sample from the Gaussian distribution $\mathcal{N}(0, N_b^{-1})$ where N_b is the number of nodes that feed into node b. For the hidden layer, we use the link function

$$g(x) = A \cdot \tanh(Sx) + \epsilon \cdot x$$

where A is the amplitude of the function, S determines the slope at the origin, and ϵ is a small constant,. This is a recommended link function for neural networks [23]. We use the values $A = 1.7159$ and $S = 2/3$, which satisfy the equalities $g(1) = 1$ and $g(-1) = -1$ [24]. Training is faster when the small linear term $\epsilon = 10^{-5}$ is included in the link function. For the output layer, we use the standard sigmoid as the link function $f(\cdot)$.

With these link functions, the objective in Equation 6 becomes

$$\mathcal{L}(\alpha, \beta) = \frac{1}{m} \sum_{i,k} \log\left(1 + \exp\left(-y_k^{(i)} \cdot \alpha^{(k)} \cdot z^{(i)}\right)\right) + \frac{\lambda}{2}(||\alpha||_F^2 + ||\beta||_F^2)$$

where

$$z_h^{(i)} = 1.7159 \cdot \tanh\left(\frac{2}{3} \cdot \beta^{(h)} \cdot x^{(i)}\right) + \epsilon \cdot \beta^{(h)} \cdot x^{(i)}.$$

In SLN-MLL, the weights γ from the input to output layer are different in nature from other weights. They model the linear dependence of tags on the input feature space, while the other weights model the residuals with respect to the model provided by γ. This motivates regularizing these sets of weights differently. Empirically, separate regularization improves performance considerably.

Comparison to existing models

We compare the neural network models proposed above to existing approaches in the multilabel learning literature.

Comparison to binary relevance. Superficially, Equation 3 for binary relevance and Equation 4 for BN-MLL appear similar. Both assume that the joint distribution over tags decomposes into a number of per-tag distributions. A crucial difference between the two is that in BN-MLL the hidden units z are shared by all tags, and consequently, the parameters $\beta \in \mathbb{R}^{H\times(D+1)}$ appear in the model for each tag k. Thus, there is sharing of information amongst all these models, which is not the case in binary relevance.

We also note that BN-MLL is different from a binary relevance model with a neural network as the base classifier, which we refer to as NN-BR. In an NN-BR model, each tag has a *separate* set of hidden units. Such a model captures nonlinear relationships between the inputs and each tag, but does *not* capture any shared information amongst the tags.

Comparison to existing neural network models. The BN-MLL approach has an architecture identical to that studied in [13]. However, there are several important algorithmic differences, which turn out to make a considerable practical difference. First, we use log loss rather than a ranking loss as proposed previously. Log loss is a more tractable function to minimize, and is a convex surrogate to the multilabel 0/1 error. Second, we use L-BFGS for training, rather than stochastic gradient descent. Third, we pick the number of hidden units based on cross-validation. Fourth, we pick regularization strengths by cross-validation. The experiments described below show that these design decisions lead to improved accuracy compared to the model studied in [13], as well as compared to other existing neural network models for multilabel learning, in particular the RBF network in [14]. Moreover, we demonstrate training time on the order of minutes on a range of datasets, whereas the implementation of [13] takes on the order of hours.

The SLN-MLL architecture has to our knowledge not been studied previously for multilabel learning. We show experimentally below that it has improved performance over the BN-MLL architecture.

Comparison to probabilistic classifier chains. Recall from Equations 1 and 2 that in PCCs with logistic regression as the base model, we learn K tag models as in binary relevance, where the kth model takes values of the previous $k-1$ tags in the chain as additional input features. This is shown in Figure 4. PCCs induce a model of $p(y|x)$ that differs from binary relevance in two ways: (i) there is sharing of parameters between the models for each tag, and (ii) the model for all but the first tag in the chain is nonlinear. Consider the case $K = 2$ for simplicity. Assuming the standard tag ordering, marginalization reveals that the PCC encodes the models

$$p(y_1 = 1|x; W^{(1)}) = \sigma(W^{(1)} \cdot x)$$
$$p(y_2 = 1|x; W^{(2)}) = \sigma(W^{(2)} \cdot x)(1 - \sigma(W^{(1)} \cdot x)) + \sigma(W^{(2)} \cdot x + b^{(2)})\sigma(W^{(1)} \cdot x).$$

where $b^{(2)}$ refers to the weight corresponding to the edge from y_1 to y_2. This generalizes to more tags. The conditional $p(y_k|x)$ will involve the product of k sigmoids, with sharing of parameters from the other conditionals. Thus, a characteristic of PCC is that as we go further along the chain, we use a more powerful model for the conditional distribution.

Compared to the neural network models, PCCs induces correlations amongst and nonlinearity within the individual tag models by virtue of the chain structure of the y_k. This is similar to the motivation for adding a hidden layer to the binary relevance model. However, there are several advantages to having a hidden layer, and thus to the BN-MLL and SLN-MLL approaches:

1. Inference. In PCCs, exact inference requires exhaustive enumeration of all candidate labels, which takes 2^K time. The reason is that there are 2^K different candidate solutions. As Equation 1 shows, $p(y|x; w)$ decomposes with complex dependencies among the tags. These dependencies require iterating through all 2^K possibilities to determine the y that maximizes $p(y|x; w)$.

   In contrast, inference in BN-MLL requires $O(D \cdot H + K \cdot H)$ time and in SLN-MLL $O(D \cdot (H + K) + K \cdot H)$ time, which is linear in the number of tags. This allows the model to be applied on datasets with a large number of tags, unlike PCCs. Empirically, we find that on datasets with a moderate number of tags, inference using the neural network model is a couple of orders of magnitude faster than PCCs.

   There is a proposed greedy inference scheme for classifier chains [8], which may also be applied to PCCs. We run through the ordering $\pi(\cdot)$, and select the most probable tag value for each tag. This also has linear inference time, but we will demonstrate its empirical performance is worse than that of the neural model. A recently proposed solution based on beam search [25] provides a tradeoff between inference time and accuracy. We show in our experiments that the neural network model outperforms this method in terms of both accuracy and runtime.

2. Tag ordering. The hidden layer eliminates the need for determining a tag ordering. For a PCC, we have to pick amongst the $K!$ possible orderings to construct our chain, not all of which may perform comparably. While there have been approaches to picking such an ordering [25], they do not possess guarantees, and require the training of multiple models on the training data.

3. Interpretability. The weights associated with the hidden units encode correlations between input features and tags, which lends interpretability to the final model. In PCCs, it is non-trivial to extract such information from the learned model parameters.

Results and Discussion

Experimental design

We report experiments on the standard benchmark multilabel data sets listed in Table 1. From [26], we selected all the data sets with fewer than 100 tags and at most about 1000

input features. Each of these data sets has a pre-defined test set, so reported results are on those sets.

We also provide results on a new multilabel dataset that we call Movie. This dataset contains subtitle and genre information taken from IMDB for 1,040 movies. Most of the movies are major Hollywood releases, and about 80% of them were released between 1997 and 2006 [27,28]. We took the subtitle collection compiled by [28], which includes English subtitles for 1,184 movies, and represented each movie as a bag-of-words (unigrams).[1] To do this, we discarded words that occurred in fewer than 50 movies, about 5% of the dataset. We then represented each movie as a term-frequency vector over the remaining 4,904 terms. We paired these movies with 12 of the genre tags obtained by [28], namely family, fantasy, crime, romance, scifi, comedy, horror, adventure, thriller, mystery, drama, action. Each tag is active for at least 100 movies. We discarded movies with missing data or with no active tags, resulting in 1,040 examples.

We compare against binary relevance, a strong baseline on a number of loss functions [1,8], kernel dependency estimation (KDE) [2], and the probabilistic classifier chain (PCC) method. For PCCs, we use the formulation described in [1], with the original ordering of tags found in the data sets. We evaluate the performance of greedy inference for PCCs, and where feasible, also of exhaustive inference. We also implemented the beam search method proposed in [25], which can be seen as a middle ground between greedy and exhaustive inference. Results for these methods are based on our implementation, but agree with previously reported results. We also obtained results for BPMLL, the model used in [13], based on the original implementation provided by the authors.[2]

For all methods, we use ℓ_2 regularization. For all methods except BPMLL, we conduct 5-fold cross validation on the training set to pick the strength(s) of regularization in the range $\{2^{-12}, 2^{-11}, \ldots, 2^{12}\}$, as well as the number of hidden units $H \in \{1, 5, 10, \ldots, 75\}$. We then retrain on the entire training set with the selected parameter settings, and report performance on the test set. For BPMLL, we use the same parameter settings used by its authors.

Evaluation is based on the measures of success listed in Table 2. The experimental results in the next section answer four questions:

1. Do the hidden layer neural network methods outperform previous multilabel learning methods?

2. Do direct connections from input to output nodes improve performance?

3. Is good performance achieved with a reasonable number of hidden nodes?

[1]The subtitles of a movie are a full transcript of the dialog in the movie. More sophisticated representations are possible, but feature selection for text mining is not the focus of this paper. We will make this dataset available for other researchers to use before the current paper is published.

[2]The authors' implementation of BPMLL is available at `http://lamda.nju.edu.cn/Default.aspx?Page=code_BPMLL`. Our results for the Yeast dataset using it agree with the results published by the authors.

4. Do the weights on hidden nodes reveal meaningful structure in the data?

In a nutshell, the answer to each question is yes.

Experimental Results and Analysis

How many hidden units are needed for good performance?

Figure 5 shows the performance of the BN-MLL and SLN-MLL methods for different numbers H of hidden units, as measured on the provided test set for each standard multilabel dataset. In general 20 hidden units are sufficient to discover and benefit from rich latent structure in the data. Figure 5 shows that with the same number of units, SLN-MLL performs better than BN-MLL. The difference is greater for small values of H. As H increases, the difference in performance between SLN-MLL and BN-MLL diminishes.

Do neural networks perform better than alternative methods?

Results are shown in Tables 3 to 8. Both neural network models outperform the baseline methods on all datasets. Moreover, the improvement is seen consistently across all evaluation metrics. As mentioned earlier, binary relevance is a strong baseline that performs well for many evaluation metrics. We observe that BN-MLL and SLN-MLL outperform binary relevance on every evaluation metric for all the datasets, unlike all other methods considered. It is important to note that the results for all methods use appropriate hyperparameter values selected using cross validation as described above.

Table 9 shows the results on the new movie dataset. Binary relevance is a strong baseline on this dataset, and outperforms the PCC methods. However, again, the SLN-MLL model performs best. This finding is further evidence of its ability to learn useful nonlinear structure complementary to that learned by other methods.

Some of the differences between methods may be not statistically significant. Assessing the statistical significance of differences is difficult, as each method is evaluated on a single test set given by the predefined train-test split. The use of these predefined splits is to allow direct comparisons to previous work, the majority of which reports results on the same test sets. We are not aware of a general method to estimate significance in such a regime. However, we do note that the variation in results across multiple runs of the SLN-MLL method is much smaller than its advantage over previous methods. Since there is a reasonably large number of examples in each dataset, this suggests that the improvement observed with SLN-MLL is not due to random chance.

Does sharing of hidden units help?

A natural question is whether the improvements shown by the neural network models over baselines like binary relevance are solely due to nonlinearity. To demonstrate that this is not the sole explanation for the results, Tables 3 to 8 also provide results for binary

relevance with a neural network as the base classifier (NN-BR). Recall from Section that the difference between NN-BR and the models proposed in this paper is that with binary relevance, there is a separate set of hidden units for each tag. In the BN-MLL and SLN-MLL models, the hidden units are shared by all tags, so they may capture tag correlations. The results show the expected patterns. In particular, NN-BR outperforms BR with logistic regression as the base classifier, due to its use of nonlinearity, but NN-BR is in turn considerably outperformed by the BN-MLL and SLN-MLL models, indicating that there is value in sharing the hidden units.

Interpretation of hidden units

Figure 6 visualizes hidden units learned from the Emotions dataset. This dataset has tags that measure the emotional content of examples, such as whether a song is happy, quiet, etc. Each hidden unit is seen to capture clearly different structure, and most units are readily interpretable. In particular, the first unit appears to represent songs that have joyful emotions. The second unit appears to capture positive correlations between the tags 'quiet-still' and 'sad-lonely,' and negative correlations between these tags and the tags 'happy-pleased' and 'relaxing-calm.'

Next, consider the Enron dataset, which has a set of tags associated with email messages. These tags represent characteristics of the messages along dimensions such as the email's content, broad topic (business or personal), emotional tone, etc. Table 10 shows the tags and input features that have the highest positive correlations with four hidden units, annotated with the possible semantic structure captured by that hidden unit. Clearly the hidden units do capture meaning in the data.

Conclusions

The results above show that feedforward neural networks with a hidden layer, trained using backpropagation, have several important advantages for multilabel learning. The advantages include tractable test-time inference and removing the need for fixing an ordering of tags. Further, the hidden units capture nonlinear latent structure that both improves classification performance and allows for interpretation.

Experimental results on several datasets show that the SLN-MLL neural network method systematically outperforms major existing methods. For this reason, SLN-MLL should be used as a baseline method in future research on multilabel learning. The method is easy to implement, and code is available at the website at `https://github.com/abhishek-kumar/NNForMLL`.

Acknowledgments

References

1. Dembczyński K, Cheng W, Hüllermeier E (2010) Bayes optimal multilabel classification via probabilistic classifier chains. In: ICML.

2. Weston J, Chapelle O, Elisseeff A, Schölkopf B, Vapnik V (2002) Kernel dependency estimation. In: NIPS.

3. Zhang ML, Zhou ZH (2007) ML-KNN: A lazy learning approach to multi-label learning. Pattern Recognition 40: 2038–2048.

4. Ji S, Tang L, Yu S, Ye J (2008) Extracting shared subspace for multi-label classification. In: KDD.

5. Tsoumakas G, Katakis I (2007) Multi-label classification: An overview. International Journal of Data Warehousing and Mining 3: 1–13.

6. Tsoumakas G, Katakis I, Vlahavas IP (2010) Mining multi-label data. In: Data Mining and Knowledge Discovery Handbook, Springer. pp. 667–685.

7. Sorower MS (2010) A literature survey on algorithms for multi-label learning. Technical report, Oregon State University, Corvallis, OR, USA.

8. Read J, Pfahringer B, Holmes G, Frank E (2011) Classifier chains for multi-label classification. Machine Learning 85: 333–359.

9. Kohonen T (1982) Self-organized formation of topologically correct feature maps. Biological Cybernetics 43: 59-69.

10. Bishop CM (1995) Neural Networks for Pattern Recognition. New York, NY, USA: Oxford University Press, Inc.

11. Salakhutdinov R, Mnih A, Hinton G (2007) Restricted Boltzmann machines for collaborative filtering. In: ICML.

12. Socher R, Pennington J, Huang EH, Ng AY, Manning CD (2011) Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In: EMNLP.

13. Zhang M, Zhou Z (2006) Multilabel neural networks with applications to functional genomics and text categorization. IEEE Transactions on Knowledge and Data Engineering 18: 1338–1351.

14. Zhang ML (2009) ML-RBF: RBF neural networks for multi-label learning. Neural Processing Letters 29: 61–74.

15. White H (1982) Maximum likelihood estimation of misspecified models. Econometrica 50.

16. Buja A, Stuetzle W, Shen Y (2005) Loss functions for binary class probability estimation and classification: Structure and applications. Technical report, University of Pennsylvania.

17. Reid MD, Williamson RC (2010) Composite binary losses. Journal of Machine Learning Research 11: 2387–2422.

18. Reid MD, Williamson RC (2011) Information, divergence and risk for binary experiments. Journal of Machine Learning Research 12: 731–817.

19. Sontag ED (1992) Feedforward nets for interpolation and classification. Journal of Computer and System Sciences 45: 20 - 48.

20. Ripley BD (1994) Neural networks and related methods for classification. Journal of the Royal Statistical Society Series B (Methodological) 56: pp. 409-456.

21. Liu DC, Nocedal J (1989) On the limited memory BFGS method for large scale optimization. Mathematical Programming 45: 503–528.

22. Le QV, Ngiam J, Coates A, Lahiri A, Prochnow B, et al. (2011) On optimization methods for deep learning. In: ICML. pp. 265-272.

23. LeCun Y, Bottou L, Orr G, Muller K (1998) Efficient backprop. In: Orr G, K M, editors, Neural Networks: Tricks of the Trade. Springer.

24. LeCun Y (1989) Generalization and network design strategies. In: Pfeifer Z R annoted Schreter, Fogelman F, Steels L, editors, Connectionism in Perspective. Zurich, Switzerland: Elsevier. An extended version was published as a technical report of the University of Toronto.

25. Kumar A, Vembu S, Menon AK, Elkan C (2012) Learning and inference in probabilistic classifier chains with beam search. In: ECML/PKDD (1). pp. 665-680.

26. Tsoumakas G, Spyromitros-Xioufis E, Vilcek J, Vlahavas I (2011) Mulan: A Java library for multi-label learning. Journal of Machine Learning Research 12: 2411–2414.

27. Itamar E (1997) Using movie titles for creating statistical alignment models. Master's Thesis, Technion Israel Institute of Technology.

28. Wortman J (2010) Film classification using subtitles and automatically generated language factors. Master's Thesis, Technion Israel Institute of Technology.

Figures

Figure 1. Neural network view of the binary relevance model. x_i represents an input node and y_i represents an output node.

Figure 2. Neural network architecture for BN-MLL.

Figure 3. Neural network architecture for SLN-MLL.

Figure 4. Graphical model of the PCC. In contrast to the nodes in the neural networks presented earlier, each node above is stochastic.

Figure 5. Performance of neural network methods with various number of hidden nodes. Success is measured on the test set and H denotes the number of hidden units used for training. For each value of H, regularization strengths are selected using cross validation.

Figure 6. Correlations between the six tags of the Emotions dataset, as seen by four of the hidden nodes in a trained SLN-MLL model. Each subfigure shows, for one hidden unit h, the uncentered covariance matrix $\alpha_h {\alpha_h}^T$ of the vector α_h of weights from h to the output nodes (one for each tag). Each entry of this matrix measures the affinity between a pair of tags as measured by h. Green indicates positive, red indicates negative, and white no correlation. Each hidden unit captures a different structure, most of which are readily interpretable.

Tables

Table 1. Details of benchmark multilabel data sets [26].

Dataset	m_{train}	m_{test}	d	K
Emotions	391	202	72	6
Scene	1211	1196	294	6
Yeast	1500	917	103	14
Enron	1123	579	1001	53

For each dataset, m is the number of examples, d the number of features, and K the number of tags.

Table 2. Measures used for evaluation.

Measure	**Definition**
Subset 0/1 loss	$\ell_{0/1}(y, \hat{y}) = [y \neq \hat{y}]$
Hamming loss	$\ell_h(y, \hat{y}) = \frac{1}{K} \sum_{i=1}^{K} [y_i \neq \hat{y}_i]$, and
Ranking loss	$\ell_r(y, \hat{y}) = \sum_{(i,j):y_i > y_j} \left([s_i < s_j] + \frac{1}{2}[s_i = s_j] \right)$
Normalized ranking loss	$\ell_r(y, \hat{y}) = \frac{1}{(\sum_{k=1}^{K} y_k) \cdot (K - \sum_{k=1}^{K} y_k)} \sum_{(i,j):y_i > y_j} ([s_i < s_j])$
One error	$\ell_{\text{One}}(y, \hat{y}) = 1 - [y_{k^*}]$, $k^* = \text{argmax}_k \hat{y}_k$
Average precision	$\ell_{\text{Prec}}(y, \hat{y}) = \frac{1}{\sum_{k=1}^{K} y_k} \cdot \sum_{k,k'=1}^{K} \frac{y_k}{\text{rank}(k)} \cdot [\text{rank}(k') \leq \text{rank}(k)]$

These measures capture various desirable properties in multilabel classifier. The target and predicted labels (sets of tags) are denoted by y and $\hat{y}$ respectively, and s_i is the score assigned by a classifier to tag i. $\hat{y}$ is obtained by thresholding probabilistic scores at 0.5.

Table 3. Subset 0/1 Loss.

	BR	BRNN	KDE	PCC_G	PCC_{15}	PCC_E	BN-MLL	SLN-MLL	BPMLL
Emotions	0.7921	0.7277	0.7822	0.7475	0.6832	0.6832	0.6782	**0.6683**	0.9158
Yeast	0.8462	0.8430	0.8397	0.7895	**0.7634**	**0.7634**	0.8288	0.8277	0.8441
Scene	0.5309	0.4323	0.6204	0.4022	0.3863	0.3903	0.3712	**0.3654**	0.8469
Enron	0.8774	0.8687	0.9016	0.8670	**0.8497**	N/A	0.8618	0.8618	0.9989

Table 4. Hamming Loss.

Emotions	0.2261	0.2030	0.2236	0.2368	0.2261	0.2261	0.1914	**0.1897**	0.3292
Yeast	0.1989	0.2026	0.1984	0.2131	0.2092	0.2092	0.1988	**0.1983**	0.2089
Scene	0.1086	0.0977	0.1204	0.1145	0.1113	0.1013	0.0877	**0.0853**	0.2957
Enron	0.0463	0.0464	0.0465	0.0465	**0.0462**	N/A	**0.0462**	**0.0462**	0.1107

Table 5. Ranking Loss.

Emotions	1.2822	1.1089	1.4307	1.3911	1.3218	1.3218	1.0841	**1.0495**	3.3391
Yeast	6.4209	6.5115	6.4384	7.7121	7.5071	7.5070	6.3740	**6.3740**	6.5081
Scene	0.4548	0.4540	0.5084	0.6120	0.5978	0.5578	0.3905	**0.3871**	2.5137
Enron	**12.938**	13.1746	14.7219	13.0743	13.0846	N/A	13.0936	13.0356	15.9154

Table 6. Normalized Ranking Loss.

Emotions	0.1763	0.1546	0.1950	0.1932	0.1805	0.1828	0.1531	**0.1467**	0.4421
Yeast	0.1732	0.1752	0.1738	0.2015	0.1976	0.1989	**0.1709**	0.1720	0.1746
Scene	0.0856	0.0846	0.0955	0.1124	0.0939	0.1127	**0.0725**	0.0727	0.4808
Enron	**0.0729**	0.0730	0.0808	0.0736	0.0736	N/A	0.0731	**0.0729**	0.0876

Table 7. One Error.

Emotions	0.3020	**0.2525**	0.3465	0.3812	0.3267	0.3564	0.2574	0.2624	0.5247
Yeast	0.2290	0.2410	0.2366	0.2497	0.2661	0.2574	0.2366	**0.2245**	0.2366
Scene	0.2592	0.2383	0.2784	0.2684	0.2660	0.2860	0.2124	**0.2115**	0.4214
Enron	**0.2263**	0.2269	0.2349	0.2263	0.2264	N/A	0.2321	0.2322	0.3212

Table 8. Average Precision.

Emotions	0.7847	0.8165	0.7655	0.7549	0.7817	0.7691	0.8140	**0.8172**	0.5799
Yeast	0.7576	0.7550	0.7553	0.7252	0.7307	0.7261	**0.7605**	0.7599	0.7506
Scene	0.8458	0.8550	0.8330	0.8293	0.8400	0.8228	0.8719	**0.8721**	0.5541
Enron	0.7081	0.7084	0.7004	0.7079	0.7081	N/A	**0.7096**	0.7087	0.6350

Table 9. Results on the new movie dataset.

Loss	BR	BRNN	KDE
Hamming Loss	0.1502±0.0063	0.1498±0.0067	0.1555±0.0061
Subset 0/1 Loss	0.8260±0.0144	0.8260±0.0086	0.8500±0.0153
Ranking Loss	2.7452±0.2663	2.7636±0.1119	2.7788±0.2798
Normalized Rank	0.1075±0.0096	0.1077±0.0102	0.1089±0.0099
One Error	0.1808±0.0286	0.1801±0.0255	0.1856±0.0299
Average Precision	0.8092±0.0158	0.8130±0.0178	0.8064±0.0156

	Greedy PCC	PCC_{15}	BN-MLL	SLN-MLL	BPMLL
Hamming Loss	0.1910±0.0119	0.1899±0.0079	0.1889±0.0045	**0.1493±0.0057**	0.6456±0.0311
Subset 0/1 Loss	0.8808±0.0232	0.8439±0.0286	0.8837±0.0271	**0.8250±0.0189**	0.9846±0.0097
Ranking Loss	4.3856±0.3558	4.3938±0.2967	4.2673±0.1487	**2.6808±0.2608**	10.9913±0.5010
Normalized Rank	0.1803±0.0182	0.1803±0.0275	0.1756±0.0102	**0.1054±0.0092**	0.8198±0.0192
One Error	0.2654±0.0373	0.2433±0.0439	0.2558±0.0317	**0.1769±0.0297**	0.8971±0.0848
Average Precision	0.7311±0.0211	0.7397±0.0302	0.7370±0.0183	**0.8129±0.0133**	0.2560±0.0139

Since this is a new dataset with no predefined training/testing split, we report the mean and standard deviation for each success measure using 5-fold cross validation. The best result for each measure is shown in boldface, while results that are worse than that of the binary relevance method are shown in a small typeface.

Table 10. Top tags and input features (words) for four hidden units, while training on the Enron dataset.

Top Tags	**Top Words**	**Semantic Topic**
shame, internal operations, anger, Fwd, dislike/scorn	legislature, association, democrats, settlement, plants, situation, school	PUBLIC PRESSURE
talking points, internal projects, URL, hope/anticipation	coal, government, contracts, feel, bob	MEETING/PLANNING
trip report, secrecy, news article, humour, sympathy/support	yesterday, david, structure, association	BUSINESS TRIP
press release, newsletter, admiration, sadness, hope	biomass, department, meeting, megawatt, california	CALIFORNIA ENERGY CRISIS

Each row represents a single hidden unit, and is annotated by hand with a semantic topic.