DEEP BELIEF NETS FOR NATURAL LANGUAGE CALL–ROUTING

Ruhi Sarikaya, Geoffrey E. Hinton, Bhuvana Ramabhadran

IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
{sarikaya,bhuvana}@us.ibm.com

Department of Computer Science
University of Toronto, Toronto, Canada
hinton@cs.toronto.edu

ABSTRACT

This paper considers application of Deep Belief Nets (DBNs) to natural language call routing. DBNs have been successfully applied to a number of tasks, including image, audio and speech classification, thanks to the recent discovery of an efficient learning technique. DBNs learn a multi-layer generative model from unlabeled data and the features discovered by this model are then used to initialize a feed-forward neural network which is fine-tuned with backpropagation. We compare a DBN-initialized neural network to three widely used text classification algorithms; Support Vector machines (SVM), Boosting and Maximum Entropy (MaxEnt). The DBN-based model gives a call–routing classification accuracy that is equal to the best of the other models even though it currently uses an impoverished representation of the input.

Index Terms— Call–Routing, Deep Learning, DBN, RBM.

1. INTRODUCTION

Natural language call–routing is a major speech application that has found widespread use. Almost all customer contact centers for large companies use such an automated system. The task in call–routing is to understand the speaker's request and take the appropriate action. Typically, call–routing requires two statistical models. The first performs speech transcription, which maps what the caller says to text. The second is the Action Classification (AC) model that maps the transcription generated by the speech recognizer to the call-types that are used to determine the appropriate action. Machine learning techniques are quite successful in text classification when provided with sufficient labeled data, but as the complexity of the task increases the amount of training data required for reasonable performance can become large. This increases the cost and time taken to deploy the natural language understanding system. To facilitate rapid deployment we explore a new way of learning a multilayer neural network that discovers good features by fitting a generative model to unlabeled data and therefore requires much less labeled data to achieve good performance.

Recently, there has been increasing interest in deep belief networks (DBNs) because of the invention of an efficient layer-by-layer learning technique. The building block of a DBN is a probabilistic model called a Restricted Boltzmann Machine (RBM), which is used to discover one layer of features at a time. To learn a DBN, RBMs are applied recursively with the feature activations produced by one RBM acting as the data for training the next RBM in the stack. DBNs have been used as generative models of many different forms of data in such diverse areas as image classification, speech recognition and information retrieval [8, 11, 9]. Deep networks typically have higher modeling capacity than shallow networks with the same number of parameters, but they are harder to train, both as stochastic top-down generative models and as deterministic bottom-up discriminative models. For generative training, it is usually very difficult to infer the posterior distribution over the multiple layers of latent (hidden) variables, and for discriminitive training using back-propagation, learning can be very slow with multiple hidden layers and overfitting can also be a serious problem. The recursive training method for DBNs solves the inference problem, and the use of features found by the DBN to initialize a multilayer, feed-forward neural network significantly decreases both the time taken for discriminitive training and the amount of overfitting [3].

An RBM is a generative model that can learn stochastic binary features which are good for modeling the higher-order statistical structure of a dataset. Even though these features are discovered without considering the discriminative task for which they will be used, some of them are typically very useful for classification as well as for generation. A subsequent stage of discriminative fine-tuning can then slightly change the features to make them even more useful for discrimination with much less overfitting than occurs with purely discriminative training. This is particularly helpful when the number of labeled training examples is relatively small. In this regime, it has been shown that classifiers based on generative models can outperform discriminative classifiers, even without making use of additional unlabeled data [6].

This paper is organized as follows: Section 2 introduces RBMs. Section 3 describes how to use train a stack of RBMs recursively and how to use the resulting DBN to initialize a feed-forward neural network that can be discriminatively fine-tuned to optimize classification. Section 4 summarizes the other widely used discriminative classifiers. Section 5 presents the experimental results and discussion followed by the conclusions and future work in Section 6.

2. RESTRICTED BOLTZMANN MACHINES

A restricted Boltzmann machine [2] is a two-layer undirected bipartite graphical model where the first layer consists of observed data variables (or visible units), and the second layer consists of latent variables (or hidden units). The visible and hidden layers are fully inter-connected via connections with symmetric undirected weights, but there are no intra-layer connections within either the visible or the hidden layer. A typical RBM model topology is shown in Fig. 1.

The weights and biases of an RBM determine the energy of a

joint configuration of the hidden and visible units $E(v, h)$,

$$E(v, h; \theta) = -\sum_{i=1}^{V}\sum_{j=1}^{H} v_i h_j w_{ij} - \sum_{i=1}^{V} b_i v_i - \sum_{j=1}^{H} a_j h_i \quad (1)$$

with model parameters $\theta = \{W, b, a\}$ and $v_i, h_j \in \{0, 1\}$. W are the symmetric weight parameters with $V \times H$ dimensions, b are the visible unit bias parameters, a are the hidden unit bias parameters. The network assigns a probability to every possible visible-hidden vector pair via the the energy function,

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})} \quad (2)$$

The normalization term or partition function, Z, is obtained by summing over all possible pairs of visible and hidden vectors.

$$Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \quad (3)$$

The probability that the model assigns to a visible vector, $\mathbf{v}$, is obtained by marginalizing over the space of hidden vectors,

$$p(\mathbf{v}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \quad (4)$$

The simplest RBMs use Bernouilli-distributed units (*i. e.* stochastic binary units), but they can be generalized to any distribution in the exponential family [9]. However, some combinations of distributions for the visible and hidden units are very hard to train (see [1] for more details). In this paper, we restrict ourselves to binary units for all of the experiments, but in the final discussion we describe a type of visible unit that might be more appropriate for modeling word-count data.

The derivative of the log probability of a visible vector with respect to the weights is given by:

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} \quad (5)$$

where the angle bracket denotes the expectation with respect to the distribution specified in the subscript. Following the gradient of the log likelihood we obtain the update rule for the weights as,

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}) \quad (6)$$

where ϵ is the learning rate. The lack of hidden–hidden connections makes the first expectation easy to compute. Given a visible vector, $\mathbf{v}$, the hidden units are conditionally independent and the conditional distribution of hidden unit j is given by:

$$p(h_j = 1 \mid \mathbf{v}) = \sigma(a_j + \sum_i v_i w_{ij}) \quad (7)$$

where σ is the logistic sigmoid function $\sigma(x) = 1/(1 + \exp(x))$. It is therefore easy to get an unbised sample of $\langle v_i h_j \rangle_{data}$. Similarly, because there are no visible–visible connections, we can easily get an unbiased sample of the state of a visible unit, i, given a hidden vector, $\mathbf{h}$:

$$p(v_i = 1 \mid \mathbf{h}) = \sigma(b_i + \sum_j h_j w_{ij}) \quad (8)$$

Unfortunately, it is exponentially expensive to compute $\langle v_i h_j \rangle_{model}$, exactly so the contrastive divergence (CD) approximation to the gradient is used by replacing $\langle v_i h_j \rangle_{model}$ with $\langle v_i h_j \rangle_{recon}$, which is a lot easier and faster to compute [10]. $\langle v_i h_j \rangle_{recon}$ is computed by setting the visible units to a random training vector. Then the binary states of the hidden units are computed using Eqn. 7, followed by computing the binary states of the visible units using Eqn. 8. The computed visible states are a 'reconstruction' of the original visible vector. The new learning rule is a crude approximation to following the gradient of the log probability of the training data, but it works well in practice and is adequate for discovering good features.

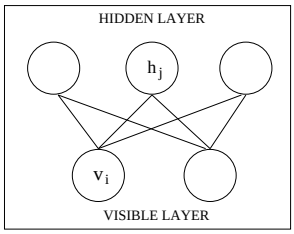

Figure 1: RBM architecture.

3. LEARNING AND USING DEEP BELIEF NETS

After RBM_1 has been trained on the data, its learned parameters, θ_1, define $p(\mathbf{v}, \mathbf{h}|\theta_1)$, $p(\mathbf{v}|\theta_1)$, $p(\mathbf{v}|\mathbf{h}, \theta_1)$, and $p(\mathbf{h}|\mathbf{v}, \theta_1)$ via the Eqns. 7 and 8. The parameters of RBM_1 also define a prior distribution over hidden vectors, $p(\mathbf{h}|\theta_1)$, which is obtained by marginalizing over the space of visible vectors. This allows $p(\mathbf{v}|\theta_1)$ to be written as:

$$p(\mathbf{v}|\theta_1) = \sum_{\mathbf{h}} p(\mathbf{h}|\theta_1) p(\mathbf{v}|\mathbf{h}, \theta_1) \quad (9)$$

The idea behind training a DBN by training a stack of RBMs is to keep the $p(\mathbf{v}|\mathbf{h}, \theta_1)$ defined by RBM_1, but to improve $p(\mathbf{v})$ by replacing $p(\mathbf{h}|\theta_1)$ by a better prior over the hidden vectors. To improve $p(\mathbf{v})$, this better prior must have a smaller KL divergence than $p(\mathbf{h}|\theta_1)$ from the "aggregated posterior" which is the equally weighted mixture of the posterior distributions over the hidden vectors of RBM_1 on all N of the training cases:

$$\frac{1}{N} \sum_{\mathbf{v} \in \text{train}} p(\mathbf{h}|\mathbf{v}, \theta_1) \quad (10)$$

The analagous statement for Gaussian mixture models is that the updated mixing proportion of a component should be closer to the average posterior probability of that component over all training cases.

Now consider training RBM_2 by using samples from the aggregated posterior of RBM_1 as training data. It is easy to ensure that the distribution which RBM_2 defines over its visible units is identical to $p(\mathbf{h}|\theta_1)$: we simply initialize RBM_2 to be an upside-down version of RBM_1 in which the roles of visible and hidden units have been swapped. So RBM_2 has $\mathbf{h}$ as a visible vector and $\mathbf{h_2}$ as a hidden vector. Then we train RBM_2 which makes $p(\mathbf{h}|\theta_2)$ be a better model of the aggregated posterior than $p(\mathbf{h}|\theta_1)$.

After training RBM_2, we can combine the two RBMs to create a hybrid of a directed and an undirected model. $p(\mathbf{h}|\theta_2)$ is defined by the undirected RBM_2, but $p(\mathbf{v}|\mathbf{h}, \theta_1)$ is defined by directed connections from the first hidden layer to the visible units. In this hybrid model, which we call a deep belief net, exact inference of $p(\mathbf{h}|\mathbf{v}, \theta_1, \theta_2)$ is no longer easy because the prior over the hidden vectors is no longer defined by θ_1. However, it is proved in [7] that if we perform approximate inference for the first hidden layer by using 7, there is a variational lower bound on the log probability of

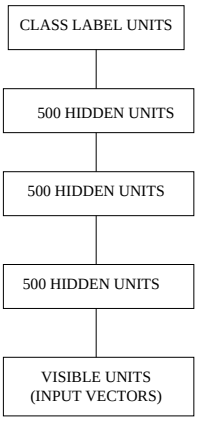

Figure 2: Stacking RBMs to create a deep network. This architecture is used in our experiments.

the training data that is improved every time we add another layer to the DBN, provided we add it in the appropriate way.

After training a stack of RBMs, the bottom up recognition weights of the resulting DBN can be used to initialize the weights of a multi-layer feed-forward neural network, which can then be discriminatively fine-tuned by backpropagating error derivatives. The feed-forward network is given a final "softmax" layer that computes a probability distribution over class labels and the derivative of the log probability of the correct class is backpropagated to train the incoming weights of the final layer and to discriminatively fine-tune the weights in all lower layers.

Deep belief networks (DBNs) have yielded impressive classification performance on several benchmark classification tasks, beating the state-of-the-art in several cases [11]. In principal, adding more layers improves modeling power, unless the DBN already perfectly models the data. In practice, however, little is gained by using more than about 3 hidden layers. We use the architecture shown in Fig. 2. It has three hidden layers that are pre-trained, one at a time, as the hidden layers in a stack of three RBMs without making any use of the class labels.

4. TRADITIONAL CLASSIFIERS

4.1. Maximum Entropy

The MaxEnt method is a flexible statistical modeling framework that has been used in widely in many areas of natural language processing [14]. The MaxEnt allows the combination of multiple overlapping information sources [15, 14]. The information sources are combined as follows:

$$P(C|W) = \frac{e^{\sum_i \lambda_i f_i(C,W)}}{\sum_{C'} e^{\sum_j \lambda_j f_j(C',W)}}, \qquad (11)$$

which describes the probability of a particular class C (e.g. action class) given the word sequence W spoken by the caller. Notice that the denominator includes a sum over all classes C', which is essentially a normalization factor for probabilities to sum to 1. The f_i are indicator functions, or *features*, which are "activated" based on computable features on the word sequence, for example if a particular word or word pair appears, or if the parse tree contain a particular tag, etc. The MaxEnt models are trained using the improved iterative scaling algorithm [15] with Gaussian prior smoothing [14] using a single universal variance parameter of 2.0.

4.2. Boosting

Boosting is an iterative method for improving the accuracy of any given learning algorithm. The premise of Boosting is to produce a very accurate prediction rule by combining moderately inaccurate (weak) rules. The algorithm operates by learning a weak rule at each iteration so as to minimize the training error rate. A specific implementation of the Boosting is AdaBoost is described in [4]. Boostong has been applied to a number of natural language processing tasks in the past.

4.3. Support Vector Machines

SVMs are derived from the theory of structural risk minimization [13]. SVMs learn the boundaries between samples of the two classes by mapping these sample points into a higher dimensional space. In the high dimensional space a hyperplane separating these regions is found by maximizing the margin between closest sample points belonging to competing classes. Much of the flexibility and classification power of SVM's resides in the choice of kernel. Some of the commonly used kernels are linear, polynomial and radial basis functions. In this work, we chose linear kernels to train the SVM since computationally it is faster compared to other kernels yet there is no significant difference in performance for the current task.

5. EXPERIMENTAL RESULTS AND DISCUSSION

The call–routing task we study in this paper is from a call–center customer hotline that gives technical assistance for a Fortune–500 company [16]. The call–routing system selects one of the 35 call–types. The training data has 27K automatically transcribed utterances amounting to 178K words. This data is split into {1K, 2K, 3K, 4K, 5K, 6K, 7K, 8K, 9K, 10K} and 27K sets. The purpose of this split is to investigate various training data sizes and their effects on the learning methods. We also have two separate datasets containing about 3.2K and 5.6K sentences used as development and test data, respectively. All of these datasets are hand–labeled with call–types. In all the classification methods employed here we used vectors of individual word counts as the inputs to the models. For the DBNs, the counts were clipped at 1 to allow them to be modeled by binary units.

In our experiments with the development data we found that hidden layers of 500 $\rightarrow$ 500 $\rightarrow$ 500 provided slightly better results than the other hidden layer sizes that we tried. The model architecture is shown in Fig. 2. The individual RBM models were trained in an unsupervised fashion using contrastive divergence learning with 100 passes (epochs) through the training dataset. The weights of each RBM were were initialized with small random values sampled from a zero-mean normal distribution with standard deviation 0.01 and updated using a learning rate of 0.01/batch-size, momentum of 0.9, and a weight decay of 0.001.

For the discriminative fine-tuning, we use stochastic gradient descent (SGD) and we also set the number of iterations through the training data to 100. This number was determined by using early stopping according to the validation set classification error. To reduce computation time, we select the SGD learning rate, momentum parameter and other parameters by maximizing the AC accuracy on a separate development set.

In Table 1, we present the results on the test data for SVMs, MaxEnt, Boosting and DBNs. Various classifier parameters (e.g.

Action Classification Accuracy (%)				
Labeled Data	MaxEnt	SVM	Boosting	DBN
1K	76.0	77.8	79.6	78.1
2K	80.4	82.2	83.6	82.6
3K	82.2	84.3	85.1	84.4
4K	83.5	85.3	84.6	85.5
5K	84.6	86.2	85.9	86.2
6K	85.5	87.0	86.3	87.0
7K	86.2	87.7	86.3	87.8
8K	86.5	88.0	87.2	88.0
9K	87.2	88.5	87.5	88.7
10K	87.6	88.5	87.7	88.7
27K	89.7	90.3	88.1	90.3

Table 1: PACKAGE SHIPMENT TASK: AC accuracy for various classifiers.

smoothing priors for MaxEnt learning, and kernel selection for SVMs) are tuned on the development data. Each classifier is trained using the amount of labeled data given in the first column. Looking first at the traditional classifiers, we notice that the SVM classifier obtained 77.8% accuracy using 1K labeled data. The corresponding figures for the MaxEnt classifier and the Boosting based classifier are 76.0% and 79.6% respectively. Not only for 1K labeled data but also for 2K and 3K data, Boosting provides the best performance. However, for larger amounts of training data, the SVM consistently outperformed both MaxEnt and Boosting. DBNs performed as well as or slightly better than SVMs for all sizes of training set. When trained on all of the training data, they had identical performance achieving 90.3% accuracy.

Our current implementation of the first RBM in the stack uses as many binary visible units as the number of distinct words in the training set, and it treats all word counts greater than 1 as if they were 1. All the other methods make use of word counts greater than 1. Word multiplicity affects about 16% of the training utterances and 17% of the test utterances.

6. CONCLUSIONS AND FUTURE WORK

We successfully applied Deep Belief Nets (DBNs) to a natural language call–routing task. DBNs use unsupervised learning to discover multiple layers of features that are then used in a feed-forward neural network and fine-tuned to optimize discrimination. Unsupervised feature discovery makes DBNs far less prone to overfitting than feedforward neural networks initialized with random weights and it also makes it easier to train neural networks with many hidden layers.

DBNs produce better classification results than several other widely used learning techniques, outperforming Maximum Entropy and Boosting based classifiers. Their performance is almost identical to SVMs which are the best of the other techniques that we investigated.

There are several extensions to our model that we are currently pursuing to improve its performance. One of them is to implement "replicated softmax" visible units in the first RBM [12]. These have already been shown to give excellent generative models of word-count data and they have a natural way of dealing with the word multiplicity issue. Since they capture more information about the input data, they stand a good chance of improving performance. Another possibility is to concatenate the top layer features learned by the DBN with the original inputs and use an SVM on the concatenated vectors. Finally, we plan to investigate how much the DBN is helped by having additional unlabeled training data that is not included in the labeled training set.

7. REFERENCES

[1] G. E. Hinton, A Practical Guide Training Restricted Boltzmann Machines, *University of Toronto Machine Learning Technical Report*, UTML TR 2010–003.

[2] G. E. Hinton, Training products of experts by minimizing contrastive divergence, *Neural Computation*, vol. 14, pp. 1771–1800, 2002.

[3] D. Erhan, Y. Bengio, A. Courville, P. Manzagol and P. Vincent, Why Does Unsupervised Pre-training Help Deep Learning?, *Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[4] R. E. Schapire and Y. Singer, Boostexter: A boosting based system for text categorization, *Machine Learning*, vol. 39, no. 2/3, pp. 135–168, 2000.

[5] G. E. Hinton, and R. Salakhutdinov,"Reducing the dimensionality of data with neural networks," *Science*, vol. 313. no. 5786, 2006, pp. 504–507.

[6] A. Y. Ng and M. I. Jordan, On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes, *Advances in Neural Information Processing Systems*, vol. 11, 2002.

[7] G. E. Hinton, S. Osindero and Y. W. Teh, A Fast Learning Algorithm for Deep Belief Nets, Advances in *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[8] G. E. Hinton,"Learning multiple layers of representation," *TRENDS in Cognitive Sciences*, vol. 11. no. 10, pp. 428–434. 2007.

[9] M. Welling, M. Rosen-Zvi and G. E. Hinton,"Exponential fanily harmoniums with an application to information retrieval," In *Advances in Neural Information Processing Systems*, pp. 1481–1488. Cambridge, MA. MIT Press. 2005.

[10] G. E. Hinton, "Training product of experts by minimizing constrastive divergence," *Neural Computation*, vol. 14. no. 18, 2002, pp. 1527–1554.

[11] G. E. Dahl, M. Ranzato, A. Momamed and G. E. Hinton, "Phone Recognition with the Mean-Covariance Restricted Boltzmann Machines," *Advances in Neural Information Processing Systems NIPS*, 2010.

[12] R. R. Salakhutdinov and G.E. Hinton, "Replicated softmax: An undirected topic model," *Advances in Neural Information Processing Systems*, vol. 22, 2007.

[13] V. Vapnik, "The Nature of Statistical Learning Theory", *Springer–Verlag*, NY, USA, 1995.

[14] S. Chen and R. Rosenfeld, "A survey smoothing techniques for ME models", *IEEE Trans. SAP*, 8(1):37–50, 2001.

[15] S. D. Pietra, V. D. Pietra and J. Lafferty, "Inducing features of random fields", *IEEE Trans. Pattern. Analysis Mach. Int.*, 19(4):380–93, 1997.

[16] R. Sarikaya, H-K. J. Kuo V. Goel, and Y. Gao, "Exploiting Unlabeled Data Using Multiple Classifiers for Improved Natural Language Call-Routing ", In Proc. *Interspeech*, Lisbon Portugal, September 2005.