SIMPLIFIED LEARNING WITH BINARY ORTHOGONAL CONSTRAINTS

Qiang Huang

School of Informatics, University of Edinburgh

ABSTRACT

Deep architecture based Deep Brief Nets (DBNs) has shown its data modelling power by stacking up several Restricted Boltzmann Machines (RBMs). However, the multiple-layer structure used in DBN brings expensive computation, and furthermore leads to slow convergence. This is because the pre-training stage is usually implemented in a data-driven way, and class information attached to the training data is only used for fine-tuning. In this paper, we aim to simplify a multiple-layer DBN to a one-layer structure. We use class information as a constraint to the hidden layer during pre-training. For each training instance and its corresponding class, a binary sequence will be generated in order to adapt the output of hidden layer. We test our approaches on four data sets: basic MNIST, basic negative MNIST, rotation MNIST and rectangle (tall vs. wide rectangles). The obtained results show that the adapted one-layer structure can compete with a three-layer DBN.

Index Terms— Deep brief network, adapted hidden layer, DBM, pre-training, target dependent constraints

1. INTRODUCTION

The use of deep learning techniques [1, 2] have brought successes in academic researches and industrial applications [3, 4, 5]. In deep neural network, deep architectures are often constructed by unsupervised pretraining and stacking of restricted Boltzmann machines. RBMs working as generative models have been used to model many different types of data, images[6], speech[7], bags of words for document representation[8].

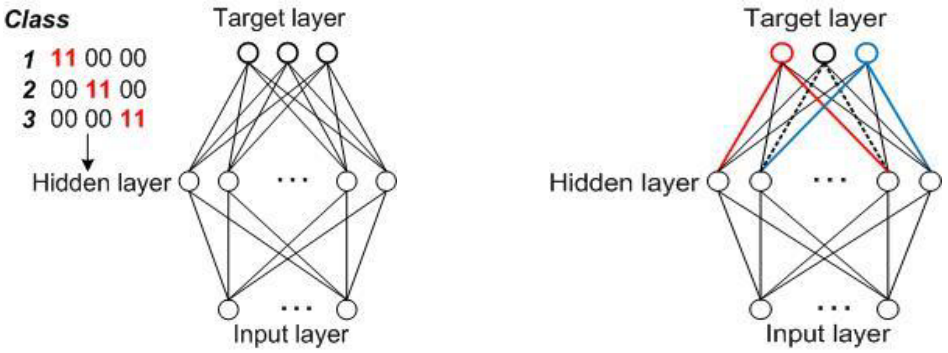
Previous studies have shown that a single RBM has limited data modeling capability, whereas a deep model, formed by stacking up several RBMs, presents great data modeling power [9]. Hinton et.al. proposed an efficient algorithm to train DBN by greedily training each layer of RBM using hidden activations in the previous layer [6]. It has been shown that the variational bound of the data log-likelihood is guaranteed under this greedy layer-wise learning framework, suggesting that stacking another layer of RBM will not deteriorate the models generative power. By stacking up RBMs, instead of getting a multi-layers Boltzmann machine, a hybrid model is constructed[9, 6]. When training RBM of each layer in DBN, contrastive divergence (CD) [10] method was proposed to replace the Markov chain. In the case of CDk, negative phase samples from the model are drawn by performing alternating Gibbs sampling only k (often k=1) times instead of reaching the equilibrium [11].

However, as learning is performed using stochastic gradient, it may converge to a local solution. It is generally not feasible to compare different local optima analytically. [12, 13] recently showed that depending on initialization and learning parameters the resulting RBMs can highly vary even for a small data set. Moreover, when processing condense data, such as negative transformation of MNIST data[14], it is still hard for the current stochastic learning method, such as CD, to avoid overfitting. In addition, although deep architecture brings benefits, some simple structures could be more desirable in order to reduce computation time for some practical applications, e.g. real-time speech process and image tracking.

Currently, there have been some work to handle the problems mentioned above. For overfitting, [2] used dropout to prevent complex co-adaptations on the training data. On each presentation of each training case, each hidden unit is randomly omitted from the network with a reasonable probability, e.g. 0.5. [15] introduced the use of mean reduction by subtracting the mean of the data prior to learning the parameters. However, these work still rely on multilayer network instead of showing the effectiveness of their on one-layer network. [16] presented an enhanced gradient method to tackle the greedy search for metaparameters in order to avoid stuck and even diverge.

In this paper, we will introduce a semi-supervised learning method by adapting the outputs of hidden units and reduce a multi-layer network to a one-layer structure simultaneously. Our work is motivated in two aspects:

- For pre-training, the parameters corresponding to each hidden layer are generally optimized in an unsupervised way. This means there will be no any prior knowledge on each hidden neuron at this stage, although the manual labellings will be used for fine-tuning. However, intuitively, the use of possible prior knowledge might be able to bring some benefits when they working as constraints to parameters optimisation.
- In a multiple-layer DBN, after pre-training and fine-



(a) Prop1 (b) Prop2

Fig. 1. Two proposed approaches: Prop1 and Prop2. Both of them use one-layer DBN structure.

tuning, we can easily find the trained filter matrices between the bottom hidden layer and the input layer are actually sparse. This is because the bottom layer is in charge of extracting the most basic and distinct features from the training data. The sparsity of filter matrices thus means there possibly exist some spaces for features mapped from higher layers.

The remainder of this paper is organised as follows: section 2 introduces the theoretical framework of our approaches. Then, section 3 describes the datasets to be used and experimental setup. Next, the evaluation performances are discussed in section 4. Finally, we draw the conclusion of this paper and give our future work in section 5.

2. THEORETICAL FRAMEWORK

For a neural network with $L, l \in \{1, ..., L\}$, hidden layers, let $\boldsymbol{v}^l$ denote the vector of inputs into layer l and $\boldsymbol{h}^l$ denote the vector of output from layer l. $\boldsymbol{W}^l$ and $\boldsymbol{b}^l$ are the weights and biases at layer l. The operation of the neural network can be described as:

$$h_i^{l+1} = \mathbf{w}_i^{l+1}\mathbf{x}^l + b_i^{l+1} \quad (1)$$

$$v_i^{l+1} = f(h^{l+1}) \quad (2)$$

where f represents an activation function. To optimise the parameters, contrastive divergence (CD) is generally used to perform stochastic steepest ascent in the training data:

$$\triangle w_{ij} = \varepsilon(< v_i h_j >_{data} - < v_i h_j >_{model}) \quad (3)$$

where ε is a learning rate.

During the optimisation of metaparameters in pretraining, it is not clear which hidden units will have more dependencies to a target unit than others. We call this case as "stochasticity" of hidden units and assume that some hidden units naturally have stronger dependencies to some specific target units

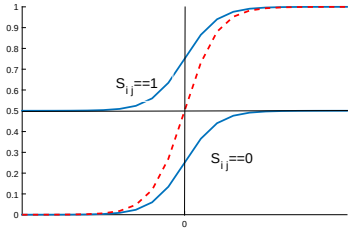


Fig. 2. activation function $f(x)$

(classes) than others and reducing the stochasticity can bring some benefits. We thus apply class information, as an additional constraint, to the hidden layer by selecting a number of hidden units and assign them directly to a specific class.

To conduct the hidden units selection and assignment, we propose two approaches. The first approach (Prop1), as shown in figure 1(a), uses a set of binary orthogonal sequence **S** to select a fixed number of hidden neurons for each class. "1" in the binary sequence means a strong dependency between the hidden and target unit, while "0" means a less dependency. In order to reflect different dependencies, we linearly interpolated between the output of each hidden unit (h_j) and its corresponding binary value S_{ij}.

$$h_j = p * h_j + (1 - p) * S_{ij} \quad (4)$$

$$1 \leq i \leq \#class, 1 \leq j \leq dimH,$$

where $dimH$ denotes the dimension of hidden layer, p denotes the interpolation coefficient and set to be 0.5 in this paper. The use of above equation can essentially be viewed as a change of activation function, namely:

$$f(x) = \begin{cases} p * f(x) + (1 - p) & \text{if } S_{ij} == 1 \\ p * f(x) & \text{otherwise} \end{cases} \quad (5)$$

Figure 2 shows the change of activation function.

The second approach (Prop2), as shown in figure 1(b), selects hidden units according to their values as the inputs to the target layer. In comparison with Prop1, Prop2 does not relies on a set of pre-defined orthogonal binary sequences. The implementation of Prop2 contains four steps:

1. Given an instance and its corresponding class c, compute the input to the target unit t_c from each hidden unit h_j using $z_{c,j} = W_{c,j} * h_j$, where $W_{c,j}$ represents the weight of a link between the c^{th} target unit and the j^{th} hidden unit

2. Select N top-ranked hidden units with respect to $z_{c,j}$, e.g. $N = \frac{\#\ hidden\_units}{\#\ class}$

3. Build a $dimH$-dimension binary sequence, where its bits corresponding to the top-ranked hidden units are set to be "1" if selected, or set to be "0"

4. Compute h_j using equation 4 and adapt W with $\triangle w$ obtained using equation 3

3. DATA AND EXPERIMENTAL SETUP

3.1. Data

In our experiments, we use the MNIST dataset (MNIST-basic) and its variants, as well as rectangles. MNIST is a standard computer vision task that provides images containing 28x28 gray-scale pixels representing ten handwritten digits (0 to 9)[14]. The variants consists of two more modifications to the MNIST dataset, including images of rotated digits (MNIST-rot) and the negative images of MNIST. In MNIST-rot, the digits were rotated by an angle generated uniformly between 0 and 2π radians. In MNIST-neg, the values of pixels in original mnist images are flipped. In rec., a classification task is to identify whether a rectangle contained in an image has a larger width or length. The rectangle can be situated anywhere in the 28 x 28 pixel image.

3.2. Setup

In our experiments, we follow a common way to split each dataset into three parts: a training for pre-training and fine-tuning the parameters (10000 for *MNIST*s and 1000 for *rec*), a valid part for early stopping 2000 for *MNIST*s and 200 for *rec*, and a test part for final evaluation (50000 for both *MNIST*s and *rec*). For a comparison, we also run two baselines, DBN-1 and DBN-3, on the four data sets, where DBN-1 has one hidden layer and DBN-3 has three. The number of hidden units and learning rate are hyper parameters and are tuned by a grid search among a small set of values on the validation set: number of hidden units $\in \{500, 600, 800, 1000\}$, learning rate $\in \{0.01, 0.1, 0.5, 0.8, 1\}$ for both pre-training and fine-tuning. The weight cost is set to be 0.0002. For DBN-3, its top hidden layer contains 2000 hidden units. The number of epochs is 200 for pre-training and 200 for fine-tuning, respectively. As a further comparison, we also set drop-out rate $\in \{0, 0.2, 0.5, 0.7, 0.9\}$ only for DBN-1 and DBN-3. The "sigmoid" function is selected as the activation function for all experiments in this paper. In all experiments we apply no further data preprocessing except that we follow Vincent *et al*.[17] to select the models based on the validation set performance by early-stopping during fine-tuning.

4. RESULTS AND ANALYSIS

Figures 3, 4, 5 and 6 show the classification results on four data sets using different methods, respectively. Both of our proposed approaches, Prop1 and Prop2, can yield salient improvements on the four data sets in comparison with DBN-1. Even if comparing with DBN-3, Prop1 still outperforms it on three data sets: rectangle, MNIST-basic and MNIST-basic-neg, and can also compete with it on MNIST-rot. Comparing with DBN-3, Prop2 also works better on the classification of rectangles and the negative MNIST data. The difference on classification performance between Prop1 and Pro2 is mainly because that Prop1 uses completely orthogonal binary sequence as a constraint to reduce the possible interferences from other classes. As aforementioned in section 1, this one-layer structure can be viewed as a combination of stochastic and deterministic DBN.

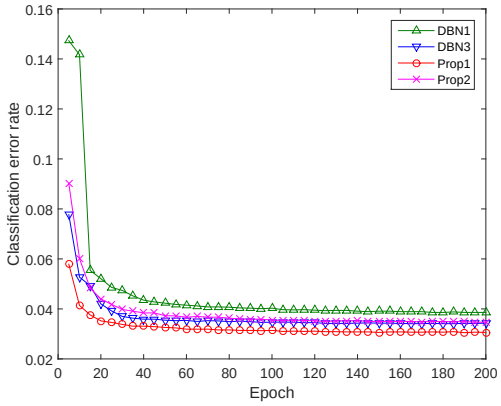


Fig. 3. MNIST-basic

In our experiments, for MNIST-rot, both of our approaches only outperform DBN-1. The possible reasons is the number of neurons we selected in the hidden layer is not big enough to cover the variations caused by rotation. For *MNIST-neg*, we find that DBN3 fails to classify the data without using dropout, although it can work well on *MINST-basic*, standard digit images with a very sparse energy distribution. As aforementioned in section 1, when the input data mean is near 1, such as *MNIST-neg*, learning is considerably worse

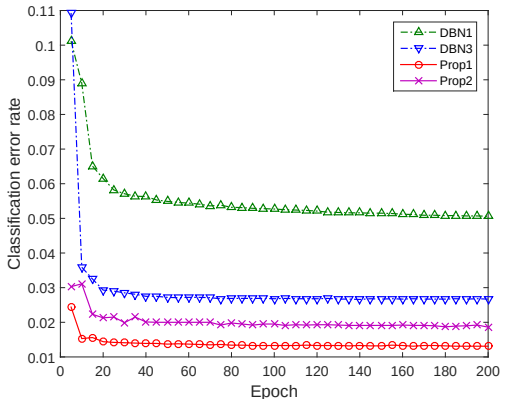


Fig. 4. rectangle

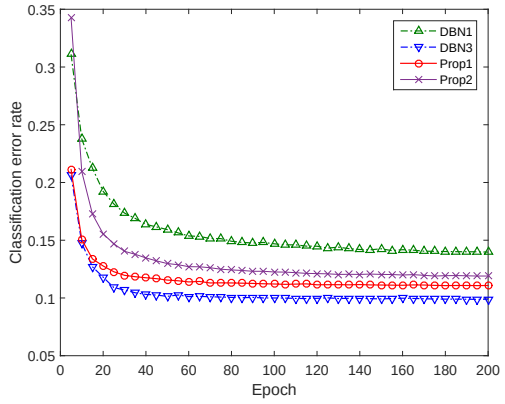


Fig. 5. MNIST-rotate

and even can not convergence. As a comparison, our two approaches can well handle this issue. It is mainly because we make the hidden units work in two states. The one state is that some hidden units work more deterministically to a target unit than other hidden units, which work stochastically with their output to the target being decreased.

It is clear that, after training a multiple-layer neural network, each neuron will have more dependencies to some specific classes and less dependencies to others. However, the selected activation function, such as *sigmoid*, corresponding to each neuron is generally kept same shape during the whole optimization. This means it will equally treat the input after the feature vectors (v) from different classes, multiplying by weight matrix W. The use of our two approaches makes it possible let the active function of each neuron lift or reduce its response when the input coming in from related or unrelated classes. Figure 2 shows, during pre-training, the active function works in two different states to distinguish the input from related and unrelated classes.

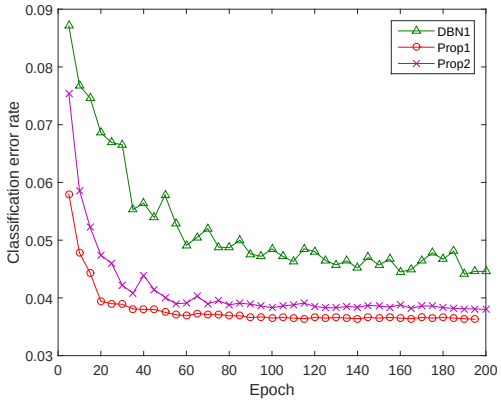


Fig. 6. MNIST-basic-neg

Table 1. Comparison with DBN1 and DBN-3 after using dropout

Data set	MNIST-basic	basic-neg	rot.	rec.
DBN1	0.0396	0.0446	0.1383	0.0506
DBN3	0.0365	0.90	0.1001	0.0261
DBN1 +drop	0.0339	0.0405	0.1198	0.0301
DBN3 +drop	0.0313	0.0378	**0.0989**	0.0198
Prop1	**0.0301**	**0.0363**	0.1101	**0.0121**
Prop2	0.0338	0.0380	0.1187	0.0187

A further comparison was given in Table 1 after DBN-1 and DBN-3 using "dropout". It is clear that, except MNIST-rot, the use of our method (Prop1) still outperforms DBN-3 and DBN-1 with "dropout" on the other three data sets.

5. CONCLUSION

The use of our proposed methods can yield better performances on three data sets in comparison with DBN-3 and a closed performance to DBN-3 on the data set of MNIST-rot. The reason is that we make the hidden units work in different states with using class information of each training instance. This adaptation can let us obtain good performances on four data sets only using one-layer structure.

In the future work, we will further test our approaches on more data sets and different network formats, such as autoencoder based neural network; moreover, we will also consider to use and design more robust activation functions to apply possible deterministic factors to the stochastic characteristics of data.

Acknowledgement: This work is supported by Natural Speech Technology (NST), a grant from the UK Engineering and Physical Sciences Research Council (EP/I031022/1).

6. REFERENCES

[1] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, 2006.

[2] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012.

[3] G. E. Dahl, D. Yu, and L. Deng, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 20, pp. 33–42, 2012.

[4] Y. Bengio, R Ducharme, V. Pascal, and J. Christian, "A neural probabilistic language model," *Journal of Machine Learning*, vol. 3, pp. 1137–1155, 2003.

[5] I. Goodfellow, M. Mirza, and A. C. Courville, "Multi-prediction deep boltzmann machines," in *Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 548–556.

[6] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.

[7] A. R. Mohamed, G. Dahl, and G. E. Hinton, "Deep belief networks for phone recognition," in *NIPS'22 workshop on deep learning for speech recognition*, 2009.

[8] R. R. Salakhutdinov and G. E. Hinton, "Replicated softmax: An undirected topic model," in *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[9] Y. Bengio, "Learning deep architectures for ai," *Foundations and Trends in Machine Learning archive*, vol. 2, pp. 1–127, 2009.

[10] M. A. Carreira-Perpignan and Hinton. G. E., "On contrastive divergence learning," in *Artificial Intelligence and Statistics*, 2005.

[11] J. How and T. Yu, "Sparse maximum entropy deep belief nets," in *International Joint Conference on neural network*, 2013, pp. 1–6.

[12] abd Muller A. Schulz, H. and S. Behnke, "Investigating convergence of restricted boltzmann machine learning," in *NIPS 2010 Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[13] A. Fischer and C. Igel, "Empirical analysis of the divergence of gibbs sampling based learning algorithms for restricted boltzmann machines," in *Proceedings of the 20th international conference on Artificial neural networks*, 2010, pp. 208–217.

[14] L.C. Yan and C. Corte, "The mnist database of handwritten digits," 1998.

[15] Y.C. Tang and I. Sutskever, "Data normalization in the learning of rbms," Tech. Rep., Department of Computer Science, University of Toronto, 2011.

[16] K. H. Cho, T. Raiko, and A. Ilin, "Enhanced gradient for training restricted boltzmann machines," *Neural Compute*, vol. 25, pp. 805–831, 2013.

[17] P. Vincent, H. Larchelle, I. Lajoie, and P. A. Bengio, Y. an Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.