Improved classification based on Deep Belief Networks

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

For better classification generative models are used to initialize the model and model features before training a classifier. Typically it is needed to solve separate unsupervised and supervised learning problems. Generative restricted Boltzmann machines and deep belief networks are widely used for unsupervised learning. We developed several supervised models based on DBN in order to improve this two-phase strategy. Modifying the loss function to account for expectation with respect to the underlying generative model, introducing weight bounds, and multi-level programming are applied in model development. The proposed models capture both unsupervised and supervised objectives effectively. The computational study verifies that our models perform better than the two-phase training approach.

1 Introduction

Restricted Boltzmann machine (RBM), an energy-based model to define an input distribution, is widely used to extract latent features before classification. Such an approach combines unsupervised learning for feature modeling and supervised learning for classification. Two training steps are needed. The first step, called pre-training, is to model features used for classification. This can be done by training RBM that captures the distribution of input. The second step, called fine-tuning, is to train a separate classifier based on the features from the first step [12]. This two-phase training approach for classification is also used for deep networks. Hinton et al. (2006) proposed deep belief networks (DBN) that are built with stacked RBMs, and trained in a layer-wise manner [9]. Two-phase training based on a deep network consists of DBN and a classifier on top of it.

The two-phase training strategy has three possible problems. 1) It requires two training processes; one for training RBMs and one for training a classifier. 2) It is not guaranteed that the modeled features in the first step are useful in the classification phase since they are obtained independently of the classification task. 3) It is an effort to decide which classifier is the best for each problem. Therefore, there is a need for a method that can conduct feature modeling and classification concurrently [12].

To resolve these problems, recent papers suggest to transform RBM to a model that can deal with both unsupervised and supervised learning. Since RBM calculate the joint and conditional probabilities, the suggested prior models combine a generative and discriminative RBM. Consequently, this hybrid discriminative RBM is trained concurrently for both objectives by summing the two contributions [11, 12]. In a similar way a self-contained RBM for classification is developed by applying the free-energy function based approximation to RBM, which was used for a supervised learning method, reinforcement learning [5]. However, these approaches are limited to transforming RBM that is a shallow network.

In this study, we developed alternative models to solve a classification problem based on DBN. Viewing the two-phase training as two separate optimization problems, we applied optimization modeling techniques in developing our models. Our first approach is to design new objective functions.

Submitted to 29th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

We design an expected loss function based on $p(h|x)$ built by DBN and the loss function of the classifier. Second, we introduce constraints that bound the DBN weights in the feed-forward phase. The constraints keep a good representation of input as well as regularize the weights during updates. Third, we applied bi-level programming to the two-phase training method. The bi-level model has a loss function of the classifier in its objective function but it constrains the DBN values to the optimal to phase-1. This model searches possible optimal solutions for the classification objective only where DBN objective solutions are optimal.

Our main contributions are several classification models combining DBN and a loss function in a coherent way. In the computational study we verify that the suggested models perform better than the two-phase method.

2 Literature Review

The two-phase training strategy has been applied to many classification tasks on different types of data. Two-phase training with RBM and support vector machine (SVM) has been explored in classification tasks on images, documents, and network intrusion data by Xing et al. (2005), Norouzi et al. (2009), Salama et al. (2011), and Dahl et al. (2012) [18, 14, 15, 4]. Logistic regression replacing SVM has been explored in Mccallum et al. (2006) and Cho et al. (2011) [13, 3]. Gehler et al. (2006) used the 1-nearest neighborhood classifier with RBM to solve a document classification task [7]. Hinton et al. (2006) suggested DBN consisting of stacked RBMs that is trained in a layer-wise manner. Two-phase method using DBN and deep neural network has been studied to solve various classification problems such as image and text recognition in Hinton et al. (2006), Bengio et al. (2007), and Sarikaya et al. (2014) [9, 16, 2]. All these papers rely on two distinct phases, while our models assume a holistic view of both aspects.

Many studies have been conducted to improve the problems of two-phase training. Most of the research has been focused on transforming RBM so that the modified model can achieve generative and discriminative objectives at the same time. Schmah et al. (2009) proposed a discriminative RBM method, and subsequently classification is done in the manner of a Bayes classifier [17]. However, this method cannot capture the relationship between the classes since the RBM of each class is trained separately. Larochelle et al. (2008, 2012) proposed a self-contained discriminative RBM framework where the objective function consists of the generative learning objective $p(x, y)$, and the discriminative learning objective, $p(y|x)$, [11, 12]. Both distributions are derived from RBM. Similarly, Elfwing et al. (2015) proposed a self-contained discriminative RBM method for classification. The free-energy function based approximation is applied in the development of this method, which is initially suggested for reinforcement learning [5]. This prior paper relying on RBM conditional probability while we handle general loss functions. Our models also hinge on completely different principles.

3 Background

Restricted Boltzmann Machines RBM is an energy-based probabilistic model, which is a restricted version of Boltzmann machines (BM) that is a log-linear Markov Random Field. It has visible nodes x corresponding to input and hidden nodes h matching the latent features. The joint distribution of the visible nodes $x \in \mathbb{R}^J$ and hidden variable $h \in \mathbb{R}^I$ is defined as

$$p(x, h) = \frac{1}{Z}e^{-E(x,h)},\; E(x, h) = -hWx - ch - bx$$

where $W \in \mathbb{R}^{I \times J}$, $b \in \mathbb{R}^J$, and $c \in \mathbb{R}^I$are the model parameters, and Z is the partition function. Since units in a layer are independent in RBM, we have the following form of conditional distributions:

$$p(h|x) = \prod_{i=1}^{I} p(h_i|x),\; p(x|h) = \prod_{j=1}^{J} p(x_j|h).$$

For binary units where $x \in \{0, 1\}^J$ and $h \in \{0, 1\}^I$, we can write

$$p(h_i = 1|h) = \sigma(c_i + W_i x),\; p(x_j = 1|h) = \sigma(b_j + W_j x)$$

where $\sigma()$ is the sigmoid function. In this manner RBM with binary units is an unsupervised neural network with a sigmoid activation function. The model calibration of RBM can be done by minimizing negative log-likelihood through gradient descent. RBM takes advantage of having the above conditional probabilities which enable to obtain model samples easier through a Gibbs sampling method. Contrastive divergence (CD) makes Gibbs sampling even simpler: 1) start a Markov chain with training samples, and 2) stop to obtain samples after k steps. It is shown that CD with a few steps performs effectively [1, 8].

Deep Belief Networks DBN is a generative graphical model consisting of stacked RBMs. Based on its deep structure DBN can capture a hierarchical representation of input data. Hinton et al. (2006) introduced DBN with a training algorithm that greedily trains one layer at a time. Given visible unit x and ℓ hidden layers the joint distribution is defined as [1, 10]

$$p(x, h^1, \cdots, h^\ell) = p(h^{\ell-1}, h^\ell) \left(\prod_{k=1}^{\ell-2} p(h^k | h^{k+1}) \right) p(x|h^1).$$

Since each layer of DBN is constructed as RBM, training each layer of DBN is the same as training a RBM.

Classification is conducted by initializing a network through DBN training [2, 10]. A two-phase training can be done sequentially by: 1) pre-training, unsupervised learning of stacked RBM in a layer-wise manner, and 2) fine-tuning, supervised learning with a classifier. Each phase requires solving an optimization problem. Given training dataset $D = \{(x^{(1)}, y^{(1)}), \dots, (x^{(|D|)}, y^{(|D|)})\}$ with input x and label y, the pre-training phase solves the following optimization problem at each layer k

$$\min_{\theta_k} \quad \frac{1}{|D|} \sum_{i=1}^{|D|} \left[-log\, p(x_k^{(i)}; \theta_k) \right]$$

where $\theta_k = (W_k, b_k, c_k)$ is the RBM model parameter that denotes weights, visible bias, and hidden bias in the energy function, and $x_k^{(i)}$ is visible input to layer k corresponding to input $x^{(i)}$. Note that in layer-wise updating manner we need to solve ℓ of the problems from the bottom to the top hidden layer. For the fine-tuning phase we solve the following optimization problem

$$\min_{\phi} \quad \frac{1}{|D|} \sum_{i=1}^{|D|} \left[\mathcal{L}(\phi; y^{(i)}, h(x^{(i)})) \right] \tag{1}$$

where $\mathcal{L}()$ is a loss function, h denotes the final hidden features at layer ℓ, and ϕ denotes the parameters of the classifier. Here for simplicity we write $h(x^{(i)}) = h(x_\ell^{(i)})$. When combining DBN and a feed-forward neural networks (FFN) with sigmoid activation, all the weights and hidden bias parameters among input and hidden layers are shared for both training phases. Therefore, in this case we initialize FFN by training DBN.

4 Proposed models

We model an expected loss function for classification. Considering classification of two phase method is conducted on hidden space, the probability distribution of the hidden variables obtained by DBN is used in the proposed models. The two-phase method provides information about modeling parameters after each phase is trained. Constraints based on the information are suggested to prevent the model parameters from deviating far from good representation of input. Optimal solution set for unsupervised objective of the two-phase method is good candidate solutions for the second phase. Bi-level model has the set to find optimal solutions for the phase-2 objective so that it conducts the two-phase training at one-shot.

DBN fitting plus loss model We start with a naive model of summing pre-trainning and fine-tuning objectives. This model conducts the two-phase training strategy simultaneously; however, we need to add one more hyperparameter ρ to balance the impact of both objectives. The model (DBN+loss) is defined as

$$\min_{\theta_{Loss}, \theta_{DBN}} \quad \mathbb{E}_{\mathbf{y},\mathbf{x}}[\mathcal{L}(\theta_{Loss}; \mathbf{y}, h(\mathbf{x}))] + \rho\, \mathbb{E}_{\mathbf{x}}[-\, log\, p(\mathbf{x}; \theta_{DBN})]$$

and empirically based on training samples D,

$$\min_{\theta_{Loss},\theta_{DBN}} \quad \frac{1}{|D|}\sum_{i=1}^{|D|}\left[\mathcal{L}(\theta_{Loss}; y^{(i)}, h(x^{(i)})) - \rho\, log\, p(x^{(i)}; \theta_{DBN})\right] \tag{2}$$

where $\theta_{Loss}, \theta_{DBN}$ are the underlying parameters. Note that $\theta_{Loss} = \phi$ from (1) and $\theta_{DBN} = (\theta_k)_{k=1}$. This model has already been proposed by Larochelle et al. (2008, 2012) if the classification loss function is based on the RBM conditional distribution [11, 12].

Expected loss model with DBN boxing We first design an expected loss model based on conditional distribution $p(h|x)$ obtained by DBN. This model conducts classification on the hidden space. Since it minimizes the expected loss, it should be more robust and thus it should yield better accuracy on data not observed. The mathematical model that minimizes the expected loss function is defined as

$$\min_{\theta_{Loss},\theta_{DBN}} \quad \mathbb{E}_{\mathbf{y},\mathbf{h}|\mathbf{x}}[\mathcal{L}(\theta_{Loss}; \mathbf{y}, h(\theta_{DBN}; \mathbf{x}))]$$

and empirically based on training samples D,

$$\min_{\theta_{Loss},\theta_{DBN}} \quad \frac{1}{|D|}\sum_{i=1}^{|D|}\left[\sum_h p(h|x^{(i)})\mathcal{L}(\theta_{Loss}; y^{(i)}, h(\theta_{DBN}; x^{(i)}))\right].$$

With notation $h(\theta_{DBN}; x^{(i)}) = h(x^{(i)})$ we explicitly show the dependency of h on θ_{DBN}. We modify the expected loss model by introducing a constraint that sets bounds on DBN related parameters with respect to their optimal values. This model has two benefits. First, the model keeps a good representation of input by constraining parameters fitted in the unsupervised manner. Also, the constraint regularizes the model parameters by preventing them from blowing up while being updated. Given training samples D the mathematical form of the model (EL-DBN) reads

$$\begin{aligned}\min_{\theta_{Loss},\theta_{DBN}} \quad & \frac{1}{|D|}\sum_{i=1}^{|D|}\left[\sum_h p(h|x^{(i)})\mathcal{L}(\theta_{Loss}; y^{(i)}, h(\theta_{DBN}; x^{(i)}))\right]\\ \text{s.t.} \quad & |\theta_{DBN} - \theta^*_{DBN}| \le \delta\end{aligned}$$

where θ^*_{DBN} are the optimal DBN parameters and δ is a hyperparameter. This model needs a pre-training phase to obtain the DBN fitted parameters.

Expected loss model with DBN classification boxing Similar to the DBN boxing model, this expected loss model has a constraint that the DBN parameters are bounded by their optimal values at the end of both phases. This model regularizes parameters with those that are fitted in both the unsupervised and supervised manner. Therefore, it can achieve better accuracy even though we need an additional training to the two-phase trainings. Given training samples D the model (EL-DBNOPT) reads

$$\begin{aligned}\min_{\theta_{Loss},\theta_{DBN}} \quad & \frac{1}{|D|}\sum_{i=1}^{|D|}\left[\sum_h p(h|x^{(i)})\mathcal{L}(\theta_{Loss}; y^{(i)}, h(\theta_{DBN}; x^{(i)}))\right]\\ \text{s.t.} \quad & |\theta_{DBN} - \theta^*_{DBN-OPT}| \le \delta\end{aligned} \tag{3}$$

where $\theta^*_{DBN-OPT}$ are the optimal values of DBN parameters after two-phase training and δ is a hyperparameter.

Feed-forward network with DBN boxing We also propose a model based on boxing constraints where FFN is constrained by DBN output. The mathematical model (FFN-DBN) based on training samples D is

$$\begin{aligned}\min_{\theta_{Loss},\theta_{DBN}} \quad & \frac{1}{|D|}\sum_{i=1}^{|D|}\left[\mathcal{L}(\theta_{Loss}; y^{(i)}, h(\theta_{DBN}; x^{(i)}))\right]\\ \text{s.t.} \quad & |\theta_{DBN} - \theta^*_{DBN}| \le \delta.\end{aligned} \tag{4}$$

Feed-forward network with DBN classification boxing Given training samples D this model (FFN-DBNOPT), which is a mixture of (3) and (4), reads

$$\begin{aligned} \min_{\theta_{Loss}, \theta_{DBN}} \quad & \frac{1}{|D|} \sum_{i=1}^{|D|} \left[\mathcal{L}(\theta_{Loss}; y^{(i)}, h(\theta_{DBN}; x^{(i)})) \right] \\ \text{s.t.} \quad & |\theta_{DBN} - \theta^*_{DBN-OPT}| \leq \delta. \end{aligned}$$

Bi-level model We also apply bi-level programming to the two-phase training method. This model searches optimal solutions to minimize the loss function of the classifier only where DBN objective solutions are optimal. Possible candidates for optimal solutions of the first level objective function are optimal solutions of the second level objective function. This model (BL) reads

$$\begin{aligned} \min_{\theta_{Loss}, \theta^*_{DBN}} \quad & \mathbb{E}_{\mathbf{y},\mathbf{x}}[\mathcal{L}(\theta_{Loss}; \mathbf{y}, h(\theta^*_{DBN}; \mathbf{x}))] \\ \text{s.t.} \quad & \theta^*_{DBN} = \arg\min_{\theta_{DBN}} \ \mathbb{E}_{\mathbf{x}}[-log\, p(\mathbf{x}; \theta_{DBN})] \end{aligned}$$

and empirically based on training samples,

$$\begin{aligned} \min_{\theta_{Loss}, \theta^*_{DBN}} \quad & \frac{1}{|D|} \sum_{i=1}^{|D|} \left[\mathcal{L}(\theta_{Loss}; y^{(i)}, h(\theta^*_{DBN}; x^{(i)})) \right] \\ \text{s.t.} \quad & \theta^*_{DBN} = \arg\min_{\theta_{DBN}} \ \frac{1}{|D|} \sum_{i=1}^{|D|} \left[-log\, p(x^{(i)}; \theta_{DBN}) \right]. \end{aligned}$$

One of the solution approaches to bi-level programming is to apply KKT conditions to the lower level problem. After applying KKT to the lower level, we obtain

$$\begin{aligned} \min_{\theta_{Loss}, \theta^*_{DBN}} \quad & \mathbb{E}_{\mathbf{y},\mathbf{x}}[\mathcal{L}(\theta_{Loss}; \mathbf{y}, h(\theta^*_{DBN}; \mathbf{x}))] \\ \text{s.t.} \quad & \nabla_{\theta_{DBN}} \mathbb{E}_{\mathbf{x}}[-log\, p(\mathbf{x}; \theta_{DBN})|_{\theta^*_{DBN}}] = 0. \end{aligned}$$

Furthermore, we transform this constrained problem to an unconstrained problem with a quadratic penalty function:

$$\min_{(\theta_{Loss}, \theta^*_{DBN})} \quad \mathbb{E}_{\mathbf{y},\mathbf{x}}[\mathcal{L}(\theta_{Loss}; \mathbf{y}, h(\theta^*_{DBN}; \mathbf{x}))] + \frac{\mu}{2} ||\nabla_{\theta_{DBN}} \mathbb{E}_{\mathbf{x}}[-log\, p(\mathbf{x}; \theta_{DBN})]|_{\theta^*_{DBN}}||^2 \tag{5}$$

where μ is a hyperparameter. The gradient of the objective function is derived in the appendix.

5 Computational study

To evaluate the proposed models classification tasks on three datasets were conducted: the MNIST hand-written images [1], the KDD'99 network intrusion dataset (NI)[2], and the isolated letter speech recognition dataset (ISOLET) [3]. The experimental results of the proposed models on these datasets were compared to those of the two-phase method.

In FFN, the sigmoid functions in the hidden layers and the softmax function in the output layer were chosen with negative log-likelihood as a loss function of the classifiers. The size and the number of the hidden layers was selected differently depending on the datasets (optimally for each case). We first implemented the two-phase method to obtain the best configuration of the hidden units and layers, and then applied this configuration to the proposed models.

Implementations were done in Theano. The mini-batch gradient descent algorithm was used to solve the optimization problems of each model. To calculate the gradients of each objective function of the models Theano's built-in functions, 'theano.tensor.grad', was used. We denote by DBN-FFN the two-phase approach.

[1] http://yann.lecun.com/exdb/mnist/
[2] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[3] https://archive.ics.uci.edu/ml/datasets/ISOLET

Table 1: Results on MNIST

Model	Test error (%)	
	Shallow network	Deep network
DBN-FFN	**1.17 %**	**1.14 %**
DBN+loss	1.61 %	1.64 %
EL-DBN	1.35 %	1.30 %
EL-DBNOPT	1.17 %	1.13 %
FFN-DBN	1.17 %	1.29 %
FFN-DBNOPT	**1.16 %**	**1.09 %**
BL	1.61 %	1.72 %

Table 2: Results on NI

Model	Test error rate		
	20 % training	30 % training	40 % training
DBN-FFN	**7.41 %**	**7.19 %**	**7.31 %**
DBN+loss	7.29 %	7.30 %	7.35 %
EL-DBN	8.35 %	7.69 %	7.69 %
EL-DBNOPT	7.34 %	7.18 %	7.31 %
FFN-DBN	7.53 %	7.45 %	7.56 %
FFN-DBNOPT	7.32 %	**7.14 %**	7.31 %
BL	**7.19 %**	7.21 %	**7.08** %

5.1 MNIST

The task on the MNIST is to classify ten digits from 0 to 9 given by 28×28 pixel hand-written images. The dataset is divided in 60,000 samples for training and validation, and 10,000 samples for testing. The hyperparameters are set as: 1) hidden units at each layer are 500 or 1000, 2) training epochs for pre-training and fine-tuning range from 100 to 900, 3) learning rates for pre-training are 0.01 or 0.05, and these for fine-tuning range from 0.1 to 2, 4) batch size is 50, and 5) ρ of the DBN+loss and μ of the BL model are diminishing during iterations.

DBN-FFN with four-hidden layers of size, 784-1000-1000-1000-1000-10, was the best, and subsequently we compared it to the proposed models with the same size of the network. In Table 1, the best test error rate was achieved by FFN-DBNOPT, 1.09%. Furthermore, the models with the DBN classification constraints, EL-DBNOPT and FFN-DBNOPT, perform better than the two-phase method. This shows that DBN classification boxing constraints regularize the model parameters by keeping a good representation of input.

5.2 Network Intrusion

The classification task on NI is to distinguish between normal and bad connections given the related network connection information. The preprocessed dataset consists of 41 input features and 5 classes, and 4,898,431 examples for training and 311,029 examples for testing. The experiments were conducted on 20%, 30%, and 40% subsets of the whole training set, which were obtained by stratified random sampling. Hyperparameters are set as: 1) hidden units at each layer are 13, 15, or 20, 2) training epochs for pre-training and fine-tuning range from 100 to 900, 3) learning rates for pre-training are 0.01 or 0.05, and these for fine-tuning are from 0.1 to 2, 4) batch size is 1000, and 5) ρ of the DBN+loss and μ of the BL are diminishing during iterations.

On NI the best structure of DBN-FFN was 41-15-15-5 for the 20% and the 30% training set, and 41-15-15-15-5 for the 40 % training set. Table 2 shows the experimental results of the proposed models with the same network as the best DBN-FFN. BL produces the best test error, 7.08%. This showed that the model being trained concurrently for unsupervised and supervised purpose can achieve better accuracy than the two-phase method. Furthermore, both EL-DBNOPT and FFN-DBNOPT yield similar to, or lower error rates than DBN-FFN in all of the three subsets.

Table 3: Results on ISOLET

Model	Test error rate
DBN-FFN	**3.12 %**
DBN+loss	4.09 %
EL-DBN	3.38 %
EL-DBNOPT	3.44 %
FFN-DBN	**3.12 %**
FFN-DBNOPT	**3.12 %**
BL	3.96 %

5.3 ISOLET

The classification on ISOLET is to predict which letter-name was spoken among the 26 English alphabets given 617 input features of the related signal processing information. The dataset consists of 5,600 for training, 638 for validation, and 1,559 examples for testing. Hyperparameters are set as: 1) hidden units at each layer are 400, 500, or 800, 2) training epochs for pre-training and fine-tuning are from 100 to 900, 3) learning rates for pre-training are from 0.001 to 0.05, and these for fine-tuning are from 0.05 to 1, 4) batch size is 20, and 5) ρ of the DBN+loss and μ of the BL model are diminishing during iterations.

In this experiment the deep network performed worse than the shallow network. One possible reason for this is its small size of training samples. The one hidden layer with 500 units was the best for DBN-FFN. Table 3 shows the experimental results of the proposed models with the same hidden layer setting. DBN-FFN and DBN classification boxing models achieve the same accuracy.

6 Conclusions

DBN+loss showed worse accuracy than two-phase training in all of the experiments. Aggregating two unsupervised and supervised objectives without a specific treatment is not effective. Second, the models with DBN optimal boxing, EL-DBN and FFN-DBN, performed worse than DBN-FFN. Regularizing the model parameters with unsupervised learning is not so effective in solving a supervised learning problem. Third, the models with DBN classification boxing, EL-DBNOPT and FFN-DBNOPT, performed no worse than DBN-FFN in all of the experiments. This shows that classification accuracy can be improved by regularizing the model parameters with the values trained for unsupervised and supervised purpose. One drawback of this approach is that one more training phase to the two-phase approach is necessary. Last, BL showed that one-step training can achieve a better performance than two-phase training. Even though it worked in one instance, improvements to current BL can be made such as applying different solution search algorithms, supervised learning regularization techniques, or different initialization strategies.

References

[1] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[2] Y. Bengio and P. Lamblin. Greedy layer-wise training of deep networks. *In Advances in Neural Information Processing Systems (NIPS)*, 20(1):153–160, 2007.

[3] K. Cho, A. Ilin, and T. Raiko. Improved learning algorithms for restricted Boltzmann machines. *In International Conference on Artificial Neural Networks (ICML)*, 28:10–17, 2011.

[4] G. E. Dahl, R. P. Adams, and H. Larochelle. Training restricted Boltzmann machines on word observations. *In International Conference on Machine Learning (ICML)*, 29:679–686, 2012.

[5] S. Elfwing, E. Uchibe, and K. Doya. Expected energy-based restricted Boltzmann machine for classification. *Neural Networks*, 64:29–38, 2015.

[6] A. Fischer and C. Igel. An introduction to restricted Boltzmann machines. *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, 7441:14–36, 2012.

[7] P. V. Gehler, A. D. Holub, and MaxWelling. The rate adapting Poisson (RAP) model for information retrieval and object recognition. *In International Conference on Machine Learning (ICML)*, 23:337–344, 2006.

[8] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14 (8):1771–1800, 2002.

[9] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[10] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–54, 2006.

[11] H. Larochelle and Y. Bengio. Classification using discriminative restricted Boltzmann machines. *In International Conference on Machine Learning (ICML)*, 25:536–543, 2008.

[12] H. Larochelle, M. Mandel, R. Pascanu, and Y. Bengio. Learning algorithms for the classification restricted Boltzmann machine. *Journal of Machine Learning Research*, 13:643–669, 2012.

[13] A. Mccallum, C. Pal, G. Druck, and X. Wang. Multi-conditional learning : generative / discriminative training for clustering and classification. *In National Conference on Artificial Intelligence (AAAI)*, 21(1): 433–439, 2006.

[14] M. Norouzi, M. Ranjbar, and G. Mori. Stacks of convolutional restricted Boltzmann machines for shift-invariant feature learning. *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, pages 2735–2742, 2009.

[15] M. Salama, H. Eid, and R. Ramadan. Hybrid intelligent intrusion detection scheme. *Advances in Intelligent and Soft Computing*, pages 293–303, 2011.

[16] R. Sarikaya, G. E. Hinton, and A. Deoras. Application of deep belief networks for natural language understanding. *In IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(4):778–784, 2014.

[17] T. Schmah, G. E. Hinton, R. S. Zemel, S. L. Small, and S. Strother. Generative versus discriminative training of RBMs for classification of fMRI images. *In Advances in Neural Information Processing Systems (NIPS)*, 21:1409–1416, 2009.

[18] E. P. Xing, R. Yan, and A. G. Hauptmann. Mining associated text and images with dual-wing Harmoniums. *In Conference on Uncertainty in Artificial Intelligence (UAI)*, 21:633–641, 2005.

7 Appendix

7.1 Approximation of DBN probability in the proposed models

DBN defines the joint distribution of the visible unit x and the ℓ hidden layers, $h^1, h^2, \cdots, h^\ell$ as

$$p(x, h^1, \cdots, h^\ell) = p(h^{\ell-1}, h^\ell)\left(\prod_{k=0}^{\ell-2} p(h^k|h^{k+1})\right) \text{ with } h^0 = x.$$

DBN plus loss model From Eq. (2), $p(x)$ in the second term of the objective function is approximated as

$$p(x; \theta_{DBN}) = \sum_{h^1, h^2, \cdots, h^\ell} p(x, h^1, \cdots, h^\ell) \approx \sum_{h^1} p(x, h^1).$$

Expected loss models $p(h|x)$ in the objective function is approximated as

$$\begin{aligned}
p(h^\ell|x) &\approx p(h^\ell|x, h^1, \cdots, h^\ell) \\
&= \frac{p(h^\ell, h^{\ell-1}, \cdots, h^1, x)}{p(h^{\ell-1}, h^{\ell-2}, \cdots, h^1, x)} \\
&= \frac{p(h^{\ell-1}, h^\ell)\left(\prod_{k=0}^{\ell-2} p(h^k|h^{k+1})\right)}{p(h^{\ell-2}, h^{\ell-1})\left(\prod_{k=0}^{\ell-3} p(h^k|h^{k+1})\right)} \\
&= \frac{p(h^{\ell-1}, h^\ell) p(h^{\ell-2}|h^{\ell-1})}{p(h^{\ell-2}, h^{\ell-1})} \\
&= \frac{p(h^{\ell-1}, h^\ell) p(h^{\ell-2}, h^{\ell-1})}{p(h^{\ell-2}, h^{\ell-1}) p(h^{\ell-1})} \\
&= p(h^\ell|h^{\ell-1}).
\end{aligned}$$

Bi-level model From Eq. (5), $\nabla_{\theta_{DBN}} log\, p(x)$ in the objective function is approximated for $i = 0, 1, \cdots, \ell$ as

$$\begin{aligned}
[\nabla_{\theta_{DBN}} log\, p(x)]_i &= \frac{\partial\, log\, p(x)}{\partial\, \theta^i_{DBN}} \\
&= \frac{\partial\, log\, \left(\sum_{h^1, h^2, \cdots, h^\ell} p(x, h^1, h^2, \cdots, h^\ell)\right)}{\partial\, \theta^i_{DBN}} \\
&\approx \frac{\partial\, log\, (\sum_{h^{i+1}} p(h^i, h^{i+1}))}{\partial\, \theta^i_{DBN}}
\end{aligned} \tag{6}$$

where $\theta_{DBN} = (\theta^0_{DBN}, \theta^2_{DBN}, \cdots, \theta^i_{DBN}, \cdots \theta^\ell_{DBN})$. The gradient of this approximated quantity is then the Hessian matrix of the underlying RBM.

7.2 Derivation of the gradient of the bi-level model

We write the approximated $||\nabla_{\theta_{DBN}} - log\, p(x)||^2$ at the layer i as

$$\begin{aligned}
||[\nabla_{\theta_{DBN}} - log\, p(x)]_i||^2 &\approx ||\frac{\partial\, -log\, (\sum_{h^{i+1}} p(h^i, h^{i+1}))}{\partial\, \theta^i_{DBN}}||^2 \\
&= \left[\left(\frac{\partial - log\, p(h^i)}{\partial \theta^i_{11}}\right)^2 + \left(\frac{\partial - log\, p(h^i)}{\partial \theta^i_{12}}\right)^2 + \cdots + \left(\frac{\partial - log\, p(h^i)}{\partial \theta^i_{nm}}\right)^2\right]
\end{aligned}$$

where m and n denote dimensions of h^i and h^{i+1} and θ^i_{pq} denotes the p^{th} and q^{th} component of the θ^i_{DBN}. The gradient of the approximated $||\nabla_{\theta_{DBN}} - log\, p(x)||^2$ at the layer i is

$$\begin{aligned}
\frac{\partial}{\theta^i_{pq}}\left(\sum_{p,q}\left(\frac{\partial - log\, p(h^i)}{\partial \theta^i_{pq}}\right)^2\right) &= 2\Big[\left(\frac{\partial - log\, p(h^i)}{\partial \theta^i_{11}}\right)\left(\frac{\partial^2 - log\, p(h^i)}{\partial \theta^i_{11}\theta^i_{pq}}\right) + \left(\frac{\partial - log\, p(h^i)}{\partial \theta^i_{12}}\right)\left(\frac{\partial^2 - log\, p(h^i)}{\partial \theta^i_{12}\partial \theta^i_{pq}}\right) + \\
&+ \left(\frac{\partial - log\, p(h^i)}{\partial \theta^i_{pq}}\right)\left(\frac{\partial^2 - log\, p(h^i)}{\partial \theta^i_{pq}\partial \theta^i_{pq}}\right) \cdots + \left(\frac{\partial - log\, p(h^i)}{\partial \theta^i_{nm}}\right)\left(\frac{\partial^2 - log\, p(h^i)}{\partial \theta^i_{nm}\theta^i_{pq}}\right)\Big] \quad \text{for } p = 1, ...n, q = 1, ...m
\end{aligned}$$

This shows that the gradient of the approximated $||\nabla_{\theta_{DBN}} - log\, p(x)||^2$ in (5) is then the Hessian matrix times the gradient of the underlying RBM. The stochastic gradient of $-log\, p(x)$ of RBM with binary input x and hidden unit h with respect to $\theta_{DBN} w_{pq}$ is

$$\frac{\partial RBM}{\partial w_{pq}} = p(h_p = 1|x)x_q - \sum_x p(x)p(h_p = 1|x)x_q$$

where RBM denotes $-log\, p(x)$ [6]. We derive the Hessian matrix with respect to w_{pq} as

$$\begin{aligned}
\frac{\partial^2 RBM}{\partial w_{pq}^2} &= \frac{\partial}{w_{pq}}[p(h_p = 1|x)x_q)] - \sum_x \frac{\partial}{w_{pq}}[p(x)p(h_p = 1|x)x_q)] \\
&= \sigma(\widetilde{net_p})(1 - \sigma(\widetilde{net_p}))x_q^2 - \sum_x [\frac{\partial p(x)}{\partial w_{pq}} p(h_p = 1|x)x_q + p(x)\sigma(\widetilde{net_p})(1 - \sigma(\widetilde{net_p}))x_q^2], \\
\frac{\partial^2 RBM}{\partial w_{pk} \partial w_{pq}} &= \frac{\partial}{w_{pk}}[p(h_p = 1|x)x_q)] - \frac{\partial}{w_{pk}}[\sum_x p(x)p(h_p = 1|x)x_q)] \\
&= \sigma(\widetilde{net_p})(1 - \sigma(\widetilde{net_p}))x_q x_k - \sum_x [\frac{\partial p(x)}{\partial w_{pk}} p(h_p = 1|x)x_q + p(x)\sigma(\widetilde{net_p})(1 - \sigma(\widetilde{net_p}))x_q x_k], \\
\frac{\partial^2 RBM}{\partial w_{kq} \partial w_{pq}} &= \frac{\partial}{w_{kq}}[p(h_p = 1|x)x_q)] - \frac{\partial}{w_{kq}}[\sum_x p(x)p(h_p = 1|x)x_q] \\
&= - \sum_x [\frac{\partial p(x)}{\partial w_{kq}} p(h_p = 1|x)x_q + p(x)\frac{\partial}{\partial w_{kq}}[p(h_p = 1|x)x_q]], \\
\frac{\partial^2 RBM}{\partial w_{kp} \partial w_{pq}} &= - \sum_x [\frac{\partial p(x)}{\partial w_{kp}} p(h_p = 1|x)x_q + p(x)]
\end{aligned}$$

where $\sigma()$ is the sigmoid function, $\widetilde{net_p}$ is $\sum_q w_{pq}x_q + c_p$, and c_p is the hidden bias. Based on what we derive above we can calculate the gradient of approximated $||[\nabla_{\theta_{DBN}} - log\, p(x)]_i||^2$.