Hindawi
Computational Intelligence and Neuroscience
Volume 2018, Article ID 6401645, 9 pages
https://doi.org/10.1155/2018/6401645

Research Article

A Reweighted Scheme to Improve the Representation of the Neural Autoregressive Distribution Estimator

Zheng Wang and Qingbiao Wu

School of Mathematical Sciences, Zhejiang University, HangZhou, Zhejiang, China

Correspondence should be addressed to Qingbiao Wu; qbwu@zju.edu.cn

Received 21 April 2018; Revised 29 October 2018; Accepted 21 November 2018; Published 23 December 2018

Academic Editor: Saeid Sanei

The neural autoregressive distribution estimator(NADE) is a competitive model for the task of density estimation in the field of machine learning. While NADE mainly focuses on the problem of estimating density, the ability for dealing with other tasks remains to be improved. In this paper, we introduce a simple and efficient reweighted scheme to modify the parameters of the learned NADE. We make use of the structure of NADE, and the weights are derived from the activations in the corresponding hidden layers. The experiments show that the features from unsupervised learning with our reweighted scheme would be more meaningful, and the performance of the initialization for neural networks has a significant improvement as well.

1. Introduction

Feature learning is one of the most important tasks in the field of machine learning. A meaningful feature representation could be the foundation of the other procedures. Among the various methods, the restricted Boltzmann machine (RBM), which is a powerful generative model, has shown its ability to learn useful representations from many different types of data [1, 2].

RBM models the higher-order correlations between dimensions of the input. It is often used as a feature extractor, or the building blocks of various deep models, for instance, deep belief nets. In the latter case, the learned representations are fed to another RBM in the higher layer, and the deep architecture often leads to better performance in many fields [3–5]. Its variants [6–8]also have the capability to deal with various kinds of tasks.

While RBM has lots of advantages, it is not suited for the problem of estimating distribution, in other words, estimating the joint probability of the observation. To estimate the joint probability of a given observation, a normalization constant must be computed, which is intractable even for a moderate size of input. To deal with the problem, some other ways must be used to approximate the normalization constant, for example, annealed importance sampling [9, 10], which is complex and computational costing.

The neural autoregressive distribution estimator (NADE) [11] is a powerful model for estimating the distribution of data, which is inspired by the mean-field procedure of RBM. Computing the joint probability under NADE can be done exactly and efficiently. NADE and its variants [12–17] have been shown to be state-of-the-art joint density models for a variety of datasets.

While NADE mainly focuses on the distribution of the data, it also can be regarded as an alternative model to extract features from data.

Reweight approaches have made a lot of achievements in the field of machine learning. In some models of ensemble learning, such as AdaBoost [18], the importance of each sample in dataset would be reweighted to achieve better results. In some deep generative models, reweight approaches have been proposed to adjust the importance weights for the procedure of importance sampling [19, 20]. With the reweight approaches, the estimation of the gradients would be more accurate.

In this paper, we deal with the feature learned by NADE and propose a novel method to improve the quality of the representation via a simple reweighted scheme of the weights learned by NADE. The proposed method remains the structure of the model, and the procedure of computation remains simple and tractable.

The remainder of the paper is structured as follows. In Section 2, we review the important architecture of RBM and

NADE, which is the foundation of our method and experiments. In Section 3, we introduce and analyze the reweighted scheme to improve the quality of features learned by NADE. In Section 4, we present a similar method for the case of initialization. We provide the experimental evaluation and demonstrate the results in Section 5. Finally, we make a conclusion in Section 6.

2. Review of RBM and NADE

In this section, we review the basic RBM model and emphasize the relationship between RBM and NADE.

A restricted Boltzmann machine is a kind of Markov random field that contains one layer of visible units $\mathbf{v} \in \{0,1\}^D$ and one layer of hidden units $\mathbf{h} \in \{0,1\}^H$. The two layers are connected with each other, and there are no connections intralayer.

The energy of the state $\{\mathbf{v}, \mathbf{h}\}$ is defined as

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i=1}^{D} b_i v_i - \sum_{i=1}^{D}\sum_{j=1}^{H} W_{ij} v_i h_j - \sum_{j=1}^{H} c_j h_j, \tag{1}$$

where $\{W_{ij}\}$ are the connecting weights between layers and $\{b_i, c_j\}$ are the biases of each layer.

The probability of a visible state is

$$p(\mathbf{v}; \theta) = \frac{1}{Z(\theta)} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}; \theta)), \tag{2}$$

where $Z(\theta) = \sum_{\mathbf{v},\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}; \theta))$ is the normalization constant.

Due to the intractability of the normalization constant, RBM is less competitive in the task of estimating distribution.

For a given observation, the distribution can be written as

$$p(\mathbf{v}) = \prod_{i=1}^{D} p\left(v_i \mid \mathbf{v}_{<i}\right), \tag{3}$$

where $\mathbf{v}_{<i}$ denotes the subvector of the observation before the i-th dimension. To evaluate the conditional distribution $p(v_i \mid \mathbf{v}_{<i})$, a factorial distribution $q(v_i, \mathbf{v}_{>i}, \mathbf{h} \mid \mathbf{v}_{<i})$ is used to approximate $p(\mathbf{v}_i, \mathbf{v}_{>i}, \mathbf{h} \mid \mathbf{v}_{<i})$:

$$\begin{aligned} q\left(v_i, \mathbf{v}_{>i}, \mathbf{h} \mid \mathbf{v}_{<i}\right) &= \mu_i(i)^{v_i} \left(1 - \mu_i(i)\right)^{1-v_i}, \\ &\prod_{j>i} \mu_j(i)^{v_j} \left(1 - \mu_j(i)\right)^{1-v_j}, \\ &\prod_{k} \tau_k(i)^{h_k} \left(1 - \tau_k(i)\right)^{1-h_k}. \end{aligned} \tag{4}$$

The minimization of the KL divergence between these two distributions leads to two important equations:

$$\begin{aligned} \tau_k(i) &= \mathrm{sig}\left(c_k + \sum_{j \geq i} W_{kj} \mu_j(i) + \sum_{j<i} W_{kj} v_j\right), \\ \mu_j(i) &= \mathrm{sig}\left(b_j + \sum_{k} W_{kj} \tau_k(i)\right), \end{aligned} \tag{5}$$

where $\mathrm{sig}(x) = 1/(1 + \exp(-x))$ is the sigmoid function.

The main structure of NADE is inspired by the mean-field procedure [21], and results in the following equations:

$$\begin{aligned} p\left(v_i = 1 \mid \mathbf{v}_{<i}\right) &= \mathrm{sig}\left(b_i + \left(\mathbf{V}^T\right)_{i,\cdot} \mathbf{h}_i\right), \\ \mathbf{h}_i &= \mathrm{sig}\left(\mathbf{c} + \mathbf{W}_{\cdot,<i} \mathbf{v}_{<i}\right), \end{aligned} \tag{6}$$

where $(\mathbf{V}^T)_{i,\cdot}$ represents the i-th row in the transpose of matrix $\mathbf{V}$ and $\mathbf{W}_{\cdot,<i}$ represents the first i-1 columns of matrix $\mathbf{W}$, which connects the input with the corresponding hidden layers.

These two equations indicate that NADE acts like a feed-forward neural network, and the training procedure of NADE can be cast into the same framework as the common neural network while the cost function is the average negative log-likelihood of the training set. The gradient of the cost function with respect to each parameter can be derived exactly by backpropagation, and the minimization of the cost function can be done using simple stochastic gradient descent. In contrast, the gradient with respect to each parameter in RBM must be approximated by sampling from Markov chains [22–27]. Experiments have shown that NADE often outperforms other models in the task of estimating distribution, while the performance of NADE in some other tasks such as the unsupervised learning of features and initialization of neural networks is not so excellent. In this paper, we mainly deal with these two problems.

3. A Reweighted Scheme for Features

The features are totally determinated by the learned weight $\mathbf{W}$ and the bias $\mathbf{c}$ wherever in RBM or NADE. To improve the features, we try to modify the corresponding parameters learned by the model while keeping the structure of NADE.

A direct idea is to take advantage of the conditional probability computed by NADE. Consider the probability of one dimension of the input conditioned on the other dimensions; to measure the importance of the specified dimension, we clamp the states of the other dimensions and simply compare the probabilities of two cases as follows:

$$\begin{aligned} \frac{p(v_i = 1, \mathbf{v}_{\neq i})}{p(v_i = 0, \mathbf{v}_{\neq i})} &= \frac{p\left(v_i = 1 \mid \mathbf{v}_{\neq i}\right) \cdot p(\mathbf{v}_{\neq i})}{p\left(v_i = 0 \mid \mathbf{v}_{\neq i}\right) \cdot p(\mathbf{v}_{\neq i})} \\ &= \frac{p\left(v_i = 1 \mid \mathbf{v}_{\neq i}\right)}{p\left(v_i = 0 \mid \mathbf{v}_{\neq i}\right)} = \omega_i. \end{aligned} \tag{7}$$

In this case, we define ω_i as the weight score for the i-th dimension of the input. Large or small value of ω_i indicates that the probabilities of these two cases vary drastically, and we should pay more attention to this specified dimension. This reweighted scheme never works in practice because of the huge amount of computation. For each dimension of every observation, we must compute two feed-forward procedures and it is impractical.

To deal with the problem, we approximate the conditional probabilities $p(v_i = 1 \mid \mathbf{v}_{\neq i})$ and $p(v_i = 0 \mid \mathbf{v}_{\neq i})$ by the fixed-order conditional probabilities $p(v_i = 1 \mid \mathbf{v}_{<i})$ and $p(v_i = 0 \mid \mathbf{v}_{<i})$, which is compatible with the original

structure of NADE. This approximation drastically reduces the cost of computation by a factor of H, which is the size of each hidden layer.

We further replace $\omega_i = (p(v_i = 1 \mid \mathbf{v}_{<i}))/(p(v_i = 0 \mid \mathbf{v}_{<i}))$ by $\omega_i = |p(v_i = 1 \mid \mathbf{v}_{<i}) - p(v_i = 0 \mid \mathbf{v}_{<i})|$ to control the instability of ω_i. Thus, we use each ω_i to modify the corresponding weight in the matrix $\mathbf{W}$, in other words, the i-th column of $\mathbf{W}$. We may pay more attention to the dimensions in which the probabilities change intensely in two different cases. These dimensions should have larger weights to generate the feature representation.

The final reweighted scheme is represented as

$$\omega_i = \left| p\left(v_i = 1 \mid \mathbf{v}_{<i}\right) - p\left(v_i = 0 \mid \mathbf{v}_{<i}\right) \right|, \tag{8}$$

$$\widetilde{\omega}_i = \begin{cases} \omega_i, & \omega_i < \tau, \\ \tau, & \text{otherwise}, \end{cases} \tag{9}$$

$$k = \sum_i^D \widetilde{\omega}_i, \tag{10}$$

$$\widehat{\omega}_i = \frac{D\widetilde{\omega}_i}{k}, \tag{11}$$

$$\mathbf{h} = \text{sig}\left(\mathbf{c} + \sum_i^D \widehat{\omega}_i \mathbf{W}_{\cdot,i} v_i \right), \tag{12}$$

where $\tau \in [0, 1]$ is the threshold to control the difference, D is the size of input, $\widehat{\omega}_i$ is the weight score of the i-th dimension, $\mathbf{W}_{\cdot,i}$ is the i-th column of $\mathbf{W}$, and $\mathbf{h}$ is the final reweighted feature.

In the reweighted procedure, equation (8) computes the difference between the probabilities in two cases for each dimension. Equation (9) controls the scale of the weight. Equations (10) and (11) normalize the weight.

While this reweighted scheme seems plausible, it seldom improves the features. It may be explained that the reweighted score does change the activation in each dimension of the feature $\mathbf{h}$, while it does not change the relative magnitude of the activations, which may be more important for a better representation.

In order to resolve this problem, we prefer to deal with the rows of $\mathbf{W}$ rather than columns, and we would again utilize the structure provided by NADE. For each dimension of the input, the NADE provides a corresponding hidden layer, which could be used to modify the learned features. In this case, we pay more attention to the dimensions of the hidden layers which are over saturated or inactivated. These ideas lead to the following new reweighted scheme:

$$\widetilde{\mathbf{h}} = \frac{\sum_i^D \mathbf{h}_i}{D}, \tag{13}$$

$$\widetilde{\omega}_j = \begin{cases} \varepsilon_{\text{upper}}, & \widetilde{h}_j > \tau_{\text{upper}}, \\ \varepsilon_{\text{lower}}, & \widetilde{h}_j < \tau_{\text{lower}}, \\ 1, & \text{otherwise}, \end{cases} \tag{14}$$

$$k = \sum_i^H \widetilde{\omega}_j, \tag{15}$$

$$\widehat{\omega}_j = \frac{H\widetilde{\omega}_j}{k}, \tag{16}$$

$$\widehat{c}_j = c_j \cdot \widehat{\omega}_j, \tag{17}$$

$$\widehat{\mathbf{W}}_{j,\cdot} = \widehat{\omega}_j \mathbf{W}_{j,\cdot}, \tag{18}$$

$$\mathbf{h} = \text{sig}\left(\widehat{\mathbf{c}} + \sum_i^D \widehat{W}_{\cdot,i} v_i \right), \tag{19}$$

where D is the size of input, $\mathbf{h}_i$ is the i-th hidden layer in NADE, $\{\varepsilon_{\text{upper}}, \varepsilon_{\text{lower}}\}$ are the relative weights, $\{\tau_{\text{upper}}, \tau_{\text{lower}}\}$ are thresholds which control the value of activation, $\widetilde{h}_j$ is the j-th unit in the normalized hidden layer $\widetilde{\mathbf{h}}$, and $\widehat{\mathbf{W}}_{j,\cdot}$ and $\mathbf{W}_{j,\cdot}$ represent the j-th row for each matrix.

We conclude the reweighted procedure in Algorithm 1.

This reweighted scheme for features deserves explanation a bit more. As the activations of each hidden layer form a corresponding vector of the same size, in step one, we sum the vectors and normalize it to obtain the result $\widetilde{\mathbf{h}}$. Thus, $\widetilde{\mathbf{h}}$ is the average value of the activations for each dimension of the hidden layer, and it is a measure for how activated the dimension is during the feed-forward procedure in NADE.

We then introduce two thresholds $\{\tau_{\text{upper}}, \tau_{\text{lower}}\}$ to control the activations. The unit is considered to be over saturated if the activation is larger than the upper threshold τ_{upper}, and the corresponding dimension of this unit is endowed with a weight $\varepsilon_{\text{upper}}$. Similarly, we give a weight $\varepsilon_{\text{lower}}$ to the dimensions where the activations are smaller than the lower threshold τ_{lower}. It should be noted that the weight $\varepsilon_{\text{lower}}$ and $\varepsilon_{\text{upper}}$ are relative values compared with the standard value 1. In practice, the weight $\varepsilon_{\text{upper}}$ should be smaller than 1 while $\varepsilon_{\text{lower}}$ should be larger than 1. We emphasize the importance of this procedure. The over saturated unit often affects the performance, and via this step, we set a smaller weight to alleviate this situation. While units with activation values close to zero are considered to be inactivated, these units should be kept inactivated, and some other units may even become inactivated after reweight. In this case, $\mathbf{W}_{j,\cdot}\mathbf{v} + c_j$ is negative, and a large weight for this dimension confirms the situation. According to our point of view, this procedure forces the sparsity of the representation, which often leads to better performance.

We assume that the original reweighted score for each dimension is just one and normalizes the reweighted score to keep it reasonable.

Here, we emphasize that our aim is to improve the features. When we meet the problem of estimating distribution, the original weight of NADE should be utilized since the original weight is the optimal result for the maximum likelihood cost function. Our scheme is unsuitable for density estimation.

Given: An observation $\mathbf{v}$, weight matrix $\mathbf{W}$, and bias $\mathbf{c}$ from the trained NADE
Output: New feature corresponds to the observation
(1) Sum the activations of all hidden layers and normalize it to get a vector $\widetilde{\mathbf{h}}$ using equation (13)
(2) For each dimension j in vector $\widetilde{\mathbf{h}}$, compare $\widetilde{h}_j$ with the thresholds $\{\tau_{\text{upper}}, \tau_{\text{lower}}\}$ and set the corresponding weight to $\widetilde{\omega}_j$ using equation (14)
(3) Normalize the weight scores using equations (15) and (16)
(4) Modify $\{\mathbf{W}, \mathbf{c}\}$ using equations (17) and (18)
(5) Compute the new feature using equation (19)
(6) Choose the best feature by validation.

Algorithm 1: A reweighted scheme for new feature.

4. A Reweighted Scheme for Initialization

The weight learned by RBM or NADE can be used to initialize the weight of another neural network, which is one of the advantages of this kind of models. The further neural network may be used in other tasks such as classification.

We have proposed the reweighted scheme for features, which is applicable for each observation, while this method is unsuitable for initialization.

To solve this problem, we compute the reweighted score for each sample in the training set and take the average of them to obtain a new reweighted score for the weight matrix and bias. This procedure can be represented as

$$\widehat{\boldsymbol{\omega}} = \frac{\sum_{k=1}^{N} \widehat{\boldsymbol{\omega}}_k}{N}, \tag{20}$$

$$\widehat{c}_j = c_j \cdot \widehat{\omega}_j, \tag{21}$$

$$\widehat{\mathbf{W}}_{j,\cdot} = \widehat{\omega}_j \mathbf{W}_{j,\cdot}, \tag{22}$$

where N is the number of samples in the training set, $\widehat{\boldsymbol{\omega}}_k$ is the reweighted score vector corresponding to the k-th training sample.

The complete process is concluded in Algorithm 2.

5. Experiments

In this section, we show the experimental results on several binary datasets with the reweighted scheme for both features and initialization. For the training procedure of NADE, a fixed order of dimensions of the input must be chosen in the beginning. Since experiments have shown that the ordering does not have a significant impact on the performance of NADE [11], for each dataset, the ordering is chosen independently and is kept the same during all the experiments on it. Furthermore, hyperparameters of NADE remain unchanged in order to select the hyperparameters of the reweighted scheme. Our implementation of the NADE model is based on the code provided by Larochelle and Murray [11].

5.1. Results on Learned Features. To test whether the reweighted scheme has improved the learned features, we perform some experiments on classification.

We note that our main purpose is to evaluate the proposed reweighted scheme rather than pursuing the best performance for classification, and we only use a moderate size of model to reduce the cost of computation. For each dataset, we first train a NADE and use Algorithm 1 to obtain the improved features. This procedure is processed for all the samples in training set, validation set, and test set which results in all new three corresponding sets. We then train a neural network with single hidden layer as the classifier on the learned features. The performance is measured by the classification error rate on test set. We further experiment on the features without reweighted scheme to obtain a standard result for comparison. A RBM with the same size of NADE is also trained and the classification result is used as reference.

We experiment on twelve different datasets from the UCI repository: Adult, Binarized-MNIST, Connect-4, Convex, DNA, Mushrooms, Newsgroups, OCR-letters, RCV1, Rectangles, SVHN, and Web. We list the details about these datasets in Table 1. The experimental results are shown in Table 2. We have chosen the best result for reweighted scheme among the results corresponding to different hyperparameters. We find that the classification error for features with reweighted scheme is lower than the one without reweighted, which proves the improvement on the original features. Features from the reweighted scheme may be more meaningful.

To further verify our method, we replace the neural network classifier with SVM, RandomForest, and AdaBoost and perform additional experiments. These experiments are implemented via LIBSVM [28] and scikit-learn. The results are shown in Tables 3–5. During all the experiments, the parameters of the classifiers have been optimized by grid search and validation which would give the best performance. The features with our reweighted scheme again outperform the original ones, and it confirms the effectiveness of our method.

Experimental results for different weights $\{\varepsilon_{\text{upper}}, \varepsilon_{\text{lower}}\}$ on OCR-letters dataset are shown in Table 6. In this series of experiments, we train a NADE on the dataset at first, the learning rate is set to 0.001, the decrease constant is set to 0, the size of hidden layer is 100, and we use tied weight in NADE. That is, we set $\mathbf{V} = \mathbf{W}$ in equation (6). Next, we keep $\tau_{\text{upper}} = 0.73$ and $\tau_{\text{lower}} = 0.27$ and only modify the reweighted parameters to explore the performance of the reweighted scheme. The results have demonstrated that the reweighted scheme has a decisive role in improving the features. An unreasonable reweighted scheme often leads to

Given: A set of training samples $\mathbf{v}_k$, weight matrix $\mathbf{W}$, and bias $\mathbf{c}$ from the trained NADE
Output: A reweighted new weight matrix $\widehat{\mathbf{W}}$ and bias $\widehat{\mathbf{c}}$ for pretraining the neural network
(1) For training sample $\mathbf{v}_k$, compute the corresponding reweighted score vector using equations (13)–(16)
(2) Using equation (20) to compute the final reweighted score vector
(3) Using equations (21) and (22) to obtain $\widehat{\mathbf{W}}$ and $\widehat{\mathbf{c}}$
(4) Choose the best weight matrix and bias by validation.

Algorithm 2: A reweighted scheme for initialization.

Table 1: Details about the twelve datasets.

Dataset	Input size	Number of classes	Training	Validation	Testing
Adult	123	2	5000	1414	26147
Binarized-MNIST	784	10	50000	10000	10000
Connect-4	126	3	16000	4000	47557
Convex	784	2	6000	2000	50000
DNA	180	3	1400	600	1186
Mushrooms	112	2	2000	500	5624
Newsgroups	5000	20	9578	1691	7505
OCR-letters	128	26	32152	10000	10000
RCV1	150	2	40000	10000	150000
Rectangles	784	2	1000	200	50000
SVHN	1024	11	594388	10000	26032
Web	300	2	14000	3188	32561

Table 2: Classification result via neural network on twelve datasets.

Dataset	Input size	Feature size	Size of NN	NADE	Reweighted-NADE	RBM
Adult	123	100	200	0.16074	**0.16013**	0.16598
Binarized-MNIST	784	300	400	0.0247	**0.0243**	0.0250
Connect-4	126	100	200	0.22436	**0.22143**	0.23367
Convex	784	300	400	0.27742	**0.26124**	0.29242
DNA	180	100	200	0.15683	**0.15008**	0.15093
Mushrooms	112	100	200	0.0082	**0.0059**	0.0066
Newsgroups	5000	1000	2000	0.30286	**0.26156**	0.30007
OCR-letters	128	100	200	0.1459	0.1409	**0.1361**
RCV1	150	100	200	0.05461	**0.04674**	0.06034
Rectangles	784	300	400	0.09566	**0.08782**	0.10204
SVHN	1024	300	400	0.08455	**0.08025**	0.09050
Web	300	150	200	0.02189	**0.02079**	0.02515

Table 3: Classification result via SVM on twelve datasets.

Dataset	NADE + SVM	Reweighted-NADE + SVM	RBM + SVM
Adult	0.15876	**0.15753**	0.16495
Binarized-MNIST	0.0374	**0.0349**	0.0358
Connect-4	0.27130	**0.26680**	0.28696
Convex	0.30036	**0.28570**	0.31466
DNA	0.13406	**0.12985**	0.14418
Mushrooms	0.01085	**0.00871**	0.00996
Newsgroups	0.29714	**0.26862**	0.28208
OCR-letters	0.1851	0.1802	**0.1692**
RCV1	0.06041	**0.05325**	0.07041
Rectangles	0.10836	**0.09864**	0.11526
SVHN	0.09346	**0.08839**	0.09842
Web	0.02110	**0.01969**	0.02420

Table 4: Classification result via Random Forest on twelve datasets.

Dataset	NADE + RF	Reweighted-NADE + RF	RBM + RF
Adult	0.16399	**0.16069**	0.16587
Binarized-MNIST	0.0301	**0.0265**	0.0312
Connect-4	0.26783	**0.25443**	0.28364
Convex	0.29632	**0.28502**	0.30658
DNA	0.16948	**0.15346**	0.15936
Mushrooms	0.01227	**0.01014**	0.01174
Newsgroups	0.32192	**0.29460**	0.31459
OCR-letters	0.1986	0.1863	**0.1708**
RCV1	0.07265	**0.06249**	0.08258
Rectangles	0.11816	**0.10624**	0.12646
SVHN	0.09838	**0.09339**	0.10283
Web	0.02647	**0.02368**	0.02773

Table 5: Classification result via AdaBoost on twelve datasets.

Dataset	NADE + AB	Reweighted-NADE + AB	RBM + AB
Adult	0.16013	**0.15677**	0.16445
Binarized-MNIST	0.0313	**0.0274**	0.0309
Connect-4	0.23912	**0.22737**	0.25494
Convex	0.29420	**0.28206**	0.30082
DNA	0.14250	**0.13238**	0.14587
Mushrooms	0.01049	**0.00853**	0.00978
Newsgroups	0.30753	**0.28115**	0.29474
OCR-letters	0.1681	0.1571	**0.1432**
RCV1	0.06847	**0.06072**	0.07509
Rectangles	0.10522	**0.09476**	0.11762
SVHN	0.08866	**0.08374**	0.09438
Web	0.02279	**0.02147**	0.02512

Table 6: Classification result for different weights on OCR-letters.

Upper-weight	Lower-weight	Error
0.8	0.8	0.1477
0.9	0.9	0.1472
1.0	1.0	0.1459
1.1	1.1	0.1445
1.2	1.2	0.1459
1.1	0.9	0.1470
1.2	0.8	0.1465
0.9	1.1	0.1447
0.8	1.2	0.1427
0.7	1.3	0.1425
0.6	1.4	**0.1411**

a worse result than the one without reweighted scheme. We have found that setting the lower weight $\varepsilon_{\text{lower}}$ larger than 1 and the upper weight $\varepsilon_{\text{upper}}$ smaller than 1 seems to be a reasonable reweighted scheme. In the previous sections, we have already explained this manner that a smaller value for upper weight makes the over saturated unit to be less saturated, which is beneficial for the representation, while a larger value for lower weight preserves the inactivated unit and forces the feature to be sparse.

It should also be noted that the weights $\{\varepsilon_{\text{upper}}, \varepsilon_{\text{lower}}\}$ are relative weight compared with the standard weight 1. Thus, the weight must be controlled and a too large or too small weight leads to a terrible result.

Another factor which influences the performance of the reweighted scheme is the thresholds $\{\tau_{\text{upper}}, \tau_{\text{lower}}\}$ that explicitly control the unit to be saturated or inactivated. Results for different thresholds $\{\tau_{\text{upper}}, \tau_{\text{lower}}\}$ on OCR-letters dataset are shown in Table 7. As the same as what we have done before, we only modify these two thresholds during this series of experiments, and we set the upper weight to 0.6 and the lower weight to 1.4. Results have also demonstrated the importance of the thresholds. On the one hand, the upper threshold controls the proportion of units which are seen to be over saturated, and a larger value of the upper threshold leads to a smaller proportion of these units. On the other hand, the lower threshold controls the proportion of units which are seen to be inactivated, and a smaller value of the lower threshold

Table 7: Classification result for different thresholds on OCR-letters.

Upper-threshold	Lower-threshold	Error
0.55	0.45	0.1434
0.58	0.42	0.1434
0.61	0.39	0.1447
0.64	0.36	0.1431
0.67	0.33	**0.1409**
0.70	0.30	0.1424
0.73	0.27	0.1411
0.76	0.24	0.1418
0.79	0.21	0.1433

leads to a smaller proportion of these units. These units would be even more inactivated after the reweighted scheme.

From our point of view, these two thresholds depend more on the dataset rather than the specification. Still, as a conservative strategy, we prefer to set the upper threshold in the range from 0.5 to 0.8, while the lower threshold in the range from 0.5 to 0.2.

To further investigate the features, we examine and analyze the value of activations of all the features in test set of OCR-letters. Figure 1 shows the number of units corresponding to each value of activations from 0 to 1 with a step of 0.01. Units whose value of activation under 0.01 are ignored to keep the figure balance since these units make up a large majority of all units. The number of these units before reweight is 562453 and 575928 after reweight, which proves that the policy we proposed does keep the inactivated units and does even force the features more sparse. A significant decrease of the over saturated units is shown in figure, which accords with our purpose.

We also investigate the average value of activation for each dimension in the feature. The results are shown in Figure 2. We find that in the NADE features after reweight, the over saturated dimensions are restrained, while the inactivated dimensions are kept or even more inactivated. The average features before and after reweight are similar while the NADE features and RBM features vary dramatically. The difference between NADE features and RBM features is due to the intrinsic difference between the model NADE and RBM.

5.2. Results on Initialization. The reweighted scheme we have proposed also improves the performance of the neural network by initialization, which we would show here.

To test the performance, we train a NADE for each dataset, and a RBM with same size is also trained. We then use the learned weight matrix $\mathbf{W}$ and the bias $\mathbf{c}$ to initialize the parameters of the neural network classifier. Then, the neural network classifier is trained on the corresponding dataset. To evaluate our reweighted scheme, the parameters after reweight are also used as the initialization of another neural network classifier with same size. Finally, the performance is measured by the classification error.

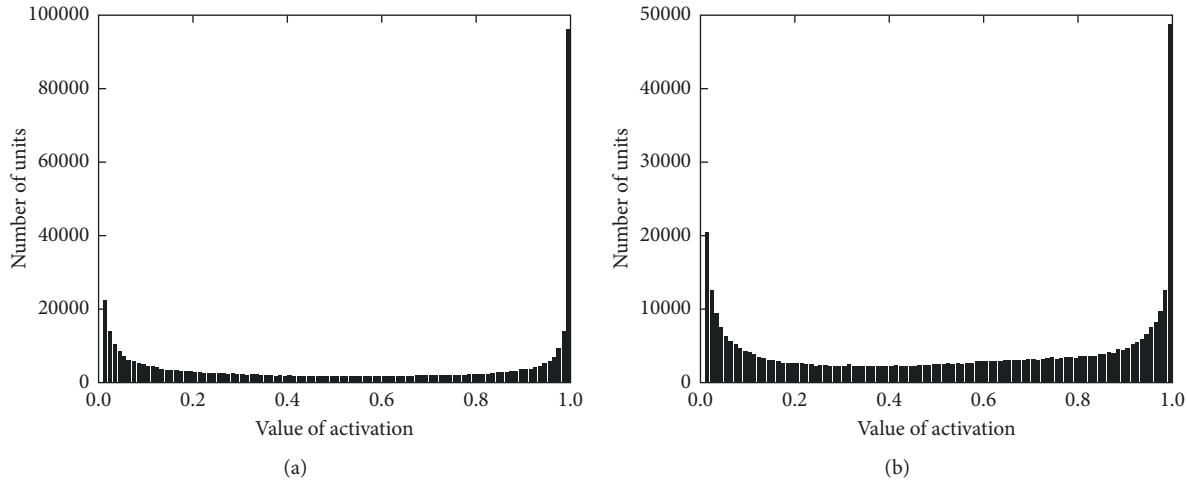

Figure 1: The number of units corresponding to each value of activations on OCR-letters. (a) Features before reweight. (b) Features after reweight.

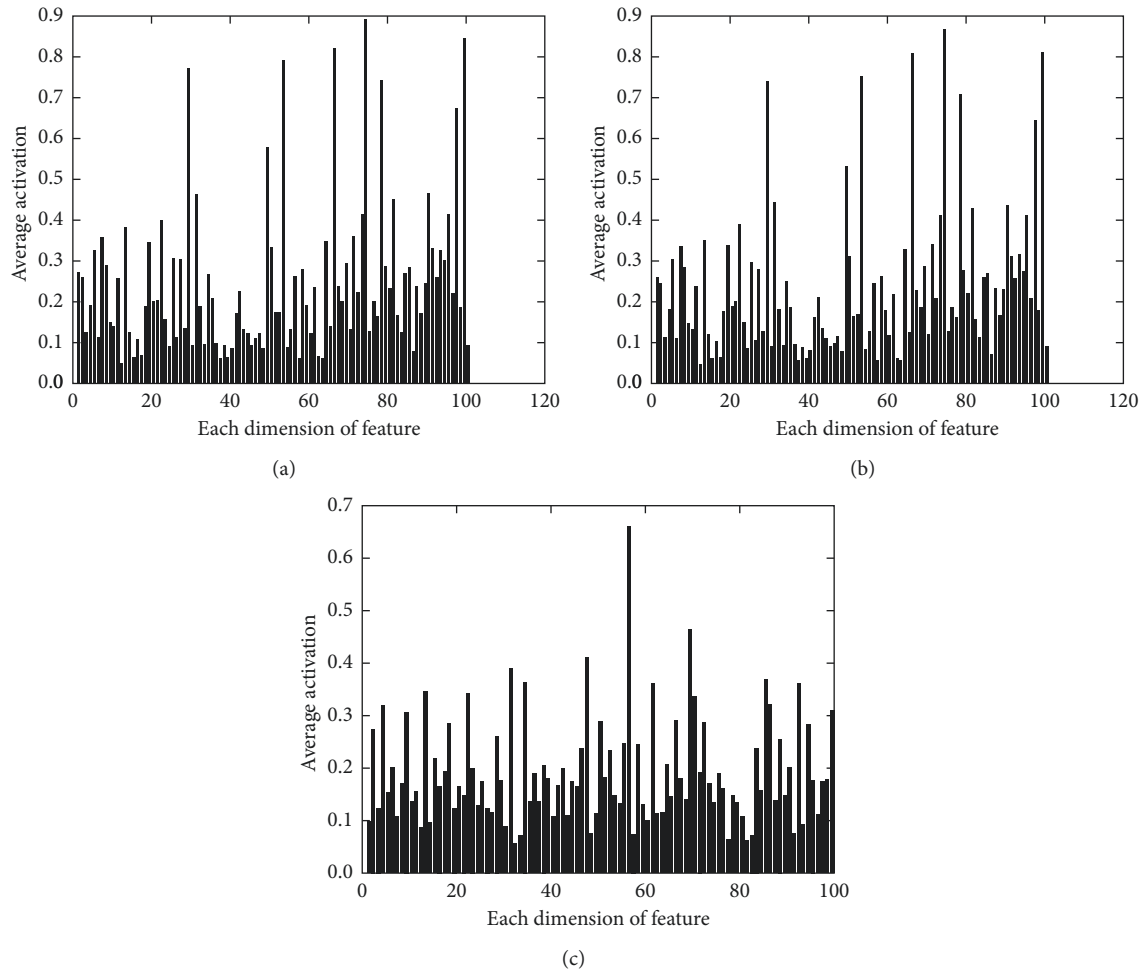

Figure 2: Average value of activations for each dimension on OCR-letters. (a) Features before reweight. (b) Features after reweight. (c) Features from RBM.

We have shown the results in Table 8. As before, we perform experiments on the same twelve datasets and with the same hyperparameters. This time, from the results, we could see that the reweighted scheme for initialization has made a more significant improvement on the classification performance compared with the original NADE parameters.

Table 8: Classification result for initialization on twelve datasets.

Dataset	Input size	Hidden layer size	NADE	Reweighted-NADE	RBM
Adult	123	100	0.15921	**0.15814**	0.16009
Binarized-MNIST	784	300	0.0216	**0.0198**	0.0205
Connect-4	126	100	0.19326	**0.18474**	0.19725
Convex	784	300	0.25672	**0.23898**	0.26394
DNA	126	100	0.06998	**0.05986**	0.06324
Mushrooms	112	100	0.00729	**0.00498**	0.00605
Newsgroups	5000	1000	0.34231	**0.28767**	0.3291
OCR-letters	128	100	0.1669	0.1601	**0.1374**
RCV1	150	100	0.05131	**0.04449**	0.05567
Rectangles	784	300	0.09062	**0.08338**	0.09948
SVHN	1024	300	0.08405	**0.07810**	0.08889
Web	300	150	0.01477	**0.01366**	0.01492

In most of the datasets, the difference between the errors of the reweighted-NADE and NADE is much larger than the one between NADE and RBM, which demonstrates the efficiency of the proposed reweighted scheme. In OCR-letters, the classification performance for reweighted-NADE is not as good as RBM, and this can be explained as the inherent difference between the parameters learned by NADE and RBM, which is hard to eliminate only via reweighted scheme. Anyhow, the proposed scheme always surpasses the one without reweighted.

In order to make the experiments more complete, we experiment on various weights $\{\varepsilon_{\text{upper}}, \varepsilon_{\text{lower}}\}$ on the Web dataset and the results are shown in Table 9. For NADE, on this dataset, the learning rate is set to 0.005, the decrease constant is set to 0, the size of hidden layer is 150, and the weight is untied. Upper threshold and lower threshold are kept to 0.67 and 0.33. The heuristic reweighted method, which sets the lower weight larger than 1 and the upper weight smaller than 1, once again proved to be effective. While this time, the proper weights are more far away from the standard weight 1. This can be explained by the effect of the average. Since we compute the average of the reweighted score of all the training samples, a more discriminated reweighted scheme maintains the differences among the dimensions in the final reweighted score vector. In other words, we prefer a larger value for lower weight and a smaller value for upper weight when dealing with the problem of initialization.

Results about various thresholds on dataset Web are shown in Table 10. Upper weight and lower weight are set to 0.6 and 1.4, respectively. We make a similar conclusion to the one in previous section. The thresholds depend more on the dataset and we prefer a conservative strategy.

Table 9: Classification result with initialization for different weights on web.

Upper-weight	Lower weight	Error
0.8	0.8	0.01409
0.9	0.9	0.01477
1.0	1.0	0.01477
1.1	1.1	0.01480
1.2	1.2	0.01480
1.1	0.9	0.01446
1.2	0.8	0.01486
0.9	1.1	0.01477
0.8	1.2	0.01471
0.7	1.3	0.01468
0.6	1.4	0.01418
0.5	1.5	0.01455
0.4	1.6	0.01375
0.3	1.7	**0.01366**
0.2	1.8	0.01437

Table 10: Classification result with initialization for different thresholds on web.

Upper-threshold	Lower threshold	Error
0.55	0.45	0.01425
0.58	0.42	0.01434
0.61	0.39	0.01449
0.64	0.36	0.01446
0.67	0.33	**0.01418**
0.70	0.30	0.01431
0.73	0.27	0.01464
0.76	0.24	0.01446
0.79	0.21	0.01452

6. Conclusion

In this paper, we have proposed a simple and novel reweighted scheme to modify the learned parameters of NADE. We make use of the activations in hidden layers of the learned NADE model and set appropriate thresholds to control the proportions of the over saturated and inactivated units. In order to achieve better results, a heuristic reweighted method is proposed. The original parameters are modified and normalized. The reweighted parameters are used to generate better features or to improve the performance of the initialization for a neural network. The experiments have shown the effectiveness of the reweighted scheme, and there are evident improvements in both two important tasks in the field of machine learning.

Data Availability

All the datasets used in this paper are publicly available and could be obtained from http://archive.ics.uci.edu/ml/datasets.html.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (Grant nos. 11771393 and 11632015) and Zhejiang Natural Science Foundation (Grant no. LZ14A010002).

References

[1] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[2] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[3] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[4] R. Salakhutdinov and I. Murray, "On the quantitative analysis of deep belief networks," in *Proceedings of 25th International Conference on Machine Learning*, pp. 872–879, ACM, Helsinki, Finland, July 2008.

[5] J. Schmidhuber, "Deep learning in neural networks: an overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[6] C. Marc-Alexandre and H. Larochelle, "An infinite restricted boltzmann machine," *Neural Computation*, vol. 28, no. 7, pp. 1265–1288, 2016.

[7] A. C. Courville, B. James, and Y. Bengio, "A spike and slab restricted Boltzmann machine," in *Proceedings of AISTATS*, vol. 1, p. 5, Fort Lauderdale, FL, USA, October 2011.

[8] G. E. Hinton and R. R. Salakhutdinov, "Replicated softmax: an undirected topic model," in *Proceedings of 22nd International Conference on Neural Information Processing Systems*, pp. 1607–1614, Vancouver, MB, Canada, December 2009.

[9] Y. Burda, R. B. Grosse, and R. Salakhutdinov, "Accurate and conservative estimates of MRF log-likelihood using reverse annealing," 2015, https://arxiv.org/abs/1412.8566.

[10] R. M. Neal, "Lingua::EN::Titlecase," *Statistics and Computing*, vol. 11, no. 2, pp. 125–139, 2001.

[11] H. Larochelle and I. Murray, "The neural autoregressive distribution estimator," in *Proceedings of AISTATS*, vol. 1, p. 2, Fort Lauderdale, FL, USA, October 2011.

[12] H. Larochelle and S. Lauly, "A neural autoregressive topic model," in *Proceedings of 22nd International Conference on Neural Information Processing Systems*, pp. 2708–2716, Lake Tahoe, Nevada, December 2012.

[13] I. Murray and R. R. Salakhutdinov, "Evaluating probabilities under high-dimensional latent variable models," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 1137–1144, Vancouver, MB, Canada, December 2009.

[14] T. Raiko, L. Yao, K. Cho, and Y. Bengio, "Iterative neural autoregressive distribution estimator nade-k," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 325–333, Montreal, QC, Canada, 2014.

[15] B. Uria, I. Murray, and H. Larochelle, "Rnade: the real-valued neural autoregressive density-estimator," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 2175–2183, Lake Tahoe, NV, USA, December 2013.

[16] Y. Zheng, R. S. Zemel, Y.-J. Zhang, and H. Larochelle, "A neural autoregressive approach to attention-based recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 67–79, 2014.

[17] Y. Zheng, Yu-J. Zhang, and H. Larochelle, "Topic modeling of multimodal data: an autoregressive approach," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1370–1377, Columbus, OH, USA, June 2014.

[18] Y. FreundR. E. Schapire et al., "Experiments with a new boosting algorithm," in *Proceedings of 13th International Conference on Machine Learning*, vol. 96, pp. 148–156, Bari, Italy, July 1996.

[19] J. . Bornschein and Y. Bengio, "Reweighted wake-sleep," 2014, https://arxiv.org/abs/1406.2751.

[20] Y. Burda, R. Grosse, and R. Salakhutdinov, "Importance weighted autoencoders," 2015, https://arxiv.org/abs/1509.00519.

[21] L. K. Saul, T. Jaakkola and M. I. Jordan, Mean field theory for sigmoid belief networks," *Journal of Artificial Intelligence Research*, vol. 4, no. 1, pp. 61–76, 1996.

[22] K. H. Cho, T. Raiko, and I. Alexander, "Parallel tempering is efficient for learning restricted Boltzmann machines," in *Proceedings of 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, Barcelona, Spain, July 2010.

[23] G. Hinton, "A practical guide to training restricted Boltzmann machines," *Momentum*, vol. 9, no. 1, p. 926, 2010.

[24] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[25] J. Martens and I. Sutskever, "Parallelizable sampling of markov random fields," in *Proceedings of AISTATS*, pp. 517–524, Sardinia, Italy, May 2010.

[26] R. R. Salakhutdinov, "Learning in markov random fields using tempered transitions," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 1598–1606, Vancouver, MB, Canada, December 2009.

[27] T. Tieleman, "Training restricted Boltzmann machines using approximations to the likelihood gradient," in *Proceedings of 25th International Conference on Machine Learning*, pp. 1064–1071, ACM, Helsinki, Finland, July 2008.

[28] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, 2001.