This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2689790, IEEE Transactions on Big Data

A Generative Model for Sparse Hyperparameter Determination

Zhiqiang Wan, *Student Member, IEEE,* Haibo He, *Senior Member, IEEE,* and Bo Tang, *Student Member, IEEE*

Abstract—Sparse autoencoder is an unsupervised feature extractor and has been widely used in the machine learning and data mining community. However, a sparse hyperparameter has to be determined to balance the trade-off between the reconstruction error and the sparsity of sparse autoencoder. Traditional sparse hyperparameter determination method is time-consuming, especially when the dataset is large. In this paper, we derive a generative model for sparse autoencoder. Based on this model, we derive a formulation to determine the sparse hyperparameter effectively and efficiently. The relationship between the sparse hyperparameter and the average activation of sparse autoencoder hidden units is also presented in this paper. Experimental results and comparative studies over numerous datasets demonstrate the effectiveness of our method to determine the sparse hyperparameter.

Index Terms—Sparse autoencoder, sparse hyperparameter, feature extractor, generative model.

1 Introduction

Suitable feature representation of input data is important for the performance of machine learning algorithms [1]. When developing machine learning algorithms, researchers normally spend much time designing or selecting suitable features to improve the learning performance. Although this "feature engineering" process, in general, can leverage domain knowledge to improve the learning performance, it is typically labor-intensive [2]. Therefore, it is highly desirable for machine learning algorithms to learn features autonomously from input data.

In general, supervised learning method and unsupervised learning method are two major categories of approaches to learn features from input data. Supervised learning method requires label information while unsupervised learning method does not. Convolutional neural network (CNN), which is a locally connected neural network, is a supervised learning method to extract features. It has been implemented in image processing [3], [4], [5], video processing [6], and natural language processing [7]. Autoencoder, on the other hand, is an unsupervised learning method with a fully connected neural network to extract features [8], [9]. It is trained to reconstruct input data at its output layer [10], and the output of its hidden units can be treated as features of input data.

Recently, many autoencoder variants have been proposed. For instance, denoising autoencoder (DAE), widely used in machine learning community [11], [12], [13], is trained by reconstructing original undistorted data from partially corrupted input data. The learned features are robust to partial corruption of the input pattern [14]. Despite corrupting input data, imposing penalty term on autoencoder is also an effective way to learn robust features. For example, contractive autoencoder (CAE) imposes the Frobenius norm of the Jacobian matrix of the encoder activation on autoencoder [15]. This penalty term penalizes features which are sensitive to small input variations. Autoencoders have also been used for feature learning for imbalanced data processing [16], [17]. For instance, in [18], a stacked dual autoencoder approach with different activation functions has been developed to capture different characteristics of the imbalanced data for improved classification performance.

Sparse autoencoder can learn structured features by imposing sparse penalty on hidden units of autoencoder [19], [20]. Introducing sparsity is an efficient way to represent information with a small number of activated hidden units [21], [22]. Besides, sparse structure has been observed in natural images which can generally be represented in terms of a small number of features [23]. When natural images are filtered with log-Gabor filters, the output distributions typically show sparse structures [24].

Sparse autoencoder has been widely used in the machine learning and data mining community. For instance, Poultney *et al.* [25] use sparse autoencoder to extract stroke-like features from handwriting digits dataset. Le *et al.* [26] propose a nine-layer locally connected sparse autoencoder to learn face features from unlabeled data. These features are robust to translation and scaling. Chen *et al.* [27] use a stacked sparse autoencoder to identify and classify overvoltage waveforms in power distribution systems. Sparse autoencoder can autonomously extract discriminative features from the ferroresonance overvoltage waveforms. Tao *et al.* [28] propose a stacked sparse autoencoder to learn spectral-spatial features from hyperspectral imagery which is intrinsically defined in spectral and spatial domains.

Although sparse autoencoder has been widely used, a sparse hyperparameter has to be determined to balance the trade-off between the reconstruction error and the sparsity of sparse autoencoder. Many researchers have explored how to determine the sparse hyperparameter. For example, Chen

• *This work was supported by National Science Foundation (NSF) under grant ECCS 1053717 and CCF 1439011. Corresponding author: H. He, Tel: 1-401-874-5844; Fax: 1-401-782-6422.*
Z. Wan, H. He and B. Tang were with the Department of Electrical, Computer and Biomedical Engineering, University of Rhode Island, 02881. E-mail: {zwan, he, btang}@ele.uri.edu

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2689790, IEEE Transactions on Big Data

et al. [27] and Sun *et al.* [29] search for the sparse hyperparameter within a large interval. Unlike these searching methods, Gong *et al.* [30] introduce a multiobjective evolutionary algorithm to determine the sparse hyperparameter. The evolutionary algorithm obtains a set of Pareto optimal solutions and selects a solution from the knee areas on the Pareto optimal front. Although the above methods can choose a suitable sparse hyperparameter, they are time-consuming, especially when the dataset is large. Besides, they do not show the relationship between the sparse hyperparameter and the sparsity.

In this paper, in order to efficiently determine the sparse hyperparameter, a generative model is derived for sparse autoencoder from a probabilistic perspective. In the generative model, the distribution of the reconstruction error follows Gaussian distribution, and the distribution of sparse autoencoder hidden units follows a sparse prior distribution. This distribution has high probability near zero and low probability anywhere else. It is introduced by Olshausen *et al.* [31] for sparse coding where the linear combination of an overcomplete basis set, which follows the sparse distribution, is used to represent an image. The shape of the sparse distribution is determined by a coefficient, and its expectation can be expressed in terms of this coefficient. Therefore, we can derive the relationship between this coefficient and the average activation of hidden units. Based on this relationship, a knee region is defined to choose this coefficient. Finally, the sparse hyperparameter can be determined by this coefficient and the variance of the reconstruction error.

The main contributions of this paper are in twofold. First, we derived the relationship between the sparse hyperparameter and the sparsity of sparse autoencoder. Second, we proposed an effective method to determine the sparse hyperparameter.

The rest of this paper is organized as follows. In Section 2, we derive a generative model for sparse autoencoder and propose a sparse hyperparameter determination method. In Section 3, we evaluate the validity of our method to estimate the average activation of hidden units. In Section 4, we evaluate the validity of our method to determine the sparse hyperparameter. Section 5 is a conclusion.

2 Sparse Autoencoder Model

2.1 Structure of autoencoder

Autoencoder, which is a fully connected neural network, can learn features autonomously from unlabeled input data. The structure of autoencoder is shown in Fig. 1. The encoder encodes input data into hidden layer while the decoder decodes information from hidden layer to output layer. The output of hidden units can be treated as features of input data.

The output of hidden units is shown in Eq.(1) where $W^{'}$ is the weight matrix, and $b^{'}$ is the bias vector. In Eq.(1), $x^{(j)}$ denotes a sample of input data, and $a^{(j)}$ denotes the output of hidden units. The activation function $f(t)$ is the sigmoid function where $f(t) = 1/\left[1 + exp(-t)\right]$.

$$a^{(j)} = f\left(W^{'}x^{(j)} + b^{'}\right) \tag{1}$$

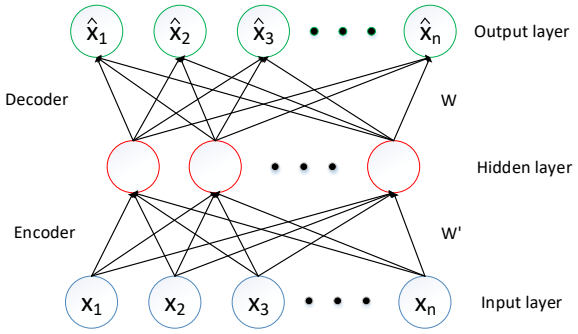


Fig. 1. Structure of autoencoder

The output of autoencoder is shown in Eq.(2) where W is the weight matrix, and b is the bias vector. In Eq.(2), $\hat{x}^{(j)}$ denotes the output of autoencoder when input data $x^{(j)}$ is fed into autoencoder.

$$\hat{x}^{(j)} = f\left(Wa^{(j)} + b\right) \tag{2}$$

Autoencoder is trained to minimize the error between input data and the output of autoencoder. Therefore, the cost function of autoencoder can be expressed as Eq.(3) where m denotes the number of samples in input data, $x^{(j)}$ denotes a sample of input data, and $a^{(j)}$ denotes the output of hidden units when given the input $x^{(j)}$. The optimal weight matrix and bias vector can be obtained by minimizing the cost function.

$$J\left(W, b, W^{'}, b^{'}\right) = \frac{1}{2m}\sum_{j=1}^{m}\left[x^{(j)} - f\left(Wa^{(j)} + b\right)\right]^2 \tag{3}$$

2.2 Generative model for sparse autoencoder

In order to extract structured features, a sparse penalty is imposed on hidden units of autoencoder. We can use a generative model to describe sparse autoencoder from a probabilistic perspective.

In Eq.(4), the input data x is represented by the output of sparse autoencoder $\hat{x}$ and additive Gaussian white noise v [32]

$$\begin{aligned} x &= \hat{x} + v \\ &= f\left(Wa + b\right) + v \end{aligned} \tag{4}$$

where a is the output of hidden units. We can rewrite Eq.(4) from a probabilistic perspective and get Eq.(5)

$$\begin{aligned} P(x \mid a, W, b) &= \mathcal{N}\left(x \mid f\left(a, W, b\right), \sigma\right) \\ &= \frac{1}{Z}\exp\left[-\frac{\left(x - f\left(Wa + b\right)\right)^2}{2\sigma^2}\right] \end{aligned} \tag{5}$$

where Z is a normalizing constant, and σ^2 denotes the variance of the Gaussian white noise.

In order to derive a generative model for sparse autoencoder, we should derive the conditional distribution $P(x \mid W, b)$ which is shown in Eq.(6)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2689790, IEEE Transactions on Big Data

$$P(x \mid W, b) = \int P(x \mid a, W, b)P(a)d(a) \tag{6}$$

where $P(a)$ denotes the prior distribution of hidden units. Since the hidden units are independent, we can rewrite the prior distribution as Eq.(7) where k is number of hidden units.

$$P(a) = \Pi_{i=1}^{k} P(a_i) \tag{7}$$

When the sparse penalty is imposed on the hidden units, the output of hidden units should be near zero with high probability. Therefore, the average activation of hidden unit i defined in Eq.(8) should be a small value near zero. In Eq.(8), m denotes the number of samples in input data, and $a_i^{(j)}$ denotes the output of hidden unit i.

$$\rho_i = \frac{1}{m} \sum_{j=1}^{m} a_i^{(j)} \tag{8}$$

From the probabilistic perspective, when the sparse penalty is imposed, the prior distribution $P(a_i)$ should have a peak at zero and have low probability anywhere else. A suitable form of this prior distribution is introduced by Olshausen *et al.* [31] and shown in Eq.(9)

$$P(a_i) = \frac{1}{Z_\beta} \exp\left[-\beta S(a_i)\right] \tag{9}$$

where the coefficient β determines the shape of the prior distribution, $S(a_i)$ corresponds to the sparse penalty, and Z_β is a normalizing constant.

Both L1 norm [33] and L2 norm [34] are widely used sparse penalty. However, Jiang *et al.* [35] mention that sparse autoencoder with L2 norm achieves better classification accuracy than that with L1 norm. Therefore, L2 norm is used as sparse penalty in this paper. Substituting L2 norm sparse penalty $S(a_i) = a_i^2$ into Eq.(9), we can get the prior distribution $P(a_i)$ in Eq.(10) where the normalizing constant Z_β equals to $\sqrt{\pi/4\beta}$.

$$P(a_i) = \frac{1}{Z_\beta} \exp\left(-\beta a_i^2\right) \tag{10}$$

From the probabilistic perspective, sparse autoencoder is trained to make the conditional distribution $P(x \mid W, b)$ close to the input data distribution $P^*(x)$ by choosing suitable weight matrix W and bias vector b. In order to make these two distributions close, we can minimize the KL divergence between distribution $P(x \mid W, b)$ and $P^*(x)$. The KL divergence is shown in Eq.(11).

$$\begin{aligned} & D\left(P^*(x) \parallel P(x \mid W, b)\right) \\ &= \int P^*(x) \log\left[\frac{P^*(x)}{P(x \mid W, b)}\right] dx \\ &= \int P^*(x) \log\left[P^*(x)\right] dx - \int P^*(x) \log\left[P(x \mid W, b)\right] dx \end{aligned} \tag{11}$$

Since distribution $P^*(x)$ is constant across our choice of weight matrix W and bias vector b, the first term in Eq.(11) is constant. Therefore, minimizing the KL divergence is equivalent to maximize $< \log\left[P(x \mid W, b)\right] >$ where $< . >$ denotes the expectation over input data.

In order to minimize the KL divergence, we have to estimate distribution $P(x \mid W, b)$. However, in Eq.(6), the integral over a to obtain $P(x \mid W, b)$ is generally intractable. Fortunately, due to the sparse assumption of distribution $P(a)$, the shape of distribution $P(a)$ has a peak at zero. Therefore, distribution $P(x, a \mid W, b) = P(x \mid a, W, b)P(a)$ has a fairly tightly peaked maximum. We can approximate this integral by the maximum value of $P(x, a \mid W, b)$ [31] and obtain an approximate solution in Eq.(12).

$$\left(W, b, W^{'}, b^{'}\right)^* = \underset{W,b,W',b'}{\operatorname{argmax}} < \max_a \log\left[P(x \mid a, W, b)P(a)\right] > \tag{12}$$

Substituting the conditional distribution $P(x \mid a, W, b)$ and prior distribution $P(a)$ into $\log\left[P(x \mid a, W, b)P(a)\right]$, we can get Eq.(13).

$$\begin{aligned} & \log\left[P(x \mid a, W, b)P(a)\right] \\ &= \log\left\{\frac{1}{Z} \exp\left[-\frac{\left[x - f\left(Wa + b\right)\right]^2}{2\sigma^2}\right]\right\} \\ &+ \log\left[\Pi_{i=1}^{k} \frac{1}{Z_\beta} \exp\left(-\beta a_i^2\right)\right] \\ &= \log\frac{1}{Z} - \frac{\left[x - f\left(Wa + b\right)\right]^2}{2\sigma^2} + \log\frac{1}{Z_\beta^k} - \sum_{i=1}^{k} \beta a_i^2 \\ &= \frac{-1}{2\sigma^2}\left\{\left[x - f\left(Wa + b\right)\right]^2 + 2\sigma^2\beta \sum_{i=1}^{k} a_i^2\right\} - \log Z_\beta^k Z \end{aligned} \tag{13}$$

In Eq.(13), the second term $\log Z_\beta^k Z$ is a constant. In the first term, $\left[x - f\left(Wa + b\right)\right]^2$ is the reconstruction error, and $2\sigma^2\beta \sum_{i=1}^{k} a_i^2$ is the sparse penalty term. Therefore, the sparse hyperparameter λ equals to $2\sigma^2\beta$ where σ^2 is the variance of the reconstruction error, and β is the coefficient determining the shape of the prior distribution. We can rewrite Eq.(12) in the form of the cost function in Eq.(14) where m denotes the number of samples in input data, and $a^{(j)}$ is the output of hidden units. Maximizing Eq.(12) is equivalent to minimize this cost function.

$$\begin{aligned} J(W, b, W^{'}, b^{'}) = & \frac{1}{2m} \sum_{j=1}^{m} \left[x^{(j)} - f\left(Wa^{(j)} + b\right)\right]^2 \\ & + \frac{1}{2m}\lambda \sum_{j=1}^{m} \sum_{i=1}^{k} \left[a_i^{(j)}\right]^2 \end{aligned} \tag{14}$$

In Eq.(14), the first term is the reconstruction error, and the second term is the sparse penalty term. The sparse hyperparameter λ has to be determined to balance the trade-off between the reconstruction error and the sparse penalty term.

In order to determine the sparse hyperparameter, first of all, we derive the relationship between the sparse hyperparameter and the average activation of hidden units. This relationship is shown in Eq.(15)

$$E(a_i) = \rho_i = \frac{1}{\sqrt{\pi\beta}} \tag{15}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2689790, IEEE Transactions on Big Data

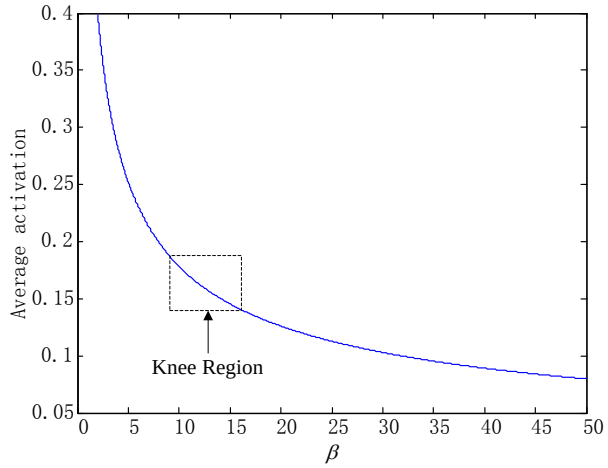


Fig. 2. Theoretical average activation of hidden units

where β is the coefficient determining the shape of the prior distribution. The average activation ρ_i equals to the expectation of the sparse prior distribution $P(a_i)$. Therefore, Eq.(15) can be used to estimate the average activation of hidden units. In addition, it provides an explicit guidance to determine the sparse hyperparameter.

The relationship between the average activation and the coefficient β is shown in Fig. 2. When β is small, the average activation is high, and hidden units become less sparse. When β is large, good sparsity can be obtained. However, the sparse hyperparameter λ, which equals to $2\sigma^2\beta$, will become large. When we minimize the cost function in Eq.(14), large λ will result in large reconstruction error.

Inspired by multi-objective optimization [36], in Fig. 2, we define a knee region from 8 to 16. When β is larger than 16, increasing β will benefit sparsity little. When β is smaller than 8, decreasing β will worsen sparsity a lot. The knee region is defined to make β as small as possible while obtaining good sparsity. Choosing β inside the knee region balances the trade-off between the reconstruction error and the sparsity. However, when β is chosen outside the knee region, it will result in either large reconstruction error or high average activation.

In order to determine the sparse hyperparameter λ, which equals to $2\sigma^2\beta$, we just need to determine β and σ^2. In Fig. 2, β is chosen from the center of the knee region to balance the trade-off between the reconstruction error and the sparsity. The variance σ^2 can be estimated during the training process.

3 Average Activation Estimation

3.1 Experiment on MNIST dataset

In order to evaluate the validity of our method to estimate the average activation of hidden units, sparse autoencoder is trained on MNIST dataset [37] which contains handwritten digits from 0 to 9. The sparse autoencoder has 784 input units and 169 hidden units. The sparse hyperparameter λ varies from 0.1 to 0.5 with step 0.1. After the training process, the average activation of hidden units is calculated. This experimental average activation is compared with the theoretical average activation which can be calculated by Eq.(15) where β equals to $\lambda/2\sigma^2$.

TABLE 1
Average activation of hidden units on MNIST dataset

λ	Experimental activation ρ	β	Theoretical activation ρ'
0.1	0.2893	3.90	0.2859
0.2	0.2204	7.09	0.2120
0.3	0.1850	9.83	0.1800
0.4	0.1621	12.28	0.1610
0.5	0.1432	14.50	0.1482

TABLE 2
Average activation of hidden units on a natural image dataset

λ	Experimental activation ρ	β	Theoretical activation ρ'
0.02	0.3122	4.13	0.2776
0.04	0.2344	6.61	0.2194
0.06	0.1955	8.62	0.1922
0.08	0.1762	10.93	0.1707
0.10	0.1502	11.69	0.1650

The experimental result is shown in Table 1. When λ increases from 0.1 to 0.5, the average activation of hidden units decreases. Besides, the experimental average activation matches the theoretical result well. This experimental result demonstrates that Eq.(15) can effectively estimate the average activation of hidden units.

3.2 Experiment on a natural image dataset

We also evaluate the validity of our method on a natural image dataset used in [38]. In this experiment, the sparse autoencoder is trained on $10,000$ image patches whose size is 8×8. These patches are randomly sampled from 10 natural images whose size is 512×512. The sparse autoencoder has 64 input units and 25 hidden units. The sparse hyperparameter λ varies from 0.02 to 0.10 with step 0.02.

The experimental result is shown in Table 2. When λ increases from 0.02 to 0.10, the average activation of hidden units decreases. Besides, the experimental average activation matches the theoretical result well. This experimental result demonstrates that Eq.(15) can effectively estimate the average activation of hidden units.

Comparing Table 1 with Table 2, we find that, in order to obtain similar sparsity, the sparse hyperparameter λ has a big difference in these two datasets. In traditional sparse hyperparameter determination method, we do not know the relationship between the sparse hyperparameter λ and the average activation of hidden units. However, in our method, this relationship can be determined by Eq.(15).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2689790, IEEE Transactions on Big Data

4 Sparse hyperparameter determination

Traditional sparse hyperparameter determination method, which searches for a suitable sparse hyperparameter within a large range, is time-consuming. However, our method based on the generative model can efficiently determine the sparse hyperparameter. In order to determine the sparse hyperparameter λ, firstly, we choose β from the center of the knee region in Fig. 2. Secondly, in each training iteration, we estimate the variance of the reconstruction error σ^2. Finally, the sparse hyperparameter is calculated based on $\lambda = 2\sigma^2\beta$. In the sparse hyperparameter determination process, we need to estimate the variance of the reconstruction error in each training iteration. We find that, when β is chosen from the knee region, the standard variance of sparse autoencoder increases about 20% in comparison with that of autoencoder without sparse penalty. Therefore, we can train an autoencoder without sparse penalty and estimate the standard variance of reconstruction error σ_1. Then, the sparse hyperparameter is determined by $\lambda = 2 \times (1.2\sigma_1)^2 \beta$.

4.1 Experiment on MNIST dataset

4.1.1 Setup

The validity of our sparse hyperparameter determination method is evaluated on MNIST dataset containing handwritten digits from 0 to 9. The dataset is separated into unlabeled data containing digits from 5 to 9 and labeled data containing digits from 0 to 4. The unlabeled data is used to train the sparse autoencoder to learn features. Once the sparse autoencoder has been trained, the weight matrix and bias vector of the sparse autoencoder remains unchanged. Then, the sparse autoencoder is used to extract features from the labeled data which is separated into training set and test set. Finally, these features are fed into the softmax classifier. L2 norm is used as weights regularization term in the experiment. The regularization parameters for sparse autoencoder and softmax classifier are 0.003 and 0.0001 respectively.

The size of the input image is 28×28, so the sparse autoencoder has 784 input units. The number of hidden units is determined based on three criteria. First, in order to extract features from input data, the number of hidden units should be smaller than the number of input units. Second, since the output of hidden units is fed into a softmax classifier, the number of hidden units should be larger than the number of the output of the softmax classifier. Third, in order to display the features learned by sparse autoencoder in a square, the number of hidden units is chosen as the square of an integer.

4.1.2 Sparse autoencoder with 169 hidden units

In order to determine the sparse hyperparameter, first of all, we train an autoencoder without sparse penalty and estimate the standard variance of reconstruction error σ_1, which equals to 0.1083. Then, $\beta = 12$ is chosen from the center of the knee region in Fig. 2. Therefore, the sparse hyperparameter chosen by our method equals to $2 \times (1.2 \times \sigma_1)^2 \times \beta = 0.4054$. In order to evaluate the validity of our method, we also choose some β outside the knee region. Specifically, $\beta_1 = 8$ and $\beta_2 = 16$ are chosen from the edge of the knee region. Their corresponding sparse hyperparameters are $\lambda_1 = 0.2702$ and $\lambda_2 = 0.5405$. Besides, we choose $\beta_3 = 0$, $\beta_4 = 4$, and $\beta_5 = 20$, which are outside the knee region. Their corresponding sparse hyperparameters are $\lambda_3 = 0$, $\lambda_4 = 0.1351$, and $\lambda_5 = 0.6756$. Then, the sparse autoencoder is trained with these sparse hyperparameters to extract features respectively. Finally, these features are fed into the softmax classifier.

TABLE 3
Classification accuracy of sparse autoencoder with 169 hidden units

β	λ	σ	Classification accuracy
0	0	0.1083	97.26%
4	0.1351	0.1147	97.79%
8	0.2702	0.1224	97.96%
12	**0.4054**	**0.1279**	**98.01%**
16	0.5405	0.1329	98.00%
20	0.6756	0.1378	97.88%

Experimental classification accuracy is shown in Table 3. The classification accuracy of sparse autoencoder is better than that of autoencoder without sparse penalty. When β increases from 0 to 12, the classification accuracy increases from 97.26% to 98.01%. When β continues to increase from 12 to 20, the classification accuracy decreases from 98.01% to 97.88%. In Table 3, $\beta = 12$ chosen by our method achieves the best accuracy. When $\beta = 8$ and $\beta = 16$ are chosen from the edge of the knee region, the classification accuracy is close to the best accuracy. This result demonstrates that the optimal β should be in the knee region, and the optimal accuracy is close to 98.01%. When β is chosen from the knee region, the sparse autoencoder can achieve good classification accuracy. However, when β is chosen outside the knee region, the classification accuracy becomes worse. Table 3 also shows that, when $\beta = 12$ is chosen from the knee region, the standard variance increase about 20% in comparison with the standard variance when $\beta = 0$.

Although we cannot guarantee $\beta = 12$ is the optimal β, its classification accuracy is close to the optimal accuracy. Therefore, we can choose $\beta = 12$ instead of finding the optimal β. The same idea can be found in optimizing the deep neural network. Since the deep neural network has many local minima, optimizing algorithm is easy to stick in these local minima. Struggling to find the global minimum is very hard and sometimes impossible. Fortunately, the performance of most local minima is close to the performance of the global minimum. In order to simplify the optimization process, we can just find a good local minimum instead of the global minimum [39].

Traditional sparse hyperparameter determination method [27] searches for a suitable sparse hyperparameter within a large range. For example, in this experiment, the sparse hyperparameter λ may be chosen from 0.1 to 1 with step 0.1. Therefore, the traditional method has to train ten sparse autoencoders and compare their classification performance. Then, the traditional method can find a suitable sparse hyperparameter 0.4, but this searching

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2689790, IEEE Transactions on Big Data

process is time-consuming. Moreover, in general, we do not know the range and the step when choosing the sparse hyperparameter. Therefore, it will take more time to choose a suitable sparse hyperparameter. The advantage of our method is that we can directly determine a suitable sparse hyperparameter.

Features learned by sparse autoencoder with different sparse hyperparameters are shown in Fig. 3. In Fig. 3a, autoencoder without sparse penalty learns unstructured features. When the sparse constraint is imposed on autoencoder, sparse autoencoder can learn structured features. However, when sparse hyperparameter is large, many features become similar. Fig. 3d shows the features learned by the sparse autoencoder whose sparse hyperparameter is determined by our method. These features are structured and have few similar features.

Fig. 4 shows the activation of hidden units when a sample from MNIST dataset is fed into autoencoder and sparse autoencoder. Fig. 4a shows the activation of autoencoder hidden units. Most of these hidden units are active. In Fig. 4b, when sparse penalty is introduced, most of the hidden units become inactive.

4.1.3 Sparse autoencoder with 196 hidden units

We also evaluate the validity of our method on sparse autoencoder with 196 hidden units. When we train an autoencoder without sparse penalty, the standard variance of reconstruction error σ_1 equals to 0.1058. Then, $\beta = 12$ is chosen from the center of the knee region in Fig. 2. Therefore, the sparse hyperparameter chosen by our method equals to $2\times(1.2 \times \sigma_1)^2 \times \beta = 0.3869$. In order to evaluate the validity of our method, we also choose some β outside the knee region. Specifically, $\beta_1 = 8$ and $\beta_2 = 16$ are chosen from the edge of the knee region. Their corresponding sparse hyperparameters are $\lambda_1 = 0.2579$ and $\lambda_2 = 0.5158$. We also choose $\beta_3 = 0$, $\beta_4 = 4$, and $\beta_5 = 20$, which are outside the knee region. Their corresponding sparse hyperparameters are $\lambda_3 = 0$, $\lambda_4 = 0.1290$, and $\lambda_5 = 0.6448$. Then, the sparse autoencoder is trained with these sparse hyperparameters to extract features respectively. Finally, these features are fed into the softmax classifier.

Experimental classification accuracy is shown in Table 4. The classification accuracy of sparse autoencoder is better than that of autoencoder without sparse penalty. When β increases from 0 to 12, the classification accuracy increases from 97.30% to 98.16%. When β continues to increase from 12 to 20, the classification accuracy decreases from 98.16% to 97.92%. In Table 4, $\beta = 12$ chosen by our method achieves the best accuracy. When $\beta = 8$ and $\beta = 16$ are chosen from the edge of the knee region, the classification accuracy is close to the best accuracy. This result demonstrates that the optimal β should be in the knee region, and the optimal accuracy is close to 98.16%. When β is chosen from the knee region, the sparse autoencoder can achieve good classification accuracy. However, when β is chosen outside the knee region, the classification accuracy becomes worse. Table 4 also shows that, when $\beta = 12$ is chosen from the knee region, the standard variance increase about 20% in comparison with the standard variance when $\beta = 0$. Although we cannot guarantee $\beta = 12$ is the optimal β, its classification accuracy is close to the optimal accuracy.

TABLE 4
Classification accuracy of sparse autoencoder with 196 hidden units

β	λ	σ	Classification accuracy
0	0	0.1058	97.30%
4	0.1290	0.1128	97.90%
8	0.2579	0.1207	98.09%
12	**0.3869**	**0.1265**	**98.16%**
16	0.5158	0.1318	98.06%
20	0.6448	0.1364	97.92%

Features learned by sparse autoencoder with different sparse hyperparameters are shown in Fig. 5. In Fig. 5a, autoencoder without sparse penalty learns unstructured features. When the sparse constraint is imposed on autoencoder, sparse autoencoder can learn structured features. However, when sparse hyperparameter is large, many features become similar. Fig. 5d shows the features learned by the sparse autoencoder whose sparse hyperparameter is determined by our method. These features are structured and have few similar features.

Comparing Fig. 5 with Fig. 3, we can find that, when the number of hidden units increases, more features become similar. This result demonstrates that some hidden units become redundant. Pruning these redundant units can simplify the sparse autoencoder and will not affect its performance.

Fig. 6 shows the activation of hidden units when a sample from MNIST dataset is fed into autoencoder and sparse autoencoder. Fig. 6a shows the activation of autoencoder hidden units. Most of these hidden units are active. In Fig. 6b, when sparse penalty is introduced, most of the hidden units become inactive.

4.2 Experiment on SVHN dataset

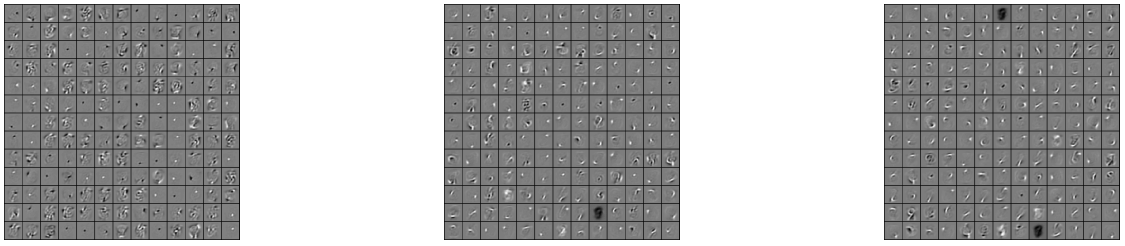
4.2.1 Setup

The validity of our method is also evaluated on Street View House Numbers (SVHN) Dataset [40]. SVHN is a real-world image dataset which is obtained from house numbers in Google Street View images. SVHN has 10 categories and contains 73257 training samples and 26032 testing samples. In our experiment, these samples are converted to grey-scale images. Since the size of the input image is 32×32, the sparse autoencoder has 1024 input units. Besides, the number of hidden units is chosen as 196. The sparse autoencoder is trained on the training samples. Once it has been trained, the weight matrix and bias vector of the sparse autoencoder remains unchanged. Then, the sparse autoencoder is used to extract features from training and testing samples. Finally, these features are fed into the softmax classifier.

4.2.2 Evaluation result

In order to determine the sparse hyperparameter, first of all, we train an autoencoder without sparse penalty and estimate the standard variance of reconstruction error σ_1, which equals to 0.056. Then, $\beta = 12$ is chosen from

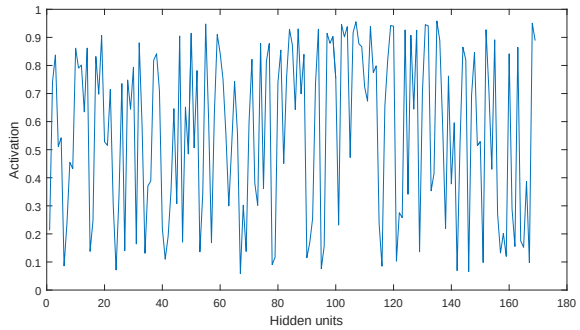
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2689790, IEEE Transactions on Big Data



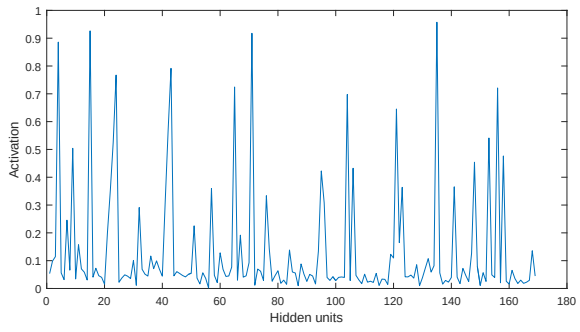
(a) $\lambda = 0$ (b) $\lambda = 0.1351$ (c) $\lambda = 0.2702$

(d) $\lambda = 0.4054$ (e) $\lambda = 0.5405$ (f) $\lambda = 0.6756$

Fig. 3. Features learned by sparse autoencoder with 169 hidden units. When sparse hyperparameter λ is small, the learned features are unstructured. When sparse hyperparameter λ is large, many features become similar. The sparse hyperparameter $\lambda = 0.4054$ is chosen by our method. The learned features are structured and have few similar features.



(a) $\lambda = 0$



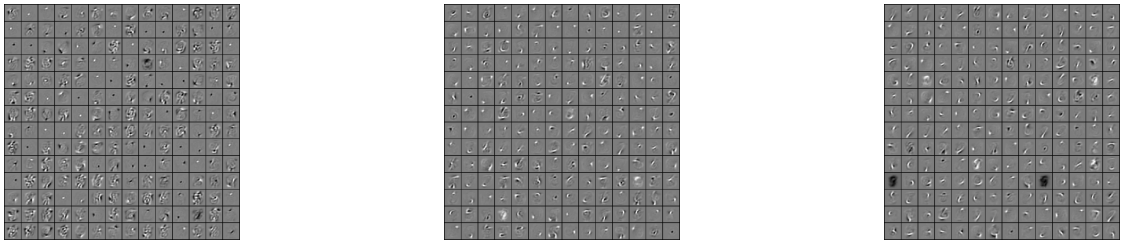
(b) $\lambda = 0.4054$

Fig. 4. Activation of 169 hidden units when a sample from MNIST dataset is fed into (a)autoencoder and (b)sparse autoencoder.

the center of the knee region in Fig. 2. Therefore, the sparse hyperparameter chosen by our method equals to $2\times(1.2 \times \sigma_1)^2 \times \beta = 0.1077$. In order to evaluate the validity of our method, we also choose some β outside the knee region. Specifically, $\beta_1 = 8$ and $\beta_2 = 16$ are chosen from the edge of the knee region. Their corresponding sparse hyperparameters are $\lambda_1 = 0.0718$ and $\lambda_2 = 0.1436$. Besides, we choose $\beta_3 = 0$, $\beta_4 = 4$, and $\beta_5 = 20$, which are outside the knee region. Their corresponding sparse hyperparameters are $\lambda_3 = 0$, $\lambda_4 = 0.0359$, and $\lambda_5 = 0.1796$. Then, the sparse autoencoder is trained with these sparse hyperparameters to extract features respectively. Finally, these features are fed into the softmax classifier.

Experimental classification accuracy is shown in Table 5. SVHN is a real-world large-scale image dataset and is much more difficult than MNIST dataset. The classification accuracy shown in Table 5 is consistent with the classification accuracy shown in [35]. The classification accuracy of sparse autoencoder is better than that of autoencoder without sparse penalty. When β increases from 0 to 12, the classification accuracy increases from 27.40% to 43.10%. When β continues to increase from 12 to 20, the classification accuracy decreases from 43.10% to 42.69%. In Table 5, $\beta = 12$ chosen by our method achieves the best accuracy. When $\beta = 8$ and $\beta = 16$ are chosen from the edge of the knee region, the classification accuracy is close to the best accuracy. This result demonstrates that the optimal β should be in the knee region, and the optimal accuracy is close to 43.10%. When β is chosen from the knee region, the sparse autoencoder can achieve good classification accuracy. However, when β is chosen outside the knee region, the classification accuracy becomes worse. Table 5 also shows that, when $\beta = 12$ is chosen from the knee region, the standard variance increase about 20% in comparison with the standard variance when $\beta = 0$.

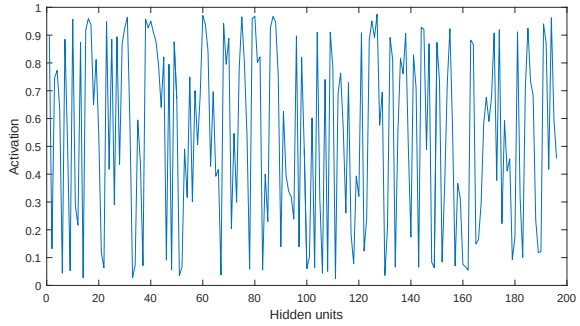
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2689790, IEEE Transactions on Big Data



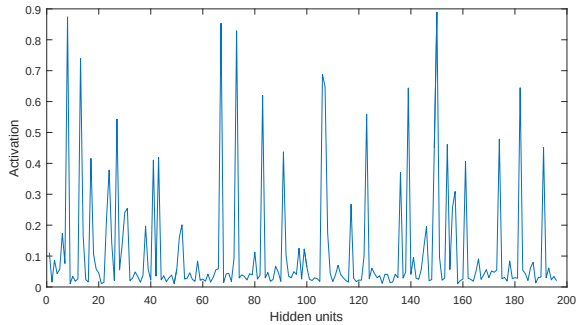
(a) $\lambda = 0$ (b) $\lambda = 0.1290$ (c) $\lambda = 0.2579$

(d) $\lambda = 0.3869$ (e) $\lambda = 0.5158$ (f) $\lambda = 0.6448$

Fig. 5. Features learned by sparse autoencoder with 196 hidden units. When sparse hyperparameter λ is small, the learned features are unstructured. When sparse hyperparameter λ is large, many features become similar. The sparse hyperparameter $\lambda = 0.3869$ is chosen by our method. The learned features are structured and have few similar features.



(a) $\lambda = 0$



(b) $\lambda = 0.3869$

Fig. 6. Activation of 196 hidden units when a sample from MNIST dataset is fed into (a)autoencoder and (b)sparse autoencoder.

TABLE 5
Classification accuracy on SVHN dataset

β	λ	σ	Classification accuracy
0	0	0.056	27.40%
4	0.0359	0.058	40.78%
8	0.0718	0.061	43.02%
12	**0.1077**	**0.063**	**43.10%**
16	0.1436	0.065	42.93%
20	0.1796	0.067	42.69%

5 Conclusion

In this paper, we derived a generative model for sparse autoencoder and proposed a sparse hyperparameter determination method. In addition, we derived the relationship between the sparse hyperparameter and the average activation of hidden units. Two experiments were performed in this paper. One evaluated the validity of our method to estimate the average activation of hidden units. The other evaluated the validity of our method to determine the sparse hyperparameter. Although we cannot guarantee the sparse hyperparameter chosen by our method is the optimal one, its classification accuracy is close to the optimal accuracy. In the experiment, we found that the sparse autoencoder has redundant hidden units. In the future research, we will investigate how to prune these redundant units to simplify sparse autoencoder.

Acknowledgments

We would like to thank Jun Yan for the constructive discussions and comments during the development process of this research.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2689790, IEEE Transactions on Big Data

REFERENCES

[1] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[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, Aug 2013.

[3] F. Wu, Z. Wang, Z. Zhang, Y. Yang, J. Luo, W. Zhu, and Y. Zhuang, "Weakly semi-supervised deep learning for multi-label image annotation," *IEEE Transactions on Big Data*, vol. 1, no. 3, pp. 109–122, Sept 2015.

[4] W. Zhang, R. Li, T. Zeng, Q. Sun, S. Kumar, J. Ye, and S. Ji, "Deep model based transfer and multi-task learning for biological image analysis," *IEEE Transactions on Big Data*, vol. PP, no. 99, pp. 1–1, 2016.

[5] K. Chen and Z. Zhang, "Learning to classify fine-grained categories with privileged visual-semantic misalignment," *IEEE Transactions on Big Data*, vol. PP, no. 99, pp. 1–1, 2016.

[6] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[7] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

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

[9] S. Lange and M. Riedmiller, "Deep auto-encoder neural networks in reinforcement learning," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, July 2010, pp. 1–8.

[10] A. Ng, "Sparse autoencoder," *CS294A Lecture Notes*, vol. 72, pp. 1–19, 2011.

[11] T. Ishii, H. Komiyama, T. Shinozaki, Y. Horiuchi, and S. Kuroiwa, "Reverberant speech recognition based on denoising autoencoder." in *Interspeech*, 2013, pp. 3512–3516.

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

[13] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder." in *Interspeech*, 2013, pp. 436–440.

[14] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*. ACM, 2008, pp. 1096–1103.

[15] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the 28th International Conference on Machine Learning*, 2011, pp. 833–840.

[16] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[17] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," *Proc. Int. Joint Conf. Neural Networks (IJCNN08)*, pp. 1322–1328, 2008.

[18] W. W. Ng, G. Zeng, J. Zhang, D. S. Yeung, and W. Pedrycz, "Dual autoencoders features for imbalance classification problem," *Pattern Recogn.*, vol. 60, no. C, pp. 875–889, Dec. 2016.

[19] J. Xu, L. Xiang, Q. Liu, H. Gilmore, J. Wu, J. Tang, and A. Madabhushi, "Stacked sparse autoencoder (ssae) for nuclei detection on breast cancer histopathology images," *IEEE Transactions on Medical Imaging*, vol. 35, no. 1, pp. 119–130, Jan 2016.

[20] M. Shu and A. Fyshe, "Sparse autoencoders for word decoding from magnetoencephalography," in *Proceedings of the third NIPS Workshop on Machine Learning and Interpretation in NeuroImaging (MLINI)*, 2013.

[21] A. L. Barth and J. F. Poulet, "Experimental evidence for sparse firing in the neocortex," *Trends in Neurosciences*, vol. 35, no. 6, pp. 345–355, 2012.

[22] O. J. Hulme, M. Skov, M. J. Chadwick, H. R. Siebner, and T. Z. Ramsøy, "Sparse encoding of automatic visual association in hippocampal networks," *NeuroImage*, vol. 102, pp. 458–464, 2014.

[23] D. J. Field, "What is the goal of sensory coding?" *Neural Computation*, vol. 6, no. 4, pp. 559–601, 1994.

[24] D. Field, "Scale-invariance and self-similar wavelet transforms: an analysis of natural scenes and mammalian visual systems," *in Wavelets, Fractals and Fourier Transforms: New Developments and New Applications, Oxford University Press*, pp. 151–193, 1993.

[25] C. Poultney, S. Chopra, Y. L. Cun *et al.*, "Efficient learning of sparse representations with an energy-based model," in *Advances in Neural Information Processing Systems*, 2006, pp. 1137–1144.

[26] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 8595–8598.

[27] K. Chen, J. Hu, and J. He, "A framework for automatically extracting overvoltage features based on sparse autoencoder," *IEEE Transactions on Smart Grid*, vol. PP, no. 99, pp. 1–1, 2016.

[28] C. Tao, H. Pan, Y. Li, and Z. Zou, "Unsupervised spectral spatial feature learning with stacked sparse autoencoder for hyperspectral imagery classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 12, pp. 2438–2442, Dec 2015.

[29] K. Sun, S. H. Huang, D. S. H. Wong, and S. S. Jang, "Design and application of a variable selection method for multilayer perceptron neural network with lasso," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–11, 2016.

[30] M. Gong, J. Liu, H. Li, Q. Cai, and L. Su, "A multiobjective sparse feature learning model for deep neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 12, pp. 3263–3277, Dec 2015.

[31] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[32] C. M. Bishop, "Pattern recognition," *Machine Learning*, 2006.

[33] C. Li, X. Yu, T. Huang, G. Chen, and X. He, "A generalized hopfield network for nonsmooth constrained convex optimization: Lie derivative approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 2, pp. 308–321, Feb 2016.

[34] A. Majumdar and R. K. Ward, "Classification via group sparsity promoting regularization," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, April 2009, pp. 861–864.

[35] N. Jiang, W. Rong, B. Peng, Y. Nie, and Z. Xiong, "An empirical analysis of different sparse penalties for autoencoder in unsupervised feature learning," in *2015 International Joint Conference on Neural Networks (IJCNN)*, July 2015, pp. 1–8.

[36] K. Deb, "Multi-objective optimization," in *Search Methodologies*. Springer, 2014, pp. 403–449.

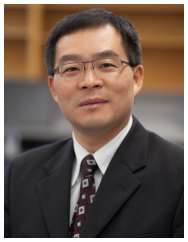
[37] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[38] B. A. Olshausen *et al.*, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

[39] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The loss surfaces of multilayer networks," *arXiv preprint arXiv:1412.0233*, 2014.

[40] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

Zhiqiang Wan (S'16) received his B.S. degree from the Harbin Institute of Technology, Harbin, China, in 2012. He received his M.S. degree in the School of Electrical and Electronics Engineering, Huazhong University of Science and Technology (HUST), Wuhan, China, in 2015. He is presently working towards his Ph.D. degree in the School of Electrical, Computer and Biomedical Engineering, University of Rhode Island (URI), Kingston, Rhode Island, USA. His current research interests includes deep learning, robotic and machine learning.



Haibo He (SM11) received the BS and MS degrees in electrical engineering from the Huazhong University of Science and Technology, China, in 1999 and 2002, respectively, and the PhD degree in electrical engineering from Ohio University in 2006. He is currently the Robert Haas Endowed chair professor at the Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island. His current research interests include computational intelligence, machine learning, data mining, and various applications. He has published one sole-author research book (Wiley), edited one book (Wiley-IEEE) and six conference proceedings (Springer), and authored and coauthored more than 250 peer-reviewed journal and conference papers. He was the general chair of the IEEE Symposium Series on Computational Intelligence (SSCI 2014). He received the IEEE International Conference on Communications Best Paper Award (2014), IEEE CIS Outstanding Early Career Award (2014), K. C. Wong Research Award, Chinese Academy of Sciences (2012), US National Science Foundation CAREER Award (2011), and Providence Business News “Rising Star Innovator” Award (2011). He is currently the Editor-in-Chief of the IEEE Transactions on Neural Networks and Learning Systems.

Bo Tang (S14) received the BS degree from Central South University, Changsha, China, in 2007, and the MS degree from the Institute of Electronics, Chinese Academy of Science, Beijing, China, in 2010. Currently, he is working toward the PhD degree in the Department of Electrical, Computer and Bio-medical Engineering, University of Rhode Island, Kingston, RI. His current research interests include statistical machine learning, data mining, computational intelligence, computer vision, and cloud computing. He is a student member of the IEEE.