LEARNING DEEP NEURAL NETWORK USING MAX-MARGIN MINIMUM CLASSIFICATION ERROR

*Ziyong Feng, Zenghui Sun, Lianwen Jin**

School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China
f.ziyong@mail.scut.edu.cn, sunfreding@gmail.com, lianwen.jin@gmail.com*

ABSTRACT

Deep neural networks (DNNs) have recently achieved state-of-the-art performance on various tasks, such as image classification, handwriting recognition, text spotting, and speech recognition. Most of these DNNs use softmax regression and cross-entropy loss to calculate the loss function for optimization. However, the loss function is merely expected to raise the output of the true class and reduce others without distinction. In this paper, we propose a new max-margin minimum classification error (M^3CE) training method, which is inspired by the traditional minimum classification error, but is more appropriate for training DNNs. The proposed M^3CE aims not only to increase the posteriori of the true class but also to decrease the output of the most confused class, which can cover any shortage of the cross-entropy loss. We evaluate the M^3CE on two popular datasets, MNIST and CIFAR-10. Experimental results show that the M^3CE complements cross-entropy efficiently and achieves better performance.

***Index Terms*—** Deep neural network, softmax regression, minimum classification error, max-margin.

1. INTRODUCTION

Recently, deep neural networks (DNNs) have attracted significant attention, because they can automatically and simultaneously discover low- and high-level features, and have achieved promising results on various databases. Compared with traditional neural networks, DNNs implemented using stacking restricted Boltzmann machines (RBMs) [1] and regularized auto-encoders [2] perform much better owing to the use of a deeper multilayer structure and various new technologies such as greedy layer-wise unsupervised pre-training and the backpropagation (BP) algorithm. Moreover, considering the topological structure, convolutional neural networks (CNNs) [3] are constructed using a convolutional and pooling operation, but are trained without layer-wise unsupervised pre-training [4]. Owing to their specialized structure, CNNs are more suitable for computer vision tasks. Therefore, on various databases, they have achieved state-of-the-art performance [5], [6], [7]. However, many parameters are needed to tune a deep CNN given the vast data. This leads to high computational cost even when using extremely fast graphics processing units (GPUs) [4] [8]. When using either RBMs or CNNs, fine-tuning plays an important role in the entire training procedure and an appropriate loss function can improve training speed as well as the final performance.

Generally, softmax, which provides a posteriori distribution, is used as the output layer of a DNN, while cross-entropy is employed as the criterion for constructing the loss function, given that the posteriori distribution from softmax should be close to the target distribution (one-hot code). The loss function calculated by softmax and cross-entropy can be interpreted as a generalization in terms of logistic loss from a two-class problem to a multiclass problem. Although this loss function is widely used for training DNNs and has been evaluated on different tasks and networks, it treats all outputs of false classes consistently. To reduce the misclassification probability, the minimum classification error (MCE) was proposed as a criterion for traditional neural networks [9]. The MCE was applied to several recognition tasks and achieved the best performance for speech recognition at the time. However, because the sigmoid function can easily be saturated, it is not appropriate for current DNNs.

In this paper, we present a max-margin minimum classification error (M^3CE) derived from the traditional MCE, and which is more suitable for DNNs. Instead of a sigmoid function, M^3CE employs a ReLU-like function to backpropagate more discriminative information to low-level layers. We also combine M^3CE with cross-entropy according to gradient analysis. We evaluated the M3CE on two deep learning benchmark datasets, MNIST [3] and CIFAR-10 [10]. The experiments show that M^3CE complements cross-entropy well and achieves higher state-of-the-art performance on CIFAR-10.

The rest of the paper is organized as follows. In Section 2, we review related work on different loss functions for neural networks. Then, we introduce the proposed M^3CE and analyze it in Section 3. Section 4 presents the experimental

results on MNIST and CIFAR-10. Section 5 concludes the paper.

2. RELATE WORK

The mean-squared error (MSE) was used as a typical criterion function for shallow neural networks in the 1990s. Given a training set $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$, $\mathbf{x}_n \in \mathbb{R}^D$, $y_n \in \{1, 2, ..., K\}$, it expects the outputs of the last layer to be close to the targets. The loss function is defined as

$$L_{MMSE} = \frac{1}{N}\sum_{n=1}^{N} \|f(\mathbf{x}_n; \Theta) - \mathbf{y}_n\|_2^2, \tag{1}$$

where $f(\cdot; \Theta)$ is the prediction function parameterized by network parameter Θ, and $\mathbf{y}_n$ is the K-dimensional target vector whose true label entry is one, while the other labels are zero. Generally, a sigmoid function is employed as the output activation function. However, for classification tasks, this cannot provide a posteriori distribution.

With the advent of deep learning, it is standard to use softmax and cross-entropy to construct the loss function for classification problems [4], [11]. The loss function is defined as

$$L_{CE} = -\frac{1}{N}\sum_{n=1}^{N} \mathbf{y}_n^T \log \mathbf{p}_n, \tag{2}$$

where $\log(\cdot)$ is an element-wise logarithmic function, and $\mathbf{p}_n \in \mathbb{R}^K$ is the output of softmax. Suppose $\mathbf{z}_n \in \mathbb{R}^K$ is the input for softmax; the k-th entry p_{nk} of $\mathbf{p}_n$ is defined as

$$p_{nk} = \frac{\exp z_{nk}}{\sum_{j=1}^{K} \exp z_{nj}}. \tag{3}$$

Obviously, unlike the sigmoid function, softmax can provide class probabilities, which are more suitable for classification problems. By combining softmax with the cross-entropy function, a neural network can be trained quickly and achieves higher performance [12], [13], as compared with MSE.

To minimize misclassification probability, Juang and Katagiri [9] incorporated the MCE into artificial neural networks. There are three steps in defining an MCE function: (1) define a discriminant function $g_k(\mathbf{z})$ for each class k; (2) give a misclassification measure for each sample; and (3) specify a loss function for the measure in (2). The final loss function can be the average of the loss of each sample. In [14], the misclassification measure is defined as

$$d_k(\mathbf{z}) = \frac{1}{N_k}\sum_{l \in S_k} (g_l(\mathbf{z}) - g_k(\mathbf{z})), \tag{4}$$

where $S_k = \{l | g_l(\mathbf{z}) > g_k(\mathbf{z})\}$ contains the confused classes, and N_k denotes the number of confused classes. However, because of the variation in N_k, the misclassification measure is not differentiable, which is not desirable in gradient descent. Therefore, Juang and Katagiri [9] proposed a more appropriate misclassification measure,

$$d_k(\mathbf{z}) = -g_k(\mathbf{z}) + \max_{l \neq k} g_l(\mathbf{z}). \tag{5}$$

In step (3), the loss function of the misclassification measure is specified as logistic loss,

$$\ell_k(d_k) = \frac{1}{1 + \exp(-\alpha d_k + \beta)}. \tag{6}$$

Finally, the MCE loss function is defined as the empirical average loss,

$$L_{MCE} = \frac{1}{N}\sum_{n=1}^{N} \ell_{y_n}(d_{y_n}(\mathbf{z}_n)). \tag{7}$$

3. MAX-MARGIN MCE

Based on the definition of MCE, we employ softmax regression as the discriminant function. Therefore, the misclassification measure (5) is redefined as

$$\begin{aligned} d_k(\mathbf{z}) &= -p_k + p_q \\ &= -\frac{z_k}{\sum_{j=1}^{K} \exp z_j} + \frac{z_q}{\sum_{j=1}^{K} \exp z_j}, \end{aligned} \tag{8}$$

where k is the index of the true class and $q = \mathrm{argmax}_{l \neq k} p_l$ denotes the most confused output of softmax. Generally, logistic loss (6) is used and the gradient of $\mathbf{z}$ is calculated for use in the BP algorithm.

$$\begin{aligned} \frac{\partial \ell_k(d_k)}{\partial \mathbf{z}} &= \frac{\partial \ell_k(d_k)}{\partial d_k(\mathbf{z})} \cdot \frac{\partial d_k(\mathbf{z})}{\partial \mathbf{z}} \\ &= \alpha \ell_k (1 - \ell_k) \cdot \frac{\partial d_k(\mathbf{z})}{\partial \mathbf{z}}. \end{aligned} \tag{9}$$

Remarkably, when $\mathbf{z}$ is misclassified, ℓ_k is close to one and $\alpha \ell_k (1 - \ell_k)$ tends to zero. Thus, $\frac{\partial \ell_k(d_k)}{\partial \mathbf{z}}$ is close to 0 and no gradient is backpropagated to previous layers, which is harmful to the training procedure.

It is worth noting that the traditional activation function, the sigmoid function, is replaced by a non-saturating activation function ReLU [4], [15], which can speed up the training procedure and prevent gradient diffusion. Therefore, we use a ReLU-like non-saturating function as the loss function of the misclassification measure,

$$\ell_k(d_k) = \max(0, 1 + d_k). \tag{10}$$

When a correctly classified sample becomes a misclassification sample, d_k increases from -1 to 1. Thus, the definition (10) of ℓ_k (called the max-margin loss) can cause the worst sample to have the maximum loss without saturation. Since $1 + d_k \geq 0$, Eq. (10) can be simplified as

$$\ell_k(d_k) = 1 + d_k. \tag{11}$$

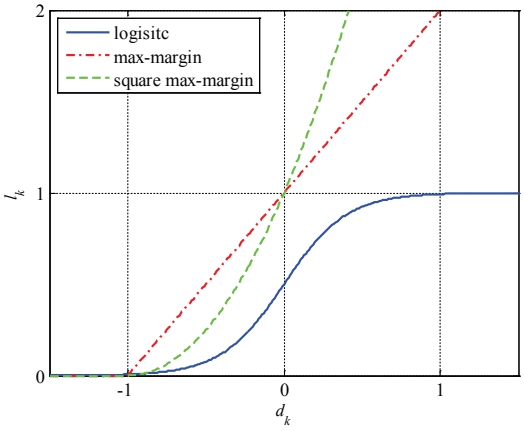

Fig. 1. Graphical depiction of logistic loss and our proposed max-margin loss values.

To set a higher loss for misclassification samples, we extend this as

$$\ell_k(d_k) = (1 + d_k)^\gamma, \tag{12}$$

where γ is a positive integer. If we set $\gamma = 2$, we obtain the squared max-margin loss. To use backpropagation, we compute the gradient of Eq. (12):

$$\frac{\partial \ell_k(d_k)}{\partial \mathbf{z}} = \gamma(1 + d_k)^{\gamma-1} \cdot \frac{\partial d_k(\mathbf{z})}{\partial \mathbf{z}}, \tag{13}$$

and

$$\frac{\partial d_k(\mathbf{z})}{\partial z_j} = \begin{cases} p_j\epsilon_k, & j \neq k \text{ and } j \neq q \\ p_j(\epsilon_k - 1), & j = k \\ p_j(\epsilon_k + 1), & j = q \end{cases}, \tag{14}$$

where $\epsilon_k = -d_k(\mathbf{z}) = p_k - p_q$ is the *margin*. Unlike Eq. (9), the gradient is not blocked. In Fig. 1, we compare the traditional logistic loss with our proposed max-margin loss values. Both Fig. 1 and Eq. (13) show that max-margin loss provides more error information.

To simplify the analysis, we set $\gamma = 1$ and substitute Eq. (14) into Eq. (13):

$$\frac{\partial l_k(d_k)}{\partial z_j} = \begin{cases} p_j\epsilon_k, & j \neq k \text{ and } j \neq q \\ p_j(\epsilon_k - 1), & j = k \\ p_j(\epsilon_k + 1), & j = q \end{cases}. \tag{15}$$

We also revisit the gradients of softmax with cross-entropy,

$$\frac{\partial l_k(d_k)}{\partial z_j} = \begin{cases} p_j - 1, & j = k \\ p_j, & j \neq k \end{cases}. \tag{16}$$

Obviously, the gradient with respect to the true class is negative in Eq. (16), but this depends on the margin ϵ_k in Eq. (15). If $\epsilon_k \in (0, 1)$, only the gradient of the target class is negative. However, when $\epsilon_k \in (-1, 0)$, all gradients are negative except for the confused class, which backpropagates the error signal to the bottom layers, possibly causing unpredictable mistakes. Hence, we combine cross-entropy and M^{3}CE:

$$L = aL_{CE} + bL_{MM}, \tag{17}$$

where the formulation of L_{MM} is the same as Eq. (7), and the misclassification measure and loss function are defined as in Eqs. (8) and (12), respectively. Thus, the total gradient combines the gradients of each loss function.

4. EXPERIMENTS

The proposed M^{3}CE-CE was evaluated on two benchmark datasets, MNIST and CIFAR-10. We designed different CNNs for these datasets, each of which is explained below. We randomly initialized the weights and biases and set the base learning rate to 0.01 and momentum to 0.9. For training the CNNs using cross-entropy, the learning rates were annealed during training by a factor of 10. All models were trained by the widely used Caffe [8]. For comparison, we set up three experiments: (1) using cross-entropy (CE) on its own, which is equivalent to setting $b = 0$ in loss function (17); (2) employing cross-entropy for pre-training and then tuning the model with M^{3}CE ($a = 1, b = 0 \rightarrow a = 0, b = 1$), referred to as M^{3}CE-CE switched (M^{3}CE-CEs); and (3) using cross-entropy and M^{3}CE-CE simultaneously ($a = 1$, $b = 1$), referred to as M^{3}CE-CE combined (M^{3}CE-CEc).

4.1. MNIST

MNIST [3] is a standard benchmark dataset used in machine learning. It contains ten classes of handwritten digits with 60,000 training images and 10,000 test images.

LeNet5 [3] is a typical CNN, originally designed specifically for MNIST. Therefore, we adopted the following CNN: 1x28x28-20C5-MP2-50C5-MP2-DR0.5-500N-DR0.5-10N, expressed in the style used by Cireşan et al. [5] to describe CNN architecture. The efficient regularization technique dropout [4] was applied in the top layers with a dropout rate of 0.5, denoted as DR0.5. We set the mini-batch size to 128. Fig. 2 shows the results of CE, M^{3}CE-CEs, and M^{3}CE-CEc. By incorporating M^{3}CE, the network converges more quickly and achieves a lower error rate of 0.45%, reducing the error of cross-entropy to 11.8%. In other words, M^{3}CE complements the cross-entropy loss function.

We compared our method with previous works using a single network without data augmentation. According to Table 1, all results are fairly similar, but both our method and Maxout Networks achieved the lowest error rate. This is noteworthy given that we used a very simple CNN architecture, LeNet5.

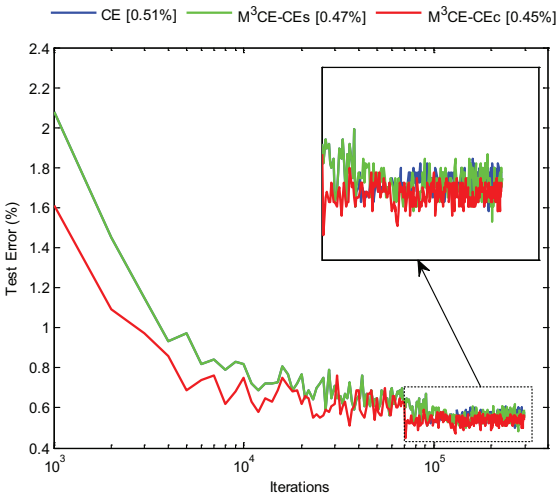

Fig. 2. Test error rates on MNIST dataset.

Table 1. Test error rates on MNIST

Method	Test Error(%)
DSN-Softmax [16]	0.51
Stochastic Pooling [17]	0.47
Network in Network [18]	0.47
Maxout Networks [19]	**0.45**
M^{3}CE-CEc	**0.45**

4.2. CIFAR-10

The CIFAR-10 dataset [10] consists of ten classes of natural images with 50,000 training images and 10,000 test images, which are color images of size 32×32. Because the CIFAR-10 is more challenging than MNIST, we employed a deeper CNN: 3x24x24-64C3-64C3-MP2-DR0.25-128C3-128C3-MP2-DR0.25-256C3-256C3-256C3-256C3-MP2-DR0.25-1024N-DR0.5-1024N-DR0.5-10N, inspired by VGG-Net [20]. We set the mini-batch size to 64. For data preprocessing, we used global contrast normalization and ZCA whitening, which have been adopted in [18] and [19]. Besides, random cropping with a crop size of 24 and flipping can be viewed as data augmentation to be applied in the training phase to prevent overfitting. In the test phase, we merely cropped the center of each test image.

We obtained test error rates of 7.44% using cross-entropy, but got 7.31% and 7.18% using M^{3}CE-CEs and M^{3}CE-CEc, respectively. We sampled five images to illustrate the effect of the proposed method, as shown in Fig. 3. In the first row, a dog image is misclassified as a frog and the probability of dog is lower than that of frog and cat. However, if we train the CNN by M^{3}CE-CEc, the probabilities of frog and cat are decreased and the probability of dog is increased. This means our method can provide more discriminative information than pure cross-entropy. This is consistent with the assumptions and motivations for this work.

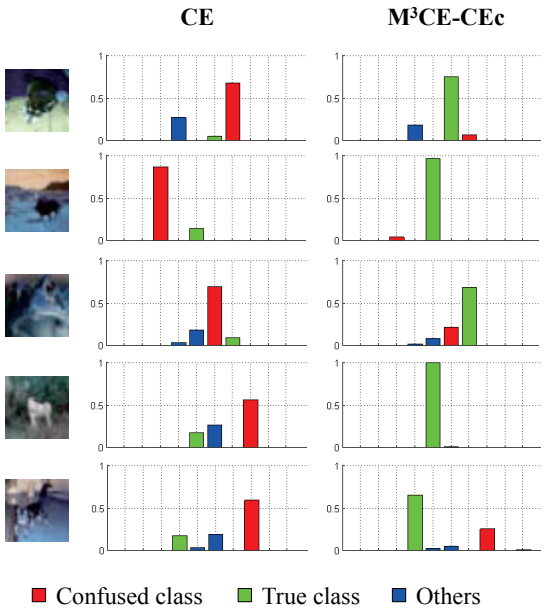

Fig. 3. Comparison of softmax outputs trained by CE and M^{3}CE-CEc.

Table 2. Test error rates on CIFAR-10

Method	Test Error(%)
Maxout Networks [19]	9.38
DropConnect [21]	9.32
Network in Network [18]	8.81
DSN [16]	8.22
CE	7.44
M^{3}CE-CEs	7.31
M^{3}CE-CEc	**7.18**

In Table 2, we compare our approaches with other methods. Since we employ a deeper CNN (14 layers), the basic method, CE, outperforms the other methods. It's also found that M^{3}CE-CEc improves the performance further using the proposed M^{3}CE. Our methods surpass those in previous works with M^{3}CE-CEc achieving the best performance.

5. CONCLUSION

In this paper, we proposed a novel max-margin MCE for training DNNs. M^{3}CE is derived from traditional MCE, but provides more discriminative information. We modified MCE by introducing a max margin criterion inspired by ReLU to be more suitable for training DNNs. We combined cross-entropy and the proposed M^{3}CE together to achieve better performance. We evaluated the proposed method M^{3}CE-CEc on two standard benchmark datasets, MNIST and CIFAR-10 and achieved promising results.

6. REFERENCES

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

[2] 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*, 2008, pp. 1096–1103.

[3] 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, 1998.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Neural Information Processing Systems*, 2012, pp. 1097–1105.

[5] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer Vision and Pattern Recognition, 2012 IEEE Conference on*. IEEE, 2012, pp. 3642–3649.

[6] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1915–1929, 2013.

[7] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Neural Information Processing Systems*, 2014, pp. 1988–1996.

[8] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[9] B. H. Juang and S. Katagiri, "Discriminative learning for minimum error classification," *IEEE Trans Signal Proc*, vol. 40, no. 12, pp. 3043–3054, 1992.

[10] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, University of Toronto, 2009.

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *arXiv preprint arXiv:1409.4842*, 2014.

[12] M. J. J. Holt and S. Semnani, "Convergence of back-propagation in neural networks using a log-likelihood cost function," *Electronics Letters*, vol. 26, no. 23, pp. 1964–1965, 1990.

[13] J. J. Hopfield, "Learning algorithms and probability distributions in feed-forward and feed-back networks.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 84, no. 23, pp. 8429–8433, 1987.

[14] S. Amari, "A theory of adaptive pattern classifiers," *Electronic Computers, IEEE Transactions on*, , no. 3, pp. 299–307, 1967.

[15] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 807–814.

[16] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," *arXiv preprint arXiv:1409.5185*, 2014.

[17] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *arXiv preprint arXiv:1301.3557*, 2013.

[18] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[19] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proceedings of The 30th International Conference on Machine Learning*, 2013, pp. 1319–1327.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[21] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 1058–1066.