Soft-Margin Softmax for Deep Classification

Xuezhi Liang[1,2,3], Xiaobo Wang[1,3(✉)], Zhen Lei[1,3], Shengcai Liao[1,3], and Stan Z. Li[1,2,3]

[1] Center for Biometrics and Security Research and National Laboratory of Pattern Recognition Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2] Center for Internet of Things, Chinese Academy of Sciences, Wuxi, China
[3] University of Chinese Academy of Sciences, Beijing, China
xzliang@cbsr.ia.ac.cn, {xiaobo.wang,zlei,scliao,szli}@nlpr.ia.ac.cn

Abstract. In deep classification, the softmax loss (Softmax) is arguably one of the most commonly used components to train deep convolutional neural networks (CNNs). However, such a widely used loss is limited due to its lack of encouraging the discriminability of features. Recently, the large-margin softmax loss (L-Softmax [1]) is proposed to explicitly enhance the feature discrimination, with hard margin and complex forward and backward computation. In this paper, we propose a novel soft-margin softmax (SM-Softmax) loss to improve the discriminative power of features. Specifically, SM-Softamx only modifies the forward of Softmax by introducing a non-negative real number m, without changing the backward. Thus it can not only adjust the desired continuous soft margin but also be easily optimized by the typical stochastic gradient descent (SGD). Experimental results on three benchmark datasets have demonstrated the superiority of our SM-Softmax over the baseline Softmax, the alternative L-Softmax and several state-of-the-art competitors.

Keywords: CNN · Softmax · L-Softmax · SM-Softmax · Classification

1 Introduction

Classification is a fundamental yet still challenging problem in machine learning and computer vision community. Over the past years, convolutional neural networks (CNNs) have shown significant improvements in many classification tasks, such as hand-written digit recognition [2], object recognition [3,4] and face recognition [5,6]. To train a deep CNN model, large scale training set and the end to end learning framework are indispensable. Facing the increasingly more complex data, CNNs can continuously be improved with dropout [3], deeper structure [7], new non-linear activations [8], regularization [9], stochastic pooling [10] and so on. Besides the above efforts, a renewed trend towards boosting the classification performance is to learn discriminative features with well-designed loss functions. However, this is non-trivial since a new loss function usually should be easily optimized by the typical stochastic gradient descent.

Intuitively, the learned features are good if their intra-class compactness and inter-class separability are well maximized. Based on such idea, the contrastive

D. Liu et al. (Eds.): ICONIP 2017, Part II, LNCS 10635, pp. 413–421, 2017.
https://doi.org/10.1007/978-3-319-70096-0_43

loss [5] and triplet loss [11] were proposed to enlarge the inter-class distinction as well as alleviate the intra-class variance. However, the number of training pairs and triplets are needed to be elaborately selected. The complexity can go up to $O(N^2)$ where N is the total number of training samples. Considering that CNNs often handle large scale training sets, the training processing may be inefficient. The hinge loss was adopted in [12] for classification. However, it is usually unstable to learn discriminative features. The softmax loss is widely used in many CNNs due to its simplicity and probabilistic interpretation. Despite its popularity, current softmax loss does not explicitly encourage the intra-class compactness and inter-class separability. The center loss was introduced in [13] and was combined with the softmax loss to enhance the intra-class compactness. It has achieved a promising performance on face recognition task. However, as pointed out in [14], combing a Euclidean based loss with softmax loss to construct a joint supervision may not be optimal. The Sparsemax [15] designed a new activation function similar to the softmax, but able to output sparse probabilities. The L-softmax loss [1] was developed to explicitly enforce the angle margin between different classes. However, the angle margin is a hard one since the corresponding parameter should be an integer. Moreover, the forward and backward computation of L-Softmax are complex.

In this paper, inspired by the recent work [1], we propose a novel soft-margin softmax (SM-Softmax) loss to effectively learn the discriminative features. Specifically, rather than introducing a hard angle margin as the work [1] does, we design a soft distant margin to enlarge the intra-class compactness and inter-class separability. In this way, we only need to change the forward computation of Softmax, without modifying the backward computation. Thus our SM-Softmax loss can be easily optimized by the standard stochastic gradient descent. Moreover, the designed soft distant margin theoretically contains all the hard angle margin in L-Softmax [1] and the degenerative margin (0) in Softmax. Thus the proposed SM-Softmax not only inherits all merits from Softmax and L-Softmax but also learns features with large soft margin between different classes. For clarity, the contribution of this paper can be summarized as follows:

- We design a new simple and powerful loss function namely SM-Softmax to strengthen the intra-class compactness and inter-class separability between learned features.
- We show that the proposed SM-Softmax loss is trainable and can be directly optimized by the typical stochastic gradient descent (SGD).
- Extensive experiments on MNIST, CIFAR10/CIFAR10+ and CIFAR100 datasets demonstrate the superiority of our SM-Softmax over the baseline Softmax, the alternative L-Softmax and several state-of-the-art methods.

2 Related Work

To learn discriminative CNNs features, existing works can be mainly classified into two categories: (1) Improving the deep CNN structures; (2) Designing better loss functions.

CNN Structures: The NiN [17] was instantiated the misro neural network with a multi-layer perceptron to enhance model discriminability for local patches within the receptive field. The Maxout [18] was designed for leveraging the dropout technique by enforcing the output to be the max of a set of inputs. The FitNet [19] was to address the network compression by introducing intermediate-level hints. The DSN [20] aimed to simultaneously minimizes classification error while making the learning process of hidden layers direct and transparent. The All-CNN [21] consisted solely of convolution layers by simply replacing the max-pooling into convolutional layer with increased layer. The R-CNN [22] resorted to a recurrent CNN for visual classification by incorporating recurrent connections into each convolutional layer. The GenPool [23] generalized the pooling operations in current CNNs to play a central role. Although existing CNN structures have achieved promising results for classification, they still suffer from the limited discrimination problem because of softmax loss.

Loss Functions: Currently many loss functions including contrastive loss [5], triplet loss [11], center loss [13], L-Softmax loss [1], softmax loss *etc.* have been used to train the CNNs. To make inter-class dispension and intra-class compactness as much as possible. The work [5] combined the softmax loss and the contrastive loss to jointly supervise the CNNs, with pairs of training samples as inputs. The work [11] adopted the triplet loss to encourage a distance constraint, requiring three (or a multiple of three) training samples as input at a time. The work [13] developed the center loss and fused it with softmax loss to learn discriminative features. The softmax loss is widely used in many CNNs and it can be written as follows:

$$L_{Softmax} = -\log\left(\frac{e^{\boldsymbol{W}_{y_i}^T \boldsymbol{x}_i}}{\sum_j^K e^{\boldsymbol{W}_j^T \boldsymbol{x}_i}}\right) = -\log\left(\frac{e^{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\cos(\theta_{y_i})}}{\sum_j^K e^{\|\boldsymbol{W}_j\|\|\boldsymbol{x}_i\|\cos(\theta_j)}}\right), \qquad (1)$$

where $\mathbf{x}_i$ denotes the deep feature of the i-th training sample. y_i is its corresponding label. $\mathbf{W} = [\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_K]^T$ is the parameters of the last fully connected layer, which can be also seen as the classifiers. K is the total number of classes. θ_j is the angle between the vector $\mathbf{W}_j$ and $\mathbf{x}_i$. The L-Softmax [1] loss employed a hard angle margin constraint in the original softmax loss, encouraging angular decision margin between classes to learn more discriminative features. Specifically, it can be formulated as:

$$L_{L-Softmax} = -\log\left(\frac{e^{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\cos(a\theta_{y_i})}}{e^{\|\boldsymbol{W}_{y_i}\|\|\boldsymbol{x}_i\|\cos(a\theta_{y_i})} + \sum_{j\neq y_i}^K e^{\|\boldsymbol{W}_j\|\|\boldsymbol{x}_i\|\cos(\theta_j)}}\right), \qquad (2)$$

where a is an integer that is closely related to the classification margin.

3 Soft-Margin Softmax Loss

To formulate our soft-margin softmax (SM-Softmax) loss, we first give a simple example to describe our intuition. Consider the binary classification and we have

Algorithm 1: Training a L-layers CNN supervised by SM-softmax loss.

```
Input: Training data $\{\mathbf{x}_i\}$. Initialized parameters $\mathbf{\Theta}$ in convolution layers.
       Parameters $\mathbf{W}$ in SM-Softmax loss layer. Hyperparameter $m$.
while not converged do
    Compute the forward propagation by the modified soft-margin Softmax (5);
    Compute the standard backward propagation;
    Update the parameters $\mathbf{W}$;
    Update the parameters $\mathbf{\Theta}$.
end
Output: The parameters $\mathbf{\Theta}$ and the weight $\mathbf{W}$.
```

a sample $\boldsymbol{x}$ from class 1. The original softmax classifier is to enforce $\boldsymbol{W}_1^T\boldsymbol{x} > \boldsymbol{W}_2^T\boldsymbol{x}$ (*i.e.*, $\|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(\theta_1) > \|\boldsymbol{W}_2\|\|\boldsymbol{x}\|\cos(\theta_2)$) to classify $\boldsymbol{x}$ correctly. To make the classification more rigorous, the work L-Softmax [1] introduces an angle margin as

$$\|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(\theta_1) \geq \|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(a\theta_1) > \|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(\theta_2), \tag{3}$$

and uses the intermediate value $\|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(a\theta_1)$ to replace $\|\boldsymbol{W}_1\|\|\boldsymbol{x}\|\cos(\theta_1)$ in the training. In that way, the class 1 and class 2 are explicitly separated. However, to make $\cos(a\theta_1)$ expand into Taylor series, a should be a positive integer. In other words, this margin cannot go through all possible angles and is a *hard* one. Moreover, the forward and backward computation are complex due to the angle margin involved. To address these issues, we here introduce a *soft* margin and simply let

$$\boldsymbol{W}_1^T\boldsymbol{x} \geq \boldsymbol{W}_1^T\boldsymbol{x} - m > \boldsymbol{W}_2^T\boldsymbol{x}, \tag{4}$$

where m is a non-negative real number and is a distant margin. In the training phase, we employ $\boldsymbol{W}_1^T\boldsymbol{x} - m$ to replace $\boldsymbol{W}_1^T\boldsymbol{x}$, thus our multi-class soft-margin softmax (SM-Softmax) classifier can be defined as:

$$s_i = \frac{e^{\boldsymbol{W}_{y_i}^T\boldsymbol{x}_i - m}}{e^{\boldsymbol{W}_{y_i}^T\boldsymbol{x}_i - m} + \sum_{j\neq y_i}^{K} e^{\boldsymbol{W}_j^T\boldsymbol{x}_i}}. \tag{5}$$

Finally, the soft-margin softmax (SM-Softmax) loss is formulated as

$$L_i = -\log\left(\frac{e^{\boldsymbol{W}_{y_i}^T\boldsymbol{x}_i - m}}{e^{\boldsymbol{W}_{y_i}^T\boldsymbol{x}_i - m} + \sum_{j\neq y_i} e^{\boldsymbol{W}_j^T\boldsymbol{x}_i}}\right). \tag{6}$$

Obviously, when m is set to zero, the SM-Softmax loss becomes identical to the original softmax loss. The advantages of the soft margin (4) can be summarized into two aspects. One is that the soft margin m can go through all the possible desired margins, and includes the hard margin a. The other one is that the SM-Softmax loss is easy to implement since it only changes the forward computation of Softmax. For clarity and completeness, we summarize the major optimization scheme in Algorithm 1.

4 Experiments

Following the protocol in [1], we demonstrate the effectiveness of the proposed SM-Softmax loss on three benchmark datasets and compare it with the baseline Softmax, the alternative L-Softmax [1] and several state-of-the-art competitors.

4.1 Dataset Description

Three benchmark datasets adopted in the experiments are those widely used for evaluating the performance of deep classification, including:
MNIST [24] is a dataset of handwritten digits (from 0 to 9) composed of 28×28 pixel gray scale images. It consists 60k training images and 10k test images. We scaled the pixel values to the [0,1] range before inputting to the CNN architecture.
CIFAR10 [25] is a set of natural color images of 32×32 pixels. It contains 50k training samples and 10k test samples. We adopt two commonly used comparison protocols on this dataset. We first compare our SM-Softmax with others under no data augmentation. For the data augmentation, we follow the standard technique in [1] for training, that is, 4 pixels are padded on each side, and a 32×32 crop is randomly sampled from the padded image or its horizontal flip. In test, we only evaluate the single view of the original 32×32 image. In addition, we subtract the per-pixel mean computed over the training set from each image before putting the images into the network.
CIFAR100 [25] is with the same size and format as the CIFAR10 dataset, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR100 are grouped into 20 superclasses.

Table 1. The detailed CNN architecture used in our work. Conv1.X, Conv2.X, Conv3.X denote convolution units that may contain multiple convolution layers. *E.g.,* $[3 \times 3,64] \times 4$ denotes 4 cascaded convolution layers with 64 filter of size 3×3.

Layer	Mnist	Cifar10/Cifar10+	Cifar100
conv0.X	$[3 \times 3,64] \times 1$	$[3 \times 3,64] \times 1$	$[3 \times 3,96] \times 1$
conv1.X	$[3 \times 3,64] \times 3$	$[3 \times 3,64] \times 4$	$[3 \times 3,96] \times 4$
Pool1	2×2 Max, Stride 2		
conv2.X	$[3 \times 3,64] \times 3$	$[3 \times 3,96] \times 4$	$[3 \times 3,192] \times 4$
Pool2	2×2 Max, Stride 2		
conv3.X	$[3 \times 3,64] \times 3$	$[3 \times 3,64] \times 4$	$[3 \times 3,384] \times 4$
Pool3	2×2 Max,Stride 2		
Fully connected	256	256	512

4.2 Compared Methods

We compare our SM-Softmax loss with the hinge loss, the commonly used Softmax loss, the recently proposed L-Softmax [1] and several state-of-the-art methods including the CNN [16], the DropConnect [2], the FitNet [19], the NiN [17], the Maxout [18], the DSN [20], the ALL-CNN [21], the R-CNN [22], the ResNet [4], the GenPool [23]. The results of all the compared methods are cropped from the original paper [1].

4.3 Implementation Details

We use Caffe [26] libary with our modifications to implement the proposed SM-Softmax loss. For the adopted CNN architecture, we follow the design philosophy of VGG-net [27], as the work [1] does. Specifically, for convolution layers, the kernel size is 3×3 and 1 padding (if not specified) to keep the feature map unchanged. For pooling layers, if the feature map size is halved, the number of filter is doubled to keep preserve the time complexity per layer. The detailed CNN architecture for each dataset are described in Table 1. For all experiments, We adopt the Relu [28] as the activation function and batch size is 128. We train all our models on a Nvidia Titan-X GPU and use the Caffe deep learning framework. CNN training is done with SGD with momentum 0.9 and weight decay of 0.0005. For the training, two stepwise strategy is adopted. We first train our CNN network supervised by softmax loss to obtain a good initialization. Then, we fine tune the CNN network supervised by our SM-Softmax loss based on the pre-trained model, with a small learning rate 0.01.

4.4 Performance Comparison

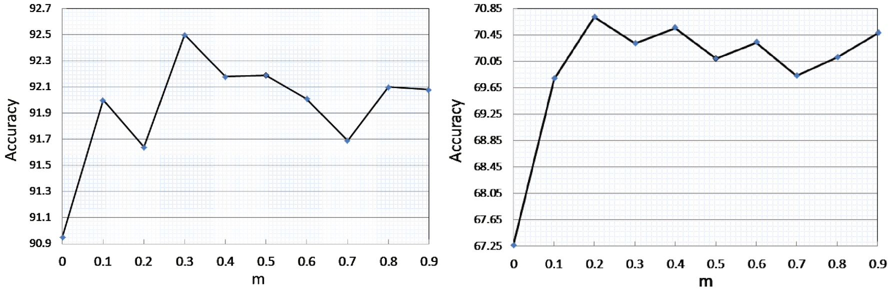
Table 2 provides the quantitative comparison among all the competitors on three benchmark datasets. The bold numbers in each column is the best performance. On MNIST, it is well-known that this dataset is typical and easy in deep classification. Almost all thee competitors can achieve under 1% error rate. The improvement of our SM-Softmax is not visibly big. On CIFAR10 and CIFAr10+, we can see that our SM-Softmax achieves about 2% improvement over the baseline Softmax and slightly better than the hard margin L-Softmax. On CIFAR100, a similar trend as that shown in MNIST and CIFAR10 is provided. In summary, the experiments have validated that the proposed SM-Softmax is significant better than Softmax due to its explicit discrimination on features, and is slightly better than the hard-margin L-Softmax because of its soft continuous margin.

4.5 Experiments on the Parameter m

The parameter m represent the soft margin between different classes. We investigate the sensitiveness of m on CIFAR10 and CIFAR100 as an example. Specifically, we vary the soft margin m from 0 to 0.9, with the stepsize of 0.1. From the

Table 2. Recognition error rate(%) on MNIST, CIFAR10 and CIFAR100 datasets. CIFAR10 denotes the performance without data augmentation, while CIFAR10+ is with data augmentation.

Method	MNIST	CIFAR10	CIFAR10+	CIFAR100
CNN [16]	0.53	N/A	N/A	N/A
DropConnect [2]	0.57	0.41	9.32	N/A
FitNet [19]	0.5	N/A	8.39	35.04
NiN [17]	0.47	10.47	8.81	35.68
Maxout [18]	0.45	11.68	9.38	38.57
DSN [20]	0.39	9.69	7.97	34.57
ALL-CNN [21]	N/A	9.08	7.25	N/A
R-CNN [22]	0.31	8.69	7.09	31.75
ResNet [4]	N/A	N/A	6.43	N/A
GenPool [23]	0.3	7.62	6.05	32.37
Hingeloss	0.47	9.91	6.96	32.90
Softmax	0.40	9.05	6.50	32.74
L-softmax [1]	0.31	7.58	5.92	29.53
SM-softmax	**0.30**	**7.50**	**5.73**	**29.28**



(a) CIFAR10

(b) CIFAR100

Fig. 1. Classification accuracy on CIFAR10 and CIFAR100 with different m.

curves in Fig. 1, we can observe that, as m grows, the accuracy rate grows gradually at the beginning and changes very slightly in a relatively large range of m. Moreover, we can clearly see that it reveals the effectiveness of our SM-Softmax ($m \neq 0$) in comparison with the baseline Softmax ($m = 0$).

5 Conclusions

This paper has proposed a novel soft-margin softmax (SM-Softmax) loss for deep classification tasks. The SM-Softmax achieves the discrimination of features by introducing a soft margin m between different classes. SM-Softmax only changes the forward of Softmax. Thus it can be easily optimized by the SGD. Extensive experiments have shown the advantages of our SM-Softmax over the baseline Softmax, the alternative L-Softmax and several state-of-the-art competitors.

Acknowledgments. This work was supported by the National Key Research and Development Plan (Grant No. 2016YFC0801002), the Chinese National Natural Science Foundation Projects #61473291, #61572501, #61502491, #61572536, #61672521 and AuthenMetric R&D Funds.

References

1. Liu, W., Wen, Y., Yu, Z.: Large-margin softmax loss for convolutional neural networks. In: ICML (2016)
2. Wan, L., Zeiler, M., Zhang, S.: Regularization of neural networks using dropconnect. In: ICML (2013)
3. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
4. He, K., Zhang, X.: Deep residual learning for image recognition. In: CVPR (2016)
5. Sun, Y., Chen, Y., Wang, X.: Deep learning face representation by joint identification-verification. In: NIPS (2014)
6. Taigman, Y., Yang, M., Ranzato, M.A.: Deepface: closing the gap to human-level performance in face verification. In: CVPR (2014)
7. Szegedy, C., Liu, W., Jia, Y.: Going deeper with convolutions. In: CVPR (2015)
8. He, K., Zhang, X., Ren, S.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: CVPR (2015)
9. Srivastava, N., Hinton, G.E., Krizhevsky, A.: Dropout: a simple way to prevent neural networks from overfitting. JMLR (2014)
10. Zeiler, M.D., Fergus, R.: Stochastic pooling for regularization of deep convolutional neural networks. arXiv preprint arXiv:1301.3557 (2013)
11. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: CVPR (2015)
12. Tang, Y.: Deep learning using linear support vector machines. arXiv preprint arXiv:1306.0239 (2013)
13. Wen, Y., Zhang, K., Li, Z., Qiao, Y.: A discriminative feature learning approach for deep face recognition. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9911, pp. 499–515. Springer, Cham (2016). doi:10.1007/978-3-319-46478-7_31
14. Liu, W., Wen, Y., Yu, Z.: SphereFace: deep hypersphere embedding for face recognition. In: CVPR (2017)
15. Martins, A., Astudillo, R.: From softmax to sparsemax: a sparse model of attention and multi-label classification. In: ICML (2016)
16. Jarrett, K., Kavukcuoglu, K., LeCun, Y.: What is the best multi-stage architecture for object recognition? In: ICCV (2009)
17. Lin, M., Chen, Q., Yan, S.: Network in network. In: ICLR (2014)
18. Goodfellow, I.J., Warde-Farley, D., Mirza, M.: Maxout Networks. In: ICML (2013)
19. Romero, A., Ballas, N.: Fitnets: Hints for thin deep nets. In: ICLR (2013)
20. Lee, C.Y., Xie, S., Gallagher, P.W.: Deeply-supervised nets. AISTATS (2015)

21. Springenberg, J.T., Dosovitskiy, A., Brox, T.: Striving for simplicity: The all convolutional net. In: ICLR (2015)
22. Liang, M., Hu, X.: Recurrent convolutional neural network for object recognition. In: CVPR (2015)
23. Lee, C.Y., Gallagher, P.W., Tu, Z.: Generalizing pooling functions in convolutional neural networks: mixed, gated, and tree. AISTATS (2016)
24. LeCun, Y. The MNIST database of handwritten digits (1998), http://yann.lecun.com/exdb/mnist/
25. Krizhevsky, A., Geoffrey, H.: Learning multiple layers of features from tiny images (2009)
26. Jia, Y., Shelhamer, E., Donahue, J.: Caffe: Convolutional architecture for fast feature embedding. In: ACM (2014)
27. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
28. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. AISTATS (2011)