IMPROVING DEEP NEURAL NETWORKS BY USING SPARSE DROPOUT STRATEGY

Hao Zheng, Mingming Chen, Wenju Liu, Zhanlei Yang, Shan Liang

National Laboratory of Pattern Recognition,
Institute of Automation, Chinese Academy of Sciences,
Beijing, P.R.China
{hao.zheng, mmchen, lwj, zhanlei.yang, sliang}@nlpr.ia.ac.cn

ABSTRACT

Recently, deep neural networks(DNNs) have achieved excellent results on benchmarks for acoustic modeling of speech recognition. By randomly discarding network units, a strategy which is called as dropout can improve the performance of DNNs by reducing the influence of over-fitting. However, the random dropout strategy treats units indiscriminately, which may lose information on distributions of units outputs. In this paper, we improve the dropout strategy by differential treatment to units according to their outputs. Only minor changes to an existing neural network system can achieve a significant improvement. Experiments of phone recognition on TIMIT show that the sparse dropout fine-tuning gets significant performance improvement.

Index Terms— dropout, sparse dropout, deep neural networks, deep learning

1. INTRODUCTION

Several years ago, Gaussian mixture models (GMMs) based acoustic models are widely used in most state-of-art automatic speech recognition (ASR) systems. To improve the performance, what most researchers tried to increase is the width of the neural networks. This approach is effective sometimes, but is not essentially. However, the work in [1] showed that an hidden Markov models (HMMs) system with pre-trained deep belief networks (DBNs) works significantly well on TIMIT data set, learning deep architecture became a research hotspot again. Experiments show that the pre-training with restricted Boltzmann machines (RBMs) method gives better initial weights for the back-propagation algorithm, which is pretty useful, especially when the number of labeled data is quite few.

From then on, DNN takes the place of GMM for modeling acoustic models. A large number of researches on DNN-HMM systems came out. Under the circumstance of big data and high-performance computing, DNN became more and more popular in many areas, such as computer vision and large vocabulary continuous speech recognition (LVCSR).

However, the data that the increasing model need is quite huge. With less amount of training data, the complex model is quite easy to over-fit. To overcome the over-fitting problem, one way is to choose larger amount of training data that covers more possible conditions, which is sometimes not easy to get. Another way is to add noise to the model [2] and the data [3], which is also the input layer of the model, making the model more robust and precise.

In this paper, we adopt sparse dropout strategy while fine-tuning deep belief networks (DBNs). The experiments on TIMIT phone recognition task show that the sparse dropout provide a 6.42% relative error reduction over a standard pre-trained DNN and a 12.90% absolute improvement over a GMM-HMM system.

The rest of this paper is organized as follows. In section 2 we describe the relations to prior work. A brief introduction of dropout strategy is given in section 3, while details of sparse dropout are described in section 4. Section 5 gives our experiment results on TIMIT set and some discussions. Finally we conclude our work and discuss our future work in the section 6.

2. RELATION TO PRIOR WORK

Dropout was first proposed by Hinton et al [4] to reduce the influence of over-fitting. Li et al gave a detailed study and a further discussion in [5]. Several works of dropout strategy on rectified linear units (ReLUs) such as [6] was then proposed. Experiments on different data sets show that the DNNs gain significant error reductions by using dropout strategy. According to their work, we made improvement on standard dropout strategy.

3. DROPOUT

The neural networks with deep size and powerful learning skills can model training data well enough. However, in most of these conditions, the models are over-fitting. Since the dis-

This research was supported in part by the China National Nature Science Foundation (No.91120303, No.61273267 and No.90820011).

tributions of test set and training set are different, especially when there is only a limited amount of labeled training data. The model which performs well on training set may make different predictions on test set and usually do worse on the test set than on the training set.

The Dropout strategy introduced in [4] is a useful training method for improving neural networks by preventing co-adaptation of feature detectors. The relations between units which are trained only by limited training data are always strong. The neural unit with dropout is randomly omitted and can not rely on other hidden units being present, which decreases the relations of each units. It is prone to perform better on test set.

Dropout strategy can be considered as adding noise to the model. Experiments in [3] show that it is very effective to add noise to data(input layers, which is a part of model), which is called as de-noising auto-encoder in this paper. With adding noise, model is more likely to learn the principal information of training data rather than over-fit the details, which makes the model gets good performance on both training and test sets.

Furthermore, it is an approximation to an average of a number of neural networks. With the number of hidden units N, there are 2^N different dropout models. Model averaging, such as decision tree, is an efficient method to reduce errors. The standard way to do model averaging is to train many separate networks and apply each of these networks to test data. However, the computation price is out of burden. The dropout strategy, which is computable, makes it possible to approximate the result of model averaging.

The computation for propagation we use in this paper is:

$$\mathbf{y}^{(l+1)} = \sigma(\mathbf{W}^{(l)T}\mathbf{x}^{(l)} + \mathbf{b}^{(l)}) \;\; 1 \leq l \leq L-1 \tag{1}$$

$$\mathbf{x}^{(l)} = dropout(\mathbf{y}^{(l)}, p) \;\; 2 \leq l \leq L \tag{2}$$

$$\mathbf{x}^{(l+1)} = (1-p)\sigma(\mathbf{W}^{(l)T}\mathbf{x}^{(l)} + \mathbf{b}^{(l)}) \tag{3}$$

where l is the number of layers, and L is the total number of layers. $W^{(l)}$ and $b^{(l)}$ are the weights and biases of the layer l, respectively. $\sigma(z)$ denotes the activation function for the lth layer, while p means the probability of dropout. We use Eq. (3) instead of original dropout network during test time so that the model performs as a mean network when all the units are used in a test process.

According to [4], for regressions with linear output units, the squared errors of the mean network are always better than the average of the squared errors of the dropout networks.

4. SPARSE DROPOUT

The common method to add sparsity to a model is to adopt L1 norm penalty to regular the loss function, which is a deterministic algorithm. Deterministic algorithms work well on problems which deterministic algorithms can surely find their best solutions, which does not include the parameter estimation of DNNs. Experiments show that some random algorithms such as Stochastic Gradient Decent or heuristic methods perform well on complex non-convex problems. One reason is that the loss function of the algorithm is always statistic negatively related to how good the solution is. This ensure the expectation of the algorithm can get is to be better. Another reason is that the algorithm does not always choose the best solution according to the local condition, which may reject the local optimums.

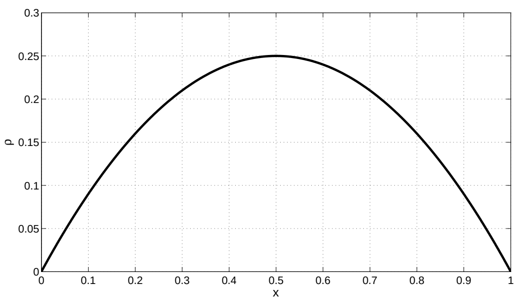

Fig. 1. Function $\rho(x)$

In dropout strategy, all the units are dropped equally, which does not respect the rule of sparsity. Different units with different values should be given differential treatments. An example of sparsity of unit output is the ReLU. By limiting negative value to zero, the unit outputs of ReLU becomes more sparse than non-zero value, which makes it sometimes gain a better performance than unit outputs with traditional sigmoid(we can regard tanh as a scaled sigmoid) function.

The output of sigmoid function is float and constant greater than zero, which means the strict judgment of sparsity is not suitable here. Sparse also means the distribution is unbalance. In this article we propose another evaluation function to appraise the sparsity of the unit output.

With satisfying the sparsity hypothesis, the expectation of value is tend to zero. However, the range of output of sigmoid function which we use as activation function is (0,1), which is to say, under the sparsity hypothesis, the expectation of unit output is prone to either 1 or 0. We use function (4) to express how sparse the unit output is.

$$\rho_i^{(l)} = x_i^{(l)}(1 - x_i^{(l)}) \tag{4}$$

where $\rho_i^{(l)}$ represents how sparse the ith unit of lth layer is; $x_i^{(l)}$ is the output of the ith unit, lth layer. The output range of sigmoid function is between zero and one, so that as shown in figure 1, the range of ρ is between 0 and 0.25. The value of ρ gets its maximum 0.25 when x equals to 0.5, which is least sparse in range (0,1).

As an indicator of degree of sparsity, the sparsity parameter ρ is also a measurement for dropout strategy. We use the probability $p^{(l)}(j) = \gamma\rho_j^{(l)}$ to take the place of global p and the formula (2) becomes

$$\mathbf{x}^{(l)} = dropout(\mathbf{y}^{(l)}, \gamma\rho^{(l)}) \tag{5}$$

where γ is a parameter to control the degree of sparse dropout. Then dropout strategy of each unit becomes a unit-dependent algorithm. With γ becomes larger, the units tend to be more sparse and the model less likely to be over-fitting, otherwise the model is easier to get convergence.

$$\mathbf{y}^{(l+1)} = (\mathbf{1} - \mathbf{p}^{(l)}) \circ \sigma(\mathbf{W}^{(l)T}\mathbf{x}^{(l)} + \mathbf{b}^{(l)}) \tag{6}$$

$$\mathbf{x}^{(l)} = dropout(\mathbf{y}^{(l)}, \gamma\rho^{(l)}) \tag{7}$$

where $\circ$ denotes element-wise product. The output of each unit should also be scaled while testing. As a general form of mean network used standard dropout strategy, we use a factor $(1 - \gamma\rho_j^{(l)})x^{(l)}$, which is the expectation of $x^{(l)}$ in formula (5), to scale the output of each unit.

5. EXPERIMENTS

Phone recognition on the TIMIT data sets was used to systematically evaluate the performance of sparse dropout fine-tuning. The training set is consisted of 3696 utterance from 462 speakers with 8 kind of different dialects. A 400 utterance of 50 speakers data set was chosen to be development set for validation purpose. The core test set with 192 utterance of 24 speakers was used for testing. Each of the 61 phone label was mapped to 3 HMM states, which means that the dimension of our DNN output layer was 183.

We used the conventional 13-dimension MFCC features, along with their first and second derivatives. 11 consecutive frames are used as input features of the network. Each dimension of the input features was normalized to be 0 mean and 1 variance. Mini-batches of size 256 were used for both pre-training and sparse dropout fine-tuning. The number of hidden layer was 4 and the unit number of each hidden layer was 2048. The sigmoid function was used as the activation function of hidden units, while soft-max function was used in output units.

The neural network was firstly pre-trained with a DBN, of which the first layer was trained for 60 epochs and others 30. The visible biases were initialized to zero and weights to random numbers sampled from a zero-mean normal distribution with standard deviation 0.01. The variance of each visible unit was fixed to be 1.0. Learning was done by minimizing contrastive divergence. Momentum started at 0.5 and was increased linearly to 0.9 . Learning rate started at 0.08 and was decreased linearly to 0.01.

The cross-entropy discriminative training was done by back-propagation algorithm to fine-tune the weights. The learning rate was started at 0.016 and scaled by a factor of 0.9 while the accuracy increase of cross validation was less than 0.1%. We also used a momentum of 0.2 to speed up learning. Maximum number of epoch iteration was set to 100.

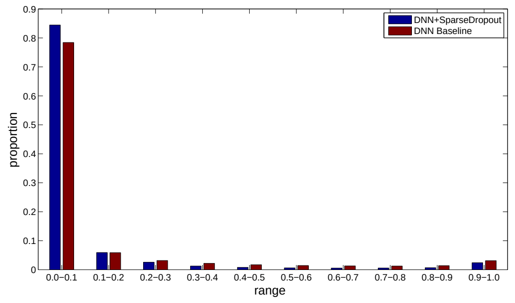

Fig. 2. Distributions of hidden layers' outputs

Table 1. Phone error rates with GMM and different kind of DNN models

	Dev	Test
GMM-Baseline	34.12	35.16
DNN-Baseline	21.70	23.69
DNN+0.4Dropout	20.87	22.91
DNN+0.5Dropout	22.33	23.70
DNN+SparseDropout	20.39	**22.26**

Table 1 lists the phone error rates obtained by GMM baseline and different kinds of DNN models. We choose the best parameter 0.4 tested in [5] and parameter 0.5 used in [1] as our dropout rates to be baselines. We have tested a list of parameter of ρ and used the best one 0.12 as sparse parameter of our result reported in Table 1. We found that the parameter from 0.8 to 1.5 gained almost the same performance, which indicated that this strategy is not very sensitive to parameter. To be fair, we give up using input dropout because all methods here are used without input dropout. From Table 1, we can see that DNN with sparse dropout achieves best performance on test set, which is about 6.42% relative improvement than DNN-Baseline and 2.92 percent relative improvement than standard dropout strategy.

Fig. 2 shows the distributions of hidden layers' outputs. The range of hidden units' outputs, which use sigmoid function in proposed, is from 0 to 1. In Fig. 2, we can see that the overall outputs of hidden units trained by sparse dropout are less than those trained by DNN-baseline, which reflects that the outputs are seem to be more sparse, especially with the larger amount of values between 0 and 0.1. The sparse dropout achieves a much less mean 0.0784 and a less variance 0.0363 than DNN-baseline, whose values are 0.1186 and 0.0527, respectively.

As a random algorithm, sparse dropout has all advantages of standard dropout strategy compared with DNN baseline.

The results of sparse dropout can keep decreasing or steady for a long time. It can work well not rely on early stopping, which needs a lot of experience to control and is with risky. It is pretty reliable to use sparse dropout.

6. CONCLUSIONS

In this paper, we applied sparse dropout strategy on DNNs to the TIMIT phone recognition task. Results show that the sparse dropout achieves not only phone error rate reduction but also unit output sparsity increase. The sparse dropout gain 6.42 percent relative improvement of PER than DNN baseline. Meanwhile, the distributions of hidden units outputs show that the outputs of units using sparse dropout strategy are prone to be more nearly to zero and more unbalanced. One of our future task is to apply sparse dropout on other type of activation functions such as Rectified Linear Unit. It seems to be a great meaningful work for that ReLU's much easier to get over-fitting. Another possible future task is to carry out experiments on LVCSR systems.

7. REFERENCES

[1] A.-r. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 14–22, 2012.

[2] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks." in *INTERSPEECH*, 2011, pp. 437–440.

[3] 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.

[4] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[5] J. Li, X. Wang, and B. Xu, "Understanding the dropout strategy and analyzing its effectiveness on lvcsr," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7614–7618.

[6] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *Proc. ICASSP*, 2013.

[7] Y. Miao and F. Metze, "Improving low-resource cd-dnn-hmm using dropout and multilingual dnn training," in *INTERSPEECH*, 2013, pp. 2237–2241.

[8] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *Proc. ICASSP*, 2013.

[9] D. Povey, "Discriminative training for large vocabulary speech recognition," *Cambridge, UK: Cambridge University*, vol. 79, 2004.

[10] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, 2011.

[11] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[12] O. Abdel-Hamid, L. Deng, and D. Yu, "Exploring convolutional neural network structures and optimization techniques for speech recognition," 2013.

[13] C. Lopes and F. Perdigão, "Phone recognition on the timit database," *Speech Technologies/Book*, vol. 1, pp. 285–302, 2011.

[14] A.-r. Mohamed, T. N. Sainath, G. Dahl, B. Ramabhadran, G. E. Hinton, and M. A. Picheny, "Deep belief networks using discriminative features for phone recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5060–5063.

[15] L. Tóth, "Phone recognition with deep sparse rectifier neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6985–6989.

[16] ——, "Convolutional deep rectifier neural nets for phone recognition," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[17] N. Srivastava, "Improving neural networks with dropout," Ph.D. dissertation, University of Toronto, 2013.

[18] S. M. Siniscalchi, P. Schwarz, and C.-H. Lee, "High-accuracy phone recognition by combining high-performance lattice generation and knowledge based rescoring," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4. IEEE, 2007, pp. IV–869.