Pruning Deep Neural Networks by Optimal Brain Damage

Chao Liu, Zhiyong Zhang, Dong Wang

Center for Speech and Language Technologies (CSLT)
Research Institute of Information Technology, Tsinghua University;
Department of Computer Science, Tsinghua University;
Tsinghua National Laboratory for Information Science and Technology
{liuc,zhangzy}@cslt.riit.tsinghua.edu.cn, wangdong99@mails.tsinghua.edu.cn

Abstract

A main advantage of the deep neural network (DNN) model lies on the fact that no artificial assumptions are placed on the data distribution and model structure, which offers the possibility to learn very flexible models. This flexibility, however, may lead to highly redundant parameters, hence demanding computation and risk of over-fitting. Network pruning cuts off unimportant connections, and therefore can be used to produce parsimonious and well generalizable models.

This paper proposes to utilize optimal brain damage (OBD) to conduct DNN pruning. OBD computes connection salience based on Hessians, and thus is sound in theory and reliable in practice. We present our implementation of OBD for DNNs, and demonstrate that the OBD pruning can produce very sparse DNNs while retaining the discriminative power of the original network to a large extent. By comparing with a simple magnitude-based pruning, we find that for weak pruned networks, pruning methods are unimportant since retraining can largely recover the function loss caused by pruning; while for highly pruned networks, sophisticated pruning methods (such as OBD) are clearly superior.

Index Terms: speech recognition, deep neural network, optimal brain damage, Hessian approximation

1. Introduction

Recent studies demonstrate that deep neural networks (DNN) are much more powerful than conventional Gaussian mixture models (GMM) in acoustic modeling for automatic speech recognition(ASR) [1, 2, 3]. A major advantage of the DNN model is that the hierarchical patterns of speech signals can be learned *automatically* from a large volume of data. In other words, there are no artificial assumptions placed on distributions of the input data and structures of the models, and all the discriminative properties (i.e., posterior distributions) are learned based on a layer-by-layer connected network.

This no-assumption property, plus the large-scale deep architecture, offers great flexibility for DNNs to learn very complex signal patterns and discriminative rules, provided that sufficient training data are available. Nowadays the DNN structure of a practical recognition system may be as huge as 8 hidden layers and 2000-4000 units per layer, amounting to more than 100 million parameters [4]. Although huge models have been demonstrated to be very powerful, most of the parameters in the model are redundant, as shown in [5] and the experimental results in Section 4. Involving a large proportion of redundant parameters in the model leads to weak generalization (over-fitting) and unnecessary computation when conducting inference.

The problem of parameter redundancy with neural networks has been studied for decades in neural computation. The principle solution is to place some heuristic or structural regularization on the network structure, so that salient weights/units are promoted. Weight decay [6] and the variants (e.g., [7]) follow this idea by introducing an ℓ-2 regularization. A more sparsity-oriented regularization is based on the ℓ-1 norm. For example, [8] applies the ℓ-1 norm on the bottleneck layer of encoder/decoder networks; the same approach was also applied to recurrent networks with rectifier-activated hidden units [9]. [10] introduces a similar regularization to train sparse deep Boltzmann machines. Another possible regularization is in the form $log(1 + v^2)$ where v is the weight/unit to regularize. This regularization, which equals to a student-t prior on v, has delivered better performance on phone recognition [11].

All the above approaches are implemented as a regularization term in the cost function. There are some other forms of regularizations, such as early stopping based on a validation set, rectifier activation that zeroes negative units [12, 13], and dropping-out that randomly discards a percentage of the network connections [14]. Although in different forms, the basic principles of these techniques are the same: introduce some priors/constraints/preferences on the network structure, so that the over-fitting problem with the 'blind training' can be partially solved.

In this paper, we focus on a particular regularization: network pruning. This technique cuts off those redundant and unimportant connections of a network and leave only the salient and essential structure. By pruning, we can learn a succinct network which is better generalizable and parsimonious in computation. In theory, the ℓ-1 regularization is able to drive part of the weights to vanish and thus leads to 'optimization-driven pruning', but in practice we seldom observe zero weights with an ℓ-1 norm, although they are indeed driven to smaller values. We therefore look for an explicit pruning which cuts off connections directly based on certain criteria.

Network pruning has been studied for decades [15, 16, 17, 18]. For DNN-based acoustic models, Yu et al. recently presented some experiments based on magnitude-based network pruning [5]. Although very simple in technique, the pruning seems quite promising: by cutting off 90% of the connections, the recognition accuracy is largely retained. We follow this direction and propose to prune DNNs by a Hessian-based approach, or optimal brain damage (OBD) [16]. Our experiments demonstrate that if we target for a very sparse network, the OBD-based pruning clearly outperforms the magnitude-based pruning.

The next section highlights the novelty of this paper compared to related works, and the OBD-based pruning is presented in Section 3. The experiments are reported in Section 4, followed by some conclusions in Section 5.

2. Related work

This paper is related to [5] in the sense that we all seek for a very sparse DNN. The difference is that [5] assumes that the magnitude (absolute value) of weights reflects salience of the associated connections, and so the pruning is performed based on $|w_{i,j}|$. We employ second order information encoded by Hessian to conduct the pruning, which is more sound in theory and powerful in practice.

Applying Hessians to network pruning has been well established in the neural computation community. Since computing Hessian matrices is very expensive, various approximations were proposed, e.g., the optimal brain damage (OBD) approach assumes a diagonal Hessian [16], and the optimal brain surgery (OBS) approximates the Hessian by outer product of a vector and its transpose. This paper applies OBD to prune DNNs. Although OBD has been well established in theory and validated in experiments, its application to DNNs is still questionable, because the DNN structure is highly complex and the diagonal assumption does not necessarily hold. From this perspective, one of the contributions of this paper is to confirm that the diagonal Hessian approximation is useful for DNNs.

3. Pruning with Diagonal Hessian

We start by presenting the diagonal Hessian approach, or OBD. The derivation is mostly the same as in [16] though the cost function is cross entropy in our case.

3.1. Optimal brain damage

Given a network whose connections are represented by a weight vector $w \in \mathcal{R}^K$ where K is the total number of weights in the network, a small change on the weight vector w, denoted by δw, causes a change on the cost function E as follows:

$$\delta E = E(w + \delta w) - E(w).$$

Applying the Taylor expansion on $E(w + \delta w)$, we have:

$$\delta E \approx \delta w \frac{\partial E}{\partial w} + \frac{1}{2}\delta w^T H \delta w$$

where H is the Hessian:

$$H = \begin{pmatrix} \frac{\partial E^2}{\partial w_1 \partial w_1} & \cdots & \frac{\partial E^2}{\partial w_1 \partial w_K} \\ \cdots & \cdots & \cdots \\ \frac{\partial E^2}{\partial w_K \partial w_1} & \cdots & \frac{\partial E^2}{\partial w_K \partial w_K} \end{pmatrix}.$$

For a well-trained network, there is $\frac{\partial E}{\partial w} = 0$ and so the change on E is mostly caused by the second order term. Further ignore the non-diagonal elements of H, we have:

$$\delta E \approx \frac{1}{2}\sum_{i=1}^{K} h_{i,i} \delta {w_i}^2$$

where $h_{i,i} = \frac{\partial E^2}{\partial {w_i}^2}$. Note that cutting off a connection w_i is equal to change the associated weight to 0, therefore the cost of pruning w_i is:

$$\delta E_i \approx \frac{1}{2} h_{i,i} {w_i}^2. \quad (1)$$

The OBD pruning scheme is as follows: first train the network until the training converges, and then compute δE_i for all the connections i. This computation is based on the entire database although it is possible to use mini batches. So the computing cost is roughly another iteration of the BP process. Prune the network according to δE_i either by a threshold on δE_i or to a percentage of the whole connections. An interesting property of OBD is that if the Hessian components are set to equal, the OBD pruning falls back to the weight magnitude-based pruning.

3.2. Hessian back propagation

We now describe how to compute the Hessian component $h_{i,i}$. The procedure is a back propagation (BP) proposed in [16], although we derive the procedure with the cross entropy cost.

Concentrating on a particular layer $m-1$ and the next layer m. Let the tuple (m, i, j) denote a connection from the j-th unit of layer $m-1$ to the i-th unit of layer m, and further denote the associated weight by $w^m_{i,j}$. The activation of the i-th unit at layer m is given by:

$$a^m_i = f(y^m_i) \quad y^m_i = \sum_j w^m_{i,j} a^{m-1}_j$$

where f is the logistic function for hidden units and the softmax function for output units. Assuming that the k-th connection is (m, i, j), the Hessian component $h_{k,k}$ is derived as:

$$h_{k,k} = \frac{\partial^2 E}{\partial {w^m_{i,j}}^2} = \frac{\partial^2 E}{\partial (y^m_i)^2}(a^m_j)^2$$

where the second order term can be computed in a back-propagation style:

$$\frac{\partial^2 E}{\partial (y^m_i)^2} = f'(y^m_i)^2 \sum_l w^2_{l,i} \frac{\partial^2 E}{\partial (y^{m+1}_l)^2} + f''(y^m_i)\frac{\partial E}{\partial a^m_i}.$$

Note that the second-order BP procedure requires an accompanying conventional BP to compute the first-order derivatives. The BP procedure starts from the output layer M. Considering that the cost is cross entropy and the output activation is softmax, the initialization of the BP procedure can be derived:

$$\frac{\partial E}{\partial (y^M_i)} = \sum_n \{a^M_i(n) - t_i(n)\}$$

$$\frac{\partial^2 E}{\partial (y^M_i)^2} = \sum_n a^M_i(n)(1 - a^M_i(n))$$

where n indexes the training samples, and $t_i(n)$ is the i-th component of the label of the n-th sample.

3.3. Training heuristics

The OBD pruning is then straightforward: first train the network until convergence, and then sweep over the training data (an extra iteration) by running the second order BP. For each mini batch, accumulate the cost of each connection according to (1). Prune the network according to the cost once all the data have been processed, and then retrain the sparse network until convergence.

Directly application of the above procedure, however, did not work for DNNs in our experiments. Careful analysis reveals two problems. First, we found that some Hessian values $h_{k,k}$ are extremely large in some mini batches, which leads to an unrecoverable bias. This can be attributed to the odd cliffs in the cost function which in turn may be caused by the complex parameter space of DNNs. We set a ceiling value θ for $h_{k,k}$ and round those large values to θ, for which was empirically set to 0.001 in our experiments.

In addition, we observe quite some negative values of $h_{k,k}$. This is not consistent with the assumption that the network has been well trained. This might be an indicator that the diagonal approximation is not very appropriate for complex networks such as DNNs. We enforce a non-negative constraint on the Hessian to remedy the problem, i.e., setting negative values to zeros in the Hessian.

Sparsity					
Total Net.	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
0.50	0.76	0.55	0.58	0.48	0.44
0.25	0.57	0.25	0.32	0.26	0.21
0.12	0.42	0.08	0.14	0.15	0.10
0.06	0.32	0.04	0.07	0.10	0.06

Table 1: Sparsity of different layers in the OBD-based pruned network.

4. Experiments

4.1. Databases and settings

The experiments were conducted with a Chinese continuous speech database that was obtained from the Tencent corporation. This database was mostly recorded through online applications and thus can largely reflect practical usage scenarios, such as complex speaker varieties and multiple channel and environment conditions. We chose 100 hours of speech data for model training (including 7,000 utterances as the validation set), and another 17 hours of speech data (27,000 utterances) for testing.

The 40-dimensional Fbank feature was used in DNN training and recognition. The DNN input involves a window of 11 frames of Fbank features that are reduced to 200 dimensions by an LDA transform. The DNN architecture involves 4 hidden layers and each layer consists of 1200 units. The output layer is composed of 3580 units, corresponding to 3580 tied context dependent states. The Kaldi toolkit[1] was used to conduct the DNN training, and we largely follow the wsj s5 recipe based on a single GPU. Specifically, the training starts from a randomly initialized DNN, and then the stochastic gradient descendent (SGD) approach was adopted to conduct optimization with the mini batch size set to 256 frames. The initial learning rate is 0.008, and it is started to be halved once the accuracy improvement on a held-out cross validation (cv) dataset is less than 0.5, and the training stops if the cv accuracy improvement is less than 0.1.

Once the DNN was trained, it is used as the acoustic model of an HMM-DNN hybrid ASR system. A 110k-words lexicon plus a 3-gram word-based language model was used in decoding.

4.2. Network classification accuracy

In the first experiment, we study the discriminative power of DNNs before and after pruning. We compare the magnitude-based pruning and the OBD pruning, and investigate the performance with/without retraining. The pruning was conducted by setting a global threshold on the magnitude of weights (magnitude-based approach) or δE_i (OBD-based approach), leading to pruned networks with various sparsity degrees, as shown in Table 1, where the sparsity of the entire network and each layer are presented.

The performance of the pruning approaches is evaluated in terms of classification accuracy (CA) of frames. Table 2 and Table 3 present the results on the training set and the validation set respectively.

We can observe that on the training set, the OBD pruning is consistently better than the magnitude-based pruning, particularly when the pruning is harsh. On the validation set, the advantage of OBD is still evident, although the difference is a bit reduced. In addition, we can see that the retraining is highly important for both the two pruning approaches. By re-optimizing the pruned sparse networks, the accuracy loss caused by pruning is largely recovered, and the discrepancy between the OBD pruning and the magnitude-based pruning is reduced. Nevertheless, the OBD pruning is still clearly superior in the cases where the pruning is rigorous.

[1] http://kaldi.sourceforge.net/

	CA%			
	Magnitude		OBD	
Sparsity	-	+	-	+
1.00	47.63	-	47.63	-
0.50	45.02	46.70	46.55	47.43
0.25	33.87	46.83	41.13	47.06
0.12	12.84	44.66	27.41	45.34
0.06	1.31	41.81	14.43	42.92

Table 2: Classification accuracy on the training set with magnitude-based and OBD-based pruning. '-' means without retraining, and '+' means after retraining.

	CA%			
	Magnitude		OBD	
Sparsity	-	+	-	+
1.00	45.32	-	45.32	-
0.50	43.88	45.32	44.79	45.29
0.25	34.81	45.29	41.06	45.24
0.12	13.63	44.60	28.56	44.96
0.06	1.25	42.66	15.34	43.80

Table 3: Classification accuracy on the validation set with magnitude-based and OBD-based pruning. '-' means without retraining, and '+' means after retraining.

4.3. Speech recognition performance

In this experiment, the pruned-and-retrained DNNs are used to perform speech recognition. The performance is evaluated in word error rates (WER). Table 4 presents the results. We observe that with either pruning approach, the power of the DNN is largely remained even though the pruned structure is highly sparse. In the case of extremely sparse networks, the OBD pruning clearly outperforms the magnitude-based pruning. This seems indicate that optimization is more important than pruning, so if there is sufficient opportunity to perform optimization, the intrinsic redundancy in the network structure can be utilized to support the functions of the pruned neurons and the entire network will not be damaged much. This is the case of weak pruning. In heavy pruning, there are few redundant neurons left and the retraining has little chance to recover the functions of the pruned neurons. In this case, a clever pruning approach such as OBD is more desirable: it prunes out less important neurons/weights so that the major discriminative power is affected as little as possible. This is the case of the heavy pruning in our experiments.

5. Conclusions

We present in this paper an OBD-based approach for DNN pruning. An assumption of this approach is that the Hessian matrix is diagonal, which is not necessarily true for DNNs whose parameter space is highly complex. Fortunately, with some practical designs such as value ceiling and non-negative constraining, the OBD pruning works well in producing very sparse networks, and outperforms the simple magnitude-based pruning. In the case of modest sparse networks, pruning approach is less important since the retraining can recover most of the functions of the pruned weights. Future work involves neuron pruning instead of weight pruning, online pruning instead of batch pruning, and OBD pruning with ℓ-1 regularization. An-

	WER%	
Sparsity	Magnitude	OBD
1.00	24.04	
0.50	24.01	24.14
0.25	24.36	24.32
0.12	25.41	25.09
0.06	27.64	26.49

Table 4: WER results on the speech recognition task.

other unaddressed problem is the highly inefficient computation with sparse matrices, which requires much effort in research and engineering.

6. Acknowledgements

This research was supported by the National Science Foundation of China (NSFC) under the project No. 61371136. It was also supported by the Tencent Corporation, Sinovoice Corporation and the Huilan Ltd.

7. References

[1] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Large vocabulary continuous speech recognition with context-dependent DBN-HMMs," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4688–4691.

[2] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[3] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013.

[4] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero, "Recent advances in deep learning for speech research at Microsoft," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013.

[5] D. Yu, F. Seide, G. Li, and L. Deng, "Exploiting sparseness in deep neural networks for large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4409–4412.

[6] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 4, 1992, pp. 950–957.

[7] R. Setiono, "A penalty function approach for prunning feedforward neural networks," *Neural computation*, vol. 9, no. 1, pp. 185–204.

[8] M. Ranzato, Y. Boureau, and Y. LeCun, "Sparse feature learning for deep belief networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2007, pp. 1185–1192.

[9] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, 2012.

[10] H. Lee, "Unsupervised feature learning via sparse hierarchical representations," Ph.D. dissertation, 2010.

[11] G. S. Sivaram and H. Hermansky, "Multilayer perceptron with sparse hidden outputs for phoneme recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5336–5339.

[12] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier networks," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume*, vol. 15, 2011, pp. 315–323.

[13] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013.

[14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," The Computing Research Repository (CoRR), 2012.

[15] J. Sietsma and R. J. Dow, "Neural net pruning-why and how," in *Neural Networks, 1988., IEEE International Conference on*. IEEE, 1988, pp. 325–333.

[16] Y. LeCun, J. S. Denker, S. A. Solla, R. E. Howard, and L. D. Jackel, "Optimal brain damage." in *Advances in Neural Information Processing Systems (NIPS)*, vol. 2, 1989, pp. 598–605.

[17] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems (NIPS)*, 1993, pp. 164–164.

[18] J. Langford, L. Li, and T. Zhang, "Sparse online learning via truncated gradient," *The Journal of Machine Learning Research*, vol. 10, pp. 777–801, 2009.