EXPLOITING SPARSENESS IN DEEP NEURAL NETWORKS FOR LARGE VOCABULARY SPEECH RECOGNITION

Dong Yu[1], *Frank Seide*[2], *Gang Li*[2], *Li Deng*[1]

[1]Microsoft Research, Redmond, USA
[2]Microsoft Research Asia, Beijing, P.R.C
{dongyu,fseide,ganl,deng}@microsoft.com

ABSTRACT

Recently, we developed context-dependent deep neural network (DNN) hidden Markov models for large vocabulary speech recognition. While reducing errors by 33% compared to its discriminatively trained Gaussian-mixture counterpart on the switchboard benchmark task, DNN requires much more parameters. In this paper, we report our recent work on DNN for improved generalization, model size, and computation speed by exploiting parameter sparseness. We formulate the goal of enforcing sparseness as soft regularization and convex constraint optimization problems, and propose solutions under the stochastic gradient ascent setting. We also propose novel data structures to exploit the random sparseness patterns to reduce model size and computation time. The proposed solutions have been evaluated on the voice-search and switchboard datasets. They have decreased the number of nonzero connections to one third while reducing the error rate by 0.2-0.3% over the fully connected model on both datasets. The nonzero connections have been further reduced to only 12% and 19% on the two respective datasets without sacrificing speech recognition performance. Under these conditions we can reduce the model size to 18% and 29%, and computation to 14% and 23%, respectively, on these two datasets.

Index Terms— speech recognition, deep belief networks, deep neural networks, sparseness

1. INTRODUCTION

Recently, we have witnessed the resurrection of artificial neural network (NN) hidden Markov model (HMM) hybrid systems for speech recognition. This mainly attributes to the discovery of the strong modeling ability of deep neural networks (DNNs[1]) and the availability of high-speed general purpose graphical processing units (GPGPUs) for training DNNs. A notable advance is the context-dependent DNN-HMMs (CD-DNN-HMMs) in which DNNs directly model the senones (i.e., tied CD phone states) and approximate their emission probabilities in HMM speech recognizers [1].

CD-DNN-HMMs have been shown to be highly promising. They have achieved 16% [1] and 33% [2,3] relative recognition error reduction over strong, discriminatively trained CD-GMM (Gaussian mixture model)-HMMs, respectively, on a voice search (VS) task [4] and the switchboard (SWB) phone-call transcription task [5].

Unfortunately, CD-DNN-HMMs use much more parameters than the corresponding CD-GMM-HMMs due to the large number of layers used in DNNs and the direct modeling of as many as 20,000 or more senones to achieve high recognition accuracy. In addition, the lower layers in DNNs are shared across all the states and need to be computed even if only one state is active in the search path. It is thus of practical importance to reduce the DNN model size so that fast computation can be possible and better generalization can be achieved. In this paper we attack this problem by exploiting parameter sparseness and formulating the task of enforcing sparseness as soft regularization and convex constraint optimization problems. We compare the performance under the stochastic gradient ascent (SGA) setting, which, to our best knowledge, is the only scalable training procedure for DNNs at present. We further propose data structures to exploit the seemingly random sparseness patterns to save storage and to speed up decoding.

[1]We define DNNs as multi-layer perceptrons (MLPs) with many more hidden layers than used before. DNNs are more difficult to train and may benefit from procedures such as deep belief network (DBN) pretraining.

2. CD-DNN-HMM

The CD-DNN-HMM combines the discriminative modeling power of DNN with the sequential modeling power of HMM. The basic idea behind CD-DNN-HMMs has been known since 1990s [6]. CD-DNN-HMMs differ from previous work in that they directly model tied CD phone states and do so using DNNs. Empirical evaluation [1,2] indicated that both these two aspects are critical for CD-DNN-HMMs to outperform the speaker-independent state-of-the-art CD-GMM-HMMs on large vocabulary speech recognition tasks.

A DNN models the posterior probability $P_{\mathbf{s}|\mathbf{o}}(s|o)$ of a class s given an observation vector o, as a stack of $(L+1)$ layers of log-linear models. The first L layers, $\ell = 0...L-1$, model hidden binary units h^ℓ given input vectors v^ℓ as Bernoulli distribution

$$P^\ell_{\mathbf{h}|\mathbf{v}}(h^\ell|v^\ell) = \prod_{j=1}^{N^\ell} \frac{e^{z_j^\ell(v^\ell)\cdot h_j^\ell}}{e^{z_j^\ell(v^\ell)\cdot 1} + e^{z_j^\ell(v^\ell)\cdot 0}}\ ,\ 0 \le \ell < L \quad (1)$$

and the top layer L models the class posterior as multinomial distribution

$$P^L_{\mathbf{s}|\mathbf{v}}(s|v^L) = \frac{e^{z_s^L(v^L)}}{\sum_{s'} e^{z_{s'}^L(v^L)}} = \mathrm{softmax}_s(z^L(v^L)) \quad (2)$$

where $z^\ell(v^\ell) = (W^\ell)^T v^\ell + a^\ell$ is the activation at layer ℓ, W^ℓ and a^ℓ are the weight matrices and bias vectors at layer ℓ, and h_j^ℓ and $z_j^\ell(v^\ell)$ are the j-th component of h^ℓ and $z^\ell(v^\ell)$, respectively.

The precise modeling of $P_{\mathbf{s}|\mathbf{o}}(s|o)$ is infeasible as it requires integration over all possible values of h^ℓ across all layers. An effective practical trick is to replace the marginalization with the "mean-field approximation" [7]. Given observation o, we set $v^0 = o$ and choose conditional expectation $E^\ell_{\mathbf{h}|\mathbf{v}}\{\mathbf{h}^\ell|v^\ell\} = \sigma\big(z^\ell(v^\ell)\big)$ as input $v^{\ell+1}$ to the next layer, where $\sigma_j(z) = 1/(1+e^{-z_j})$ is the sigmoid function.

Given T training samples $o(t)$ and the associated ground-truth labels $s(t)$, DNNs are often trained to maximize the total log conditional probability

$$D = \sum_{t=1}^{T} \log P_{\mathbf{s}|\mathbf{o}}(s|o), \tag{3}$$

The only feasible procedure to train DNNs on a large amount of training samples so far is the *error back-propagation* (BP) procedure with stochastic gradient ascent (SGA). In its basic form,

$$\frac{\partial D}{\partial W^\ell} = \sum_t v_t^\ell (\omega_t^\ell e_t^\ell)^T \quad ; \quad \frac{\partial D}{\partial a^\ell} = \sum_t \omega_t^\ell e_t^\ell \tag{4}$$

where the error signals $e^L = (\log \mathrm{softmax})'(z^L(v^L))$, $e^{\ell-1} = W^\ell \cdot \omega^\ell \cdot e^\ell$, and $\omega^\ell = \mathrm{diag}\left(\sigma'(z^\ell(v^\ell))\right)$ if $0 \le \ell < L, \omega^\ell = 1$, otherwise, the component-wise derivatives $\sigma_j'(z) = \sigma_j(z) \cdot (1 - \sigma_j(z))$, $(\log \mathrm{softmax})_j'(z) = \delta_{s(t),j} - \mathrm{softmax}_j(z)$, and δ is Kronecker delta.

DNNs are more difficult to converge to good solutions and computationally more demanding to train than the shallow MLPs. It is only recently that training DNNs has become feasible, with the easy access to high-speed GPGPUs and the discovery of effective weight initialization techniques, in particular "deep belief network" pretraining [8]. The detailed steps on training a CD-DNN-HMM system can be found in [1].

3. EXPLOITING SPARSENESS

It is our experience that recognition accuracy of CD-DNN-HMMs typically increases with the number of hidden units and layers, as long as the training process is controlled by a held-out set. For example, in the SWB task, we continue to see small but consistent accuracy improvement even with 9 hidden layers [2]. Resulting optimal models, however, are large. As shown in Section 4, the CD-DNN-HMMs trained using 24 hours of VS data and 300 hours of SWB data have 19 and 45 million parameters, respectively. They each correspond to 12 and 2 times the size of the corresponding traditional CD-GMM-HMMs.

Fortunately, inspection of fully connected DNNs after the training has shown that a large portion of all connections have very small weights. Fig. 1 illustrates the distribution of weight magnitudes of a typical 7-hidden-layer DNN. In this specific example, the magnitude of 70% of weights is below 0.1. This inspired us to reduce model size by removing connections with small weight magnitude so that we can work with deeper and wider DNNs more effectively. Note that we did not observe similar patterns on bias parameters. This is expected since nonzero bias terms indicate the shift of hyperplanes from the origin. Since the number of bias parameters is very small compared to that of weight parameters, keeping bias parameters intact does not affect the final model size in a noticeable way.

The task of enforcing sparseness can be formulated as a multi-objective optimization problem since we want to maximize the log conditional probability D and minimize the number of nonzero weights at the same time. This two-objective optimization problem can be converted into a single objective optimization problem with soft regularization and convex constraint formulations. Note that researchers have investigated the MLP weight sparseness problems in the past. The most well known work [9, 10] pruned the weights after training converges based on the second-order derivatives. Unfortunately, these algorithms are difficult to scale up to large training set we typically use in speech recognition and their advantages vanish if additional training iterations are carried out upon the pruned weights.

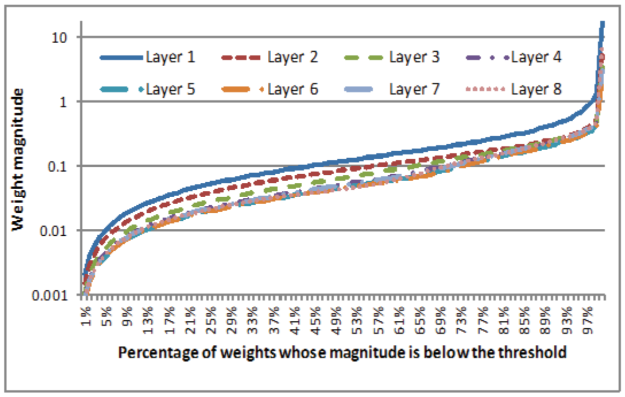

Fig. 1. The weight magnitude distribution of a 7-hidden-layer DNN illustrated as the percentage of weights whose magnitude is below a threshold.

3.1. Soft Regularization Formulation

In the soft regularization formulation, we convert the problem to maximizing the criterion

$$D_0 = \sum_{t=1}^{T} \log P_{\mathbf{s}|\mathbf{o}}(s|o) - \alpha \cdot \|W\|_0, \tag{5}$$

where the L_0 matrix norm $\|W\|_0$ is the number of nonzero weights, and α is a balancing parameter. We further approximate D_0 as

$$D_1 = \sum_{t=1}^{T} \log P_{\mathbf{s}|\mathbf{o}}(s|o) - \alpha \cdot \|W\|_1, \tag{6}$$

by replacing L_0-norm with L_1-norm.

Maximizing D_1 can be solved by following the sub-gradient

$$\frac{\partial D_1}{\partial W^\ell} = \frac{\partial D}{\partial W^\ell} - \alpha \cdot \mathrm{sgn}(W^\ell) \tag{7}$$

where $\mathrm{sgn}(\cdot)$ is the sign function applied element-wise.

Unfortunately, the SGA method with sub-gradient (7) usually does not generate precise sparse solutions. To enforce a sparse solution, one often truncates the solutions after each K steps by forcing parameters with magnitude smaller than a threshold θ to zero [11]. This truncation step, however, is somewhat arbitrary and is not a direct derivation from optimizing D_1. In addition, K is difficult to select correctly. In general, it is not desirable to take a small K (e.g., 1), especially when the minibatch size is small, since in that case each SGA update step only slightly modifies weights. When a parameter is close to zero it remains so after several SGA updates and will be rounded back to zero if K is not sufficiently large. Consequently, truncation can be done only after (a reasonably large) K steps in the hopes that nonzero coefficients have sufficient time to go above θ. On the other hand, a large K means that every time the parameters are truncated, D will be reduced and will require a similar number of steps to get the loss compensated.

3.2. Convex Constraint Formulation

In the convex constraint formulation, we maximize the log conditional probability D subject to the constraint

$$\|W\|_0 \le q \tag{8}$$

where q is a threshold value for the maximal number of nonzero weights allowed.

This constrained optimization problem is hard to solve. However, an approximate solution can be found following two observations: First, after sweeping through the full training set several times the weights become relatively stable — they tend to remain either large or small magnitudes. Second, in a stabilized model, the importance of the connection is approximated well by the magnitudes of the weights (times the magnitudes of the corresponding input values, but these are relatively uniform within each layer since on the input layer, features are normalized to zero-mean and unit-variance, and hidden-layer values are probabilities). This leads to the very simple yet efficient and effective Algorithm 1. After Step 2 sparseness constraint (8) is enforced. However, the log conditional probability D is reduced due to connection pruning, esp. when the degree of sparseness is high (i.e., q is small). It is thus important to apply Step 3 and continue training the DNN, which tends to converge much faster than the original training.

Algorithm 1 Main Steps to Train a Sparse DNN

```
1: Train a fully connected DNN by sweeping through the full training set several times using labels from forced alignment.
2: Keep only the connections whose weight magnitudes are in top $q$.
3: Continue training the DNN with the sparseness pattern generated from Step 2 unchanged.
```

We have developed an empirically effective approach to keeping the same sparse connections (and thus same sparseness constraint) in Step 3. We can either mask the pruned connections or round weights with magnitude below $\min\{0.02, \theta/2\}$ to zero, where θ is the minimal weight magnitude that survived the pruning and 0.02 is determined by the sparseness pattern shown in Fig. 1. The masking approach is cleaner but requires storage of a huge masking matrix. In our implementation we used the rounding alternative. Note that it is important to round only weights smaller than $\min\{0.02, \theta/2\}$, instead of θ, to zero. This is because the weights may shrink and be suddenly removed and it is desirable to keep the effect of this removal to minimum without sacrificing the degree of sparseness.

Compared to the sub-gradient solution associated with the soft regularization formulation, Algorithm 1 carries several benefits: First, threshold q can be easily determined by examining the weights after Step 1. Determining the truncation parameter θ in the sub-gradient solution, however, is very difficult. Second, we can mask or truncate the weights after each SGA update in Step 3 without performance degradation. However, in the sub-gradient solution, it is very difficult and sometimes impossible to find a good update number K. Third, in the sub-gradient solution, we need to tune the balancing parameter α while there is no such parameter in Algorithm 1.

3.3. Data Structure

The sparse weights learned generally have random patterns. Here we propose data structures to effectively exploit the sparse weights to reduce model size and to speed up decoding calculation ($W^T v$). One data structure example is depicted in Figure 2. The basic idea is simple: only store and calculate with the nonzero-weights (nzws). To speed up the calculation we stored the indexes and actual weights into adjacent groups so that they can be retrieved efficiently with good locality. A slightly different but almost equally efficient data structure is to group pairs of indexes and weights. With the proposed data structure, each column can be multiplied with the input vector in parallel. To further speed up the calculation, parallelization can also happen within each column.

Note that the data structure shown in Figure 2 is just one implementation. The best data structure depends heavily on the hardware architecture chosen and the trade-off between storage size and computation speed. For example, the index block i_ks can be further compressed by keeping the delta indexes (requires only one byte per index). Further more, if streaming SIMD extension (SSE) instructions are used, we can group frames into batches of four and store nonzero weights row-first to achieve similar computation speedup.

The saving of storage from using the data structure shown in Figure 2 is obvious. For an $N \times M$ single-precision matrix with $x\%$ nonzero-weights, the normal matrix requires $4 \times N \times M$ bytes. With the proposed data structure, it requires $2 + 6 \times M(1 + x\% \times N)$ bytes, which takes less space when $x\% < 2/3 - 1/N$.

The speedup of calculation depends heavily on the implementation and hardware used. For a naive matrix-vector multiplication (i.e., SSE is not used), it requires $N \times M$ multiplications and summation, and $2 \times N \times M$ memory accesses. With the proposed data structure, it requires only $x\% \times N \times M$ multiplications and summations, and $3 \times x\% \times N \times M$ memory accesses.

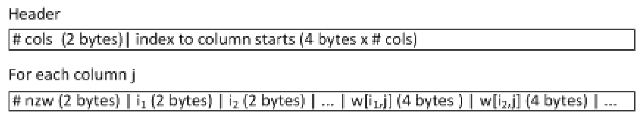
Header
# cols (2 bytes) | index to column starts (4 bytes x # cols)
For each column j
# nzw (2 bytes) | i_1 (2 bytes) | i_2 (2 bytes) | ... | w[i_1,j] (4 bytes) | w[i_2,j] (4 bytes) | ...

Fig. 2. A data structure that can exploit the random sparseness pattern in weight matrices to save storage and speed up calculation, where nzws = nonzero-weights

4. EXPERIMENTAL RESULTS

Evaluation was done on the VS and SWB datasets.

4.1. Dataset Description

The VS dataset contains US-wide business and location search queries with a 24-hour (or 32,057-utterance) training set, a 6.5-hour (or 8,777-utterance) development set, and a 9.5-hour (or 12,758-utterance) test set. For the sake of easy comparisons, we have used the same lexicon, tri-gram language model (LM), and 39-dim MFCC features as in previous studies [1]. The LM contains 65K word unigrams with a perplexity of 117 and OOV rate of 6% on the test set. The total number of senones and number of mixtures in this dataset is 761 and 24 respectively and were optimized for CD-GMM-HMM systems trained under maximum likelihood (ML) criterion.

The SWB dataset used in this study contains the standard 309-hour Switchboard-I training set [5], and the NIST 2000 Hub5 and RT03S (FSH portion) evaluation sets. The system uses 13-dim PLP features with windowed mean-variance normalization and up to third-order derivatives, reduced to 39 dimensions by HLDA. The 3-state speaker-independent crossword triphones share 9304 40-mixture senones optimized for the ML-trained GMM-HMM system.

4.2. Results and Discussions

To obtain the experimental results shown in Tables 1 and 2 we have followed the steps in [1] to build the fully-connected CD-DNN-HMMs and steps in Alg. 1 for the sparse models. In all the setups, DNNs were first pre-trained using the DBN pretraining algorithm [8] generatively and then fine-tuned discriminatively using the BP algorithm. Additional details can be found in [1, 2]. CD-DNN-HMMs

Table 1. *Model size, computation time, and percent query error rate (QER) with and without sparseness constraints on the VS dataset. CD-DNN-HMMs contain 5 hidden layers each with 2048 nodes trained using CD-DNN-HMM alignment. 'nz' means 'nonzero.' The OOV rate for both the dev and test sets is about 6%.*

acoustic model	# of nz params	Model Size	Calc Time	Dev QER	Test QER
GMM MPE	1.5M	-	-	34.5	36.2
CD-DNN-HMM	19.2M	100%	100%	28.0	30.4
sparse: 67% nz	12.8M	101%	80%	27.9	30.3
sparse: 46% nz	8.8M	69%	55%	27.7	30.1
sparse: 31% nz	6.0M	47%	37%	27.7	30.1
sparse: 21% nz	4.0M	32%	25%	27.8	30.2
sparse: 12% nz	2.3M	18%	14%	27.9	30.4
sparse: 5% nz	1.0M	8%	6%	29.7	31.7

Table 2. *Model size, computation time, and percent word error rate (WER) with and without sparseness constraints on the SWB dataset. CD-DNN-HMMs contain 7 hidden layers each with 2048 nodes; trained using CD-DNN-HMM alignment. 'nz' means 'nonzero.'*

acoustic model	# of nz params	Model Size	Calc Time	Hub5'00 SWB	RT03S FSH
GMM, BMMI	29.4M	-	-	23.6	27.4
CD-DNN	45.1M	100%	100%	16.4	18.6
sparse: 69% nz	31.1M	104%	83%	16.2	18.5
sparse: 52% nz	23.6M	78%	62%	16.1	18.5
sparse: 34% nz	15.2M	51%	41%	16.1	18.4
sparse: 24% nz	11.0M	36%	29%	16.2	18.5
sparse: 19% nz	8.6M	29%	23%	16.4	18.7
sparse: 15% nz	6.6M	22%	18%	16.5	18.7

outperform discriminatively trained CD-GMM-HMMs by 16% and 33% relative error reduction, respectively, on the VS and SWB datasets. Note that in both cases the size of the CD-DNN-HMM is much larger than that of the corresponding CD-GMM-HMM.

We do not present the results obtained by the sub-gradient algorithm associated with the soft regularization formulation in Tables 1 and 2 due to its limitations discussed in Section 3. Because of the requirement to tune many parameters in the sub-gradient algorithm and the fact that training DNNs are expensive, we only applied this algorithm to several configurations on the VS dataset. From all the configurations we have tried, the sub-gradient algorithm consistently under-performs Alg. 1 by 1-2% in absolute terms. Furthermore, if we use θ instead of $\min\{0.02, \theta/2\}$ in Step 3 of Alg. 1 as the truncation threshold, we lose 0.1-0.3% in absolute terms.

Overall, by exploiting the sparseness property in the model, we obtained 0.2-0.3% absolute error reduction and simultaneously reduced the connections to only $1/3$ on both the VS and SWB datasets. Alternatively, we can reduce the number of weights to 12% and 19%, respectively, on the VS and SWB datasets, without sacrificing recognition accuracy. In that case, the CD-DNN-HMM is only 1.5 and 0.3 times as large as the CD-GMM-HMM on the VS and SWB datasets, respectively, and takes only 18% and 29% of the model size compared to the fully-connected models. This translates to reducing the DNN computation to only 14% and 23% of that needed by the full-connected models on the VS and SWB datasets respectively, using either the basic CPU implementation or the SSE implementation with frame batching.

5. CONCLUSION

The main focus of the research reported in this paper is to reduce the model size, speed up the computation, and improve the generalization ability of CD-DNN-HMMs. We have shown that while DNNs can become very large as more layers are added, the majority of the weights are close to zero and this is exploited to reduce the model size in this work. We formulated the task of reducing the number of connections in the DNN as soft regularization and convex constraint optimization problems and proposed solutions under the SGA setting. The method derived from the convex constraint formulation we reported in detail in this paper performs better and is easier to implement than the sub-gradient algorithm associated with the soft regularization formulation. Experiments on VS and SWB datasets demonstrated that a significant improvement in model size and computation time can be obtained with the same or sometimes slightly higher recognition accuracy.

Acknowledgment

Our thanks go to A. Gunawardana, K. Thambiratnam, J. Droppo, N. Ström, A. Stolcke, A. Jayamohan and I. Kouzminykh.

6. REFERENCES

[1] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," *IEEE Trans. Speech and Audio Proc., Special Issue on Deep Learning for Speech and Lang. Processing*, 2012.

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

[3] X. Chen F. Seide, G. Li and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *ASRU*, 2011.

[4] D. Yu, Y. C. Ju, Y. Y. Wang, G. Zweig, and A. Acero, "Automated directory assistance system - from theory to practice," in *Proc. Interspeech*, 2007, pp. 2709–2711.

[5] J. Godfrey and E. Holliman, "Switchboard-1 release 2," in *Linguistic Data Consortium*. 1997.

[6] S. Renals, N. Morgan, H. Boulard, M. Cohen, and H. Franco, "Connectionist probability estimators in HMM speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 1, pp. 161–174, 1994.

[7] C. Peterson and J. Anderson, "A mean field theory learning algorithm for neural networks," *Complex Systems*, pp. 995–1019, 1987.

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

[9] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*. 1990, pp. 598–605, Morgan Kaufmann.

[10] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems 5*. 1993, pp. 164–171, Morgan Kaufmann.

[11] J. Langford, . Li, and T. Zhang, "Sparse online learning via truncated gradient," *J. Mach. Learn. Res.*, vol. 10, pp. 777–801, 2009.