DEEP STACKING NETWORKS FOR INFORMATION RETRIEVAL

Li Deng, Xiaodong He, and Jianfeng Gao

{deng,xiaohe,jfgao}@microsoft.com

Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA

ABSTRACT

Deep stacking networks (DSN) are a special type of deep model equipped with parallel and scalable learning. We report successful applications of DSN to an information retrieval (IR) task pertaining to relevance prediction for sponsor search after careful regularization methods are incorporated to the previous DSN methods developed for speech and image classification tasks. The DSN-based system significantly outperforms the LambdaRank-based system which represents a recent state-of-the-art for IR in normalized discounted cumulative gain (NDCG) measures, despite the use of mean square error as DSN's training objective. We demonstrate desirable monotonic correlation between NDCG and classification rate in a wide range of IR quality. The weaker correlation and more flat relationship in the high IR-quality region suggest the need for developing new learning objectives and optimization methods.

Index Terms— deep stacking network, information retrieval, document ranking

1. INTRODUCTION

Deep stacking networks (DSN) are a recent information processing architecture developed from deep learning and speech processing research [3][16]. DSN has advantages over other deep models in its simplicity in learning --- not requiring stochastic gradient descent which renders parallelization of network parameter learning virtually impossible. The strength of DSN in scalable learning lies in a simple training objective --- the mean square error (MSE) between the target value and the network prediction in each module of the DSN architecture.

The simplicity of the DSN's training objective drastically facilitates its successful applications to image recognition, speech recognition, and speech understanding [4][16]. The MSE objective and classification error rate have been shown to be well correlated. For information retrieval (IR) applications, however, the inconsistency between the MSE objective and the desired objective (e.g., normalized discounted cumulative gain (NDCG) [12]) is much greater than that for the above classification-focused applications. For example, NDCG as a desirable IR objective function is a highly non-smooth function of the parameters to be learned, with a very different nature from the nonlinear relationship between MSE and classification error rate. RankNet [1], which has been successful in IR, had to use surrogate objective functions with computable gradients but their values are only loosely coupled with the desired NDCG.

We are thus interested in the answers to the following question: Is NDCG reasonably well correlated with classification rate or MSE where the relevance level in IR is used as the DSN prediction target? And further (especially if the answer is positive), can the advantage of learning simplicity in DSN be applied to improve IR quality measures such as NDCG? The main goal of the research reported in this paper is to address the above questions. Our experimental results presented in this paper provide largely positive answers to both. In addition, we explore and address some special care that need to be taken in implementing DSN learning algorithms when moving from classification to IR applications.

2. DEEP STACKING NETWORK: ARCHITECTURE

The philosophy of DSN design rests in the concept of stacking, as proposed originally in [17], where simple modules of functions or classifiers are composed first and then they are "stacked" on top of each other so as to learn complex functions or classifiers. Following this philosophy, [3] presented the basic form of the DSN architecture that consists of many stacking modules, each of which takes a simplified form of shallow multilayer perceptron using convex optimization for learning perceptron weights.

Fig. 1 gives an example of a four-module DSN, each consisting of three sub-layers and being illustrated with a separate color. Dashed lines in green denote layer duplications. "Stacking" is accomplished by concatenating all previous modules' output predictions with the original input vector to form the new "input" vector in the new module. The DSN weight parameters $\boldsymbol{W}$ and $\boldsymbol{U}$ in each module are learned efficiently from training data, which we describe below.

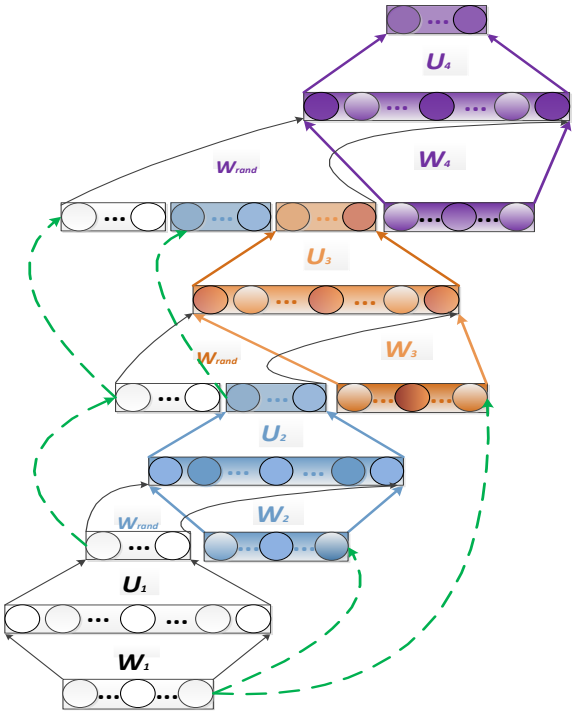

Fig. 1: An illustration of the DSN architecture.

3. DEEP STACKING NETWORK: LEARNING

In each module of the DSN, the output units are linear and the hidden units are sigmoidal nonlinear. The linearity in the output units permits highly efficient, parallelizable, and close-form estimation (a result of convex optimization) for the output network weight matrices $\boldsymbol{U}$ given the hidden units' activities. Due to the close-form constraints between the input and output weights, the input weight matrices W can also be elegantly estimated in an efficient, parallelizable, batch-mode manner. In following sections, the indices of the network weight matrices are omitted for simplification.

3.1 Basic learning algorithm

Denote training vectors by $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_i, \cdots, \boldsymbol{x}_N]$, in which each vector is denoted by $\boldsymbol{x}_i = [x_{1i}, \cdots, x_{ji}, \cdots, x_{Di}]^T$ where D is the dimension of the input vector, which is a function of the module, and N is the total number of training samples. Denote L the number of hidden units and C the dimension of the output vector. Then, the output of the a DSN module is $\boldsymbol{y}_i = \boldsymbol{U}^T \boldsymbol{h}_i$, where $\boldsymbol{h}_i = \sigma(\boldsymbol{W}^T \boldsymbol{x}_i)$ is the hidden layer output, $\boldsymbol{U}$ is an $L \times C$ weight matrix at the upper layer, $\boldsymbol{W}$ is an $D \times L$ weight matrix at the lower layer, and $\sigma(\cdot)$ is the sigmoid function. (Bias terms are implicitly represented in the above formulation if $\boldsymbol{x}_i$ and $\boldsymbol{h}_i$ are augmented with ones.)

Given target vectors $\boldsymbol{T} = [\boldsymbol{t}_1, \cdots, \boldsymbol{t}_i, \cdots, \boldsymbol{t}_N]$, where each vector is $\boldsymbol{t}_i = [t_{1i}, \cdots, t_{ji}, \cdots, t_{Ci}]^T$, the parameters $\boldsymbol{U}$ and $\boldsymbol{W}$ are learned to minimize the average of the total square error

$$E = \|\boldsymbol{Y} - \boldsymbol{T}\|^2 = \mathrm{Tr}[(\boldsymbol{Y} - \boldsymbol{T})(\boldsymbol{Y} - \boldsymbol{T})^T] \quad (1)$$

where $\boldsymbol{Y} = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_i, \cdots, \boldsymbol{y}_N]$. Note that once the lower layer weights $\boldsymbol{W}$ are fixed (e.g., by random numbers), the hidden layer values $\boldsymbol{H} = [\boldsymbol{h}_1, \cdots, \boldsymbol{h}_i, \cdots, \boldsymbol{h}_N]$ are also determined uniquely. Consequently, the upper layer weights $\boldsymbol{U}$ can be determined by setting the gradient

$$\frac{\partial E}{\partial \boldsymbol{U}} = \frac{\partial \mathrm{Tr}[(\boldsymbol{U}^T\boldsymbol{H} - \boldsymbol{T})(\boldsymbol{U}^T\boldsymbol{H} - \boldsymbol{T})^T]}{\partial \boldsymbol{U}} = 2\boldsymbol{H}(\boldsymbol{U}^T\boldsymbol{H} - \boldsymbol{T})^T \quad (2)$$

to zero, leading to the closed-form solution

$$\boldsymbol{U} = (\boldsymbol{H}\boldsymbol{H}^T)^{-1}\boldsymbol{H}\boldsymbol{T}^T \quad (3)$$

3.2 Module-bound fine tuning

The weight matrices $\boldsymbol{W}$ of the DSN in each module can be further learned using batch-mode gradient descent [18]. The computation of the error gradient makes use of Eq. (3) and proceeds by

$$\frac{\partial E}{\partial \boldsymbol{W}} = \frac{\partial\, \mathrm{Tr}[(\boldsymbol{U}^T\boldsymbol{H} - \boldsymbol{T})(\boldsymbol{U}^T\boldsymbol{H} - \boldsymbol{T})^T]}{\partial \boldsymbol{W}} \quad (4)$$

$$= \frac{\partial\, \mathrm{Tr}[([(\boldsymbol{H}\boldsymbol{H}^T)^{-1}\boldsymbol{H}\boldsymbol{T}^T]^T\boldsymbol{H} - \boldsymbol{T})([(\boldsymbol{H}\boldsymbol{H}^T)^{-1}\boldsymbol{H}\boldsymbol{T}^T]^T\boldsymbol{H} - \boldsymbol{T})^T]}{\partial \boldsymbol{W}}$$

$$= \frac{\partial\, \mathrm{Tr}[\boldsymbol{T}\boldsymbol{T}^T - \boldsymbol{T}\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{H}^T)^{-1}\boldsymbol{H}\boldsymbol{T}^T]}{\partial \boldsymbol{W}}$$

$$= -\frac{\partial\, \mathrm{Tr}[(\boldsymbol{H}\boldsymbol{H}^T)^{-1}\boldsymbol{H}\boldsymbol{T}^T\boldsymbol{T}\boldsymbol{H}^T]}{\partial \boldsymbol{W}}$$

$$= -\frac{\partial\, \mathrm{Tr}[(\boldsymbol{\sigma}(\boldsymbol{W}^T\boldsymbol{X})[\boldsymbol{\sigma}(\boldsymbol{W}^T\boldsymbol{X})]^T)^{-1}\boldsymbol{\sigma}(\boldsymbol{W}^T\boldsymbol{X})\boldsymbol{T}^T\boldsymbol{T}[\boldsymbol{\sigma}(\boldsymbol{W}^T\boldsymbol{X})]^T]}{\partial \boldsymbol{W}}$$

$$= 2\boldsymbol{X}\left[\boldsymbol{H}^T \circ (\boldsymbol{1} - \boldsymbol{H})^T \circ [\boldsymbol{H}^\dagger(\boldsymbol{H}\boldsymbol{T}^T)(\boldsymbol{T}\boldsymbol{H}^\dagger) - \boldsymbol{T}^T(\boldsymbol{T}\boldsymbol{H}^\dagger)]\right],$$

where $\boldsymbol{H}^\dagger = \boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{H}^T)^{-1}$ is pseudo-inverse of $\boldsymbol{H}$. How to initialize $\boldsymbol{W}$ in gradient descent in stacking modules can be found in [4].

3.3 Regularization in the DSN learning

During this study, we found that regularization in DSN learning is much more important for the IR task than for the speech and image classification tasks investigated earlier.

One particular difficulty in IR is the low dimensionality in the output vectors associated with each module in the DSN. For instance, in our experimental data (see details in Section 4) the output consists of only two values: one for being relevant and zero for being non-relevant. The low dimensionality weakens the stacking information provided from a lower module of the DSN to its upper module, compared with the speech tasks where the number of classes to be recognized tends to be much higher --- e.g., around 200 phone-state classes [4].

The problem of low dimensionality and the smaller amount of training data for our IR task compared with the earlier speech experiments require special care --- the implementation of effective regularization mechanisms while learning the DSN parameters as described in Subsections 3.1 and 3.2. In all experiments reported in Section 4, we implemented *L2* regularization for learning weight matrices $\boldsymbol{U}$ in Eq. (3). Regularization for learning weight matrices $\boldsymbol{W}$ is implemented by adding a separate "data reconstruction error" term in the gradient of Eq. (4) and by carefully tuning a weight parameter between this reconstruction error and the original "target error" terms.

4. EVALUATION

4.1 The IR tasks and data sets

We have recently conducted extensive experiments on a sponsored web IR task using DSN. In addition to the organic web search results, commercial search engines also provide supplementary sponsored results in response to the user's query. The sponsored results are selected from a database pooled by advertisers who bid to have their ads displayed on the search result pages. Given an input query, the search engine will retrieve relevant ads from the database, rank them, and display them at the proper place on the search result page; e.g., at the top or right hand side of the web search results [11][13]. Finding relevant ads to a query is quite similar to common web search. E.g., although the documents come from a constrained database, the task resembles typical search ranking that targets on predicting document relevance to the input query.

In this work, we learn a DSN model of ad relevance that helps improve the sponsored search system. Our relevance model is trained to distinguish relevant and irrelevant ads given a search query. Particularly, our model assigns a relevance score to an ad given a query. Then, ads are further ranked by their relevance score. Like typical IR tasks, we measure the performance of our ad relevance model by NDCG at positive 1, 3, and 10.

Our DSN-based IR system is compared with LambdaRank[2] as the baseline. The targets of these systems are generated from annotated data with "relevant" or "irrelevant" judgment for each query-ad pair, where judgments are performed by professional annotators. Our training and test sets contain 189K and 58K query ad pairs, respectively.

4.2 Features to DSN and baseline systems

The ranking features used in the network models in this study can be categorized into two main groups: text features and user click features. We use a very similar set of text features to those proposed in [9][10]. They include: 1) query length features (i.e., the number of characters and words); and 2) three sets of text matching features, each of which compares the query text to one of the three text streams of an ad (i.e., the title, description, and word-segmented display URL). Each feature set includes unigram similarities (computed using TFIDF [15] and BM25 [14]), word overlap (unigram, bigram, and skipped bigram), character overlap (unigram, bigram, and skipped bigram), etc.

The two types of user click features that we use are both derived from clickthrough logs (i.e., a list of query and clicked ad pairs). The first type is clickthrough features. Following [5], we construct for each ad a click stream that consists of a list of queries with clicks on the ad, and then extract a set of 30 features by matching the click stream to the input query. The click stream can be viewed as a description of the ad from users' perspective. The second type of click features used in our experiments is a set of translation probabilities between query and ad based on the translation models learned on the query-ad pairs extracted from clickthrough logs [6].

4.3 Experimental results

As our baseline system, we use LambdaRank [2], one of the state-of-the-art rankers. LambdaRank is a two-layer (shallow) neural net ranker that maps a feature vector $\mathbf{x}$ to a real value y, called relevance score, which indicates the relevance of the document given the query. The excellent performance of LambdaRank demonstrated earlier lies in the fact that it can directly optimize NDCG by using an implicit cost function whose gradient are specified by rules, called lambda-functions. Table 1 presents the NDCG-1, 3, and 10 results of this LambdaRank baseline in comparison with the new DSN system described in Sections 2 and 3. In all three NDCG measures, the DSN outperforms the baseline significantly.

Table 1. IR quality comparisons between a state-of-the-art baseline ranker and the DSN ranker.

IR Systems	NDCG@1	NDCG@3	NDCG@10
LambdaRank	0.331	0.347	0.382
DSN system	0.359	0.366	0.402

In Fig. 2, we show the correlation between classification error rates (which are closely correlated with the training objectives of MSE and not shown here) in the test set and the corresponding NDCG1 values. Desirable monotonic correlation is clearly evidenced, especially for NDCG values below 0.35. However, weaker correlation and a wider range of error rates can be identified for a fixed NDCG value in the high IR-quality region with NDCG above 0.35. This

indicates that the inconsistency between the training objective and the IR-quality measure becomes a critical issue in that region. In the future, it is desirable to train the model using techniques that optimizes an objective that is closely related to the end-to-end IR quality, like the discriminative training methods widely used for speech recognition [7][8] and more recent end-to-end decision-feedback training approaches applied successfully to speech translation [19].

Finally, to analyze the learning behavior of DSN, we plot in Fig. 3 the learning curves in terms of the three NDCG measures for the test set as a function of the training epoch. Each epoch is one sweep of all 189K training vectors in learning $\boldsymbol{U}$ and $\boldsymbol{W}$ in a DSN module. Seven epochs are used in each module. Thus the improvement of NDCG saturates at three to four models. No over fitting occurs due to careful regularization as described in Section 3.3.

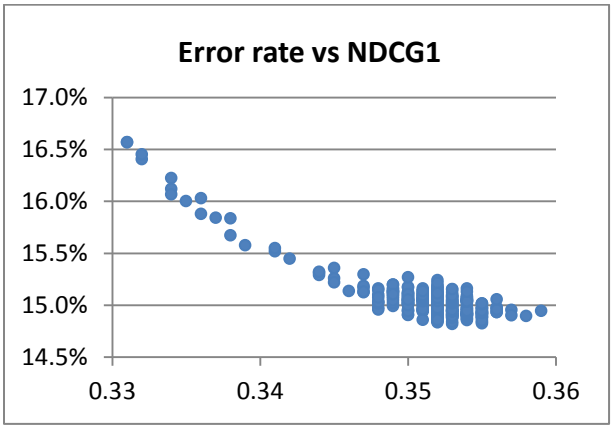

Fig. 2: Relationship between the classification error rates and the NDCG1 values on the test set.

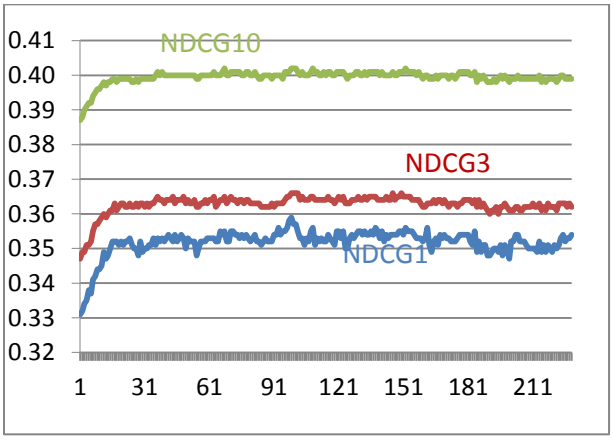

Fig. 3: Learning curves: NDCG values as a function of the training epochs cumulated over DSN modules.

6. CONCLUSION

We present in this paper the first study, to the best of our knowledge, on the use of deep learning techniques, the DSN architecture in particular, to the ad-related IR problem. We conclude from the experiments that the classification error rate, which is closely correlated with MSE as the DSN training objective, is generally correlated well with the NDCG as the IR quality measure, with the exception in the region of high IR quality. We also conclude that despite such exception the NDCG values obtained on the independent test set using MSE as the training criterion are significantly higher than the state-of-the-art baseline system.

The poorer correlation observed so far between the DSN training objective and the IR quality measure in the high IR-quality region suggests promise of further improvement of the DSN method. This would demand future research directed to the development of more suitable objective functions and new DSN learning methods. We also expect that with greater levels of IR targets than two as used in the current experiments, the effectiveness of the DSN will become stronger than reported in this paper since the stacking information from one module to another will become richer.

7. REFERENCES

[1] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G.Hullender. "Learning to rank using gradient descent." In Proc. ICML 2005.

[2] C. Burges, R. Rango, and Q. Le. "Learning to rank with non-smooth cost functions." In Proc. NIPS 2006.

[3] L. Deng and D. Yu. "Deep Convex Network: A scalable architecture for deep learning." In Proc. Interspeech 2011.

[4] L. Deng, D. Yu, and J. Platt. "Scalable stacking and learning for building deep architectures." In Proc. ICASSP 2012.

[5] J. F. Gao, W. Yuan, X. Li, K. Deng and J.Y. Nie. "Smoothing clickthrough data for web search ranking." In Proc. SIGIR 2009.

[6] J. F. Gao, X. He and J.Y. Nie. "Clickthrough-based translation models for web search: from word models to phrase models." In Proc. CIKM 2010.

[7] X. He, L. Deng, and W. Chou. "Discriminative learning in sequential pattern recognition." IEEE Signal Processing Magazine, September, 2008.

[8] X. He, L. Deng, and W. Chou. "A novel learning method for hidden Markov models in speech and audio processing." In Proc. IEEE Workshop on Multimedia Signal Processing, 2006.

[9] D. Hillard, E. Manavoglu, H. Raghavan, C. Leggetter, E. Cantú-Paz, and R. Iyer.. "The Sum of Its Parts: Reducing Sparsity in Click Estimation with Query

Segments." In *Journal of Information Retrieval*, V14(3) pp. 315-336, June, 2011.

[10] D. Hillard, S. Schroedl, E. Manavoglu, H. Raghavan and C. Leggetter. "Improving Ad relevance in sponsored search." In Proc. WSDM 2010.

[11] B. Jansen and M. Resnick. "Examining searcher perceptions of and interactions with sponsored results." In Proc. Workshop on Sponsored Search Auctions, 2005.

[12] K. Jarvelin and J. Kekalainen.. "IR evaluation methods for retrieving highly relevant documents." In Proc. SIGIR 2000.

[13] M. Richardson, E. Dominowska, and R. Ragno. "Predicting clicks: estimating the click-through rate for new ads." In Proc. WWW 2007.

[14] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. "Okapi at TREC-3." In Proceedings of the Third Text REtrieval Conference, Gaithersburg, USA, November, 1994.

[15] G. Salton; M. McGill. *Introduction to modern information retrieval*. McGraw-Hill. 1986.

[16] G. Tur, L. Deng, D. Hakkani-Tür, and X. He. "Towards deep understanding: Deep convex networks for semantic utterance classification." In Proc. ICASSP 2012.

[17] D. Wolpert. "Stacked generalization." Neural Networks, vol. 5(2), pp. 241-259, 1992.

[18] D. Yu, and L. Deng. "Accelerated parallelizable neural networks learning algorithms for speech recognition." In Proc. Interspeech 2011.

[19] Y. Zhang, L. Deng, X. He, and A. Acero. "A novel decision function and the associated decision-feedback learning for speech translation." In Proc. ICASSP 2011.