PROVABLE METHODS FOR TRAINING NEURAL NETWORKS WITH SPARSE CONNECTIVITY

Hanie Sedghi  
University of Southern California  
Los Angeles, CA 90089  
hsedghi@usc.edu

Anima Anandkumar  
University of California  
Irvine, CA 92697  
a.anandkumar@uci.edu

Abstract

We provide novel guaranteed approaches for training feedforward neural networks with sparse connectivity. We leverage on the techniques developed previously for learning linear networks and show that they can also be effectively adopted to learn non-linear networks. We operate on the moments involving label and the score function of the input, and show that their factorization provably yields the weight matrix of the first layer of a deep network under mild conditions. In practice, the output of our method can be employed as effective initializers for gradient descent.

Keywords: Deep feedforward networks, sparse connectivity, $\ell_1$-optimization, Stein’s lemma.

1 Introduction

The paradigm of deep learning has revolutionized our ability to perform challenging classification tasks in a variety of domains such as computer vision and speech. However, so far, a complete theoretical understanding of deep learning is lacking. Training deep-nets is a highly non-convex problem involving millions of variables, and an exponential number of fixed points. Viewed naively, proving any guarantees appears to be intractable. In this paper, on the contrary, we show that guaranteed learning of a subset of parameters is possible under mild conditions.

We propose a novel learning algorithm based on the method-of-moments. The notion of using moments for learning distributions dates back to Pearson (1894). This paradigm has seen a recent revival in machine learning and has been applied for unsupervised learning of a variety of latent variable models (see Anandkumar et al. (2014) for a survey). The basic idea is to develop efficient algorithms for factorizing moment matrices and tensors. When the underlying factors are sparse, $\ell_1$-based convex optimization techniques have been proposed before, and been employed for learning dictionaries (Spielman et al. (2012)), topic models, and linear latent Bayesian networks (Anandkumar et al. (2014)).

In this paper, we employ the $\ell_1$-based optimization method to learn deep-nets with sparse connectivity. However, so far, this method has theoretical guarantees only for linear models. We develop novel techniques to prove the correctness even for non-linear models. A key technique we use is the Stein’s lemma from statistics (Stein (1986)). Taken together, we show how to effectively leverage algorithms based on method-of-moments to train deep non-linear networks.

1.1 Summary of Results

We present a theoretical framework for analyzing when neural networks can be learnt efficiently. We demonstrate how the method-of-moments can yield useful information about the weights in a neural
network, and also in some cases, even recover them exactly. In practice, the output of our method can be used for dimensionality reduction for back propagation, resulting in reduced computation.

We show that in a feedforward neural network, the relevant moment matrix to consider is the cross-moment matrix between the label and the score function of the input data (i.e. the derivative of the log of the density function). The classical Stein’s result [Stein (1986)] states that this matrix yields the expected derivative of the label (as a function of the input). The Stein’s result is essentially obtained through integration by parts [Nourdin et al. (2013)].

By employing the Stein’s lemma, we show that the row span of the moment matrix between the label and the input score function corresponds to the span of the weight vectors in the first layer, under natural non-degeneracy conditions. Thus, the singular value decomposition of this moment matrix can be used as low rank approximation of the first layer weight matrix during back propagation, when the number of neurons is less than the input dimensionality. Note that since the first layer typically has the most number of parameters (if a convolutional structure is not assumed), having a low rank approximation results in significant improvement in performance and computational requirements.

We then show that we can exactly recover the weight matrix of the first layer from the moment matrix, when the weights are sparse. It has been argued that sparse connectivity is a natural constraint which can lead to improved performance in practice [Thom & Palm (2013)]. We show that the weights can be correctly recovered using an efficient \( \ell_1 \) optimization approach. Such approaches have been earlier employed for linear models such as dictionary learning [Spielman et al. (2012)] and topic modeling [Anandkumar et al. (2012)]. Here, we establish that the method is also successful in learning non-linear networks, by alluding to Stein’s lemma.

Thus, we show that the cross-moment matrix between the label and the score function of the input contains useful information for training neural networks. This result has an intriguing connection with [Alain & Bengio (2012)], where it is shown a denoising auto-encoder approximately learns the score function of the input. Our analysis here provides a theoretical explanation of why pre-training can lead to improved performance during back propagation: the interaction between the score function (learnt during pre-training) and the label during back propagation results in correctly identifying the span of the weight vectors, and thus, it leads to improved performance.

The use of score functions for improved classification performance is popular under the framework of Fisher kernels [Jaakkola et al. (1999)]. However, in [Jaakkola et al. (1999)], Fisher kernel is defined as the derivative with respect to some model parameter, while here we consider the derivation with respect to the input and refer to it as score function. Note that if the Fisher kernel is with respect to a location parameter, these two notions are equivalent. Here, we show that considering the moment between the label and the score function of the input can lead to guaranteed learning and improved classification.

Note that there are various efficient methods for computing the score function (in addition to the auto-encoder). For instance, [Sasaki et al. (2014)] point out that the score function can be estimated efficiently through non-parametric methods without the need to estimate the density function. In fact, the solution is closed form, and the hyper-parameters (such as the kernel bandwidth and the regularization parameter) can be tuned easily through cross validation. There are a number of score matching algorithms, where the goal is to find a good fit in terms of the score function, e.g [Hyvärinen (2005); Swersky et al. (2011)]. We can employ them to obtain accurate estimations of the score functions.

Since we employ a method-of-moments approach, we assume that the label is generated by a feedforward neural network, to which the input data is fed. In addition, we make mild non-degeneracy assumptions on the weights and the derivatives of the activation functions. Such assumptions make the learning problem tractable, whereas the general learning problem is NP-hard. We expect that the output of our moment-based approach can provide effective initializers for the back propagation procedure.

In a recent work we have extended Stein’s lemma by defining higher order score functions such that the cross moment between the label and higher order score function provides us with the expected value of higher order derivatives of the label with respect to the input [Janzamin et al. (2014)]. This allows us to use tensor methods on these higher order derivatives and obtain the discrimina-
tive features for various generative models such as mixture of classifiers, neural networks, sparse coding, etc. Detailed analysis of mixture of classifiers using the proposed framework can be found in (Sedghi & Anandkumar, 2014). This approach can help us to learn the weight matrix for different layers of a NN without the sparsity assumption.

1.2 RELATED WORK

In this paper, we show that the method-of-moments can yield low rank approximations for weights in the first layer. Empirically, low rank approximations of the weight matrices have been employed successfully to improve the performance and for reducing computations (Davis & Arel, 2013). Moreover, the notion of using moment matrices for dimension reduction is popular in statistics, and the dimension reducing subspace is termed as a central subspace (Cook, 1998).

We present a $\ell_1$ based convex optimization technique to learn the weights in the first layer, assuming they are sparse. Note that this is different from other convex approaches for learning feedforward neural network. For instance, (Bengio et al., 2005) show via a boosting approach that learning neural networks is a convex optimization problem as long as the number of hidden units can be selected by the algorithm. However, typically, the neural network architecture is fixed, and in that case, the optimization is non-convex.

Our work is the first to show guaranteed learning of a feedforward neural network incorporating both the label and the input. (Arora et al., 2013) considered the auto-encoder setting, where learning is unsupervised, and showed how the weights can be learnt correctly under a set of conditions. They assume that the hidden layer can be decoded correctly using a “Hebbian” style rule, and they all have only binary states. We present a different approach for learning by using the moments between the label and the score function.

2 MOMENTS OF A NEURAL NETWORK

2.1 FEEDFORWARD NETWORK WITH ONE HIDDEN LAYER

We first consider a feedforward network with one hidden layer. Subsequently, we discuss how much this can be extended. Let $y$ be the label vector generated from the neural network and $x$ be the feature vector. We assume $x$ has a well-behaved continuous probability distribution $f(x)$ such that the score function $\nabla_x \log f(x)$ exists. The network is depicted in Figure 1. Let

$$E[y|h] = \sigma_2(A_2h), \quad E[h|x] = \sigma_1(A_1x).$$

This setup is applicable to both multiclass and multilabel settings. For multiclass classification $\sigma_2$ is the softmax function and for multilabel classification $\sigma_2$ is a elementwise sigmoid function. Recall that multilabel classification refers to the case where each instance can have more than one (binary) label (Bishop et al., 2006; Tsoumakas & Katakis, 2007).

2.2 METHOD-OF-MOMENTS: LABEL-SCORE FUNCTION CORRELATION MATRIX

We hope to get information about the weight matrix using moments of the label and the input. The question is when this is possible and with what guarantees. To study the moments let us start from a simple problem. For a linear network and whitened Gaussian input $x \sim N(0, I)$, we have $y_{\text{linear}} = Ax$. In order to learn $A$, we can form the label-score function correlation matrix as

$$E[y_{\text{linear}}x^\top] = AE[xr^\top] = A.$$

Therefore, if $A$ is low dimensional, we can project $x$ into that span and perform classification in this lower dimension.

Stein’s lemma for a Gaussian random vector $x$ (Stein, 1986) states that for a function $g(\cdot)$ satisfying some mild regularity conditions we have

$$E[g(x)x^\top] = E_x[\nabla_x g(x)].$$

3
A more difficult problem is generalized linear model (GLM) of a (whitened) Gaussian $x \in \mathbb{R}^n$. In this case, $y = \sigma(Ax)$ for any nonlinear activation function $\sigma(\cdot)$ that satisfies some mild regularity conditions. Using Stein’s lemma we have
\[
\mathbb{E}[\sigma(Ax) | x^\top] = \mathbb{E}_{x'}[\nabla_{x'} \sigma(x')] A,
\]
where $x' \sim N(0, AA^\top)$. Therefore, assuming $\mathbb{E}_x[\nabla_x \sigma(Ax)]$ has full column rank, we obtain the row span of $A$. For Gaussian (and elliptical) random vector $x$, $P_{A^\top} x$ provides the sufficient statistic with no information loss. Thus, we can project the input into this span and obtain dimensionality reduction.

The Gaussian distribution assumption is a restrictive assumption. The more challenging problem is when random vector $x$ has a general probability distribution and the network has hidden layers. How can we deal with such an instance? Below we provide the method to learn such problems.

2.2.1 RESULTS

Let $x$ be a random vector with probability density function $f(x)$ and let $y$ be the output label corresponding to the network described in Equation (1). For a general probability distribution, we use score function of the random vector $x$ which provides us with sufficient statistics for $x$.

**Definition: Score function** The score of $x$ with probability density function $f(x)$ is the random vector $\nabla_x \log f(x)$.

Let
\[
M := \mathbb{E}[y (\nabla_x \log f(x))^\top],
\]
which can be calculated in a supervised setting. Note that $\nabla_x \log f(x)$ represents the score function for random vector $x$.

**Theorem 1.** In a nonlinear neural network with feature vector $x$ and output label $y$, we have
\[
M = -\mathbb{E}_x[\sigma'(\tilde{x}_2)A_2 \text{Diag}(\sigma'(\tilde{x}_1))]A_1,
\]
where $\tilde{x}_2 = A_2 \sigma_1(A_1 x)$ and $\tilde{x}_1 = A_1 x$.

**Proof.** Our method builds upon Stein’s lemma (Stein, 1986). We use Proposition 1.
The proof follows integration by parts; the result for the scalar integrals exists. Note that it is also assumed that the above expectations exist (in the sense that the corresponding boundaries of support of \(E\) and \(E_y\).)

**Proposition 1** (Stein’s lemma [Stein et al. (2004)]). Let \(x \in \mathbb{R}^n\) be a random vector with joint density function \(f(x)\). Suppose the score function \(\nabla_x \log f(x)\) exists. Consider any continuously differentiable function \(g(x) : \mathbb{R}^n \rightarrow \mathbb{R}^n\) such that all the entries of \(g(x)f(x)^\top\) go to zero on the boundaries of support of \(f(x)\). Then, we have

\[
E[g(x) (\nabla_x \log f(x))^\top] = -E[\nabla_x g(x)],
\]

Note that it is also assumed that the above expectations exist (in the sense that the corresponding integrals exist).

The proof follows integration by parts; the result for the scalar \(x\) and scalar-output functions \(g(x)\) is provided in [Stein et al. (2004)].

**Remark 1** (Connection with pre-training). The above theorem provides us with a nice closed-form. If \(B = E_x[\sigma_2'(\hat{x}_2) A_2 \text{Diag}(\sigma_1'(\hat{x}_1))\] has full column rank, we obtain the row space of \(A_1\). In deep networks auto-encoder is shown to approximately learn the score function of the input [Alain & Bengio (2012)]. It has been shown that pre-training results in better performance. Here, we are using the correlation matrix between labels and score function to obtain the span of weights. Auto-encoder appears to be doing the same by estimating the score function. Therefore, our method provides a theoretical explanation of why pre-training is helpful.

**Remark 2.** For whitened Gaussian (and elliptical) random vector, projecting the input onto row-space of \(M\) is a sufficient statistic. Empirically, even for non-Gaussian distribution, this has lead to improvements [Sun et al. 2013; Li 1992]. The moment method presented in this paper presents a low-rank approximation to train the neural networks.

So far, we showed that we can recover the span of \(A_1\). How can we retrieve the matrix \(A_1\)? Without further assumptions this problem is not identifiable. A reasonable assumption is that \(A_1\) is sparse. In this case, we can pose this problem as learning \(A_1\) given its row span. This problem arises in a number of settings such as learning a sparse dictionary or topic modeling. Next, using the idea presented in [Spielman et al. 2012], we discuss how this can be done.

### 3 Learning the Weight Matrix

In this Section, we explain how we learn the weight matrix \(A_1\) given the moment \(M\). Assuming sparsity we use [Spielman et al. 2012] method.

**Identifiability.** The first natural identifiability requirement on \(A_1\) is that it has full row rank. [Spielman et al. 2012] show that for Bernoulli-Gaussian entries under relative scaling of parameters, we can impose that the sparest vectors in the row-span of \(M\) are the rows of \(A_1\). Any vector in this space is generated by a linear combination \(w^\top M\) of rows of \(M\). The intuition is random sparsity, where a combination of different sparse rows cannot make a sparse row. Under this identifiability condition, we need to solve the optimization problem

\[
\text{minimize } \|w^\top M\|_0 \text{ subject to } w \neq 0.
\]

**\(\ell_1\) optimization** In order to come up with a tractable update, [Spielman et al. 2012] use the convex relaxation of \(\ell_0\) norm and relax the nonzero constraint on \(w\) by constraining it to lie in an affine hyperplane \(\{r^\top w = 1\}\). Therefore, the algorithm includes solving the following linear programming problem

\[
\text{minimize } \|w^\top M\|_1 \text{ subject to } r^\top w = 1.
\]
Algorithm 1 Exact recovery of sparsely-used dictionaries (Spielman et al., 2012).

\begin{algorithm}
\begin{algorithmic}
\FOR{each $j = 1, \ldots, n_x$}
\quad Solve $\min_w \|w^\top M\|_1$ subject to $(Me_j)^\top w = 1$, and set $s_j = w^\top M$.
\ENDFOR
\FOR{each $i = 1, \ldots, k$}
\quad \REPEAT
\quad \quad $l \leftarrow \arg\min_{s_l \in S} \|s_l\|_0$, breaking ties arbitrarily.
\quad \quad $v_i = s_l$.
\quad \quad $S = S \setminus \{s_l\}$.
\quad \UNTIL{Rank$(\langle v_1, \ldots, v_i \rangle) = i$}
\ENDFOR
\end{algorithmic}
\end{algorithm}

It is proved that under some additional conditions, when $r$ is chosen as a column or sum of two columns of $M$, the linear program is likely to produce rows of $A_1$ with high probability (Spielman et al., 2012). We explain these conditions in our context in Section 3.1.

By normalizing the rows of the output, we obtain a row-normalized version of $A_1$. The algorithm is shown in Algorithm 1.

We finally note that there exist more sophisticated analysis and algorithms for the problem of finding the sparsest vectors in a subspace. Anandkumar et al. (2012) provide the deterministic sparsity version of the result. Barak et al. (2012) require more computation and even quasi-polynomial time but they can solve the problem in denser settings.

### 3.1 Guarantees for Learning First Layer Weights

We have the following assumptions to ensure that the weight matrix $A_1 \in \mathbb{R}^{k \times n_x}$ is learnt correctly.

#### Assumptions

A.1 **Elementwise first layer**: $\sigma_1$ is a elementwise function.

A.2 **Nondegeneracy**: $\mathbb{E}_x [\sigma_2'(A_2\sigma_1(A_1 x)) A_2 \text{Diag}(\sigma_1'(A_1 x))]$ has full column rank\footnote{Throughout this Section, we use the notation $\sigma_1'(A_1 x)$ to denote $\sigma_1'(\tilde{x})|_{\tilde{x}=A_1x}$.}

A.3 **Score function**: The score function $\nabla_x \log f(x)$ exists.

A.4 **Sufficient input dimension**: We have $n_x > c_1 k^2 \log^2 k$ for some positive constant $c_1$.

A.5 **Sparse connectivity**: The weight matrix $A_1$ is Bernoulli($\theta$)-Gaussian. For some positive constant $\alpha$, we have $\frac{1}{\sqrt{n_x}} \leq \theta \leq \alpha \sqrt{\log k}$.

A.6 **Normalized weight matrix**: The weight matrix $A_1$ is row-normalized.

Assumption A.1 is common in deep network literature since there are only elementwise activation in the intermediate layers.

Assumption A.2 is satisfied where $A_2$ is full-rank and $\sigma_2'(A_2\sigma_1(A_1 x))$, $\text{Diag}(\sigma_1'(A_1 x))$ are non-degenerate. This is the case when the number of classes is large, i.e. $n_y \geq k$ as in imagenets. In future, we plan to consider the setting with a small number of classes using other methods like tensor methods. For non-degeneracy assumption of $\sigma_2'(\cdot)$, the reason is that we assume the functions are at least linear, i.e. their first order derivatives are nonzero. This is true for the activation function models in deep networks such as sigmoid function, piecewise linear rectifier and softmax function at the last layer.

In a deep network $k$ is usually a few thousand while $n_x$ is in the millions. Hence, Assumption A.4 is satisfied.

Assumption A.5 requires the weight matrix to be sparse and the expected number of nonzero elements in each column of $A_1$ be at most $O(\sqrt{k})$. In other words, each input is connected to at
most $O(\sqrt{k})$ neurons. This is a meaningful assumption in the deep-nets literature as it has been argued that sparse connectivity is a natural constraint which can lead to improved performance in practice (Thom & Palm, 2013).

If Assumption A.6 does not hold, we will have to learn the scaling and the bias through back propagation. Nevertheless, since the row-normalized $A_1$ provides the directions, the number of parameters in back propagation is reduced significantly. Therefore, instead of learning a dense matrix we will only need to find the scaling in a sparse matrix. This results in significant shrinkage in the number of parameters the back propagation needs to learn.

Finally we provide the results on learning the first layer weight matrix in a feedforward network with one hidden layer.

**Theorem 2.** Let Assumptions A.1 – A.5 hold for the nonlinear neural network (1), then Algorithm 1 uniquely recovers a row-normalized version of $A_1$ with exponentially small probability of failure.

For proof, see (Spielman et al., 2012).

**Remark 3** (Efficient implementation). The $\ell_1$ optimization is an efficient algorithm to implement. The algorithm involves solving $k$ optimization problems. Traditionally, the $\ell_1$ minimization can be formulated as a linear programming problem. In particular, each of these $\ell_1$ minimization problems can be written as a LP with $2(k-1)$ inequality constraints and one equality constraint. Since the computational complexity of such a method is often too high for large scale problems, one can use approximate methods such as gradient projection (Figueiredo et al., 2007; Kim et al., 2007), iterative-shrinkage thresholding (Daubechies et al., 2004) and proximal gradient (Nesterov, 1983; Nesterov et al., 2007) that are noticeably faster (Anandkumar et al., 2012).

**Remark 4** (Learning $A_2$). After learning $A_1$, we can encode the first layer as $h = \sigma_1(A_1 x)$ and perform softmax regression to learn $A_2$.

**Remark 5** (Extension to deterministic sparsity). The results in this work are proposed in the random setting where the i.i.d. Bernoulli-Gaussian entries for matrix $A_1$ are assumed. In general, the results can be presented in terms of deterministic conditions as in (Anandkumar et al., 2012). Anandkumar et al. (2012) show that the model $M = BA_1$ is identifiable when $B$ has full column rank and the following expansion condition holds (Anandkumar et al., 2012).

$$|N_B(S)| \geq |S| + d_{\max}(B), \quad \forall S \subseteq \text{Columns of } B, \quad |S| \geq 2.$$

Here, $N_B(S) := \{i \in [k] : B_{ij} \neq 0 \text{ for some } j \in S\}$ denotes the set of neighbors of columns of $B$ in set $S$. They also show that under additional conditions, the $\ell_1$ relaxation can recover the model parameters. See (Anandkumar et al., 2012) for the details.

### 3.2 Extension to Deep Networks

So far, we have considered a network with one hidden layer. Now, consider a deep $k$-node neural network with depth $d$. Let $y$ be the label vector and $x$ be the feature vector. We have

$$E[y|x] = \sigma_d(A_d \sigma_{d-1}(A_{d-1} \sigma_{d-2}(\cdots A_2 \sigma_1(A_1 x)))),$$

where $\sigma_1$ is elementwise function (linear or nonlinear). This set up is applicable to both multiclass and multilabel settings. For multiclass classification, $\sigma_d$ is the softmax function and for multilabel classification $\sigma_d$ is an elementwise sigmoid function. In this network, we can learn the first layer using the idea presented earlier in this Section to learn the first layer. From Stein’s lemma, we have

$$M = \text{Cov}[y, \nabla_x \log f(x)] = -E_x[\nabla_x y] = E[\sigma_d'(\tilde{x})A_d \sigma_{d-1}'(\tilde{x})A_{d-1} \sigma_{d-2}'(\tilde{x})A_{d-2} \cdots \sigma_2'(\tilde{x})A_2 \text{Diag}(\sigma_1'(\tilde{x}))]A_1.$$

**Assumption B.2 Nondegeneracy:**

The matrix $B = E[\sigma_d'(\tilde{x})A_d \sigma_{d-1}'(\tilde{x})A_{d-1} \sigma_{d-2}'(\tilde{x})A_{d-2} \cdots \sigma_2'(\tilde{x})A_2 \text{Diag}(\sigma_1'(\tilde{x}))]$ has full column rank.

In Assumption B.2, $\tilde{x}_i = A_i h_i$, $i \in [d]$ where $h_i$ denotes the input and the $i$-th layer.

**Theorem 3.** Let Assumptions A.1, B.2, A.3 – A.6 hold for the nonlinear deep neural network (2). Then, Algorithm 1 uniquely recovers a row-normalized version of $A_1$ with exponentially small probability of failure.
The proof follows Stein’s lemma, use of Chain rule and (Spielman et al., 2012).

In a deep network, the first layer includes most of the parameters (if a structure such as convolutional networks is not assumed) and other layers consist of a small number of parameters since there are small number of neurons. Therefore, the above result is a prominent progress in learning deep neural networks.

**Remark 6.** This is the first result to learn a subset of deep networks for general nonlinear case in supervised manner. The idea presented in (Arora et al., 2013) is for the auto-encoder setting, whereas we consider supervised setting. Also, Arora et al. (2013) assume that the hidden layer can be decoded correctly using a “Hebbian” style rule, and they all have only binary states. In addition, they can handle sparsity level up to \(k^\gamma, 0 < \gamma \leq 0.2\) while we can go up to \(\sqrt{k}\), i.e. \(\gamma = 0.5\).

**Remark 7** (Challenges in learning the higher layers). In order for \(B\) to have full column rank, intermediate layers should have square weight matrices. However, if we want to learn the middle layers, \(AA\) requires that the number of rows of the weight matrices be smaller than the number of columns in a specific manner and therefore \(B\) cannot have full column rank. In future, we hope to investigate new methods to help in overcoming this challenge.

4 CONCLUSION

We introduced a new paradigm for learning neural networks using method-of-moments. In the literature, this method has been restricted to unsupervised setting. Here, we bridged the gap and employed it for discriminative learning. This opens up a lot of interesting research directions for future investigation. First, note that we only considered the input to have continuous distribution for which the score function exists. The question is whether learning the parameters in a neural network is possible for the discrete data. Although Stein’s lemma has a form for discrete variables (in terms of finite differences) (Wei et al., 2010), it is not clear how that can be leveraged to learn the network parameters. Next, it is worth analyzing how we can go beyond \(\ell_1\) relaxation and provide guarantees in such cases. Another interesting problem arises in case of small number of classes. Note that for non-degeneracy condition, we require the number of classes to be bigger than the number of neurons in the hidden layers. Therefore, our method does not work for the cases where \(n_y < k\). In addition, in order to learn the weight matrices for intermediate layers, we need the number of rows to be smaller than the number of columns to have sufficient input dimension. On the other hand, non-degeneracy assumption requires these weight matrices to be square matrices. Hence, learning the weights in the intermediate layers of deep networks is a challenging problem. It seems tensor methods, which have been highly successful in learning a wide range of hidden models such as topic modeling, mixture of Gaussian and community detection problem (Anandkumar et al., 2014), may provide a way to overcome the last two challenges.

REFERENCES

Alain, Guillaume and Bengio, Yoshua. What regularized auto-encoders learn from the data generating distribution. arXiv preprint arXiv:1211.4246, 2012.

Anandkumar, A., Hsu, D., and Kakade, A. Javanmard S. M. Learning Topic Models and Latent Bayesian Networks Under Expansion Constraints. Preprint. ArXiv:1209.5350, Sept. 2012.

Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M., and Telgarsky, M. Tensor decompositions for learning latent variable models. J. of Machine Learning Research, 15:2773–2832, 2014.

Arora, Sanjeev, Bhaskara, Aditya, Ge, Rong, and Ma, Tengyu. Provable bounds for learning some deep representations. arXiv preprint arXiv:1310.6343, 2013.

Barak, Boaz, Brandao, Fernando GSL, Harrow, Aram W, Kelner, Jonathan, Steurer, David, and Zhou, Yuan. Hypercontractivity, sum-of-squares proofs, and their applications. In Proceedings of the forty-fourth annual ACM symposium on Theory of computing, pp. 307–326. ACM, 2012.

Bengio, Yoshua, Roux, Nicolas L, Vincent, Pascal, Delalleau, Olivier, and Marcotte, Patrice. Convex neural networks. In Advances in neural information processing systems, pp. 123–130, 2005.

Bishop, Christopher M et al. Pattern recognition and machine learning, volume 1. springer New York, 2006.
Cook, R Dennis. Principal hessian directions revisited. *Journal of the American Statistical Association*, 93(441):84–94, 1998.

Daubechies, Ingrid, Defrise, Michel, and De Mol, Christine. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on pure and applied mathematics*, 57(11):1413–1457, 2004.

Davis, Andrew and Arel, Itamar. Low-rank approximations for conditional feedforward computation. *arXiv preprint arXiv:1312.4461*, 2013.

Figueiredo, Mário AT, Nowak, Robert D, and Wright, Stephen J. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):586–597, 2007.

Hyvärinen, Aapo. Estimation of non-normalized statistical models by score matching. In *Journal of Machine Learning Research*, pp. 695–709, 2005.

Jaakkola, Tommi, Haussler, David, et al. Exploiting generative models in discriminative classifiers. In *Advances in neural information processing systems*, pp. 487–493, 1999.

Janzamin, Majid, Sedghi, Hanie, and Anandkumar, Anima. Score Function Features for Discriminative Learning: Matrix and Tensor Frameworks. *arXiv preprint arXiv:1412.2863*, Dec. 2014.

Kim, Seung-Jean, Koh, Kwangmoo, Lustig, Michael, Boyd, Stephen, and Gorinevsky, Dimitry. An interior-point method for large-scale 1 l-regularized least squares. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):606–617, 2007.

Landsman, Zinoviy and Nešlehová, Johanna. Stein’s lemma for elliptical random vectors. *Journal of Multivariate Analysis*, 99(5):912–927, 2008.

Li, Ker-Chau. On principal hessian directions for data visualization and dimension reduction: another application of stein’s lemma. *Journal of the American Statistical Association*, 87(420):1025–1039, 1992.

Nesterov, Yurii. A method of solving a convex programming problem with convergence rate o (1/k^2). In *Soviet Mathematics Doklady*, volume 27, pp. 372–376, 1983.

Sedghi, Hanie and Anandkumar, Anima. Provable Tensor Methods for Learning Mixtures of Classifiers. *arXiv preprint arXiv:1412.3046*, Dec. 2014.

Spielman, Daniel A, Wang, Huan, and Wright, John. Exact recovery of sparsely-used dictionaries. In *Conference on Learning Theory*, 2012.

Stein, Charles. Approximate computation of expectations. *Lecture Notes-Monograph Series, 7*i–164*, 1986.

Sun, Yuekai, Ioannidis, Stratis, and Montanari, Andrea. Learning mixtures of linear classifiers. *arXiv preprint arXiv:1311.2547*, 2013.