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 data, 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, ℓ_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 [20]. 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 [3] for a survey). The basic idea is to develop efficient algorithms for factorizing moment matrices and tensors. When the underlying factors are sparse, ℓ_1-based convex optimization techniques have been proposed before, and been employed for learning dictionaries [22], topic models, and linear latent Bayesian networks [2].

In this paper, we employ the ℓ_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 [23]. 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 low rank approximations during back propagation, resulting in reduced computation.

We show that in a feedforward neural network, the relevant moment matrix to consider is the covariance matrix between the label and the Fisher score of the input data (i.e. the derivative of the log of the density function). The classical Stein's result [23] 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 [19].

By employing the Stein's lemma, we show that the row span of the covariance matrix between the label and the input Fisher score corresponds to the span of the weight vectors in the first layer, under natural non-degeneracy conditions. Thus, the singular value decomposition of the covariance matrix can be used for low rank approximation of the first layer weight matrix during back propagation. 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 covariance 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 [25]. We show that the weights can be correctly recovered using an efficient ℓ_1 optimization approach. Such approaches have been earlier employed for linear models such as dictionary learning [22] and topic modeling [2]. 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 covariance matrix between the label and the Fisher score of the input contains useful information for training neural networks. This result has an intriguing connection with [1], 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 a low-rank approximation of the weights, and thus, it leads to improved performance.

The use of Fisher scores for improved classification performance is popular under the framework of Fisher kernels [12]. However, in [12], Fisher kernel is defined as the derivative with respect to some model parameter, while here we consider the derivation with respect to the input. 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 Fisher score 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. [21] 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.

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. In order to provide a guaranteed approach for learning, 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.

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 [10]. 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 [8].

We present a ℓ_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. [6] 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

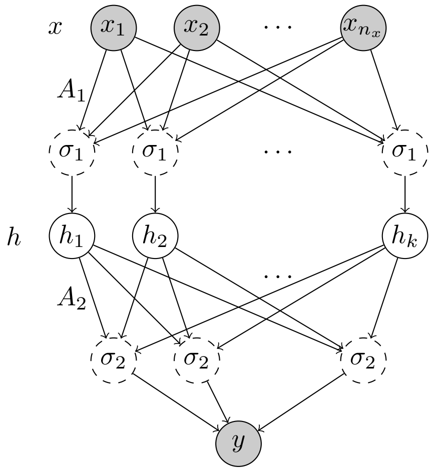

Figure 1: Graphical representation of Feedforward model $\mathbb{E}[h|x] = \sigma_1(A_1 x), \mathbb{E}[y|h] = \sigma_2(A_2 h)$.

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. [4] 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 Fisher score of the input.

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 Fisher score $\nabla_x \log f(x)$ exists. The network is depicted in Figure 1. Let

$$\mathbb{E}[y|h] = \sigma_2(A_2 h), \quad \mathbb{E}[h|x] = \sigma_1(A_1 x). \tag{1}$$

This setup is applicable to both multiclass and multilabel settings. For multiclass classification σ_2 is the softmax function and for multilabel classification σ_2 is a elementwise sigmoid function. Recall that multilabel classification refers to the case where each instance can have more than one (binary) label [7, 26].

2.2 Method-of-moments: label-input covariance 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 $x \sim \mathcal{N}(0, I)$, we have $y_{\text{linear}} = Ax$. In order to learn A, we can form the label-input covariance matrix as

$$\mathrm{Cov}(y_{\text{linear}}, x) = A\mathbb{E}[xx^\top] = A.$$

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 [14] states that

$$\mathrm{Cov}(g(x), x) = \mathbb{E}_x[\nabla_x g(x)].$$

A more difficult problem is generalized linear model (GLM) of a (whitened) Gaussian $x \in \mathbb{R}^{n_x}$. In this case, $y = \sigma(Ax)$ for any nonlinear activation function $\sigma(\cdot)$. Using Stein's lemma we have

$$\mathrm{Cov}(\sigma(Ax), x) = \mathbb{E}_x(\nabla_x y) = \mathbb{E}_{x'}[\nabla_{x'}\sigma(x'))]A,$$

where $x' \sim \mathcal{N}(0, AA^\top)$.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 centered 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 Fisher score of the random vector x which provides us with sufficient statistics for x.

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

For centered label y, 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 Fisher score for random vector x.

Theorem 1. *In a nonlinear neural network with feature vector x and output label y, we have*

$$M = -\mathbb{E}_x[\sigma_2'(\tilde{x}_2) A_2 \,\mathrm{Diag}(\sigma_1'(\tilde{x}_1))]A_1,$$

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 [23]. We use Proposition 1.

$$\begin{aligned} M &= \mathbb{E}_{x,y}[y\,(\nabla_x \log f(x))^\top] = \mathbb{E}_x\left[\mathbb{E}_y\left[y\,(\nabla_x \log f(x))^\top \,|x\right]\right] \\ &= \mathbb{E}_x[\sigma_2(A_2(\sigma_1(A_1 x)\,(\nabla_x \log f(x))^\top] \\ &= -\mathbb{E}_x[\sigma_2'(\tilde{x}_2)A_2\,\mathrm{Diag}(\sigma_1'(\tilde{x}_1))A_1] \end{aligned}$$

The second equality is a result of law of total expectation. The third equality follows from Stein's lemma as in Proposition 1 below. The last equality results from Chain rule. ∎

Proposition 1 (Stein's lemma [19])**.** *Let $x \in \mathbb{R}^{n_x}$ be centered and have density f. If x has a score function $\nabla_x \log f(x)$, for all continuously differentiable functions $g(x)$ we have*

$$\mathrm{Cov}[g(x) \otimes \nabla_x \log f(x)] = -\mathbb{E}[\nabla_x g(x)]. \tag{2}$$

The proof follows integration by parts. For details see [19].

Remark 1 (Connection with pre-training)**.** *The above Theorem provides us with a nice closed-form. If $B = \mathbb{E}_x[\sigma_2'(\tilde{x}_2)A_2\,\mathrm{Diag}(\sigma_1'(\tilde{x}_1))]$ 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 [1]. It has been shown that pre-training results in better performance. Here, we are using the covariance 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.*

Algorithm 1 Exact recovery of sparsely-used dictionaries [22].

```
for each $j = 1, \dots, n_x$ do
    Solve $\min_w \|w^\top M\|_1$ subject to $(Me_j)^\top w = 1$, and set $s_j = w^\top M$.
end for
for each $i = 1, \dots, k$ do
    repeat
        $l \leftarrow \arg\min_{s_l \in \mathcal{S}} \|s_l\|_0$, breaking ties arbitrarily.
        $v_i = s_l$.
        $\mathcal{S} = \mathcal{S} \setminus \{s_l\}$.
    until $\mathrm{Rank}([v_1, \dots, v_i]) = i$
end for
Set $\hat{A}_1 = [\frac{v_1}{\|v_1\|}, \dots, \frac{v_k}{\|v_k\|}]^\top$.
```

Remark 2 (Nonparametric learning of Fisher score)**.** *It should be noted that there exists efficient non-parametric ways to learn the Fisher score without estimating the probability density function (e.g. [16]).*

Remark 3. *For whitened Gaussian (and elliptical) random vector, projecting the input onto rowspace of M is a sufficient statistic. Empirically, even for non-Gaussian distribution, this has lead to improvements [24, 15]. 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 [22], 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. [22] method.

Identifiablity The first natural identifiability requirement on A_1 is that it has full row rank. Spielman et al. [22] show that for Bernoulli-Gaussian entries under relative scaling of parameters, we can impose that the sparsest 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

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

ℓ_1 optimization In order to come up with a tractable update, Spielman et al. [22] use the convex relaxation of ℓ_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

$$\text{minimize } \|w^\top M\|_1 \text{ subject to } r^\top w = 1.$$

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 [22]. 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. [2] provide the deterministic sparsity version of the result. Barak et al. [5] 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:** σ_1 is a elementwise function.

A.2 **Nondegeneracy:** $\mathbb{E}_x[\sigma_2'(A_2\sigma_1(A_1x))A_2 \operatorname{Diag}(\sigma_1'(A_1x))]$ has full column rank[1].

A.3 **Fisher score:** The Fisher score $\nabla_x \log f(x)$ 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(θ)-Gaussian. For some positive constant α, we have

$$\frac{2}{k} \leq \theta \leq \frac{\alpha}{\sqrt{k}\log k}.$$

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 $\sigma_2'(A_2\sigma_1(A_1x)), \operatorname{Diag}(\sigma_1'(Ax))$ are nondegenerate. 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 $\mathcal{O}(\sqrt{k})$. In other words, each input is connected to at most $\mathcal{O}(\sqrt{k})$ 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 [25].

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 $\hat{A}_1$ 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 [22].

Remark 4 (Efficient implementation)**.** *The ℓ_1 optimization is an efficient algorithm to implement. The algorithm involves solving k optimization problems. Traditionally, the ℓ_1 minimization can be formulated as a linear programming problem. In particular, each of these ℓ_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 [11, 13], iterative-shrinkage thresholding [9] and proximal gradient [17, 18] that are noticeably faster [2].*

[1]Throughout this Section, we use the notation $\sigma_1'(A_1x)$ to denote $\sigma_1'(\tilde{x})|_{\tilde{x}=Ax}$.

Remark 5 (Learning $\hat{A}_2$)**.** *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 6 (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 [2]. Anandkumar et al. [2] show that the model $M = BA_1$ is identifiable when B has full column rank and the following expansion condition holds [2].*

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

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 ℓ_1 relaxation can recover the model parameters. See [2] 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

$$\mathbb{E}[y|x] = \sigma_d(A_d \sigma_{d-1}(A_{d-1}\sigma_{d-2}(\cdots A_2\sigma_1(A_1 x)))), \tag{3}$$

where σ_1 is elementwise function (linear or nonlinear). This set up is applicable to both multiclass and mutlilabel settings. For multiclass classification, σ_d is the softmax function and for multilabel classification σ_d is a 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

$$\begin{aligned} M &= \operatorname{Cov}[y, \nabla_x \log f(x)] = -\mathbb{E}_x[\nabla_x y] \\ &= \mathbb{E}[\sigma_d'(\tilde{x}_d)A_d\sigma_{d-1}'(\tilde{x}_{d-1})A_{d-1}\sigma_{d-2}'(\tilde{x}_{d-2})A_{d-2}\cdots\sigma_2'(\tilde{x}_2)A_2 \operatorname{Diag}(\sigma_1'(\tilde{x}_1))]A_1. \end{aligned}$$

Assumption B.2 Nondegeneracy:

The matrix $B = \mathbb{E}[\sigma_d'(\tilde{x}_d)A_d\sigma_{d-1}'(\tilde{x}_{d-1})A_{d-1}\sigma_{d-2}'(\tilde{x}_{d-2})A_{d-2}\cdots\sigma_2'(\tilde{x}_2)A_2 \operatorname{Diag}(\sigma_1'(\tilde{x}_1))]$ 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* (3)*. 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 [22].

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 7. *This is the first result to learn a subset of deep networks for general nonlinear case in supervised manner. The idea presented in [4] is for the auto-encoder setting, whereas we consider supervised setting. Also, Arora et al. [4] 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 8 (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, $A.4$ 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) [27], it is not clear how that can be leveraged to learn the network parameters. Next, it is worth analyzing how we can go beyond ℓ_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 [3], may provide a way to overcome the last two challenges.

References

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

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

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

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

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

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

[7] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[8] R Dennis Cook. Principal hessian directions revisited. *Journal of the American Statistical Association*, 93(441):84–94, 1998.

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

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

[11] Mário AT Figueiredo, Robert D Nowak, and Stephen J Wright. 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.

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

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

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

[15] Ker-Chau Li. 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.

[16] Laurens Maaten. Learning discriminative fisher kernels. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 217–224, 2011.

[17] Yurii Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[18] Yurii Nesterov et al. Gradient methods for minimizing composite objective function, 2007.

[19] Ivan Nourdin, Giovanni Peccati, and Yvik Swan. Integration by parts and representation of information functionals. *arXiv preprint arXiv:1312.5276*, 2013.

[20] K. Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society, London, A.*, page 71, 1894.

[21] Hiroaki Sasaki, Aapo Hyvärinen, and Masashi Sugiyama. Clustering via mode seeking by direct estimation of the gradient of a log-density. *arXiv preprint arXiv:1404.5028*, 2014.

[22] Daniel A Spielman, Huan Wang, and John Wright. Exact recovery of sparsely-used dictionaries. In *Proc. of the Twenty-Third international joint conference on Artificial Intelligence*, pages 3087–3090, 2013.

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

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

[25] Markus Thom and Günther Palm. Sparse activity and sparse connectivity in supervised learning. *The Journal of Machine Learning Research*, 14(1):1091–1143, 2013.

[26] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13, 2007.

[27] Zhengyuan Wei, Xinsheng Zhang, and Taifu Li. On stein identity, chernoff inequality, and orthogonal polynomials. *Communications in StatisticsTheory and Methods*, 39(14):2573–2593, 2010.