Unsupervised Pre-Training Speeds up the Search for Good Features: An Analysis of a Simplified Model of Neural Network Learning

Avraham P Ruderman
ANU/NICTA
avraham.ruderman@nicta.com.au

Abstract

While unsupervised pre-training has been shown empirically to improve supervised learning of neural networks, it is still not well understood how it does this. We suggest that unsupervised pre-training helps supervised learning by speeding up the search for good features. We show that for highly structured input distributions, unsupervised pre-training can allow supervised learning algorithms to achieve higher accuracy by allowing them to search through a larger number of features in a given amount of time. The ability to search through more features can improve the accuracy of the learned predictor by allowing the learner to lower its bias and possibly also its variance. In particular, for a fixed k and for the set of features consisting of all k-conjunctions, we show theoretically and empirically that changing to a new representation based only on unsupervised data allows us to search through many more features in a given amount of time. We show that this in turn lowers the bias of the learner. However, we also show theoretically and empirically that this decrease in bias often comes at the cost of a large increase in variance. We are currently working to address this issue of increased variance.

1 Introduction

In recent years, neural networks have had great success [1]. In fields such as computer vision and speech recognition, neural network learning algorithms have achieved significant improvements over the previous state-of-the-art algorithms [2, 3].

There are a number of results explaining the good properties of neural networks in terms of their expressiveness [4] and sample complexity [5]. In contrast, an understanding of the optimization component of neural network learning is still lacking. Most of the formal results that are known about optimizing neural networks are negative, showing that there exist problems for which optimizing the parameters of neural networks is provably hard [6, 7]. These results are worst-case analyses and do not explain what has been observed empirically. While recent work has shown that for deep linear neural networks, pre-training provably speeds up learning [8], it is not clear how these results extend to networks with nonlinearities.

The only algorithm we are aware of for learning neural networks in polynomial time achieve this by bounding the fan-in of the units in the neural network [9]. However, the time complexity of the proposed algorithm is still exponential in the fan-in making it impractical for networks with all but the smallest fan-in and inputs of all but the smallest dimensions.

A great methodological breakthrough in the optimization of neural networks occurred in 2006 when it was shown that by using unsupervised pre-training, it is possible to learn deep neural networks with much better performance than shallow neural networks [10, 11].

1.1 Question: When and how does changing representation help supervised learning?

While unsupervised pre-training has allowed great improvements in results for many supervised learning tasks, there is still little formal, quantitative understanding as to why unsupervised pre-training gives such improvements. A quantitative understanding of how unsupervised pre-training helps supervised learning is important for at least two reasons:

- it would give an explicit objective for unsupervised pre-training that is directly tied to the supervised learning we ultimately care about;
- it would allow us to better characterize what problems unsupervised pre-training is helpful for and where we can expect big gains from its use.

The work of Bengio et al. [12] suggested that unsupervised pre-training helps the optimization algorithm find better local minima. As evidence for this they showed that for a neural network with a small top layer, backpropogation with pre-training was able to obtain lower training error than backpropogation on both shallow networks and randomly initialized deep networks. It has been argued though, that these results were a product of the early stopping procedure used and that in fact unsupervised pre-training acts as a regularizer by restricting the set of local optima in which backpropogation may end up [13].

However, further experiments investigating the role of unsupervised pre-training in an online learning setting on a large dataset show that the role of pre-training is not merely to prevent overfitting [13]. Instead, the experiments suggest that for non-convex problems, unsupervised pre-training helps back-propagation, which is a local optimization method, by starting it in a good location. We note that the optimization and regularization hypotheses need not be in conflict. In particular, if we consider the optimization of the regularized training loss, then better optimization can still lead to a solution with higher train error but lower test error.

In this work, we address the more general question: *When and how does a change of representation, based only on the input distribution, help supervised learning?* We note that in this work we are comparing shallow architectures without supervised pre-training to deep architectures with pre-training.

1.2 Hypothesis: For highly structured inputs, changing representation speeds up the search for good features

In this work we propose the following mechanism for a change of representation to improve supervised learning: for data with highly structured inputs, changing representation allows us to search through certain sets of features more efficiently. This in turn allows us to find better features given a fixed budget of compute time. Our results do not apply to all input distributions, only to distribution in which features can be partitioned into highly dependent groups, as is often the case for natural images and audio.

In particular, we show that for ensembles of k-conjunctions (subsection 2.3) on highly structured inputs, moving to a new representation that captures this structure (subsection 3.1) allows a learner to search through k-conjunctions more efficiently (subsection 3.3). This in turn allows the learner to find better features which lead to higher test time accuracy (subsection 3.6).

1.3 Structure of this paper

In section 2 we introduce the learning setting and k-conjunctions, the class of features we would like to consider. We also show that learning ensembles of such features can be computationally hard. In section 3 we state our main result, that a change in representation can lead to faster search over features for highly structured inputs. We also discuss the implications of this change of representation in decreasing bias but possibly increasing variance.

2 Learning setting and model

In this section we introduce the learning setting and our simplified model of neural network learning. We analyze the sample and computational complexities of learning in this simplified model. In

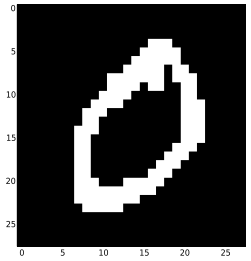

Figure 2.1: An example input from the binary version of MNIST

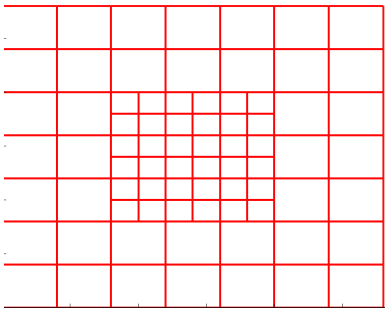

Figure 2.2: An illustration of the disjoint partition we used for the disjoint enumerator in our experiments

particular, we show that while the learning time is polynomial, it still leads to infeasible running times for all but the smallest problems. This motivates the next section where we show how learning can be sped up for highly structured input distributions.

2.1 Overview of our simplified model

Neural networks as they are typically used today are complex and often difficult to analyze. Our approach in this work is to start with a vastly simplified model of learning neural networks, one that can be more easily analyzed. Our hope is that once we have a good understanding of this simple model, we can gradually extend our analysis toward explaining the behavior of neural networks as they are used in practice.

As our simplified model, we consider the problem of learning ensembles of k-conjunctions. We list here the simplifications we make in our model.

- We are assuming the inputs are binary.
- We are assuming the encoder has zero reconstruction error and consists of sub-encodes on disjoint subsets of the features (see subsection 3.1).
- We are using conjunctions rather than thresholds as features, and assuming that these feature detectors have small fan-in (see section 2.3).
- We are assuming that only the top two layers are learned in a supervised fashion.

Our hope in presenting this work in the deep learning workshop is to get feedback as to whether our simplified model seems like a good starting point and to discuss approaches for gradually extending our results toward more powerful models and more general settings.

2.2 The learning setting

We will be dealing with the supervised learning setting in which we are given n input, output pairs

$$(x_1, y_1), \ldots, (x_n, y_n)$$

where $x_i \in \{0,1\}^d$ and $y_i \in \mathcal{Y}$.

We will use a binary version of the MNIST dataset as a running example to illustrate our point with concrete examples. To get a binary version of MNIST we simply threshold each pixel at a value of 127. An example digit from this dataset is shown in figure 2.1.

2.3 Ensembles of k-conjunctions

For $x \in \{0,1\}^d$, a k-conjunction is an expression of the form

$$(x_{j_1} = b_1 \wedge \ldots \wedge x_{j_k} = b_k)$$

where $1 \leq j_1, \ldots, j_k \leq d$ and $b_1, \ldots, b_k \in \{0, 1\}$. We generalize this definition to inputs from $\mathbb{N}$, the set of integers. In particular, we will also refer to an expression as above as a k-conjunction for any $x \in \mathbb{N}^d$ and $b_1, \ldots, b_k \in \mathbb{N}$. In this case, an example 3-conjunction would be $(x_1 = 42 \wedge x_2 = 3 \wedge x_5 = 11)$. We will consider conjunction as functions, returning 1 if the proposition is true and 0 otherwise and let

$$\mathcal{C}^k_{\mathbb{N}} := \{(x_{j_1} = b_1 \wedge \ldots \wedge x_{j_k} = b_k) : b_j \in \mathbb{N} \text{ for all } 1 \leq j \leq k\}$$

be the set of all such k-conjunctions. We will also discuss ensembles of such conjunctions. In particular, we define

$$\mathcal{F}^{k,B}_{\mathbb{N}} := \{\sum_{c \in \mathcal{C}^k_{\mathbb{N}}} w_c c : \|w\|_1 \leq B\}$$

to be the set of all linear combinations of functions from $\mathcal{C}^k_{\mathbb{N}}$ with weights having ℓ_1 norm at most B.

2.4 Computational complexity of learning k-conjunction ensembles

In this subsection we introduce a naive algorithm that learns ensembles of k-conjunctions in time

$$\mathcal{O}((2d)^k)$$

and argue that given current algorithms, it is unlikely that this upper bound on running time can be significantly improved upon without distributional assumptions. Our main result (section 3) is to show that if we add some distributional assumptions about the inputs, and if these distributional properties are appropriately captured by a new representation, then this running time can be improved upon by changing to this new representation.

The naive algorithm for learning k-conjunction ensembles simply enumerates all possible conjunctions of k inputs and then learns a linear predictor using the outputs of these conjunctions as features. Computing all k conjunctions has complexity $\mathcal{O}((2d)^k)$ since there are $\binom{d}{k} \leq \mathcal{O}(d^k)$ subsets of features of size k and for each subset there are 2^k conjunctions.

We note that in the case of the easier learning problem of learning k-juntas [14], which can be viewed as a subset of k-conjunction ensembles, the best known algorithms [15] still have a bound on their time complexity of the form $d^{\mathcal{O}(k)}$. Thus it is unlikely that a bound on running time of

$$d^{\mathcal{O}(k)}$$

can be improved upon for this problem without distributional assumptions, given the state of current algorithms.

2.5 Sample complexity of learning k-conjunction ensembles

In this subsection we analyze the Rademacher complexity (see [16] or [17]) of $\mathcal{F}^{k,B}_{\{0,1\}}$ to show that as long as the number of samples we have grows faster than

$$\mathcal{O}\left(B\sqrt{k \log d}\right),$$

then for any 1-Lipschitz loss (e.g., logistic or hinge), the difference between the training loss and test loss of any predictor from $\mathcal{F}^{k,B}_{\{0,1\}}$ goes to 0 with high probability.

First, we note that on k boolean variables, there are 2^k possible conjunctions. Further, there are $\binom{d}{k} \leq \mathcal{O}(d^k)$ possible subsets of d input variables which are of size k. Thus, the number of functions in $\mathcal{C}^k_{\{0,1\}}$ can be bounded by $|\mathcal{C}^k_{\{0,1\}}| \leq \mathcal{O}((2d)^k)$. By Massart's lemma [18], this implies that we can bound the Rademacher complexity $R_m(\mathcal{C}^k_{\{0,1\}})$ of $\mathcal{C}^k_{\{0,1\}}$ by

$$R_m(\mathcal{C}^k_{\{0,1\}}) \leq \mathcal{O}\left(\frac{\sqrt{k \log d}}{m}\right).$$

Since Rademacher complexity is invariant to linear combination of ℓ^1 norm at most 1 and since rademacher complexity scales with multiplication by scalars, we have

$$R_m(\mathcal{F}^{k,B}_{\{0,1\}}) \leq \mathcal{O}\left(B\frac{\sqrt{k \log d}}{m}\right).$$

Finally, composition with the 1-Lipschitz loss function does not increase the Rademacher complexity and so the bound follows.

3 Re-representing highly structured inputs to speed up the search for good features

In the last section we saw that given current algorithms and without distributional assumptions, the time complexity of learning ensembles of k-conjunctions is $\mathcal{O}(d^k)$ which is prohibitive for all but the smallest k and d. In this section we show that if the inputs are partitioned into groups and each group is encoded by a lossless code, then the class of k-conjunction ensembles on the encoded inputs covers all k-conjunction ensembles on the original inputs (though possibly requiring weights of larger norm to express the same function). If the number of groups is small and the code for each group is small, then learning ensembles of k-conjunctions in this new representation is more efficient than in the original representation.

As a concrete illustration, consider the change of representation that one gets by partitioning an image into non-overlapping patches, learning a code for each patch (e.g., using k-means) and then mapping each patch to an integer depending on the element of the code it is assigned to. Suppose that we wish to learn an ensemble of 3-conjunctions. Our claim is that if the encoding of image patches gives perfect reconstruction of the inputs then any function that can be expressed as an ensemble of 3-conjunctions over pixels, can also be expressed as an ensemble 3-conjunction over image patches. Further, if the number of patches is small, and the code for each patch is small then searching through all 3-conjunctions of image patches is much faster than searching through all 3-conjunctions of pixels.

3.1 Conditions on new representation

We will assume we are given a map $C : \{0,1\}^d \to \mathbb{N}^J$ to a new representation. We will denote by $s_1, \ldots, s_J$ the subsets of inputs that the component functions $C_1, \ldots, C_J$ of C depend on. We impose the following conditions on C

1. The inputs to the component functions of C form a disjoint partition of the inputs. That is, $\cup_j s_j = \{1, \ldots, d\}$ and $s_j \cap s_{j'} = \emptyset$ if $j \neq j'$.
2. The function C is one to one, i.e. viewed as a code, C is lossless.

We will refer to such a map as a *disjoint enumerator*. We acknowledge that the requirement on C to have 0 reconstruction error is very stringent. We are currently working on relaxing it to C having low reconstruction error.

We will denote by $|C_j|$ the size of the code of the j-th component function of C and by N the size of the largest such code

$$N := \max_j |C_j|.$$

3.2 k-conjunction ensembles under new representation include all k-conjunction ensembles on original inputs

In this subsection we show that any function that can be expressed as an ensemble of k-conjunctions on the inputs X can also be expressed as an ensemble of k-conjunctions on $C(X)$, the inputs encoded by a disjoint enumerator. We also quantify the amount by which the ℓ_1 norm of the ensemble weights must grow in the worst case in order to express the same function.

Our analysis rests on the following observations:

Lemma 1. *Suppose C is a disjoint enumerator, and c is a k-conjunction on $\{0,1\}^d$ then we can write*

$$c = \vee_{j=1}^{B} \tilde{c}_j \circ C = \sum_{j=1}^{B} \tilde{c}_j \circ C$$

for some $0 \leq B \leq N^k$ and $\tilde{c}_1, \ldots, \tilde{c}_B \in \mathcal{C}_{\mathbb{N}}^k$.

Proof. Suppose a conjunction c depends on inputs $x_{l_1}, \ldots, x_{l_k}$. There are at most k subsets $s_{\lambda_1}, \ldots, s_{\lambda_k}$ of $\{1, \ldots, d\}$ which $l_1, \ldots, l_k$ belong to. Now, the number of possible conjunctions on these subsets is $|s_{\lambda_1}| \cdot \ldots \cdot |s_{\lambda_k}| \leq N^k$. Now, enumerate all conjunctions $\tilde{c}_j$ on $C_{\lambda_1}(x), \ldots, C_{\lambda_k}(x)$ for which $x_{l_1}, \ldots, x_{l_k}$ have the appropriate values (this can be done since the code is one-to-one), their disjunction is equal to c. Note that there are at most N^k of these conjunctions. We now have a disjunctions of conjunctions $\vee_{j=1}^{B} \tilde{c}_j$. Since all the conjunctions in this disjunction are disjoint, the disjunction is equal to their sum. □

As a corollary we get the following result:

Proposition 2. *Suppose C is a disjoint enumerator, then for some $0 \leq B \leq N^k$, the set of functions $\mathcal{F}_{\mathbb{N}}^{B,k} \circ C$ covers $\mathcal{F}_{\{0,1\}}^{1,k}$. That is, for every $f \in \mathcal{F}_{\{0,1\}}^{1,k}$, there exists a $g \in \mathcal{F}_{\mathbb{N}}^{B,k}$ such that with probability 1 $f(X) = g(C(X))$.*

Proof. By lemma 1 □

If J, the number of codes, and N, the size of the largest code, are small, then optimizing over k-conjunction ensembles in the new representation can be much faster than optimizing over k-conjunction ensembles in the original representation. Proposition 2 guarantees that the training error of the optimum we find will be as good as the optimum on the original inputs (as long as we allow for weights of larger norm).

In the next few subsection we look at what we gain and what we lose by moving to the new representation. Subsection 3.3 looks at the reduction in time complexity. Subsection 3.4 looks at the resulting reduction in bias. Subsection 3.5 looks at the price we pay through an increase in variance. Finally, subsection 3.6 looks at the overall change in predictive performance by changing representation.

3.3 New representation reduces time complexity of learning

Under the new representation, the time complexity of learning ensembles of k-conjunctions can be bounded by

$$\mathcal{O}((NJ)^k).$$

We saw in subsection 2.4 that using a naive algorithm, the time complexity of learning ensembles of k-conjunctions has time complexity of $\mathcal{O}((2d)^k)$. We also argued that this cannot be significantly improved upon without distributional assumptions given current algorithms. In particular, this suggests that if $NJ < 2d$ then the time complexity of learning ensembles of k-conjunctions has reduced by changing representation.

We now justify the above bound on the time complexity as well as give a more refined analysis of the change in computational complexity when learning ensembles of k-conjunctions under the new representation as opposed to under the original representation. To this end we introduce the following:

Definition 1. *For a random variable X, we define*

$$\#\operatorname{supp}(X) := |\{x : P(x) > 0\}|.$$

Further, if X has d coordinates, s is a subset of $\{1, \ldots, d\}$ and X^s is the projection of X onto the coordinates specified by s, then we define

$$\#\operatorname{supp}_k(X) := \sum_{s \subset \{1,\ldots,d\}, |s| \leq k} |\operatorname{supp}(X^s)|$$

*and we will call this the k-*support of X.

Now, suppose we want to learn an ensemble of k-conjunctions, then it is sufficient to enumerate only k-conjunctions that are supported by the distribution. In particular, for a given dataset, it is sufficient to enumerate only k-conjunctions that appear in the dataset. Thus, for a given $s \subset \{1, \ldots, d\}$ of size k, we only need to enumerate all values for X^s that appear in the dataset. Thus the time complexity of listing all k-conjunctions for a given dataset and a given s is bounded by $\mathcal{O}(\#\operatorname{supp}(X^s))$ and so the complexity of enumerating all k-conjunctions on $C(X)$ is bounded by $\mathcal{O}(\#\operatorname{supp}_k(C(X)))$. It

follows that the time complexity of learning $\mathcal{F}_{\mathbb{N}}^{B,k} \circ C$ is smaller than that of the naive algorithm for learning $\mathcal{F}_{\{0,1\}}^{B,k}$ if

$$\#\operatorname{supp}_k(C(X)) \leq \#\operatorname{supp}_k(X).$$

To get the bound stated at the start of this subsection, note that

$$\#\operatorname{supp}_k(C(X)) \leq \binom{J}{k} N^k \leq (JN)^k.$$

We empirically study the k-support under the original and new representation in subsection 3.6.

3.4 New representation reduces bias

If we change representation using a disjoint enumerator then the bias of k-conjunction ensemble learners is reduced in two ways. First, for each group of inputs, all conjunctions are considered. Second, and often more significantly, the reduced time complexity of enumerating k-conjunction means one can learn conjunctions of a higher degree.

We study empirically this reduction in bias in subsection 3.6. It is important to note that in Table 2 for $k = 2$, the train accuracy for the encoded representation is lower than that on the original inputs. This suggests that the coding scheme we are using is quite inaccurate. In particular, if it were completely accurate, then we could represent the same functions with ensembles of 2-conjunctions on the new representation as we could on the original representation and therefore achieve at least as good a train accuracy. Indeed, it is possible that the accuracy of k-conjunction ensembles in the encoded representation may be significantly increased if we can reduce this reconstruction error.

3.5 New representation may increase variance

The reduction in bias described in the previous section does come at a cost of increased variance. In particular, we can give the following bound on the Rademacher complexity of learning under the new representation

$$R_m(\mathcal{F}_{\mathbb{N}}^{B,k}) \leq \mathcal{O}\left(\frac{BN^k\sqrt{k\log(NJ)}}{m}\right)$$

In particular, comparing this bound to the bound for k-conjunctions on the original representation suggests that we need N^k as many samples as we did under the original representation. Note however that we are just comparing upper bounds and this comparison need not represent the actual effects of a change of representation. In particular, while we do observe empirically an increase in variance in Table 2, the increase is not as extreme as predicted by the comparison of the upper bounds. We are currently working on methods to lower this variance.

3.6 Experiments

For our experiments we used a disjoint partition of the MNIST images as illustrated in Figure 2.2. All dictionaries were learned using the k-means algorithm. For the outer ring of 4×4 patches, we learned dictionaries of size 3, for the inner ring of 4×4 patches we learned dictionaries of size 10 and for the small 2×2 patches in the center we learned dictionaries of size 8. These choices are unlikely to be optimal for reconstruction or prediction but they nonetheless illustrate some of our points.

Empirical estimates of the k-support in the original and encoded representation for the binary MNIST dataset are compared in Table 1. Only conjunctions appearing more than 10 times in the dataset were counted.

We now present train and test accuracies of k-conjunctions under the original and coded representation. We note that these experiments are very preliminary and we have much work to do to improve these. The main point of these experiments was not to try and achieve state-of-the-art performance but rather to illustrate that the changes suggested by the theory in fact occur in practice. In particular, we want to look at the changes in bias and variance when moving from the original representation to the representation based on encoded patches.

Table 1: Estimates of k-support on original and re-represented Binary MNIST dataset

k	original representation	encoded representation
1	1.4×10^3	5.1×10^2
2	9.1×10^5	1.1×10^5
3	3.9×10^8	1.1×10^7

For our experiments we learned linear predictors under the logistic loss using stochastic gradient descent on features corresponding to k-conjunctions. We found that ℓ^1-norm and ℓ^2-norm regularization of the weights had little or no effect and so experiments reported here are of training with no regularization (except through early stopping). Train and test accuracies are presented in Table 2.

Table 2: Train and test accuracies of k-conjunction ensembles learned on the original and coded representations of the binary MNIST dataset. For 3-conjunctions we randomly selected 5000 triplets of features and enumerated all conjunctions on them

features	original representation			encoded representation		
	train	test	train - test	train	test	train - test
1-conj	0.903	0.905	-0.002	0.937	0.934	0.003
2-conj	0.994	0.979	0.015	0.992	0.970	0.022
2-conj + 3-conj [5000]	-	-	-	0.996	0.972	0.024

4 Conclusion

In this work, we have put forward a hypothesis as to how a change of representation based on unsupervised learning helps supervised learning. Our goal here was not to propose new algorithms but rather to try and understand a highly effective learning approach of which we have little understanding. We have shown that if the representation resulting from the unsupervised learning has certain structure, then one can learn k-conjunctions in the new representation faster than on the original representation. This in turn allows the learner to find better features more quickly.

However, as an explanation for the success of unsupervised pre-training, our framework is still lacking in a number of ways. First, our model makes many simplifying assumptions. Second, we learn a restricted set of neural networks. Finally, the empirical performance obtained by changing representation does not yet show an overall improvement in test accuracy like that shown by unsupervised pre-training. Thus, as future work, we intend to expand our analysis to more general settings and neural networks that are closer to those used in practice. In particular, we intend to extend our framework to

- lossy codes on real valued inputs;
- overlapping codes;
- polynomials of degree k, ensembles of k-fan-in thresholds and ensembles of trees of depth k.

Our hope is that as we work with models that are closer to neural networks used in practice they will have performance that is comparable to that of neural networks but at the same time remain amenable to analysis.

Finally, in this work, we have not discussed how the new representation is obtained, but merely assumed that it is given to us. Our work suggests that a good representation is one with non-overlapping codes of small size. In future work we intend to investigate algorithms for learning such representations.

Acknowledgements

I would like to thank the anonymous reviewers for their valuable feedback.

References

[1] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[3] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE, 2013.

[4] Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *Information Theory, IEEE Transactions on*, 39(3):930–945, 1993.

[5] Peter L Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *Information Theory, IEEE Transactions on*, 44(2):525–536, 1998.

[6] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.

[7] Avrim L Blum and Ronald L Rivest. Training a 3-node neural network is np-complete. *Neural Networks*, 5(1):117–127, 1992.

[8] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[9] Wee Sun Lee, Peter L Bartlett, and Robert C Williamson. Efficient agnostic learning of neural networks with bounded fan-in. *Information Theory, IEEE Transactions on*, 42(6):2118–2132, 1996.

[10] Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[11] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[12] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[13] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

[14] Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1):245–271, 1997.

[15] Elchanan Mossel, Ryan O'Donnell, and Rocco P Servedio. Learning juntas. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 206–212. ACM, 2003.

[16] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *The Journal of Machine Learning Research*, 3:463–482, 2003.

[17] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

[18] Pascal Massart. Some applications of concentration inequalities to statistics. In *Annales de la Faculté des Sciences de Toulouse*, volume 9, pages 245–303. Université Paul Sabatier, 2000.