Hardness and Impossibility Results in Learning Theory

Daniel Khashabi

May 13, 2017

Abstract

In this write-up I am going over relevant impossibility and hardness results related to learning theory. I go over the related literature and explain all the necessary background, hopefully, with an easy-to-understand language. Later I go over some results from the past as well as a recent result.

1 Introduction

While study of impossibility results started right after the first successful attempts to modelling “learning" [KV89], aren’t popularly studied the community, and over time a big gap is shaped between what’s done in practice and what is understood by theory.

In this write up I go over some of the existing results and tools as well as some open problems.

2 Preliminaries

We give a brief description of the relevant terminology, parameters and scenarios.

2.1 Learning Theory: Basics

I introduce basic terminology and notation for learning theory. For more more exact definitions we refer the reader to the important surveys in the literature [MRT12, SSBD14].

Basic notation. Define the input space to be $X = \mathbb{R}^d$ and the output space to be $Y = \{\pm 1\}$. Suppose we are given n training instances $S = \{\boldsymbol{x}_i, y_i\}_{i=1}^{n}$, are drawn from an unknown distribution $\mathcal{D}$, with marginal distributions $D_{\mathcal{X}}$ and $D_{\mathcal{Y}}$. Define the hypothesis class $\mathcal{H}$ to be the space of all functions $h_{\boldsymbol{w}}$ that we can use to approximate our problem. Define the expected error (or risk) to be $\text{Err}_{\mathcal{D}}(h) \triangleq \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[h_{\boldsymbol{w}}(\boldsymbol{x}) \neq y\right] = \text{P}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left(h_{\boldsymbol{w}}(\boldsymbol{x}) \neq y\right)$. Define the best hypothesis to be $h^* \triangleq \arg\min_{h\in\mathcal{H}} \text{Err}_{\mathcal{D}}(h)$ and its error $\text{OPT} = \text{Err}_{\mathcal{D}}(h^*)$. The empirical error (risk) is defined according to the training data $\widehat{\text{Err}}_{\mathcal{D}}(h) \triangleq \frac{1}{|S|}\sum_{(\boldsymbol{x},y)\sim S} h_{\boldsymbol{w}}(\boldsymbol{x}) \neq y$. In reality we choose the best hypothesis via training data S and the empirical risk: $\hat{h} \triangleq \arg\min_{h\in\mathcal{H}} \widehat{\text{Err}}_{\mathcal{D}}(h)$. Thus is commonly referred to as *empirical risk minimization* (ERM).

Realizable scenario. When output label does not have any noise and OPT is zero (i.e. there exists a plane which perfectly separates the data in the hypothesis space $\mathcal{H}$). The opposite of the realizable scenario is the *agnostic* scenario for which no assumptions are made. In the literature realizable and agnostic case are also referred to as *noise-free* and *noisy* scenarios, respectively as well.

Proper vs improper learning. If an algorithm is strictly required to output an output from the set $\mathcal{H}$ the algorithm is called *proper learning*. In some scenarios $\mathcal{H}$ is subset of another set $\mathcal{H}'$. If the algorithm is allowed to output hypothesis outside $\mathcal{H}$ inside $\mathcal{H}' \backslash \mathcal{H}$, as long as it satisfies some certain guarantees (possibly in expectation). This is often referred to as *representation-independent learning* or *improper learning*[*]. In the context of half-spaces, an improper algorithm might sometimes output a classifier that is not a half-space classifier.

The learning problem. Find h such that $\text{Err}_{\mathcal{D}}(h) \leq \text{Err}_{\mathcal{D}}(h^{*}) + \epsilon$, for some $\epsilon \in [0, 1]$. ϵ is commonly called the *excess* error. This is sometimes referred to as the *exact* learning since the algorithms gets arbitrarily close to the exact objective (in additive sense). In other words, $\text{Err}_{\mathcal{D}}(\mathcal{H}) = \min_{h \in \mathcal{H}} \text{Err}_{D}(h) = 0$.

The approximate learning problem. Find h such that $\text{Err}_{\mathcal{D}}(h) \leq \mu \text{Err}_{\mathcal{D}}(h^{*}) + \epsilon$, for some $\epsilon \in [0, 1]$ and $\mu > 1$. μ is commonly called the *approximation* ratio, which is the relaxed version version of the exact learning (i.e. $\mu = 1$)[†].

For many of the algorithms we introduce here, their performance measures will be a function of the approximation factor μ, and naturally as we decrease μ close to one, the running time should get worse. Hence we one can think of these results as interpolation between approximate learning and exact learning (when setting μ to one).

Another way of converting an approximate problem is to set $\mu = 1 + \epsilon'$:

$$\mu \leq \mu \text{OPT} + \epsilon_{old} = (1 + \epsilon')\text{OPT} + \epsilon = \text{OPT} + \epsilon_{new}, \quad \text{where } \epsilon_{new} = (1 + \text{OPT})\epsilon$$

Hence ϵ can be set small enough so that ϵ_{new} is close to what use desires.

Efficient algorithm/learner. An algorithm is efficient if it runs in poly $\left(n, \frac{1}{\epsilon}, \log \frac{1}{\delta}, d\right)$, where ϵ is the excess error, d is the input dimension and the guarantee holds with probability at least $1 - \delta$.

Agnostic learning. We say a learning algorithm *agnostically* learns $\mathcal{H}$ if for any distribution $\mathcal{D}$, with probability $\geq 1 - \delta$, it outputs a hypothesis with error less than $\text{Err}_{D}(\mathcal{H}) + \epsilon$.

Random classification noise. Define *noise level* to be $s(\boldsymbol{x}) = \min\left(\text{P}\left(Y = 1 | X = \boldsymbol{x}\right), 1 - \text{P}\left(Y = 1 | X = \boldsymbol{x}\right)\right)$ which can be simplified to $s(\boldsymbol{x}) = \frac{1 - \eta(\boldsymbol{x})}{2}$. Intuitively the smaller the noise level across all the instance is, it is easier to do classification. One way looking at this is that, each label of a noise-free sample is flipped independently with some fixed probability. In this model the noise is independent of the actual example points, which are generated according to a probability distribution. When $s(\boldsymbol{x}) = 0$ almost surely in the deterministic case. Clearly when $s > 0.5$ there is no hope of learning; hence it is almost assume that $s < 0.5$.

In the exact setting ($\mu = 1$) we say an algorithm is learnable in presence of random classification noise, if for any $\epsilon, \delta > 0$ and s it produces an output a classifier with error ϵ with probability at least $1 - \delta$, in time poly $\left(\frac{1}{\epsilon}, \log \frac{1}{\delta}, d, \frac{1}{1 - 2s}\right)$. Note that in this criterion, as $s \to 0.5$, $\frac{1}{1 - 2s}$ gets bigger.

This model first introduced by Angluin and Laird[AL88], and it is sometimes referred to as *benign (nonadversarial)* noise.

[*] Although there is nothing “improper" about it!

[†] Statistical estimation problems are mostly evaluated based on additive errors while combinatorial problems approximation problems are evaluated based on ratio of objectives. The one we study here has both factors, but we use the *approximation* term when referring to the multiplicative approximation, following the computer science community.

2.2 Learning problems

Learning half-spaces. If the hypothesis class $\mathcal{H}$ consists of half-spaces and each member of this space $h_{\boldsymbol{w}} \in \mathcal{H}$ is defined as $h_{\boldsymbol{w}}(\boldsymbol{x}) = \text{sign}\left(\langle \boldsymbol{x}, \boldsymbol{w} \rangle\right)$, or simply defined a weighted linear sum of some variables:

$$\text{sign}\left(w_1 x_1 + \ldots + w_n x_n - \theta\right)$$

Half-spaces are the cornerstone of the statistical machine learning; vast space of machine learning models are extensions of half-spaces. For example Naive Bayes, Logistic Regression and Conditional Random Fields[LMP01] (normalized exponentiated linear separators), Neural Networks (stack of linear/nonlinear separators), Support Vector Machines[CV95] (separators with preference for a big margin). Even Boolean functions can implicitly be simulated with linear functions: AND of variables ($\theta = n - \frac{1}{2}$), Or of variables ($\theta = \frac{1}{2}$), and majority-vote ($\theta = \frac{n}{2}$).

Learning intersection of half-spaces. Given the definition of a half-space $h_{\boldsymbol{w}}$, the class of "intersection of half-spaces" is defined as

$$\tilde{h}_{\boldsymbol{w}}(\boldsymbol{x}) = \bigwedge_{i=1}^{k} \text{sign}\left(w_1^i x_1 + \ldots + w_n^i x_n - \theta^i\right)$$

2.3 Assumptions

Often to show hardness of computational problems, assumptions made on hardness of existing problems. Here we provide a short list of what has been used.

P$\neq$ NP. Example works that use this assumption are [ABX08].

RP$\neq$ NP. An example usage of this assumption is for proving the hardness of proper learning.

Refuting random K-SAT is hard. The existing algorithms can only refute random instances with $\Omega\left(n^{K/2}\right)$ for $K \geq 4$ [Fei02]. A version of this assumption in [Dan15] is that, refuting K-SATwith $n^f(K)$ constraints is hard for some $f(K) = \omega(1)$.

Cryptographic assumptions. Many of the existing results are based on cryptographic assumptions. Goldreich *et al.* showed that if one-way functions (functions easy to compute but hard to invert) exist, then learning polynomial-sized circuits is hard [GGM84]. Kearns and Variant [KV89] showed other hardness results for learning automata, constant-depth threshold circuit, $\log$-depth circuit, by making cryptographic assumptions (e.g. breaking RSA).

Hardness of other algebraic problems. Kharitonov proved hardness of learning constant-depth circuits, by assuming hardness of factoring random Blum integers [Kha93]. Klivans and Sherstov showed hardness of learning intersection of polynomially many half-spaces is hard, assuming hardness of short-Vector Problem [KS06]. Feldman *et al.* showed that learning intersection of polynomially-many half-spaces is hard [FGKP06].

3 Known results

In what follows we discuss the existing results for learning half-spaces under certain assumptions. We start off by results of exact learning (i.e. $\mu = 1$) and which will be followed by results of approximate learning (i.e. $\mu > 1$).

No known algorithms can learn DNF **formulas (depth 2 circuits).** However, we can only rule out learning of circuits of depth d, for some unspecified constant d [Kha93].

No known algorithm can learn intersections of 2 half-spaces. In a stronger claim, Klivans and Sherstov [KS06] only rule out learning intersections of polynomially many half-spaces.

Learning in realizable case is *usually* easy; learning half-spaces in realizable case is always easy. In the absence of (label/output) noise (when the data is realizable, i.e. $\mathrm{OPT} = 0$) half-spaces are efficiently learnable.

One simple way of showing it is via Linear Programming. We show that the problem of half-spaces in the realizable case can be modelled as an LP. Given the training data S, we are looking for a realizable answer we need a w such that $y_i \boldsymbol{w}.\boldsymbol{x}_i > 0, \forall i = 1, \dots, m$. Define $\delta = \min_{i \in [m]} y_i \boldsymbol{w}.\boldsymbol{x}_i$ (because we are in the realizable scenario it exists). The objective can equivalently written as $y_i \boldsymbol{w}.\boldsymbol{x}_i \geq \delta, \forall i \in [m]$. To keep the notation cleaner we rewrite this equation as $y_i \frac{\boldsymbol{w}}{\delta}.\boldsymbol{x}_i \geq 1, \forall i \in [m]$. Note that for any $\boldsymbol{w}$, $\frac{\boldsymbol{w}}{\delta}$ is another solution in the solution space. Hence the objective can be just written as $y_i \boldsymbol{w}.\boldsymbol{x}_i \geq 1, \forall i \in [m]$. Given the definition of the LP the problem is easily solvable. Note that the LP has only inequality constraints (i.e. it is a feasibility LP).

There are other algorithms (with guarantees) for the realizable scenario; for example the Perceptron algorithm of Rosenblatt [Ros58] is generated to find find a correct classifier after at most $(RB)^2$ updates/iterations, where $B = \min \{ \|\boldsymbol{w}\| : \forall i \in [m], y_i \boldsymbol{w}.\boldsymbol{x}_i \geq 1 \}$ and $R = \max_i \|\boldsymbol{x}_i\|$. Note how the final bound depends on the realizablity in definition of B.

Another popular algorithm is Winnow, which unlike perceptron uses multiplicative updates. Given that the hypothesis space consists of disjunctions (or conjunctions) and assuming that the target function can be expressed with r coefficients (queries), Winnow algorithms with at most $O(r \log n)$ mistakes finds the target function [Lit88].

Learning can sometimes be hard, even in the realizable case. We argued that for half-spaces learning in the realizable case is easy. For completeness we mention that learning some concept classes (beyond half-spaces), even in the realizable case can be hard. A famous example is 3-term DNF's: Pitt and Valiant [PV88] showed that unless (randomized poly) **RP= NP**, *properly* learning 3-term DNFclass is not efficiently PAC-learnable [PV88]. There are corresponding upper-bounds in prior work by Valiant [Val84] where he shows that there is an efficient algorithm for learning any DNFwith $\leq q$ for some constant q. A common way of proving the DNFhardness result is by reduction to the 3-coloring problem [KV94].

In the 3-coloring problem, we check it is possible to color the graph with 3 colors, such that any two neighboring vertices have different colors. In the 3-term DNF, given a collection of positive and negative instance sets, S^+ and S^- respectively, we look for a 3-DNFthat would satisfy them all.

Given a 3-coloring instance, we generate binary sequences as inputs to 3-term DNF. Each term in the 3-DNF should correspond to a single color class. We generate binary vectors with length equal to number of vertices. We will encode each vertex as positive instances and each edge as negative instances. Each vertex v_i generates a binary vector of ones, with the i-th element set to 0. Each edge (v_i, v_j) generates a binary vector one 1's, with the i-th, j-th elements set to zero. Note that the number of the samples generated is polynomial in the size of input.

If there is a consistent 3-coloring for the input graph, then there is a 3-DNF consistent with our instances generated according to the input graph. Suppose the colors are red, blue and yellow. We construct a Boolean function $T_R \vee T_Y \vee T_B$, where each Boolean function $T_.$ is constructed as it follows: Let T_R be the conjunction of all the literals x_i for which the vertex v_i is not red. Among the positive instances S^+ the ones that satisfy this are the ones corresponding to vertex i. Hence any instance will be satisfied by one of the three Boolean formulas T_i. Among the negative instances none satisfy this, hence none will be satisfied by the Boolean function. Therefore all the instances $S^+ \cup S^-$ are compatible with the formula $T_R \vee T_Y \vee T_B$.

Conversely if there is a 3-DNF consistent with our instances generated according to the input graph, then there is a consistent 3-coloring for the input graph: label the clauses with colors. Then color vertex v_i to the color of the clause that is satisfied by the corresponding example in S^+. If there are multiple colors, pick a valid one arbitrarily. It is easy to show that there can't be an edge with the same color on two ends.

This proves that learning 3-DNFis at least as hard as 3-coloring.

Here a simplified argument for the hardness of general proper learning:

Given a hypothesis class $\mathcal{H}$, let $\Pi(\mathcal{H})$ be the problem of distinguishing between $\mathcal{H}$-realizable sample S and the one with $\text{Err}_S(\mathcal{H}) \geq 1/4$. Will prove that if $\mathcal{H}$ is efficiently *properly* learnable, the problem $\Pi(\mathcal{H})$ is in **RP**: In order to solve $\Pi(\mathcal{H})$ we can invoke the proper learning algorithm, on a set of uniformly sampled instances S. Let h be the output of this algorithm: (a) If S is a realizable, the $\text{Err}_S(h)$ is small. (b) If $\text{Err}_S(\mathcal{H}) \geq 1/4$ then $\text{Err}_S(h) \geq 1/4$ (since $h \in \mathcal{H}$ and setting is *proper*). This is an efficient way to decide whether $\mathcal{H}$ is realizable or not. With this in hand, we conclude that if $\Pi(\mathcal{H})$ is **NP**-hard, then $\mathcal{H}$ is not efficiently learnable, unless **NP= RP**.

Another interesting point about 3-term DNF's is that, if we remove the *proper* learnability condition (i.e. we allow the learning algorithm to sometimes output from a super-set of 3-term DNF's), it is efficiently learnable. Specifically one can efficiently learn 3-term DNF's using 3-CNF's, i.e. each DNFcan be efficiently converted to a CNF, and each CNFis easy to learn. This is a good example to show the importance of representation in hardness of learning

Proper learning of half-spaces in agnostic scenario is hard. For many functions ERM in the agnostic case is **NP**-hard; i.e. unless P = NP, there is no polynomial time approximation scheme for finding a member in the class that approximately minimizes the empirical risk on a given training sample. Ben-David *et* *al.* [BDEL03] proves this result for class of monomials (disjunctions), axis-aligned hyper-rectangles, closed balls and monotone monomials.

When restricted to proper algorithms, learning half-spaces is equivalent to *minimizing disagreement* (aka co-agnostic learning) which is a well-studied problem and it's included in Karp's celebrated work [Kar72] as an **NP**-hard problem.

Half-spaces are not efficiently properly agnostically PAC learnable. This has proved in different ways based on several commonly considered hard problems; Kalai *et al.* [KKMS08] based on hardness of learning parity functions, Feldman *et al.* [Fel06] based on hardness of shortest vector problem, Daniely and Shalev-Shwartz [DSS14] using hardness of refuting random k-SAT, Klivans and Kothari [KK14] via hardness of learning sparse parity functions (under uniform distribution), Daniely [Dan15] using hardness of refuting random k-XOR. Feldman[Fel06] has shown that for any constant $\epsilon > 0$ determining whether the best disjunction for a given $\epsilon > 0$ has error $\leq \epsilon$ or error $> \frac{1}{2} - \epsilon$ is **NP**-hard. Guruswami and Raghavendra [GR09] showed that for any $\epsilon \in (0, 1/2]$, it is **NP**-hard to find a halfspace with error bounded by $1/2 - \epsilon$.

Improper learning of half-space in the agnostic scenario is *probably* hard. Klivans and Kothari [KK14] show that any algorithm for improperly agnostically learning half-spaces requires $n^{\Omega(\log(1/\epsilon))}$ time under the assumption that k-sparse parities under uniform distribution requires $n^{\Omega(k)}$ time.

Approximate learning of half-spaces is efficiently learnable with distributional assumptions. One can simplify the problem by making distributional assumption for the generative process of data. One common assumption is uniform assumption; under this assumption Kalai *et al.* [KKMS08] presented an algorithm with $\mu = O\left(\sqrt{\log \frac{1}{\text{OPT}}}\right)$ approximation ratio. Later Awasthi *et al.* [ABL14] improved the approximation ratio to $\mu = O(1)$ under the same assumption.

4 Hardness of Improper Learning

There is a good understanding for hardness of learning in the *proper* scenario. The ideas discussed by Daniely *et al.* [DLS14] is to analyze the hardness of learning for the *improper* scenario. The idea discussed for a wide range of hypothesis spaces and the techniques are relatively general (with stronger assumptions). [‡] However we go over proving hardness results for special case of CNF's with smaller assumptions. Specifically the result of Daniely *et al.* [DSS14] where it's proved that learning DNF's are hard, with the following assumption:

Assumption 4.1. *Refuting random K-SAT formulas with $n^{\omega(1)}$ constraints is hard.*

Define $\mathsf{DNF}^{q}(n)$ be a class of DNF's with less than $q(n)$ clauses. Daniely *et al.* show that:

Theorem 4.2. *Learning $\mathsf{DNF}^{\omega(\log n)}$ is hard.*

Proof sketch. The proof uses Assumption 4.1. One other ingredient of the proof comes from [DLS14]:

[‡] A recent by Allen *et al.* [AOW15] "falsified the SRCSP assumptions" used in Danielly *et al.* [DLS14].

Theorem 4.3 ([DLS14]). *IF for any $a > 0$, it is hard to distinguish $\mathcal{H}$-realizable from scattered samples of size n^a, then learning $\mathcal{H}$ is hard.*

where *scattered* instances are defined as the set of input/output pairs with outputs assigned via independent fair coins.

Informally speaking, a classifier with short description can return a limited number of hypothesis; if the instances are scattered, all the hypothesis are likely to be perform poorly.

Define a predicate $P : \{\pm 1\}^K \to \{0, 1\}$. A P-constraint is defined as $C(x) = P(j_1 x_{i_1}, ..., j_K x_{i_K})$, with $j_i \in \{\pm 1\}$. A CSP instance is defined as collection of $P-$constraints $J = \{C_1, ..., C_m\}$ with input assignments $x \in \{\pm\}^n$. The *value* of a CSP instance J is denoted with $\mathsf{VAL}\,(J)$, the maximal fraction of constraints that can be simultaneously satisfied. A CSP instance is *satisfiable* if its value is 1.

Define $\mathsf{CSP}^{\text{rand}}_{m(n)}\,(P)$ to be the problem of distinguishing satisfiable from random P-formulas, for some function $m : \mathbb{N} \to \mathbb{N}$. The proof starts off by reducing $\mathsf{CSP}^{\text{rand}}_{n^d}\,(\mathsf{SAT}_K)$ to the problem of distinguishing realizable samples from scattered samples, and uses Theorem 4.3 to conclude the result.

5 Open Questions

Before finishing this survey I summarize a set of important open questions.

***Improper* learning half-spaces is *definitely* NP-hard.** There are some evidences for this based upon some cryptographic and average case complexity assumptions, but no strong standalone proof yet.

Lower-bounds for interesting problems: Decision Trees + intersection of two half-spaces. For these problems we have almost no satisfying lower-bound.

Exponential lower-bound for DNF's. The best known algorithms for DNF's are exponential while the known lower-bounds are polynomial.

References

[ABL14] Pranjal Awasthi, Maria Florina Balcan, and Philip M. Long. The power of localization for efficiently learning linear separators with noise. In *Proc. 36th ACM symposium on Theory of computing* (STOC), STOC '14, pages 449–458, New York, NY, USA, 2014. ACM.

[ABX08] Benny Applebaum, Boaz Barak, and David Xiao. On basing lower-bounds for learning on worst-case assumptions. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 211–220. IEEE, 2008.

[AL88] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[AOW15] Sarah R Allen, Ryan ODonnell, and David Witmer. How to refute a random csp. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 689–708. IEEE, 2015.

[BDEL03] Shai Ben-David, Nadav Eiron, and Philip M Long. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66(3):496–514, 2003.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[Dan15] Amit Daniely. Complexity theoretic limitations on learning halfspaces. *Proc. 38th ACM symposium on Theory of computing* (STOC), 2015.

[DLS14] Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. From average case complexity to improper learning complexity. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 441–448, 2014.

[DSS14] A. Daniely and S. Shalev-Shwatz. Complexity theoretic limitations on learning dnf's. *arXiv preprint arXiv:1404.3378*, 2014.

[Fei02] Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 534–543. ACM, 2002.

[Fel06] Vitaly Feldman. Optimal hardness results for maximizing agreements with monomials. In *Computational Complexity, 2006. CCC 2006. Twenty-First Annual IEEE Conference on*, pages 9–pp. IEEE, 2006.

[FGKP06] Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. New results for learning noisy parities and halfspaces. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 563–574. IEEE, 2006.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct randolli functions. In *Foundations of Computer Science, 1984. 25th Annual Symposium on*, pages 464–479. IEEE, 1984.

[GR09] Venkatesan Guruswami and Prasad Raghavendra. Hardness of learning halfspaces with noise. *SIAM Journal on Computing* (SICOMP), 39(2):742–765, 2009.

[Kar72] Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[Kha93] Michael Kharitonov. Cryptographic hardness of distribution-specific learning. In *STOC*, 1993.

[KK14] Adam Klivans and Pravesh Kothari. Embedding Hard Learning Problems Into Gaussian Space. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*, volume 28 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 793–809, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[KKMS08] Adam Tauman Kalai, Adam R Klivans, Yishay Mansour, and Rocco A Servedio. Agnostically learning halfspaces. *SIAM Journal on Computing* (SICOMP), 37(6):1777–1805, 2008.

[KS06] Adam R. Klivans and Alexander A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 553–562, 2006.

[KV89] Michael Kearns and Leslie G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. In *J. ACM*, 1989.

[KV94] Michael J Kearns and Umesh Virkumar Vazirani. *An introduction to computational learning theory*. MIT press, 1994.

[Lit88] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning*, 2(4):285–318, 1988.

[LMP01] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[MRT12] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

[PV88] Leonard Pitt and Leslie G Valiant. Computational limitations on learning from examples. *Journal of the ACM (JACM)*, 35(4):965–984, 1988.

[Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[Val84] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.