Minimizing the Maximal Loss: How and Why?

Shai Shalev-Shwartz* Yonatan Wexler†

Abstract

A commonly used learning rule is to approximately minimize the *average* loss over the training set. Other learning algorithms, such as AdaBoost and hard-SVM, aim at minimizing the *maximal* loss over the training set. The average loss is more popular, particularly in deep learning, due to three main reasons. First, it can be conveniently minimized using online algorithms, that process few examples at each iteration. Second, it is often argued that there is no sense to minimize the loss on the training set too much, as it will not be reflected in the generalization loss. Last, the maximal loss is not robust to outliers. In this paper we describe and analyze an algorithm that can convert any online algorithm to a minimizer of the maximal loss. We prove that in some situations better accuracy on the training set is crucial to obtain good performance on unseen examples. Last, we propose robust versions of the approach that can handle outliers.

1 Introduction

In a typical supervised learning scenario, we have training examples, $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$, and our goal is to learn a function $h : \mathcal{X} \to \mathcal{Y}$. We focus on the case in which h is parameterized by a vector $w \in \mathcal{W} \subset \mathbb{R}^d$, and we use h_w to denote the function induced by w. The performance of w on an example (x, y) is assessed using a loss function, $\ell : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$. A commonly used learning rule is to approximately minimize the average loss, namely,

$$\min_{w \in \mathcal{W}} L_{\text{avg}}(w) := \frac{1}{m} \sum_{i=1}^{m} \ell(w, x_i, y_i) \; . \tag{1}$$

Another option is to approximately minimize the maximal loss, namely,

$$\min_{w \in \mathcal{W}} \; L_{\max}(w) := \max_{i \in [m]} \ell(w, x_i, y_i) \; . \tag{2}$$

Obviously, if there exists $w^* \in \mathcal{W}$ such that $\ell(w^*, x_i, y_i) = 0$ for every i then the minimizers of both problems coincide. However, approximate solutions can be very different. In particular, since $L_{\max}(w) \geq L_{\text{avg}}(w)$ for every w, the guarantee $L_{\max}(w) < \epsilon$ is stronger than the guarantee $L_{\text{avg}}(w) < \epsilon$. Furthermore, for binary classification with the hinge-loss, any vector for which $L_{\max}(w) < 1$ must predict all the labels on the training set correctly, while the guarantee $L_{\text{avg}}(w) < 1$ is meaningless.

Some classical machine learning algorithms can be viewed as approximately minimizing $L_{\max}$. For example, many boosting algorithms can be viewed as coordinate descent algorithms for solving $L_{\max}$ with

*School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel.
†Orcam Ltd., Jerusalem, Israel

respect to the loss $\ell(w, x_i, y_i) = -y_i \langle w, x_i \rangle$ (here $\langle w, x_i \rangle$ is the inner product of w and x_i) and over the set $\mathcal{W}$ being the probabilistic simplex (see for example [18] and the references therein). Hard-SVM can also be viewed as solving $L_{\max}$ with respect to the loss function $\ell(w, x_i, y_i) = \lambda \|w\|^2 + \max\{0, 1 - y_i \langle w, x_i \rangle\}$.

However, minimizing L_{avg} is a more popular approach, especially for deep learning problems, in which w is the vector of weights of a neural network and the optimization is performed using variants of stochastic gradient descent (SGD). There are several reasons to prefer L_{avg} over $L_{\max}$:

1. If m is very large, it is not practical to perform operations on the entire training set. Instead, we prefer iterative algorithms that update w based on few examples at each iteration. This can be easily done for L_{avg} by observing that if we sample i uniformly at random from $[m]$, then the gradient of $\ell(w, x_i, y_i)$ with respect to w is an unbiased estimator of the gradeint of $L_{\text{avg}}(w)$. This property, which lies at the heart of the SGD algorithm, does not hold for $L_{\max}$.

2. Our ultimate goal is not to minimize the loss on the training set but instead to have a small loss on unseen examples. As argued before, approximately minimizing $L_{\max}$ can lead to a smaller loss on the training set, but it is not clear if this added accuracy will also be reflected in performance on unseen examples. Formal arguments of this nature were given in [6, 19].

3. The objective $L_{\max}$ is not robust to outliers. It is easy to see that even a single outlier can make the minimizer of $L_{\max}$ meaningless.

In this paper we tackle the aforementioned disadvantages of $L_{\max}$, and by doing so, we show cases in which $L_{\max}$ is preferable. In particular:

1. We describe and analyze a meta algorithm that can take any online learner for w and convert it to a minimizer of $L_{\max}$. A detailed description of our meta algorithm, its analysis, and a comparison to other approaches, are given in Section 2.

2. The arguments in [6, 19] rely on a comparison of upper bounds. We show that these upper bounds are not tight in many cases. Furthermore, we analyze the sample complexity of learning in situations where the training examples are divided to "typical" scenarios and "rare" scenarios. We argue that in many practical cases, our goal is to have a high accuracy on both typical and rare examples. We show conditions under which minimizing even few rare examples suffice to guarantee good performance on unseen examples from the rare scenario. In other words, few examples can have a dramatic effect on the performance of the learnt classifier on unseen examples. This is described and analyzed in Section 3.

3. Finally, in Section 4 we review standard techniques for generalizing the results from realizable cases to scenarios in which there might be outliers in the data.

To summarize, we argue that in some situations minimizing $L_{\max}$ is better than minimizing L_{avg}. We address the "how" question in Section 2, the "why" question in Section 3, and the issue of robustness in Section 4. Proofs are provided in the appendix.

2 How

In this section we describe and analyze an algorithmic framework for approximately solving the optimization problem given in (2).

Denote by $\mathcal{S}_m = \{p \in [0,1]^m : \|p\|_1 = 1\}$ the probabilistic simplex over m items. We also denote by $\Lambda : \mathcal{W} \to \mathbb{R}^m$ the function defined by

$$\Lambda(w) = (\ell(w, x_1, y_1), \ldots, \ell(w, x_m, y_m)) \ .$$

The first step is to note that the optimization problem given in (2) is equivalent to

$$\min_{w \in \mathcal{W}} \max_{p \in \mathcal{S}_m} \langle p, \Lambda(w) \rangle \ . \tag{3}$$

This is true because for every w, the p that maximizes the inner optimization is the all zeros vector except 1 in the coordinate for which $\ell(w, x_i, y_i)$ is maximal.

We can now think of (3) as a zero-sum game between two-players. The p player tries to maximize $\langle p, \Lambda(w) \rangle$ while the w player tries to minimize $\langle p, \Lambda(w) \rangle$. The optimization process is comprised of T game rounds. At round t, the p player defines $p_t \in \mathcal{S}_m$ and the w player defines $w_t \in \mathcal{W}$. We then sample $i_t \sim p_t$ and define the value of the round to be $\ell(w_t, x_{i_t}, y_{i_t})$.

To derive a concrete algorithm we need to specify how player p picks p_t and how player w picks w_t. For the w player one can use any online learning algorithm. We specify the requirement from the algorithm below.

Definition 1 (Low regret w player)**.** *Let $\ell^s : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ be a surrogate loss, namely, for every i and w we have that $\ell^s(w, x_i, y_i) \geq \ell(w, x_i, y_i)$. We say that the w player enjoys a regret of $R_w(T)$ w.r.t. ℓ^s if for every sequence of indices $i_1, \ldots, i_T$ and for every $w^\star \in \mathcal{W}$ we have that*

$$\frac{1}{T} \sum_{t=1}^{T} \ell(w_t, x_{i_t}, y_{i_t}) \ \leq \ \frac{1}{T} \sum_{t=1}^{T} \ell^s(w^\star, x_{i_t}, y_{i_t}) + R_w(T) \ . \tag{4}$$

For the p player, we use the seminal work of [2]. In particular, recall that the goal of the p player is to maximize the loss. Since the loss is in $[0,1]$, it is equivalent to minimizing 1 minus the loss, which is also in $[0,1]$. The basic idea of the construction is therefore to think of the m examples as m slot machines, where at round t the loss of pulling the arms of the different machines is according to $z_t := 1 - \Lambda(w_t) \in [0,c]^m$. Crucially, the work of [2] do not assume that z_t are sampled from a fixed distribution, but rather the vectors z_t can be chosen by an adversary. As observed in Auer et al. [2, Section 9], this naturally fits zero-sum games, as we consider here.

In [2] it is proposed to rely on the algorithm EXP3.P.1 as the strategy for the p-player. The acronym EXP3 stands for **Exp**loration-**Exp**loitation-**Exp**onent, because the algorithm balances between exploration and exploitation and rely on an exponentiated gradient framework. The "P" in EXP3.P.1 stands for a regret bound that holds with high probability. This is essential for our analysis because we will later apply a union bound over the m examples. While the EXP3.P.1 algorithm gives the desired regret analysis, the runtime per iteration of this algorithm scales with m. Here, we propose another variant of EXP3 for which the runtime per iteration is $O(\log(m))$.

To describe our strategy for the p player, recall that it maintains $p_t \in \mathcal{S}_m$. We will instead maintain another vector, $q_t \in \mathcal{S}_m$, and will set p_t to be the vector such that $p_{t,i} = \frac{1}{2} q_{t,i} + \frac{1}{2m}$. That is, p_t is a half-half mix of q_t with the uniform distribution. While in general such a strong mix with the uniform distribution can hurt the regret, in our case it only affects the convergence rate by a constant factor. On the up side, this strong exploration helps us having an update step that takes $O(\log(m))$ per iteration.

A pseudo-code of the resulting algorithm is given in Section 2.3. Observe that we use a tree structure to ensure that the update step of the p player takes $O(\log(m))$ time per iteration. The following theorem summarizes the convergence of the resulting algorithm.

Theorem 1. *Let $(x_1, y_1), \ldots, (x_m, y_m)$ be training examples which are realizable with respect to some surrogate loss, and assume that we have an online learner with regret of $R_w(T)$ with respect to the same surrogate. Suppose we run the online boosting algorithm with T, k such that $R_w(T) \leq \epsilon/8$, $T = \Omega(m \log(m/\delta)/\epsilon)$, and $k = \Omega(\log(m/\delta)/\epsilon)$, and with $\eta = 1/(2m)$. Then, with probability of at least $1 - \delta$,*

$$\max_i \frac{1}{k} \sum_{j=1}^{k} \ell(w_{t_j}, x_i, y_i) \;\leq\; \epsilon \,.$$

The proof of the theorem is given in Appendix A.

The above theorem tells us that we can easily find an ensemble of $O(\log(m)/\epsilon)$ predictors, such that the ensemble loss is smaller than ϵ for all of the examples.

We next need to show that we can construct a single predictor with a small loss. To do so, we consider two typical scenarios. The first is classification settings, in which $\ell(w, x, y)$ is the zero-one loss and the second is convex losses in which $\ell(w, x, y)$ has the form $\phi_y(h_w(x))$, where for every y, ϕ_y is a convex function.

2.1 Classification

In classification, $\ell(w, x, y)$ is the zero-one loss, namely, it equals to zero if $h_w(x) = y$ and it equals to 1 if $h_w(x) \neq y$. In this case, the value of c is 1 and we can take ϵ to be any number strictly smaller than $1/2$, say 0.499.

Observe that Theorem 1 tells us that the average loss of w_t (over $[T]$ or $\mathcal{T}$) is smaller than $\epsilon = 0.499$. Since the values of the loss are either 1 or 0, it means that the loss of more than $1/2$ of the classifiers is 0, which implies that the majority classifier has a zero loss.

Corollary 1. *Assume that $\ell(w, x, y)$ is the zero-one loss function, namely, $\ell(w, x, y) = 1[h_w(x) \neq y]$. Apply Theorem 1 with $\epsilon = 0.49$. Then, with probability of at least $1 - \delta$, the majority classifier of $h_{w_{t_1}}, \ldots, h_{w_{t_k}}$ is consistent[1] with all the examples.*

Example 1. *Consider a binary classification problem in which the data is linearly separable by a vector w^* with a margin of 1. We can use the online Perceptron algorithm as our w learner (see [16]). Then, after $\tilde{O}\left(m + \|w^*\|^2\right)$ iterations, we will find an ensemble of $O(\log(m))$ halfspaces, whose majority vote is consistent with all the examples. In Section 2.4 we compare the runtime of the method to state-of-the-art approaches. Here we just note that to obtain a consistent hypothesis using SGD one needs order of $m \|w^*\|^2$ iterations, which is significantly larger in most scenarios.*

2.2 Convex Losses

Consider now the case in which $\ell(w, x, y)$ has the form $\phi_y(h_w(x))$, where for every y, ϕ_y is a convex function. Note that this assumption alone does not imply that ℓ is a convex function of w (this will be true only if $h_w(x)$ is an affine function).

In the case of convex ϕ_y, combining Theorem 1 with Jensen's inequality we obtain:

Corollary 2. *Under the assumptions of Theorem 1, if $\ell(w, x, y)$ has the form $\phi_y(h_w(x))$, where for every y, ϕ_y is a convex function, then the predictor $h_{\mathcal{T}}(x) = \frac{1}{k} \sum_{t \in \mathcal{T}} h_{w_t}(x)$ satisfies*

$$\forall i, \quad \phi_{y_i}(h_{\mathcal{T}}(x_i)) \;\leq\; \epsilon \,.$$

[1] A consistent hypothesis is a hypothesis that makes no mistakes on the training set.

If we further assume that $h_w(x)$ is an affine function of w, and let $w_{\mathcal{T}} = \frac{1}{k}\sum_{t\in\mathcal{T}} w_t$, then we also have that

$$\forall i, \quad \phi_{y_i}(h_{w_{\mathcal{T}}}(x_i)) \;\leq\; \epsilon\;.$$

Example 2. *Consider a linear regression problem where $\mathcal{X}$ is the ℓ_2 ball of radius X, $\mathcal{W}$ is the ℓ_2 ball of radius W, $\mathcal{Y} = [0,1], h_w(x) = \max\{0, \min\{1, \langle w, x\rangle\}\}$, and $\ell(w,x,y) = |h_w(x) - y|$. For the w player we can use the online gradient descent algorithm (see [16]) with respect to the surrogate loss $\ell^s(w,x,y) = |\langle w, x\rangle - y|$. The regret bound is $R_w(T) \leq XW/\sqrt{T}$. It follows that to find a predictor with error of at most ϵ on all the examples we need that $T = \Omega\left(\frac{X^2W^2}{\epsilon^2} + \frac{m\log(m)}{\epsilon}\right)$. In contrast, to achieve the same result with SGD we need order of $\frac{X^2W^2m^2}{\epsilon^2}$ iterations.*

2.3 Pseudo-code

Below we describe a pseudo-code of the algorithm. We rely on a tree data structure for maintaining the probability of the p-player. It is easy to verify the correctness of the implementation. Observe that the runtime of each iteration is the time required to perform one step of the online learner plus $O(\log(m))$ for sampling from p_t and updating the tree structure.

Focused Online Learning (FOL)

Input:
Training examples $(x_1, y_1), \ldots, (x_m, y_m)$
Loss function $\ell : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to [0,1]$
Parameters η, T, k
Oracle access to online learning algorithm OLA
Initialization:
Tree.initialize(m) (see the Tree pseudo-code)
$w_1 =$ OLA.initialize()
Loop over $t \in \{1, \ldots, T$**:**
$(i_t, p_{i_t}) =$ Tree.sample$(1/2)$
OLA.step(x_{i_t}, y_{i_t})
Tree.update$(i_t, \exp(\eta\, \ell(w_t, x_{i_t}, y_{i_t})/p_{i_t}))$
Output:
Sample $(t_1, \ldots, t_k)$ indices uniformly from $[T]$
Output Majority/Average of $(h_{w_{t_1}}, \ldots, h_{w_{t_k}})$

Tree

initialize(m)
  Build a full binary tree of height $h = 2^{\lceil \log_2(m) \rceil}$
  Set value of the first m leaves to 1 and the rest to 0
  Set the value of each internal node to be
    the sum of its two children
  Let q_i be the value of the i'th leaf divided by
    the value of the root
sample(γ)
  Sample $b \in \{0, 1\}$ s.t. $\mathbb{P}[b = 0] = \gamma$
  If $b = 0$
    Sample i uniformly at random from $[m]$
  Else
    Set v to be the root node of the tree
    While v is not a leaf:
      Go to the left/right child by sampling
        according to their values
    Let i be the obtained leaf
  Return: $(i, \gamma/m + (1 - \gamma)q_i)$
update(i, f)
  Let v be the current value of the i'th leaf of the tree
  Let $\delta = f\, v - v$
  Add δ to the values of all nodes on
    the path from the i'th leaf to the root

2.4 Related Work

As mentioned before, our algorithm is a variant of the approach given in Auer et al. [2, Section 9], but has the advantage that the update of the p player at each iteration scales with $\log(m)$ rather than with m. Phrasing the max-loss minimization as a two players game has also been proposed by [7, 12]. These works focus on the specific case of binary classification with a linear predictor, namely, they tackle the problem $\min_{w \in \mathbb{R}^d : \|w\|_2 \leq 1} \max_{p \in \mathcal{S}_m} \sum_i p_i \langle w, x_i \rangle$. Assuming that the data is linearly separable with a margin of γ, [7] presents an algorithm that finds a consistent hypothesis in runtime of $\tilde{O}((m + d)/\gamma^2)$. For the same problem, our algorithm (with the Perceptron as the weak learner) finds a consistent hypothesis in runtime of $\tilde{O}((m + 1/\gamma^2)d)$. Furthermore, if the instances are $\bar{d}$-sparse (meaning that the number of non-zeros in each x_i is at most $\bar{d}$), then the term d in our bound can be replaced by $\bar{d}$. In any case, our bound is sometimes better and sometimes worse than the one in [7]. We note that we can also use AdaBoost [10] on top of the Perceptron algorithm for the same problem. It can be easily verified that the resulting runtime will be identical to our bound. In this sense, our algorithm can be seen as an online version of AdaBoost.

Finally, several recent works use sampling strategies for speeding up optimization algorithms for minimizing the average loss. See for example [3, 4, 21, 1].

3 Why

In this section we tackle the "Why" question, namely, why should we prefer minimizing the maximal loss instead of the average loss. For simplicity of presentation, throughout this section we deal with binary classification problems with the zero-one loss, in the realizable setting. In this context, minimizing the maximal loss to accuracy of $\epsilon < 1$ leads to a consistent hypothesis[2]. On the other hand, minimizing the average loss to any accuracy of $\epsilon > 1/m$ does not guarantee to return a consistent hypothesis. Therefore, in this context, the "why" question becomes: why should we find a consistent hypothesis and not be satisfied with a hypothesis with $L_{\text{avg}}(h) \leq \epsilon$ for some $\epsilon > 1/m$.

In the usual PAC learning model (see [17] for an overview), there is a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ and the training examples are assumed to be sampled i.i.d. from $\mathcal{D}$. The goal of the learner is to minimize $L_{\mathcal{D}}(h) := \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(h,x,y)] = \mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$. For a fixed $h \in \mathcal{H}$, the random variable $L_{\text{avg}}(h)$ is an unbiased estimator of $L_{\mathcal{D}}(h)$. Furthermore, it can be shown (Boucheron et al. [5, Section 5.1.2]) that with probability of at least $1-\delta$ over the choice of the sample $S \sim \mathcal{D}^m$ we have that:

$$\forall h \in \mathcal{H},\ \ L_{\mathcal{D}}(h) \leq L_{\text{avg}}(h) + \tilde{O}\left(\sqrt{L_{\text{avg}}(h)\frac{\text{VC}(\mathcal{H}) - \log(\delta)}{m}} + \frac{\text{VC}(\mathcal{H}) - \log(\delta)}{m}\right)$$

where $\text{VC}(\mathcal{H})$ is the VC dimension of the class $\mathcal{H}$ and the notation $\tilde{O}$ hides constants and logarithmic terms.

From the above bound we get that any h with $L_{\text{avg}}(h) = 0$ (i.e., a consistent h) guarantees that $L_{\mathcal{D}}(h) = \tilde{O}\left(\frac{\text{VC}(\mathcal{H})+\log(1/\delta)}{m}\right)$. However, we will obtain the same guarantee (up to constants) if we will choose any h with $L_{\text{avg}}(h) \leq \epsilon$, for $\epsilon = \tilde{O}\left(\frac{\text{VC}(\mathcal{H})+\log(1/\delta)}{m}\right)$. Based on this observation, it can be argued that it is enough to minimize L_{avg} to accuracy of $\epsilon = \tilde{O}\left(\frac{\text{VC}(\mathcal{H})+\log(1/\delta)}{m}\right) > \frac{1}{m}$, because a better accuracy on the training set will in any case get lost by the sampling noise.

Furthermore, because of either computational reasons or high dimensionality of the data, we often do not directly minimize the zero-one loss, and instead minimize a convex surrogate loss, such as the hinge-loss. In such cases, we often rely on a margin based analysis, which means that the term $\text{VC}(\mathcal{H})$ is replaced by B^2, where B is the restriction of the norm of the weight vector that defines the classifier. It is often the case that the convergence rate of SGD is of the same order, and therefore there is no added value of solving the ERM problem over performing a single SGD path over the data (or few epochs over the data). Formal arguments of this nature were given in [6, 19].

Despite of these arguments, we show below reasons to prefer the max loss formulation over the avg loss formulation. The first reason is straightforward: arguments that are based on worst case bounds are problematic, since in many cases the behavior is rather different than the worst case bounds. In subsection 3.1 we present a simple example in which there is a large gap between the sample complexity of SGD and the sample complexity of ERM, and we further show that the runtime of our algorithm will be much better than the runtime of SGD for solving this problem.

Next, we describe a family of problems in which the distribution from which the training data is being sampled is a mix of "typical" examples and "rare" examples. We show that in such a case, few "rare" examples may be sufficient for learning a hypothesis that has a high accuracy on both the "typical" and

[2]Recall that a consistent hypothesis is a hypothesis that makes no mistakes on the training set. We also use the term Empirical Risk Minimization (ERM) to describe the process of finding a consistent hypothesis, and use $\text{ERM}(S)$ to denote any hypothesis which is consistent with a sample S.

"rare" examples, and therefore, it is really required to solve the ERM problem as opposed to being satisfied with a hypothesis for which $L_{\text{avg}}(h)$ is small.

3.1 A Simple Example of a Gap

Consider the following distribution. Let $z_1 = (\alpha, 1)$ and $z_2 = (\alpha, -2\alpha)$ for some small $\alpha > 0$. To generate an example $(x, y) \sim \mathcal{D}$, we first sample a label y uniformly at random from $\{\pm 1\}$, then we set $x = yz_1$ with probability $1 - \epsilon$ and set $x = yz_2$ with probability ϵ. The hypothesis class is halfspaces: $\mathcal{H} = \{x \to \text{sign}(\langle w, x \rangle) : w \in \mathbb{R}^2\}$.

The following three lemmas, whose proofs are given in the appendix, establish the gap between the different approaches.

Lemma 1. *For every $\delta \in (0, 1)$, if $m \geq \frac{2\log(4/\delta)}{\epsilon}$ then, with probability of at least $1 - \delta$ over the choice of the training set, $S \sim \mathcal{D}^m$, any hypothesis in ERM(S) has a generalization error of 0.*

Lemma 2. *Suppose we run SGD with the hinge-loss and any $\eta > 0$ for less than $T = \Omega(1/(\alpha\epsilon))$ iterations. Then, with probability of $1 - O(\epsilon)$ we have that SGD will not find a solution with error smaller than ϵ.*

Lemma 3. *Running our algorithm takes $\tilde{O}\left(\frac{1}{\epsilon} + \frac{1}{\alpha}\right)$ iterations.*

3.2 Typical vs. Rare Distributions

To motivate the learning setting, consider the problem of face detection, in which the goal is to take an image crop and determine whether it is an image of a face or not. An illustration of typical random positive and negative examples is given in Figure 1 (top row). By having enough training examples, we can learn that the discrimination between face and non-face is based on few features like "an ellipse shape", "eyes", "nose", and "mouth". However, from the typical examples it is hard to tell whether an image of a watermelon is a face or not — it has the ellipse shape like a face, and something that looks like eyes, but it doesn't have a nose, or a mouth. The bottom row of Figure 1 shows some additional "rare" examples.

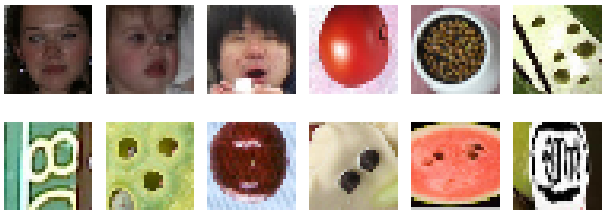

Figure 1: Top: typical positive (left) and negative (right) examples. Bottom: rare negative examples.

Such a phenomenon can be formally described as follows. There are two distributions over the examples, $\mathcal{D}_1$ and $\mathcal{D}_2$. Our goal is to have an error of at most ϵ on **both** distributions, namely, we would like to find h such that $L_{\mathcal{D}_1}(h) \leq \epsilon$ and $L_{\mathcal{D}_2}(h) \leq \epsilon$. However, the training examples that we observe are sampled i.i.d. from a mixed distribution, $\mathcal{D} = \lambda_1\mathcal{D}_1 + \lambda_2\mathcal{D}_2$, where $\lambda_1, \lambda_2 \in (0, 1)$ and $\lambda_1 + \lambda_2 = 1$. We assume that $\lambda_2 \ll \lambda_1$, namely, typical examples in the training set are from $\mathcal{D}_1$ while examples from $\mathcal{D}_2$ are rare.

Fix some ϵ. If $\lambda_2 < \epsilon$, then a hypothesis with $L_{\text{avg}}(h) \leq \epsilon$ might err on most of the "rare" examples, and is therefore likely to have $L_{\mathcal{D}_2}(h) > \epsilon$. If we want to guarantee a good performance on $\mathcal{D}_2$ we must optimize to a very high accuracy, or put another way, we would like to minimize $L_{\max}$ instead of L_{avg}. The

question is how many examples do we need in order to guarantee that a consistent hypothesis on S will have a small error on both $\mathcal{D}_1$ and $\mathcal{D}_2$. A naive approach is to require order of $\mathrm{VC}(\mathcal{H})/(\lambda_2\epsilon)$ examples, thus ensuring that we have order of $\mathrm{VC}(\mathcal{H})/\epsilon$ examples from both $\mathcal{D}_1$ and $\mathcal{D}_2$. However, this is a rough estimate and the real sample complexity might be much smaller. Intuitively, we can think of the typical examples from $\mathcal{D}_1$ as filtering out most of the hypotheses in $\mathcal{H}$, and the goal of the rare examples is just to fine tune the exact hypothesis. In the example of face detection, the examples from $\mathcal{D}_1$ will help us figure out what is an "ellipse like shape", what is an "eye", and what is a "mouth" and a "nose". After we understand all this, the rare examples from $\mathcal{D}_2$ will tell us the exact requirement of being a face (e.g., you need an ellipse like shape and either eyes or a mouth). We can therefore hope that the number of required "rare" examples is much smaller than the number of required "typical" examples. This intuition is formalized in the following theorem.

Theorem 2. *Fix $\epsilon, \delta \in (0,1)$, distributions D_1, D_2, and let $D = \lambda_1 D_1 + \lambda_2 D_2$ where $\lambda_1 + \lambda_2 = 1, \lambda_1, \lambda_2 \in [0,1]$, and $\lambda_2 < \lambda_1$. Define $\mathcal{H}_{1,\epsilon} = \{h \in \mathcal{H} : L_{D_1}(h) \leq \epsilon\}$ and $c = \max\{c' \in [\epsilon, 1) : \forall h \in \mathcal{H}_{1,\epsilon},\ L_{D_2}(h) \leq c' \Rightarrow L_{D_2}(h) \leq \epsilon\}$. Then, if*

$$m \geq \Omega\left(\frac{\mathrm{VC}(\mathcal{H})\log(1/\epsilon) + \log(1/\delta)}{\epsilon} + \frac{\mathrm{VC}(\mathcal{H}_{1,\epsilon})\log(1/c) + \log(1/\delta)}{c\,\lambda_2}\right)$$

we have, with probability of at least $1-\delta$ over the sampling of a sample $S \sim D^m$:

$$L_{D_1}(\mathrm{ERM}(S)) \leq \epsilon \quad \textit{and} \quad L_{D_2}(\mathrm{ERM}(S)) \leq \epsilon$$

The proof of the theorem is given in the appendix. The first term in the sample complexity is a standard VC-based sample complexity. The second term makes two crucial improvement. First, we measure the VC dimension of a reduced class ($\mathcal{H}_{1,\epsilon}$), containing only those hypotheses in $\mathcal{H}$ that have a small error on the "typical" distribution. Intuitively, this will be a much smaller hypothesis class compared to the original class. Second, we apply an analysis of the sample complexity similar to the "shell analysis" of [11], and assume that the error of all hypotheses in $\mathcal{H}_{1,\epsilon}$ on $\mathcal{D}_2$ is either smaller than ϵ or larger than c, where we would like to think of c as a constant.

All in all, the theorem shows that a small number of "rare" examples in the training set can have a dramatic effect on the performance of the algorithm on the rare distribution $\mathcal{D}_2$. But, we will see this effect only if we will indeed find a hypothesis consistent with all (or most) examples from $\mathcal{D}_2$, which requires an algorithm for minimizing $L_{\max}$ and not L_{avg}.

4 Robustness

In the previous section we have shown cases in which minimizing $L_{\max}$ is better than minimizing L_{avg}. However, in the presence of outliers, minimizing $L_{\max}$ might lead to meaningless results — even a single outlier can change the value of $L_{\max}$ and might lead to a trivial, non-interesting, solution. In this section we describe two tricks for addressing this problem. The first trick replaces the original sample with a new sample whose examples are sampled from the original sample. The second trick relies on slack variables. We note that these tricks are not new and appears in the literature in various forms. See for example [13, 15]. The goal of this section is merely to show how to apply known tricks to the max loss problem.

Recall that in the previous section we have shown that a small amount of "rare" examples can have a dramatic effect on the performance of the algorithm on the "rare" distribution. Naturally, if the number of outliers is larger than the number of rare examples we cannot hope to enjoy the benefit of rare examples. Therefore, throughout this section we assume that the number of outliers, denoted k, is smaller than the number of "rare" examples, which we denote by m_2.

4.1 Sub-sampling with repetitions

The first trick we consider is to simply take a new sample of n examples, where each example in the new sample is sampled independently according to the uniform distribution over the original m examples. Then, we run our algorithm on the obtained sample of n examples.

Intuitively, if there are k outliers, and the size of the new sample is significantly smaller than m/k, then there is a good chance that no outliers will fall into the new sample. On the other hand, we want that enough "rare" examples will fall into the new sample. The following theorem, whose proof is in the appendix, shows for which values of k and m_2 this is possible.

Theorem 3. *Let k be the number of outliers, m_2 be the number of rare examples, m be the size of the original sample, and n be the size of the new sample. Assume that $m \geq 10k$. Then, the probability that the new sample contains outliers and/or does not contain at least $m_2/2$ rare examples is at most $0.01 + 0.99kn/m + e^{-0.1\, nm_2/m}$.*

For example, if $n = m/(100k)$ and $m_2 \geq 1000 \, \log(100) \, k$, then the probability of the bad event is at most 0.03.

4.2 Slack variables

Another common trick, often used in the SVM literature, is to introduce a vector of slack variables, $\xi \in \mathbb{R}^m$, such that $\xi_i > 0$ indicates that example i is an outlier. We first describe the ideal version of outlier removal. Suppose we restrict ξ_i to take values in $\{0, 1\}$, and we restrict the number of outliers to be at most K. Then, we can write the following optimization problem:

$$\min_{w \in \mathcal{W}, \xi \in \mathbb{R}^m} \; \max_{i \in [m]} \; (1 - \xi_i) \, \ell(w, x_i, y_i) \;\; \text{s.t.}$$
$$\xi \in \{0, 1\}^m, \; \|\xi\|_1 \leq K \; .$$

This optimization problem minimizes the max loss over a subset of examples of size at least $m - K$. That is, we allow the algorithm to refer to at most K examples as outliers.

Note that the above problem can be written as a max-loss minimization:

$$\min_{\bar{w} \in \bar{\mathcal{W}}} \max_i \bar{\ell}(\bar{w}, x_i, y_i) \quad \text{where}$$
$$\bar{\mathcal{W}} = \{(w, \xi) : w \in \mathcal{W}, \xi \in \{0, 1\}^m, \|\xi\|_1 \leq K\} \;\; \text{and}$$
$$\bar{\ell}((w, \xi), x_i, y_i) = (1 - \xi_i) \ell(w, x_i, y_i)$$

We can now apply our framework on this modified problem. The p player remains as before, but now the $\bar{w}$ player has a more difficult task. To make the task easier we can perform several relaxations. First, we can replace the non-convex constraint $\xi \in \{0, 1\}^m$ with the convex constraint $\xi \in [0, 1]^m$. Second, we can replace the multiplicative slack with an additive slack, and re-define: $\bar{\ell}((w, \xi), x_i, y_i) = \ell(w, x_i, y_i) - \xi_i$.

This adds a convex term to the loss function, and therefore, if the original loss was convex we end up with a convex loss. The new problem can often be solved by combining gradient updates with projections of ξ onto the set $\xi \in [0,1]^m, \|\xi\|_1 \leq K$. For efficient implementations of this projection see for example [8]. We can further replace the constraint $\|\xi\|_1 \leq K$ with a constraint of $\|\xi\|_2^2 \leq K$, because projection onto the Euclidean ball is a simple scaling, and the operation can be done efficiently with an adequate data structure (as described, for example, in [20]).

A Proof of Theorem 1

A.1 Background

Bernstein's type inequality for martingales: A sequence $B_1, \ldots, B_T$ of random variables is Markovian if for every t, given B_{t-1} we have that B_t is independent of $B_1, \ldots, B_{t-2}$. A sequence $A_1, \ldots, A_T$ of random variables is a martingale difference sequence with respect to $B_1, \ldots, B_T$ if for every t we have $\mathbb{E}[A_t | B_1, \ldots, B_t] = 0$.

Lemma 4 (Hazan et al. [12, Lemma C.3] and Fan et al. [9, Theorem 2.1])**.** *Let $B_1, \ldots, B_T$ be a Markovian sequence and let $A_1, \ldots, A_T$ be a martingale difference sequence w.r.t. $B_1, \ldots, B_T$. Assume that for every t we have $|A_t| \leq V$ and $\mathbb{E}[A_t^2 | B_1, \ldots, B_t] \leq s$. Then, for every $\alpha > 0$ we have*

$$\mathbb{P}\left(\frac{1}{T}\sum_{t=1}^{T} A_t \geq \alpha\right) \leq \exp\left(-T\frac{\alpha^2/2}{s + \alpha V/3}\right)$$

In particular, for every $\delta \in (0,1)$, if

$$T \geq \frac{2(s + \alpha V/3)\log(1/\delta)}{\alpha^2}\,,$$

then with probability of at least $1-\delta$ we have that $\frac{1}{T}\sum_{t=1}^{T} A_t \leq \alpha$.

The EG algorithm: Consider a sequence of vectors, $z_1, \ldots, z_T$, where every $z_t \in \mathbb{R}^m$. Consider the following sequence of vectors, parameterized by $\eta > 0$. The first vector is $\tilde{q}_1 = (1, \ldots, 1) \in \mathbb{R}^m$ and for $t \geq 1$ we define $\tilde{q}_{t+1}$ to be such that:

$$\forall i \in [m],\ \ \tilde{q}_{t+1,i} = \tilde{q}_{t,i}\exp(-\eta z_{t,i})\,.$$

In addition, for every t define $q_t = \tilde{q}/(\sum_{i=1}^m \tilde{q}_i) \in \mathcal{S}_m$. The algorithm that generates the above sequence is known as the EG algorithm [14].

Lemma 5 (Theorem 2.22 in [16])**.** *Assume that $\eta z_{t,i} \geq -1$ for every t and i. Then, for every $u \in \mathcal{S}_m$ we have:*

$$\sum_{t=1}^{T}\langle q_t - u, z_t\rangle \ \leq\ \frac{\log(m)}{\eta} + \eta\sum_{t=1}^{T}\sum_{i=1}^{m} q_{t,i}z_{t,i}^2\,.$$

A.2 Proof

To simplify our notation we denote $\ell_i(w_t) = \ell(w_t, x_i, y_i)$. We sometimes omit the time index t when it is clear from the context (e.g., we sometime use q_i instead of $q_{t,i}$).

A.2.1 The w player

By our realizability assumption and the assumption that $R_w(T) \leq \epsilon/6$ we have that, for every $i_1, \ldots, i_T$,

$$\frac{1}{T}\sum_{t=1}^{T} \ell_{i_t}(w_t) \;\leq\; \epsilon/6 \tag{5}$$

A.2.2 The p player

Recall that $p_i = \frac{1}{2m} + \frac{q_i}{2}$. Note that, for every i,

$$\frac{1}{p_i} \leq 2m \quad \text{and} \quad \frac{q_i}{p_i} \leq 2$$

Define let $z_t = -\frac{\ell_{i_t}(w_t)}{p_{i_t}} e_{i_t}$. Observe that the p player applies the EG algorithm w.r.t. the sequence $z_1, \ldots, z_T$. Since $z_{t,i} \geq -2m$ we obtain from Lemma 5 that if $\eta \leq 1/(2m)$ then, for every $u \in \mathcal{S}_m$,

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T} \langle q_t - u, z_t\rangle \;&\leq\; \frac{\log(m)}{\eta T} + \frac{\eta}{T}\sum_{t=1}^{T}\sum_{i=1}^{m} q_{t,i} z_{t,i}^2 \\
&\leq \frac{\log(m)}{\eta T} + \frac{\eta}{T}\sum_{t=1}^{T} q_{t,i_t}\frac{\ell_{i_t}(w_t)^2}{p_{t,i_t}^2} \\
&\leq \frac{\log(m)}{\eta T} + \frac{\eta}{T}\sum_{t=1}^{T} 4m\ell_{i_t}(w_t)^2 \\
&\leq \frac{\log(m)}{\eta T} + \frac{\eta 4m}{T}\sum_{t=1}^{T} \ell_{i_t}(w_t) \\
&\leq \epsilon/8 + \frac{2}{T}\sum_{t=1}^{T} \ell_{i_t}(w_t)\;,
\end{aligned}$$

where in the last inequality we used $\eta = 1/(2m)$ and $T = \Omega(m\log(m)/\epsilon)$. Rearranging, and combining with (5) we obtain

$$\frac{1}{T}\sum_{t=1}^{T}\langle u, \frac{1}{p_{t,i_t}}\ell_{i_t}(w_t)e_{i_t}\rangle \;\leq\; \frac{1}{T}\sum_{t=1}^{T}\left(\frac{q_{t,i_t}}{p_{t,i_t}} + 2\right)\ell_{i_t}(w_t) + \epsilon/8 \;\leq\; \frac{4}{T}\sum_{t=1}^{T}\ell_{i_t}(w_t) + \epsilon/8 \leq \frac{5\epsilon}{8}\;. \tag{6}$$

A.2.3 Measure concentration

Note that, if $u = e_i$, then

$$\mathbb{E}[\langle u, \frac{1}{p_{t,i_t}}\ell_{i_t}(w_t)e_{i_t}\rangle^2 | q_t, w_t] = \sum_{j=1}^{m}\frac{p_{t,j}}{p_{t,j}^2}\ell_j(w_t)^2 u_j \leq \frac{1}{p_{t,i}} \leq 2m\;.$$

Define the martingale difference sequence $A_1, \ldots, A_T$ where $A_t = \ell_i(w_t) - \langle u, \frac{1}{p_{t,i_t}}\ell_{i_t}(w_t)e_{i_t}\rangle$. We have that $|A_t| \leq (2m+1)$ and $\mathbb{E}[A_t^2|q_t, w_t] \leq 2m$. Therefore, the conditions of Lemma 4 holds and we obtain

that if $T \geq 6m\log(m/\delta)/(\epsilon/8)^2$ then with probability of at least $1-\delta/m$ we have that $\frac{1}{T}\sum_t A_t \leq \epsilon/8$. Applying a union bound over $i \in [m]$ we obtain that with probability of at least $1-\delta$ it holds that

$$\forall i \in [m],\ \ \frac{1}{T}\sum_t \ell_i(w_t)\ \leq\ \frac{1}{T}\sum_t \langle u, \frac{1}{p_{i_t}}\ell_{i_t}(w_t)e_{i_t}\rangle + \epsilon/8\ .$$

Combining with (6) we obtain that, with probability of at least $1-\delta$,

$$\forall i \in [m],\ \ \frac{1}{T}\sum_{t=1}^{T} \ell_i(w_t)\ \leq\ \frac{6\epsilon}{8}\ .$$

Finally, relying on Bernstein's inequality (see Lemma B.10 in [17]), it is not hard to see that if $k = \Omega(\log(m/\delta)/\epsilon)$ then, with probability of at least $1-\delta$ we have that

$$\forall i \in [m],\ \ \frac{1}{k}\sum_{i=1}^{k} \ell_i(w_{t,i}) \leq \frac{1}{T}\sum_{t=1}^{T} \ell_i(w_t) + \frac{\epsilon}{4}\ ,$$

and this concludes our proof.

B Proofs of Lemmas in Section 3.1

Proof of Lemma 1. There are only 4 possible examples, so an ERM will have a generalization error of 0 provided we see all the 4 examples. By a simple direct calculation together with the union bound over the 4 examples it is easy to verify that the probability not to see all the examples is at most $4(1-\epsilon/2)^m \leq 4e^{-m\epsilon/2}$, and the claim follows. □

Proof of Lemma 2. For SGD, we can assume (due to symmetry) that y is always 1. Therefore, there are only two possible examples z_1, z_2. With probability $1-\epsilon$ the first examples is z_1. Also, w_T has always the form

$$\eta(kz_1 + rz_2) = \eta((k+r)\alpha, k-2r\alpha)\ ,$$

where k is the number of times we had a margin error on z_1 and r is the number of times we had a margin error on z_2. To make sure that $\langle w_T, z_2\rangle > 0$ we must have that

$$(k+r)\alpha^2 - 2(k-2r\alpha)\alpha > 0\ \ \Rightarrow\ \ r > k\,\frac{2\alpha-\alpha^2}{5\alpha^2} \approx k\frac{2}{5\alpha} \tag{7}$$

Note that the first example is z_1 with probability of $1-\epsilon$, hence we have that $k \geq 1$ with probability of at least $1-\epsilon$. In addition, r is upper bounded by the number of times we saw z_2 as the example, and by Chernoff's bound we have that that the probability that this number is greater than $2m\epsilon$ is at most $e^{-\epsilon/3} \approx (1-\epsilon/3)$. Therefore, with probability of $1-O(\epsilon)$ we have the requirement that m must be at least $\Omega(1/(\alpha\epsilon))$, which concludes our proof. □

Proof of Lemma 3. We have shown that $m = 1/\epsilon$ examples suffices. Specifying our general analysis to classification with the hinge-loss, it suffices to ensure that the regret of both players will be smaller than $1/2$. The regret of the sampling player is bounded by $O(m\log(m))$. As for the halfspace player, to simplify the derivation, lets use the Perceptron as the underlying player. I is easy to verify that the vector w_T has the form $kz_1 + rz_2 = ((k+r)\alpha, k-2r\alpha)$, for some integers k, r. Lets consider two regimes. The first is

the first time when r, k satisfies $\langle w_T, z_2 \rangle > 0$. As we have shown before, this happens when r is roughly $2k/(5\alpha)$. Once this happens we also have that

$$\langle w_T, z_1 \rangle = ((k+r)\alpha^2 + k - 2r\alpha) \approx (k - 2r\alpha) \gtrapprox 4k/5 > 0 \,,$$

So, the Perceptron will stop making changes and will give us an optimal halfspace. Next, suppose that we have a pair r, k for which $\langle w_T, z_2 \rangle \leq 0$. If we now encounter z_2 then we increase r. If we encounter z_1 then

$$\langle w_T, z_1 \rangle \approx (k - 2r\alpha) \gtrapprox 4k/5 > 0 \,,$$

so we'll not increase k. Therefore, k will increase only up to a constant, while r will continue to increase until roughly $2k/(5\alpha)$, and then the Perceptron will stop making updates. This implies that the regret of the Perceptron is bounded by $O(1/\alpha)$, which concludes our proof. □

C Proof of Theorem 2

We can think of the ERM algorithm as following the following three steps. First, we sample $(i_1, \ldots, i_m) \in \{1, 2\}^m$, where $\mathbb{P}[i_r = j] = \lambda_j$. Let m_1 be the number of indices for which $i_r = 1$ and let $m_2 = m - m_1$. Second, we sample $S_1 \sim D_1^{m_1}$, and define $\hat{\mathcal{H}}_1$ to be all hypotheses in $\mathcal{H}$ which are consistent with S_1. Last, we sample $S_2 \sim D_2^{m_2}$ and set the output hypotheses to be some hypothesis in $\hat{\mathcal{H}}_1$ which is consistent with S_2.

The proof relies on the following three claims, where we use C to denote a universal constant:

- **Claim 1:** With probability of at least $1 - \delta/3$ over the choice of $(i_1, \ldots, i_m)$ we have that both $m_1 \geq \lambda_1 m/2$ and $m_2 \geq \lambda_2 m/2$.
- **Claim 2:** Assuming that $m_1 \geq C \left(\frac{\mathrm{VC}(\mathcal{H}) \log(1/\epsilon) + \log(1/\delta)}{\epsilon} \right)$, then with probability of at least $1 - \delta/3$ over the choice of S_1 we have that $\hat{\mathcal{H}}_1 \subseteq \mathcal{H}_{1,\epsilon}$.
- **Claim 3:** Assume that $m_2 \geq C \left(\frac{\mathrm{VC}(\mathcal{H}_{1,\epsilon}) \log(1/c) + \log(1/\delta)}{c} \right)$, then with probability of at least $1 - \delta/3$ over the choice of S_2, any hypothesis in $\mathcal{H}_{1,\epsilon}$ which is consistent with S_2 must have $L_{D_2}(h) \leq c$.

Claim 1 follows directly from Chernoff's bound, while Claim 2-3 follows directly from standard VC bounds (see for example Shalev-Shwartz and Ben-David [17, Theorem 6.8]).

Equipped with the above three claims we are ready to prove the theorem. First, we apply the union bound to get that with probability of at least $1 - \delta$, the statements in all the above three claims hold. This means that $\hat{\mathcal{H}}_1 \subseteq \mathcal{H}_{1,\epsilon}$ hence $L_{D_1}(\mathrm{ERM}(S)) \leq \epsilon$. It also means that $\mathrm{ERM}(S)$ must be in $\mathcal{H}_{1,\epsilon}$, and therefore from the third claim and the assumption in the theorem we have that $L_{D_2}(\mathrm{ERM}(S)) \leq \epsilon$ as well, which concludes our proof.

D Proof of Theorem 3

The probability that all of the outliers do not fall into the sample of n examples is

$$(1 - k/m)^n \geq 0.99\, e^{-kn/m} \,.$$

Therefore, the probability that at least one outlier falls into the sample is at most

$$1 - 0.99\, e^{-kn/m} \leq 1 - 0.99(1 - kn/m) = 0.01 + 0.99kn/m$$

On the other hand, the expected number of rare examples in the sample is nm_2/m and by Chernoff's bound, the probability that less than half of the rare examples fall into the sample is at most $\exp(-0.1\, nm_2/m)$. Applying the union bound we conclude our proof.

References

[1] Zeyuan Allen-Zhu and Yang Yuan. Even faster accelerated coordinate descent using non-uniform sampling. *arXiv preprint arXiv:1512.09103*, 2015.

[2] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[3] Yoshua Bengio and Jean-Sébastien Senécal. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *Neural Networks, IEEE Transactions on*, 19(4):713–722, 2008.

[4] Guillaume Bouchard, Théo Trouillon, Julien Perez, and Adrien Gaidon. Accelerating stochastic gradient descent via online learning to sample. *arXiv preprint arXiv:1506.09016*, 2015.

[5] Stéphane Boucheron, Olivier Bousquet, and Gábor Lugosi. Theory of classification: A survey of some recent advances. *ESAIM: probability and statistics*, 9:323–375, 2005.

[6] Olivier Bousquet and Léon Bottou. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.

[7] Kenneth L Clarkson, Elad Hazan, and David P Woodruff. Sublinear optimization for machine learning. *Journal of the ACM (JACM)*, 59(5):23, 2012.

[8] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM, 2008.

[9] Xiequan Fan, Ion Grama, and Quansheng Liu. Hoeffding's inequality for supermartingales. *Stochastic Processes and their Applications*, 122(10):3545–3559, 2012.

[10] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[11] David Haussler, Michael Kearns, H Sebastian Seung, and Naftali Tishby. Rigorous learning curve bounds from statistical mechanics. *Machine Learning*, 25(2-3):195–236, 1996.

[12] Elad Hazan, Tomer Koren, and Nati Srebro. Beating sgd: Learning svms in sublinear time. In *Advances in Neural Information Processing Systems*, pages 1233–1241, 2011.

[13] Peter J. Huber and Elvezio M. Ronchetti. *Robust Statistics (second edition)*. J. Wiley, 2009.

[14] J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, January 1997.

[15] Ricardo A Maronna, R Douglas Martin, and Victor J Yohai. *Robust Statistics: Theory and Methods*. J. Wiley, 2006.

[16] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2011.

[17] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge university press, 2014.

[18] Shai Shalev-Shwartz and Yoram Singer. On the equivalence of weak learnability and linear separability: New relaxations and efficient boosting algorithms. *Machine learning*, 80(2-3):141–163, 2010.

[19] Shai Shalev-Shwartz and Nathan Srebro. Svm optimization: inverse dependence on training set size. In *Proceedings of the 25th international conference on Machine learning*, pages 928–935. ACM, 2008.

[20] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[21] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling. *arXiv preprint arXiv:1401.2753*, 2014.