On the Perceptron's Compression

Shay Moran* Ido Nachum† Itai Panasoff‡ Amir Yehudayoff§

Abstract

We study and provide exposition to several phenomena that are related to the perceptron's compression. One theme is deducing conclusions from this compression, for example, we describe applications in the robust concepts model suggested by Arriaga and Vempala, and we prove a "sparse" version of the separation theorem for convex bodies (which also provides a different proof of Novikoff's theorem on the convergence of the perceptron). A second theme is based on modifications of the perceptron algorithm that extend the aggressive perceptron and yield better guarantees on its output. These modifications can be useful in training neural networks as well, and we demonstrate them by exhibiting simple neural networks with low error on the MNIST database.

*School of Mathematics, Institute for Advanced Study, Princeton NJ. shaymoran1@gmail.com.

†Department of Mathematics, Technion-IIT, Israel. idon@tx.technion.ac.il.

‡Department of Mathematics, Technion-IIT, Israel. itai.panasoff@gmail.com .

§Department of Mathematics, Technion-IIT, Israel. amir.yehudayoff@gmail.com. Research supported by ISF Grant No. 1162/15.

1 Introduction

The perceptron is an abstraction of a biological neuron that was introduced in the 1950's by Rosenblatt [29], and has been extensively studied in many works (see e.g. the survey [26]). It receives as input a list of real numbers (various electrical signals in the biological case) and if the weighted sum of its input is greater than some threshold it outputs 1 and otherwise -1 (it fires or not in the biological case). More formally, a perceptron computes a function of the form $\text{sign}(w \cdot x - b)$ where $w \in \mathbb{R}^d$ is the weight vector, $b \in \mathbb{R}$ is the threshold, $\cdot$ is the standard inner product, and $\text{sign} : \mathbb{R} \to \{\pm 1\}$ is 1 on the nonnegative numbers. It is only capable of representing binary functions that are partitions of $\mathbb{R}^d$ defined by some hyperplane.

Definition. *A map $Y : \mathcal{X} \to \{\pm 1\}$ over a finite set $\mathcal{X} \subset \mathbb{R}^d$ is (linearly)[1] separable if there exists $w \in \mathbb{R}^d$ such that $\text{sign}(w \cdot x) = Y(x)$ for all $x \in \mathcal{X}$. When the Euclidean norm of w is $\|w\| = 1$, the number $\text{marg}(w, Y) = \inf_{x \in \mathcal{X}} Y(x) w \cdot x$ is the margin of w with respect to Y. The number $\text{marg}(Y) = \sup_{w \in \mathbb{R}^d : \|w\| = 1} \text{marg}(w, Y)$ is the margin of Y. We call Y an ε-partition if its margin is at least ε.*

Variants of the perceptron (neurons) are the basic building blocks of general neural networks. Typically, the sign function is replaced by some other activation function (e.g., sigmoid or rectified linear unit $\mathsf{ReLu}(z) = \max\{0, z\}$). Studying the perceptron and its variants may therefore help in understanding neural networks, their design and their training process.

Here we provide some new insights into the perceptron's behavior, survey some of the related work, and deduce some geometric applications. These ideas may also be useful in other learning contexts, as we explain below.

The Training Process

Deciding how to train a model from a list of input examples is a central consideration

[1] We focus on the linear case, when the threshold is 0. A standard lifting that adds a coordinate with 1 to every vector allows to translate the general (affine) case to the linear case. This lifting may significantly decrease the margin; e.g., the map Y on $\mathcal{X} = \{999, 1001\} \subset \mathbb{R}$ defined by $Y(999) = 1$ and $Y(1001) = -1$ has margin 1 in the affine sense, but the lift to $(999, 1)$ and $(1001, 1)$ in $\mathbb{R}^2$ yields very small margin in the linear sense. This solution may therefore cause an unnecessary increase in running time. This tax can be avoided, for example, if one has prior knowledge of $R = \max_{x \in \mathcal{X}} \|x\|$. In this case, setting the last coordinate to be R does not significantly decrease the margin. In fact, it can be avoided without any prior knowledge using the idea in Algorithm 3 below.

in any learning process. In the case of the perceptron algorithm the input examples are traversed while maintaining a hypothesis $w^{(t)}$ in a way that reduces the error on the current example:

initialize: $w^{(0)} = \vec{0}$ and $t = 0$
while $\exists i$ *with* $y_i w^{(t)} \cdot x_i \leq 0$ **do**
  $w^{(t+1)} = w^{(t)} + y_i x_i$
  $t = t + 1$
end
return $w^{(t)}$

Algorithm 1: The perceptron algorithm

It is well known that the perceptron algorithm terminates whenever its input sample is linearly separable, in which case its output represents a separating hyperplane. Novikoff analyzed the number of steps T required for the Perceptron to stop as a function of the margin of the input sample [28].

A drawback of this algorithm is that there is no guarantee on the margin of its output (e.g. the margin of the output separator may be much smaller than the optimal margin). Various works suggested and studied different variants of the basic perceptron algorithm [7, 9, 21, 16, 2, 35, 22], some of them also guarantee optimal margin (under various assumptions).

In Section 2 we suggest different variants of the perceptron that are simple to implement and do not require prior knowledge on the norm of the examples nor on the optimal margin. These variants can be seen as extensions of the aggressive perceptron [10], and are similar to ideas that appear in optimization algorithms in the context of Support Vector Machines (see e.g. [32] and references within).

In a nutshell, we suggest to replace the condition $y_i w^{(t)} \cdot x_i < 0$ by $y_i w^{(t)} \cdot x_i < \beta$ for some appropriately chosen $\beta > 0$ that may change over time and space. As we will see, different choices of β yield different guarantees. In the aggressive perceptron, β is fixed to be 1 and the output's margin is shown to be at least $\frac{\varepsilon^*}{2+R^2}$, where ε^* is the optimal margin, and R is the maximal norm of an input example.

We suggest two different ways to choose β. One that allows to remove the dependence on R in the output's margin, and one that allows to arbitrarily approach the optimal margin. A related work of Soudry et al. [34] analyzes gradient descent for a single neuron and shows convergence to the optimal separating hyperplane under certain assumptions (linearly separable, and appropriate activation and loss functions).

Our results explain some choices that are made in practice, and can potentially help to improve them. Observe that if one applies gradient descent on a neuron of the form $\mathsf{ReLu}(w \bullet x)$ with loss function of the form $\mathsf{ReLu}(\beta - f(x) w \bullet x)$ with $\beta = 0$ then one gets the same update rule as in the perceptron algorithm. Choosing $\beta = 1$ corresponds to using the hinge loss to drive the learning process. The fact that $\beta = 1$ yields provable bounds on the output's margin of a single neuron provides a theoretical explanation of the success of the hinge loss. Moreover, thinking of β as a hyper-parameter that can be carefully tuned is a common problem that needs to be addressed in order to maximize performance. We propose a couple of new options for choosing and updating β throughout the training process, which can help in solving this hyper-parameter problem (see Algorithms 4 and 3). We also explain the theoretical advantages of these options in the case of a single neuron.

We also provide some experimental data. Our experiments verify that our suggestions for choosing β can indeed yield better results. We used the MNIST database [23] of handwritten digits as a test case. We used a simple and standard neural network with one hidden layer consisting of 800 neurons and 10 output neurons (the choice of 800 is the same as in Simard et al. [33]). We trained the network by back-propagation (gradient descent). The loss function of each output neurons of the form $\mathsf{ReLu}(w \bullet G(x))$ where $G(x)$ is the output of the hidden layer is $\mathsf{ReLu}(-f(x) w \bullet G(x) + \beta)$ for different β's. This loss function is 0 if w provides a correct and confident (depending on β) classification of x and is linear in $G(x)$ otherwise. This choice updates the network even when the network classifies correctly but with less than β confidence. It has the added value of yielding simple and efficient calculations compared to other choices (like cross entropy or soft-max).[2]

We tested four values of β as shown in Figure 1. In two tests, the value of β is fixed in time[3] to be 0 and 1. In two tests, β changes with the time t in a sub-linear fashion. This choice can be better understood after reading the analysis of Algorithm 4. Roughly speaking, the analysis predicts that β should be of the form t^{1-c} for $c > 0$, and that the smaller c is, the smaller the error will be and the more time the learning process will take. This prediction is indeed verified in the experiments; it is evident that choosing β in a time-dependent manner yields better results. For comparison, the last row of

[2] An additional added value is that with this loss function there is a dichotomy, either an error occurred or not. This dichotomy can be helpful in making decisions throughout the learning process. For example, instead of choosing the batch-size to be of fixed size B, we can choose the batch-size in a dynamic but simple way: just wait until B errors occurred.

[3] Time is measured by the number of updates.

the table shows the error of the two-layer MLP of the same size that is driven by the cross-entropy loss [33].

	error on MNIST
$\beta = 0$	no convergence
$\beta = 1$	1.5 %
$\beta \approx t^{0.4}$	1.44 %
$\beta \approx t^{0.75}$	1.35 %
cross-entropy [33]	1.6 %

Table 1: results for 1 hidden layer with 800 neurons

Finally, a natural suggestion that emerges from our work is to add $\beta > 0$ as a parameter for each individual neuron in the network, and not just to the loss function. Namely, to translate the input to a ReLu neuron by β. The value of β may change during the learning process. Figuratively, this can be thought of as "internal clocks" of the neurons.

Generalization Bounds

An important aspect of a learning algorithm is its generalization capabilities; namely, its error on new examples it was not trained with (see the textbook [31] for background and definitions). One typically formalizes it by assuming that the input sample consists of i.i.d. examples drawn from an unknown distribution D on $\mathbb{R}^d$ that are labelled by some unknown function $c : \mathbb{R}^d \to \{\pm 1\}$. The algorithm is said to generalize if it outputs an hypothesis $h : \mathbb{R}^d \to \{\pm 1\}$ so that $\Pr_D[h \neq c]$ is as small as possible.

We focus on the case that c is linearly separable. A natural choice for h in this case is given by hard-SVM; namely, the halfspace with maximum margin on the input sample. It is known that if D is supported on points that are γ-far from some hyperplane then the hard-SVM choice generalizes well (see Theorem 15.4 in [31]). The proof of this property of hard-SVMs uses Radamacher complexity.

We suggest that using the perceptron algorithm, instead of the hard-SVM solution, yields a more general statement with a simpler proof. The reason is that the perceptron naturally performs compression. Similar compression-based generalization bounds for the Perceptron and other mistake-bound based algorithms were given in [17].

Theorem 1.1. *Let D be a distribution on $\mathbb{R}^d$. Let $c : \mathbb{R}^d \to \{\pm 1\}$. Let $x_1, \ldots, x_m$ be i.i.d. samples from D. Let $S = ((x_1, c(x_1)), \ldots, (x_m, c(x_m))$. If*

$$\Pr_S [\text{marg}(S) < \varepsilon] < \delta/2 \tag{1}$$

for some $\varepsilon, \delta > 0$, then

$$\Pr_S \left[\mathbb{P}_D[\pi(S) \neq c] \leq 50 \frac{\log(\varepsilon^2 m) + \log(2/\delta)}{\varepsilon^2 m} \right] \geq 1 - \delta$$

where π is the perceptron algorithm.

The theorem can also be interpreted of as a local-to-global statement in the following sense. Assume that we know nothing of c, but we get a list of m samples that are linearly separable with significant margin (this is a local condition that we can empirically verify). Then we can deduce that c is close to being linearly separable. The perceptron's compression allows to deduce more general local-to-global statements, like bounding the global margin via the local/empirical margins (this is related to [30]).

Condition (1) holds when the expected value of one over the margin is bounded from above (and may hold when c is not linearly separable). This assumption is weaker than the assumption in [31] on the behavior of hard-SVMs (that the margin is always bounded from below).

The proof of the theorem is based on viewing the perceptron as a sample compression scheme, see Section 4 for more details. It highlights the importance of compression in this context as well.

Dimension Reduction

Here we follow the theme of robust concepts presented by Arriaga and Vempala [3]. Let $\mathcal{X} \subset \mathbb{R}^d$ be of size n so that $\max_{x \in \mathcal{X}} \|x\| = 1$. Think of $\mathcal{X}$ as representing a collection of high resolution images. As in many learning scenarios, some assumptions on the learning problem should be made in order to make it accessible. A typical assumption is that the unknown function to be learnt belongs to some specific class of functions. Here we focus on the class of all ε-separated partitions of $\mathcal{X}$; these are functions $Y : \mathcal{X} \to \{\pm 1\}$ that are linearly separable with margin at least ε. Such partitions are called robust concepts in [3] and correspond to "easy" classification problems.

Arriaga and Vempala demonstrated the difference between robust concepts and non-robust concept with the following analogy; it is much easier to distinguish between "Elephant" and "Dog" than between "African Elephant" and "Indian Elephant." They proved that random projections can help to perform efficient dimension reduction for ε-separated learning problems (and more general examples). They also described "neuronal" devices for performing it, and discussed their advantages. Similar dimension reductions were used in several other works in learning e.g. [14, 15, 4, 20, 6].

We observe that the perceptron's compression allows to deduce a *simultaneous* dimension reduction. Namely, the dimension reduction works simultaneously for the entire class of robust concepts. This follows from results in Ben-David et al. [5], who studied limitations of embedding learning problems in linearly separated classes.

We now explain this in more detail. The first step in the proof is the following theorem (proved in [5]).

Theorem 1.2. *The number of ε-separated partitions of $\mathcal{X}$ is at most $(2(n+1))^{1/\varepsilon^2}$.*

The theorem is sharp in the following sense.

Example 1.3. *Let $e_1, \ldots, e_n \in \mathbb{R}^n$ be the n standard unit vectors. Every subset of the form $(e_i)_{i\in I}$ for $I \subset [n]$ of size k is $\Omega(1/\sqrt{k})$-separated, and there are $\binom{n}{k}$ such subsets.*

The example also allows to lower bound the number of updates of any perceptron-like algorithm. If there is an algorithm that given $Y : \mathcal{X} \to \{\pm 1\}$ of margin ε is able to find w so that $Y(x) = \mathrm{sign}(w \bullet x)$ for $x \in \mathcal{X}$ that can be described by at most K of the points in $\mathcal{X}$ then K should be at least $\Omega(1/\varepsilon^2)$.

The upper bound in the theorem allows to perform dimension reduction that simultaneously works well on the entire concept class. Let A be a $k \times d$ matrix with i.i.d. entries that are normally distributed $(N(0,1))$[4] with $k \geq C \log(n/\delta)/\varepsilon^4$ where $C > 0$ is an absolute constant. Given A, we can consider

$$A\mathcal{X} = \{Ax : x \in \mathcal{X}\} \subset \mathbb{R}^k$$

in a potentially smaller dimension space. The map $x \mapsto Ax$ is almost surely one-to-one on $\mathcal{X}$. So, every subset of $\mathcal{X}$ corresponds to a subset of $A\mathcal{X}$ and vice versa. The following theorem shows that it preserves all well-separated partitions.

Theorem 1.4. *With probability of at least $1-\delta$ over the choice of A, all ε-partitions of $\mathcal{X}$ are $\varepsilon/2$-partitions of $A\mathcal{X}$ and all $\varepsilon/2$-partitions of $A\mathcal{X}$ are $\varepsilon/4$-partitions of $\mathcal{X}$.*

[4]Other distributions will work just as well.

The proof of the above theorem (which is implicit in [5]) is a simple application of Theorem 1.2 together with the Johnson-Lindenstrauss lemma [18]. For completeness, it appears in Section 5.

Convex Separation and Games

Linear programming (LP) is a central paradigm in computer science and mathematics. LP duality is a key ingredient in many algorithms and proofs, and is deeply related to von Neumann's minimax theorem that is seminal in game theory [27]. Two related and fundamental geometric properties are Farkas' lemma [12], and the following separation theorem.

Theorem 1.5. *For every non empty convex sets $K, L \subset \mathbb{R}^d$, precisely one of the following holds: (i) $\mathsf{dist}(K, L) = \inf\{\|p - q\| : p \in K, q \in L\} = 0$, or (ii) there is a hyperplane separating K and L.*

We observe that the following stronger version of the separation theorem follows from the perceptron's compression (a similar version of Farkas' lemma can be deduced as well).

Lemma 1.6 (Sparse Separation)**.** *For every non empty convex sets $K, L \subset \mathbb{R}^d$ so that $\sup\{\|p - q\| : p \in K, q \in L\} = 1$ and every $\varepsilon > 0$, one of the following holds:*

(i) $\mathsf{dist}(K, L) < \varepsilon$.

(ii) There is a hyperplane $H = \{x : w \bullet x = b\}$ separating K from L so that its normal vector is "sparse":

- $\frac{w \bullet p - b}{\|w\|} > \frac{\varepsilon}{30}$ *for all* $p \in K$,
- $\frac{w \bullet q - b}{\|w\|} < -\frac{\varepsilon}{30}$ *for all* $q \in L$, *and*
- w *is a sum of at most* $(10/\varepsilon)^2$ *points in* K *and* $-L$.

The lemma is strictly stronger than the preceding separation theorem. It is proved in Section 3 below, where we also explain how this perspective yields an alternative proof of Novikoff's theorem on the convergence of the perceptron [28]. It is interesting to note that the usual proof of the separation theorem relies on a concrete construction of the separating hyperplane that is geometrically similar to hard-SVMs. The proof using the perceptron, however, does not include any "geometric construction" and yields a sparse and strong separator (it also holds in infinite dimensional Hilbert space, but it uses that the sets are bounded in norm).

These ideas have a game theoretic interpretation as well. In the associated game there are two players. A Point player whose pure strategies are points v in some finite set $V \subset \mathbb{R}^d$ so that $\max\{\|v\| : v \in V\} = 1$, and a Hyperplane player whose pure strategies are w for $w \in \mathbb{R}^d$ with $\|w\| = 1$. For a given choice of v and w, the Hyperplane player's payoff is of $P(v, w) = v \bullet w$ coins (if this number is negative, then the Hyperplane player pays the Point player). The goal of the Point player is thus to minimize the amount of coins she pays. A mixed strategy of the Point player is a distribution μ on V, and of the Hyperplane player is a (finitely supported) distribution κ on $\{w : \|w\| = 1\}$. The expected gain is

$$P(\mu, \kappa) = \mathop{\mathbb{E}}_{(v,w)\sim\mu\times\kappa} P(v, w).$$

Claim 1.7 (Sparse Strategies)**.** *Let ε^* be the minimax value of the game:*

$$\varepsilon^* = \sup_{\kappa} \inf_{\mu} P(\mu, \kappa) \geq 0.$$

There is $T \geq \frac{1}{3(\varepsilon^*)^2}$ (if $\varepsilon^* = 0$ then $T = \infty$) and a sequence of mixed strategies $\mu_1, \mu_2, \ldots, \mu_T$ of the Point player so that for all $t \leq T$, the support size of μ_t is at most t and for every mixed strategy κ of the Hyperplane player,

$$P(\mu_t, \kappa) \leq \sqrt{3/t}.$$

The last strategy in the sequence $\mu_1, \mu_2, \ldots$ guarantees the Point player a loss of at most $3\varepsilon^*$. This sequence is naturally and efficiently generated by the perceptron algorithm and produces a strategy for the Point player that is optimal up to a constant factor (see Section 3). The ideas presented in Section 2 allow to reduce the constant 3 to as close to 1 as we want, by paying in running time (see Algorithm 4).

2 Training

In this section we explore a couple of methods for training perceptrons. Part of the motivation is suggesting ideas for training other models, like neural networks.

The standard analysis of the perceptron convergence properties uses the optimal separating hyperplane (later in Section 3 we present an alternative analysis that does not use it): let

$$w^* = \mathop{\mathrm{argmax}}_{w\in\mathbb{R}^d:\|w\|=1} \mathrm{marg}(w, S),$$

where we think of S as the map from $\{x_1, \ldots, x_m\}$ to $\{\pm 1\}$ defined by $x_i \mapsto y_i$.[5] Novikov's analysis consists of the following two parts. Let $\varepsilon^* = \text{marg}(w^*, S)$ and $R = \max_i \|x_i\|$.

Part I: The projection grows linearly in time. In each iteration, the projection of $w^{(t)}$ on w^* grows by at least ε^*, since $y_i x_i \cdot w^* \geq \varepsilon^*$. By induction, we get $w^{(t)} \cdot w^* \geq \varepsilon^* t$ for all $t \geq 0$.

Part II: The norm grows sub-linearly in time. In each iteration,

$$\|w^{(t)}\|^2 = \|w^{(t-1)}\|^2 + 2y_i x_i \cdot w^{(t-1)} + \|x_i\|^2 \leq \|w^{(t-1)}\|^2 + R^2$$

(the term $2y_i x_i \cdot w^{(t-1)}$ is negative by choice). So by induction $\|w^{(t)}\| \leq R\sqrt{t}$ for all t.

Combining the two parts,

$$1 \geq \frac{w^{(t)} \cdot w^*}{\|w^{(t)}\|\|w^*\|} \geq \frac{\varepsilon^*}{R}\sqrt{t},$$

which implies that the number of iterations of the algorithm is at most $(R/\varepsilon^*)^2$.

This analysis provides no guarantees on the margin of the hyperplane the perceptron outputs (and it is not hard to construct examples where this margin is arbitrarily small). We suggest two different modifications of the algorithm that are not too costly in terms of the running time and guarantee near optimal margin. We start with an algorithm that achieves a constant approximation of the optimal margin ε^*; this version is called the aggressive perceptron [10].

initialize: $w^{(0)} = \vec{0}$ and $t = 0$
while $\exists i$ *with* $y_i w^{(t)} \cdot x_i < \beta$ **do**
  $w^{(t+1)} = w^{(t)} + y_i x_i$
  $t = t + 1$
end
return $w^{(t)}$

Algorithm 2: The aggressive perceptron algorithm

We only replaced that < 0 condition in the while loop by a $< \beta$ condition, for some $\beta > 0$. As before, by induction

$$\|w^{(t)}\|^2 = \|w^{(t-1)}\|^2 + 2y_i x_i \cdot w^{(t-1)} + \|x_i\|^2 \leq (2\beta + R^2)t$$

[5] We assume that S is consistent with a function (does not contain identical points with opposite labels).

and

$$1 \geq \frac{w^{(t)} \bullet w^*}{\|w^{(t)}\|\|w^*\|} \geq \frac{\varepsilon^*}{\sqrt{2\beta + R^2}}\sqrt{t}$$

where $R = \max_i \|x_i\|$. The number of iterations is thus at most $\frac{2\beta+R^2}{(\varepsilon^*)^2}$. In addition, by choice, for all i,

$$y_i w^{(t)} \bullet x_i \geq \beta.$$

So, since

$$\|w^{(t)}\| \leq \sqrt{(2\beta + R^2)t} \leq \frac{2\beta + R^2}{\varepsilon^*},$$

we get

$$\text{marg}(w^{(t)}, S) \geq \frac{\varepsilon^* \beta}{2\beta + R^2}.$$

Specifically, by knowing the maximal norm of the examples we can achieve the true margin up to a multiplicative constant. For example, we can take $\beta = R^2$, and get margin at least $\varepsilon^*/3$ with running time at most $3R^2/(\varepsilon^*)^2$.

It is possible to do even better than that. We do not need to know in advance the maximal norm of the examples. We can adaptively change the parameter β according to example received.

initialize: $w^{(0)} = \vec{0}$ and $t = 0$ and $\beta = 0$
while $\exists i$ *with* $y_i w^{(t)} \bullet x_i \leq \beta$ **do**
  $w^{(t+1)} = w^{(t)} + y_i x_i$
  $t = t + 1$
  if $\beta < \|x_i\|^2$ **then**
    $\beta = 4\|x_i\|^2$
  end
end
return $w^{(t)}$

Algorithm 3: The R-independent perceptron algorithm

This version of the algorithm guarantees a margin of $\varepsilon^*/3$ coupled with a running time comparable to the original algorithm *without* knowing R. Indeed, to bound the running time observe that between every a change in β occurs, as before, there could be at most $\frac{2\beta+R^2}{\varepsilon^2}$ errors (for the relevant β and R). The amount of changes in β is at

most $\lceil \log(R/r) \rceil$, where $r = \min_i \|x_i\|$. The overall running time is at most

$$\sum_{k=1}^{\lceil \log(R/r) \rceil} \frac{2 \bullet 4 \left|x_{i_k}\right|^2 + (2 \left|x_{i_k}\right|)^2}{(\varepsilon^*)^2} \leq 2 \bullet \sum_{k=1}^{\lceil \log(R/r) \rceil} \frac{3 \bullet 4^k r^2}{\varepsilon^2}$$
$$\leq 6 \bullet 4/3 \bullet 4^{\lceil \log(R/r) \rceil} \frac{r^2}{\varepsilon^2} = O((R/\varepsilon^*)^2).$$

The third and last version we propose, reaches a margin that is as close to the optimum as we wish. The algorithm depends on a parameter $1 < \alpha < 2$ that determines the guarantee on the margin of its output. The closer α is to 2, the larger the margin and the running time are. For simplicity, we assume here that $R = \max_i \|x_i\| = 1$. The idea is as follows. The analysis of the classical perceptron relies on the fact that $\|w^{(t)}\|^2 \leq t$ in each step. On the other hand, in an "extremely aggressive" version of the perceptron that always updates, one can only obtain a trivial bound $\|w^{(t)}\|^2 \leq t^2$ (as $w^{(t)}$ can be the sum of t unit vectors in the same direction). The update rule in the version below is tailored so that a bound of $\|w^{(t)}\|^2 \leq t^\alpha$ for $\alpha \in (1, 2)$ is maintained.

initialize: $w^{(0)} = \vec{0}$ and $t = 0$ and $\beta = 0$
while $\exists i$ *with* $y_i w^{(t)} \bullet x_i \leq \beta$ **do**
  $w^{(t+1)} = w^{(t)} + y_i x_i$
  $t = t + 1$
  $\beta = 0.5((t+1)^\alpha - t^\alpha - 1)$
end
return $w^{(t)}$

Algorithm 4: The ∞-perceptron algorithm

Here we use that for $t \geq 2$,

$$\|w^{(t)}\|^2 \leq \|w^{(t-1)}\|^2 + (t^\alpha - (t-1)^\alpha - 1) + \|x_i\|^2.$$

By induction, for all $t \geq 0$,

$$\|w^{(t)}\|^2 \leq t^\alpha.$$

This time

$$1 \geq \frac{w^{(t)} \bullet w^*}{\|w^{(t)}\| \|w^*\|} \geq \frac{\varepsilon^* t}{t^{\alpha/2}}.$$

So, the running time is at most $(1/\varepsilon^*)^{2/(2-\alpha)}$.

The output's margin is at least

$$\frac{0.5((t+1)^\alpha - t^\alpha - 1)}{t^{\alpha/2}}. \tag{2}$$

This is decreasing function for $t > 0$, since its derivative is at most zero (see Appendix A). So, by plugging in the upper bound on t, since $(t+1)^\alpha - t^\alpha \geq \alpha t^{\alpha-1}$ for $t \geq 0$, the output's margin is at least

$$0.5\alpha \frac{(1/\varepsilon^*)^{2(\alpha-1)/(2-\alpha)} - 1}{(1/\varepsilon^*)^{\alpha/(2-\alpha)}} = 0.5\alpha\varepsilon^* - (\varepsilon^*)^{\alpha/(2-\alpha)}.$$

So we can get arbitrarily close to the true margin by setting $\alpha = 2(1-\delta)$ for some small $0 < \delta < 0.5$ of our choice. This gives margin

$$(1-\delta)\varepsilon^* - (\varepsilon^*)^{(2-\delta)/\delta} \geq \varepsilon^*\big(1 - \delta - (\varepsilon^*)^{1/\delta}\big).$$

The running time, however, becomes $(1/\varepsilon^*)^{1/\delta}$.

When ε^* is very close to 1, the lower bound on the margin above may not be meaningful. We claim that the margin of the output is still close to ε^* even in this case. To see this, let $\tilde{w}$ be a hyperplane with margin $\tilde{\varepsilon} = (1 - \delta\ln(1/\delta))\varepsilon^*$. We can carry the argument above with $\tilde{w}$ instead of w^*, and get that the margin is at least

$$\tilde{\varepsilon}\big(1 - \delta - (\tilde{\varepsilon})^{1/\delta}\big) > (1 - 2\delta - \delta\ln(1/\delta))\varepsilon^*.$$

So we can choose δ small enough, without knowing any information on ε^*, and get an almost optimal margin.

This bound on the running time is sharp, as the following example shows.

Example 2.1. *Let $\mathcal{X} = \{\big((\sqrt{1-\varepsilon^2}, \varepsilon), 1\big), \big((-\sqrt{1-\varepsilon^2}, \varepsilon), 1\big)\}$. This two points are linearly separated with margin $\Omega(\varepsilon)$. The algorithm stops after $\Omega\Big((1/\varepsilon)^{2/(2-\alpha)}\Big)$ iterations (if ε is small enough and α close enough to 2).*

3 Convex Separation and Games

Convex Separation

We start by proving the sparse separation lemma (Lemma 1.6). Let K, L be convex sets and $\varepsilon > 0$. For $x \in \mathbb{R}^d$, let $\tilde{x}$ in $\mathbb{R}^{d+1}$ be the same as x in the first d coordinates and 1 in the last (we have $\|\tilde{x}\| \leq \|x\| + 1$). We thus get two convex bodies $\tilde{K}$ and $\tilde{L}$ in $d+1$ dimensions (using the map $x \mapsto \tilde{x}$).

Run Algorithm 2 with $\beta = 1$ on inputs that positively label $\tilde{K}$ and negatively label $\tilde{L}$. This produces a sequence of vectors $w^{(0)}, w^{(1)}, \ldots$ so that $\|w^{(t)}\| \leq \sqrt{6t}$ for all t. For

every $t > 0$, the vector $w^{(t)}$ is of the form $w^{(t)} = k^{(t)} - \ell^{(t)}$ where $k^{(t)}$ is a sum of t_1 elements of $\tilde{K}$ and $\ell^{(t)}$ is a sum of t_2 elements of $\tilde{L}$ so that $t_1 + t_2 = t$. In particular, we can write $\frac{1}{t}w^{(t)} = \alpha^{(t)}p^{(t)} - (1-\alpha^{(t)})q^{(t)}$ for $\alpha^{(t)} \in [0,1]$ where $p^{(t)} \in \tilde{K}$ and $q^{(t)} \in \tilde{L}$ (note that the last coordinate of $w^{(t)}$ equals $2\alpha^{(t)} - \frac{1}{2}$).

If the algorithm does not terminate after T steps for T satisfying $\sqrt{6/T} < \varepsilon/4$ then it follows that $\|\frac{1}{T}w^{(T)}\| < \varepsilon/4$. In particular, $|\alpha^{(T)} - 1/2| < \varepsilon/8$ and so

$$\frac{\varepsilon}{4} > \|\alpha^{(t)}p^{(t)} - (1-\alpha^{(t)})q^{(t)}\| > \frac{\|p^{(t)} - q^{(t)}\|}{2} - \frac{\varepsilon}{4},$$

which implies that $\mathsf{dist}(K, L) < \varepsilon$.

In the complementing case, the algorithm stops after $T < (10/\varepsilon)^2$ rounds. Let w be the first d coordinates of $w^{(T)}$ and b be its last coordinate. For all $p \in K$,

$$\frac{w \bullet p + b}{\|w\|} \geq \frac{1}{\|w^{(T)}\|} \geq \frac{1}{\sqrt{6T}} > \frac{\varepsilon}{30}.$$

Similarly, for all $q \in L$ we get $\frac{w \bullet q + b}{\|w\|} < -\frac{\varepsilon}{30}$.

Alternative Proof of the Perceptron's Convergence

A similar argument provides a different proof of Novikoff's theorem on the convergence of the perceptron [28]. Assume without loss of generality that all of examples are labelled positively (by replacing x by $-x$ if necessary). Also assume that $R = \max_i \|x_i\| = 1$. As in the proof above, let $w^{(0)}, w^{(1)}, \ldots$ be the sequence of vectors generated by the perceptron (Algorithm 1). Instead of arguing that the projection on w^* grows linearly with t, argue as follows. The vectors $v^{(1)}, v^{(2)}, \ldots$ defined by $v^{(t)} = \frac{1}{t}w^{(t)}$ are in the convex hull of the examples and have norm at most $\|v^{(t)}\| \leq \frac{1}{\sqrt{t}}$. Specifically, for every w of norm 1 we have $v^{(t)} \bullet w \leq \frac{1}{\sqrt{t}}$ and so there is an example x so that $x \bullet w \leq \frac{1}{\sqrt{t}}$. This implies that the running time T satisfies $\frac{1}{\sqrt{T}} \geq \varepsilon^*$ since for every example x we have $x \bullet w^* \geq \varepsilon^*$.

Game Theory

We now move to the game theoretic perspective (we prove Claim 1.7). Let $v^{(t)} = \frac{1}{t}w^{(t)}/t$ be as in the proof of Lemma 1.6 above, when we replace K by V and L by $\emptyset$. We

can interpret $v^{(t)}$ as a mixed-strategy μ_t of the Point player (the uniform distribution over some multi-subset of V of size t). Specifically, for every κ and $t > 0$,

$$P(\mu_t, \kappa) = \mathop{\mathbb{E}}_{w \sim \kappa} v^{(t)} \bullet w \leq \|v^{(t)}\| \leq \sqrt{3/t}.$$

Denote by T the stopping time. If $T = \infty$ then indeed $P(\mu_t, \kappa)$ tends to zero as $t \to \infty$. If $T < \infty$, we have $v \bullet v^{(T)} \geq \frac{1}{T}$ for all $v \in V$. We can interpret $v^{(T)}$ as a non trivial strategy for the Hyperplane player: let

$$\tilde{w} = \frac{v^{(T)}}{\|v^{(T)}\|}.$$

Thus, for every μ,

$$P(\mu, \tilde{w}) \geq \frac{1}{T\|v^{(T)}\|} \geq \frac{1}{\sqrt{3T}}.$$

In particular, $\varepsilon^* \geq \frac{1}{\sqrt{3T}}$ and so

$$T \geq \frac{1}{3(\varepsilon^*)^2}.$$

4 Generalization Bounds

Here we analyze the generalization capabilities of the perceptron. In a nutshell, these capabilities stem from the compression it performs. To formally explain this, we need the notions of sample compression schemes [24] and selection schemes [11], which correspond to learning algorithms whose output hypothesis is determined by a small subsample of the input.

Definition (Selection schemes)**.** *A selection scheme of size d consists of a compression map κ and a reconstruction map ρ such that for every input sample S:*

- *κ maps S to a sub-sample of S of size at most d.*
- *ρ maps $\kappa(S)$ to a hypothesis $\rho(\kappa(S)) : \mathcal{X} \to \{\pm 1\}$; this is the output of the learning algorithm induced by the selection scheme.*

Following Littlestone and Warmuth, David et al. showed that every selection scheme does not overfit its data [11]: Let (κ, ρ) be a selection scheme of size d. Let S be a sample of m independent examples from an arbitrary distribution D that are labelled by some fixed concept c, and let $K(S) = \rho(\kappa(S))$ be the output of the selection scheme. For a hypothesis h, let $L_D(h) = \Pr_D[h \neq c]$ denote the (true) error of h and $L_S(h) = \frac{1}{m}\sum_{i=1}^{m} 1_{h(x_i)=c(x_i)}$ denote the empirical error of h.

Theorem 4.1. *For every $\delta > 0$,*

$$\Pr_S \left[|L_D(K(S)) - L_S(K(S))| \geq \sqrt{\varepsilon \cdot L_S(K(S))} + \varepsilon \right] \leq \delta,$$

where

$$\varepsilon = 50 \frac{d \log(m/d) + \log(1/\delta)}{m}.$$

Proof of Theorem 1.1. Consider the following selection scheme of size $1/\varepsilon^2$ that agrees with the perceptron on samples with margin at least ε: If the input sample S has $\text{marg}(S) \geq \varepsilon$, apply the Perceptron (which gives a compression of size $1/\varepsilon^2$). Else, compress it to the emptyset and reconstruct it to some dummy hypothesis. The theorem now follows by applying Theorem 4.1 on this selection scheme and by the assumption that that $\text{marg}(S) \geq \varepsilon$ for $1 - \delta/2$ of the space (note that $L_S(K(S)) = 0$ when $\text{marg}(S) \geq \varepsilon$). □

5 Dimension Reduction

Following [5], this section uses the perceptron to count the collection of linearly separable partitions over a given set of points. This allows to perform simultaneous efficient dimension reduction for the entire family of well-separated halfspaces. We suggest that it may sometimes be useful to include such dimension reductions in other learning algorithms. One example we are aware of that is similar in spirit is given in [1], where the random initialization of a neural network is typically useful.

The perceptron algorithm allows us to bound the amount of possible ε-partitions on a given set.

Proof of Theorem 1.2. Given an ε-partition of the set $\mathcal{X}$, the perceptron algorithm finds a separating hyperplane after making at most $1/\varepsilon^2$ updates. It follows that every ε-partition can be represented by a multiset of $\mathcal{X}$ together with the corresponding signs. The total number of options is at most $(n+1)^{1/\varepsilon^2} \cdot 2^{1/\varepsilon^2}$. □

We now review a well known lemma by Johnson and Lindenstrauss [18]; see also [25] and references within. The lemma states that a random projection approximately preserves norms and inner products (it is proved by a couple of applications of the union bound together with analyzing the case of a single vector).

Lemma 5.1. *Let $x_1, ..., x_N \in \mathbb{R}^d$ with $\|x_i\| \leq 1$ for all $i \in [N]$. Then, for every $\varepsilon > 0$ and $0 < \delta < 1/2$,*

$$\mathbb{P}\Big[\exists i, j \in [N] \;\; |(Ax_i \bullet Ax_j) - (x_i \bullet x_j)| > \varepsilon\Big] < \delta,$$

where $k = O(\log(N/\delta)/\varepsilon^2)$ and A is a $k \times d$ matrix with i.i.d. entries that are $N(0,1)$.

We can now prove the dimension reduction.

Proof of Theorem 1.4. Every linear ε-partition can be represented by a unit vector (the normal to the hyperplane). Pick such a vector for every ε-partition. By Theorem 1.2 the total number of vectors picked is at most $(2(n+1))^{1/\varepsilon^2}$. Now, the previous lemma implies the desired result, as it suffices to preserve the $n(2(n+1)^{1/\varepsilon^2})$ inner products between the picked vectors and the n points in $\mathcal{X}$ up to accuracy $\varepsilon/2$. □

References

[1] A. Andoni, R. Panigrahy, G. Valiant and L. Zhang. Learning Polynomials with Neural Networks. *PMLR* 32(2), pages 1908–1916, 2014.

[2] J. K. Anlauf and M. Biehl. The AdaTron: An Adaptive Perceptron Algorithm. *EPL*, 1989.

[3] R.I. Arriaga and S. Vempala. An algorithmic theory of learning: Robust concepts and random projection. *Machine Learning*, 63(2), pages 161–182, 2006.

[4] N. Balcan, A. Blum and S.Vempala. On Kernels, Margins and Low-dimensional mappings. In *ALT* 2004.

[5] S. Ben-David, N. Eiron and H. U. Simon. Limitations of Learning Via Embeddings in Euclidean Half Spaces. In *JMLR* 2002.

[6] A. Blum and R. Kannan. Learning an intersection of k halfspaces over a uniform distribution. In *FOCS*, 1993.

[7] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT* , pages 144-152, 1992.

[8] Nicol'o Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9), pages 2050–2057, 2004.

[9] R. Collobert and S. Bengio. Links between perceptrons, MLPs and SVMs. *IDIAP*, 2004.

[10] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research* 7, pages 551–585, 2006.

[11] O. David, S. Moran and A. Yehudayoff. Supervised learning through the lens of compression. In *NIPS*, pages 2784-2792, 2016.

[12] G. Farkas. Uber die Theorie der Einfachen Ungleichungen. *Journal fur die Reine und Angewandte Mathematik*, 124 (124), pages 1–27, 1902.

[13] Y. Freund and R. E. Schapire. Large Margin Classification Using the Perceptron Algorithm. *Machine Learning*, pages 277-296, 1999.

[14] A. Garg, S. Har-Peled and D. Roth. On generalization bounds, projection profile, and margin distribution. In *ICML*, pages 171–178, 2002.

[15] A. Garg and D. Roth. Margin Distribution and Learning. In *ICML*, pages 210–217, 2003.

[16] C. Gentile. A New Approximate Maximal Margin Classification Algorithm. *Journal of Machine Learning Research*, pages 213-242, 2001.

[17] T. Graepel, R. Herbrich and J. Shawe-Taylor. PAC-Bayesian Compression Bounds on the Prediction Error of Learning Algorithms for Classification. *Machine Learning*, pages 55-76, 2005.

[18] W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Conference in modern analysis and probability*, 1982.

[19] R. Khardon and G. Wachman. Noise Tolerant Variants of the Perceptron Algorithm. *Journal of Machine Learning Research*, pages 227-248 , 2007.

[20] A. Klivans and R. Servedio. Learning intersections of halfspaces with a margin. In *Workshop on Computational Learning Theory*, 2004.

[21] M. Korzen and K. Klesk. Maximal Margin Estimation with Perceptron-Like Algorithm. In *ICAISC*, 2008.

[22] W. Krauth and M. Mézard. Learning algorithms with optimal stablilty in neural networks. *J. Phys. A: Math. Gen.*, 1987.

[23] Y. LeCun and C. Cortes. The MNIST database of handwritten digits. 1998.

[24] N. Littlestone and M. Warmuth. Relating data compression and learnability. *Unpublished*, 1986.

[25] J. Matousek. On variants of the Johnson–Lindenstrauss lemma. Random Structures & Algorithms, 33(2), pages 142–156, 2008.

[26] M. Mohri and A. Rostamizadeh. Perceptron Mistake Bounds. *arXiv:1305.0208*.

[27] J. von Neumann. Zur Theorie der Gesellschaftsspiele. Math. Ann. 100, pages 295–320, 1928.

[28] Albert B.J. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622, 1962.

[29] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), pages 386–408, 1958.

[30] R.E. Schapire, Y. Freund, P. Bartlett and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5), pages 1651–1686, 1998.

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

[32] S. Shalev-Shwartz, Y. Singer, N. Srebro and A. Cotter. Pegasos: Primal estimated sub-gradient solver for SVM. Mathematical programming 127, no. 1, pages 3-30, 2011.

[33] P. Y. Simard, D. Steinkraus and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR* 3, pages 958–962, 2003.

[34] D. Soudry, E. Hoffer and N. Srebro. The Implicit Bias of Gradient Descent on Separable Data. *arXiv:1710.10345*, 2017.

[35] A. Wendemuth. Learning the unlearnable. *J. Phys. A: Math. Gen.*, 1995.

A The derivative of the margin

Here we prove that the derivative of (2) is at most zero. The numerator of the derivative is 0.5 times

$$\begin{aligned}
&(\alpha(t+1)^{\alpha-1} - \alpha t^{\alpha-1})t^{\alpha/2} - \frac{\alpha}{2}t^{(\alpha-2)/2}((t+1)^{\alpha} - t^{\alpha} - 1)) \\
&= \frac{\alpha}{2}t^{(\alpha-2)/2}(2t(t+1)^{\alpha-1} - 2t^{\alpha}) + \frac{\alpha}{2}t^{(\alpha-2)/2}(-(t+1)^{\alpha} + t^{\alpha} + 1)) \\
&= \frac{\alpha t^{(\alpha-2)/2}}{2}\left((t+1)^{\alpha-1}(t-1) - t^{\alpha} + 1\right).
\end{aligned}$$

At $t = 1$, we get the value 0, so it suffices to prove that $(t+1)^{\alpha-1}(t-1) - t^{\alpha} + 1$ is a non increasing function for $t \geq 1$. Indeed, the derivative of the term inside the parenthesis is

$$\begin{aligned}
&(\alpha-1)(t+1)^{\alpha-2}(t-1) + (t+1)^{\alpha-1} - \alpha t^{\alpha-1} \\
&= (\alpha-1)\left(\frac{t-1}{(t+1)^{2-\alpha}} - t^{\alpha-1}\right) + (t+1)^{\alpha-1} - t^{\alpha-1} \\
&\leq (\alpha-1)\left(\frac{t-1}{(t+1)^{2-\alpha}} - t^{\alpha-1}\right) + (\alpha-1)t^{\alpha-2} \qquad (\alpha < 2) \\
&\leq (\alpha-1)\left(\frac{t-1}{t^{2-\alpha}} - t^{\alpha-1} + \frac{1}{t^{2-\alpha}}\right) = 0.
\end{aligned}$$