On the Perceptron's Compression

Shay Moran[1], Ido Nachum[2], Itai Panasoff[3], and Amir Yehudayoff[3(✉)]

[1] Google Brain, Princeton, NJ, USA
[2] EPFL, Lausanne, Switzerland
[3] Department of Mathematics, Technion-IIT, Haifa, Israel
`amir.yehudayoff@gmail.com`

Abstract. We study and provide exposition to several phenomena that
are related to the perceptron's compression. One theme concerns modi-
fications of the perceptron algorithm that yield better guarantees on the
margin of the hyperplane it outputs. These modifications can be useful
in training neural networks as well, and we demonstrate them with some
experimental data. In a second theme, we deduce conclusions from the
perceptron's compression in various contexts.

Keywords: Machine learning · Compression · Convex separation

1 Introduction

The perceptron is an abstraction of a biological neuron that was introduced in
the 1950's by Rosenblatt [31], and has been extensively studied in many works
(see e.g. the survey [27]). 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).

Formally, a perceptron computes a function of the form $\mathsf{sign}(w \cdot x - b)$ where
$w \in \mathbb{R}^d$ is the weight vector, $b \in \mathbb{R}$ is the threshold, $\cdot$ is the standard inner product,
and $\mathsf{sign} : \mathbb{R} \rightarrow \{\pm 1\}$ is 1 on the non-negative numbers. It is only capable of
representing binary functions that are induced by partitions of $\mathbb{R}^d$ by hyperplanes.

Definition 1. *A map* $Y : \mathcal{X} \rightarrow \{\pm 1\}$ *over a finite set* $\mathcal{X} \subset \mathbb{R}^d$ *is (lin-
early)[1] separable if there exists* $w \in \mathbb{R}^d$ *such that* $\mathsf{sign}(w \cdot x) = Y(x)$ *for all*

---

[1] We focus on the linear case, when the threshold is 0. A standard lifting that adds a coor-
dinate 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 ideas in Algorithm 3 below.

$x \in \mathcal{X}$. When the Euclidean norm of w is $\|w\| = 1$, the number $\mathsf{marg}(w, Y) = \min_{x \in \mathcal{X}} Y(x) w \cdot x$ is the margin of w with respect to Y. The number $\mathsf{marg}(Y) = \sup_{w \in \mathbb{R}^d : \|w\| = 1} \mathsf{marg}(w, Y)$ 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 $\mathsf{ReLu}(z) = \max\{0, z\}$). Therefore, studying the perceptron and its variants may help in understanding neural networks, their design and their training process.

Overview. In this paper, we provide some insights into the perceptron's behavior, survey some of the related work, deduce some geometric applications, and discuss their usefulness in other learning contexts. Below is a summary of our results and a discussion of related work, partitioned to five parts numbered (i) to (v). Each of the results we describe highlights a different aspect of the perceptron's compression (the perceptron's output is a sum of small subset of examples). For more details, definitions and references, see the relevant sections.

(i) Variants of the perceptron (Sect. 2). The well-known perceptron algorithm (see Algorithm 1 below) is guaranteed to find a separating hyperplane in the linearly separable case. However, there is no guarantee on the hyperplane's margin compared to the optimal margin ε^*. This problem was already addressed in several works, as we now explain (see also references within). The authors of [9] and [23] defined a variant of the perceptron that yields a margin of the form $\Omega(\varepsilon^*/R^2)$; see Algorithm 2 below. The authors of [10] defined the passive-aggressive perceptron algorithms that allow e.g. to deal with noise, but provided no guarantee on the margin of the output. The authors of [22] defined a variant of the perceptron that yields provable margin under the assumption that a lower bound on the optimal margin is known. The author of [17] designed the ALMA algorithm and showed that it provides almost optimal margin under the assumption that the samples lie on the unit sphere. It is worth noting that normalizing the examples to be on the unit sphere may significantly alter the margin, and even change the optimal separating hyperplane. The author of [37] defined the minimal overlap algorithm which guarantees optimal margin but is not online since it knows the samples in advance. Finally, the authors of [36] analyzed gradient descent for a single neuron and showed convergence to the optimal separating hyperplane under certain assumptions (appropriate activation and loss functions). We provide two new ideas that improve the learning process. One that adaptively changes the "scale" of the problem and by doing so improves the guarantee on the margin of the output (Algorithm 3), and one that yields almost optimal margin (Algorithms 4).

(ii) Applications for neural networks (Sect. 3). Our variants of the perceptron algorithm are simple to implement, and can therefore be easily applied in the training process of general neural networks. We validate their benefits by training a basic neural network on the MNIST dataset.

(iii) Convex separation (Sect. 4). We use the perceptron's compression to prove a sparse separation lemma for convex bodies. This perspective also suggests a different proof of Novikoff's theorem on the perceptron's convergence [30]. In addition, we interpret this sparse separation lemma in the language of game theory as yielding sparse strategies in a related zero-sum game.

(iv) Generalization bounds (Sect. 5). An important aspect of a learning algorithm is its generalization capabilities; namely, its error on new examples that are independent of the training set (see the textbook [33] for background and definitions). We follow the theme of [18], and observe that even though the (original) perceptron algorithm does not yield an optimal hyperplane, it still generalizes.

(v) Robust concepts (Sect. 6). The robust concepts theme presented by Arriaga and Vempala [3] suggests focusing on well-separated data. We notice that the perceptron fits well into this framework; specifically, that its compression yields efficient dimension reductions. Similar dimension reductions were used in several previous works (e.g. [3–6,16,21]).

Summary. In parts (i)–(ii) we provide a couple of new ideas for improving the training process and explain their contribution in the context of previous work. In part (iii) we use the perceptron's compression as a tool for proving geometric theorems. We are not aware of previous works that studied this connection. Parts (iv)–(v) are mostly about presenting ideas from previous works in the context of the perceptron's compression. We think that parts (iv) and (v) help to understand the picture more fully.

2   Variants of the Perceptron

Deciding how to train a model from a list of input examples is a central consideration in any learning process. In the case of the perceptron algorithm the input examples $(x_1, y_1), (x_2, y_2), \dots$ with $x_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$ 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**
> $\quad\mid\quad w^{(t+1)} = w^{(t)} + y_i x_i$
> $\quad\mid\quad t = t + 1$
> **end**
> return $w^{(t)}$
>
> **Algorithm 1:** The perceptron algorithm

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 [30].

The standard analysis of the perceptron convergence properties uses the optimal separating hyperplane w^* (later in Sect. 4 we present an alternative analysis that does not use it):

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

where we think of S as the map from $\{x_1, \ldots, x_m\}$ to $\{\pm 1\}$ defined by $x_i \mapsto y_i$.[2] Novikoff's analysis consists of the following two parts. Let $\varepsilon^* = \mathsf{marg}(w^*, S)$ 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 $(R/\varepsilon^*)^2$.

As discussed in Sect. 1, Algorithm 1 has several drawbacks. Here we describe some simple ideas that allow to improve it. Below we describe three algorithms, each is followed by a theorem that summarizes its main properties.

In the following, $\mathcal{X} \subset \mathbb{R}^d$ is a finite set, Y is a linear partition, $\varepsilon^* = \mathsf{marg}(Y)$ is the optimal margin, and $R = \max_{x \in \mathcal{X}} \|x\|$ is the maximal norm of a point.

In the first variant that already appeared in [9, 23], the suggestion is to replace the condition $y_i w^{(t)} \cdot x_i < 0$ by $y_i w^{(t)} \cdot x_i < \beta$ for some a priori chosen $\beta > 0$. that may change over time. As we will see, different choices of β yield different guarantees.

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

Algorithm 2: The β-perceptron algorithm

---

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

Theorem 1 ([9,23]). *The β-perceptron algorithm performs at most $\frac{2\beta+R^2}{(\epsilon^*)^2}$ updates and achieves a margin of at least $\frac{\beta\epsilon^*}{2\beta+R^2}$.*

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

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

and

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

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

$$y_iw^{(t)} \cdot x_i \geq \beta.$$

So, since

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

we get

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

To remove the dependence on R in the output's margin above, we propose to rescale β according to the observed examples.

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

Algorithm 3: The R-independent perceptron algorithm

Theorem 2. *The R-independent perceptron algorithm performs at most $\frac{10R^2}{(\varepsilon^*)^2}$ updates and achieves a margin of at least $\frac{\varepsilon^*}{3}$.*

Proof. 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 before a change in β occurs, there could be at most $\frac{2\beta+R^2}{\varepsilon^2}$ errors (as before 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

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

$$\leq 6 \cdot 4/3 \cdot 4^{\lceil \log(R/r) \rceil} \frac{r^2}{(\varepsilon^*)^2} = O((R/\varepsilon^*)^2). \square$$

Finally, if one would like to improve upon the $\frac{\varepsilon^*}{3}$ guarantee, we suggest to change β with time. To run the algorithm, we should first decide how well do we want to approximate the optimal margin. To do so, we need to choose the parameter $\alpha \in (1, 2)$; the closer α is to 2, the better the approximation is (see Theorem 3).

> **initialize:** $w^{(0)} = \vec{0}$ and $t = 0$ and $\beta = 0$ and $\alpha \in (1, 2)$
> **while** $\exists i$ with $y_i w^{(t)} \cdot x_i \leq \beta$ **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

Theorem 3. *If $R \leq 1$, the ∞-perceptron algorithm performs at most $(1/\varepsilon^*)^{2/(2-\alpha)}$ updates and achieves a margin of at least $\frac{\alpha \varepsilon^*}{2}$.*

Proof. 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.

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

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

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

The output's margin is at least

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

This is a decreasing function for $t > 0$, since its derivative is at most zero.

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.   □

Remark 1. *The bound on the running time is sharp, as the following example shows. The two points $(\sqrt{1-\varepsilon^2}, \varepsilon), (\sqrt{1-\varepsilon^2}, -\varepsilon)$ with labels $1, -1$ 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).*

Remark 2. *Algorithms 3 and 4 can be naturally combined to a single algorithm that arrives arbitrarily close to the optimal margin without assuming that $R \leq 1$.*

3   Application for Neural Networks

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 $\mathsf{ReLu}(w \cdot x)$ with loss function of the form $\mathsf{ReLu}(\beta - y_x w \cdot x)$ 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 suggests a formal evidence that supports the benefits of the hinge loss.

Moreover, in practice, β is treated as a hyper-parameter and tuning it is a common challenge that needs to be addressed in order to maximize performance. We proposed a couple of new options for choosing and updating β throughout the training process that may contribute towards a more systematic approach for setting β (see Algorithms 4 and 3). Theorems 2 and 3 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 [24] of handwritten digits as a test case with no preprocessing. We used a simple and standard neural network with one hidden layer consisting of 800/300 neurons and 10 output neurons (the choice of 800 and 300 is the same as in Simard et al. [35] and Lecun et al. [24]). We trained the network by back-propagation (gradient descent). The loss function of each output neuron of the form $\mathsf{ReLu}(w \cdot G(x))$, where $G(x)$ is the output of the hidden layer, is $\mathsf{ReLu}(-y_x w \cdot G(x) + \beta)$ 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).[3]

We tested four values of β as shown in Fig. 1. In two tests, the value of β is fixed in time[4] 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. 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 the table shows the error of the two-layer MLP of the same size that is driven by the cross-entropy loss [35]. In fact, our network of 300 neurons performed better than all the general purpose networks with 300 neurons even with preprocessing of the data that appear in http://yann.lecun.com/exdb/mnist/ (Fig. 2).

	test error
$\beta = 0$	no convergence
$\beta = 1$	1.5 %
$\beta \approx t^{0.4}$	1.44 %
$\beta \approx t^{0.75}$	1.35 %
cross-entropy [35]	1.6 %

Fig. 1. One hidden layer with 800 neurons

	test error
$\beta \approx t^{0.75}$	1.49 %
mean square error [24]	4.7 %
MSE, [distortions] [24]	3.6 %
deskewing [24]	1.6 %

Fig. 2. One hidden layer with 300 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

---

[3]   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.

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

of β may change during the learning process. Figuratively, this can be thought of as "internal clocks" of the neurons. A neuron changes its behavior as time progresses. For example, the "older" the neuron is, the less it is inclined to change.

4   Convex Separation

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 [29]. Two related and fundamental geometric properties are Farkas' lemma [12], and the following separation theorem.

Theorem 4 (Convex separation theorem). *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 (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:*

1. *$\mathsf{dist}(K, L) < \varepsilon$.*
2. *There is a hyperplane $H = \{x : w \cdot x = b\}$ separating K from L so that its normal vector is "sparse":*
    (i) $\frac{w \cdot p - b}{\|w\|} > \frac{\varepsilon}{30}$ for all $p \in K$,
    (ii) $\frac{w \cdot q - b}{\|w\|} < -\frac{\varepsilon}{30}$ for all $q \in L$, and
    (iii) w is a sum of at most $(10/\varepsilon)^2$ points in K and $-L$.

Proof. 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 $p \in K$,

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

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

The lemma is strictly stronger than the preceding separation theorem. Below, we also explain how this perspective yields an alternative proof of Novikoff's theorem on the convergence of the perceptron [30]. 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).

Alternative Proof of the Perceptron's Convergence. 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 $v^{(t)} \cdot w \leq \frac{1}{\sqrt{t}}$ and so there is an example x so that $x \cdot w \leq \frac{1}{\sqrt{t}}$. This implies that the running time T satisfies $\frac{1}{\sqrt{T}} \geq \varepsilon^*$ since for every example x we have $x \cdot w^* \geq \varepsilon^*$.

A Game Theoretic Perspective. The perspective of game theory turned out to be useful in several works in learning theory (e.g. [13,28]). The ideas above 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 $P(v, w) = v \cdot w$ 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

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

Claim (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,

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

Proof. Let $v^{(t)} = \frac{1}{t}w^{(t)}/t$ be as in the proof of Lemma 1 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 $t > 0$,

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

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 $v \cdot v^{(T)} \geq \frac{1}{T}$ 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

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

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. The ideas presented in Sect. 2 allow to reduce the constant 3 to as close to 1 as we want, by paying in running time (see Algorithm 4).

5   Generalization Bounds

Generalization is one of the key concepts in learning theory. 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 [33]). The proof of this property of hard-SVMs uses Rademacher complexity.

We suggest that using the perceptron algorithm, instead the hard-SVM solution, yields a more general statement with a simpler proof. The reason is that the perceptron can be interpreted as a sample compression scheme.

Theorem 5 (similar to [18]). *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*

$$\Pr_S\left[\mathsf{marg}(S) < \varepsilon\right] < \delta/2 \tag{2}$$

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

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

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 [32]).

Condition (2) 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 [33] on the behavior of hard-SVMs (that the margin is always bounded from below).

For the proof of Theorem 5 we will need the following.

Definition 2 (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 $L_S(h) = \frac{1}{m}\sum_{i=1}^m 1_{h(x_i)\neq c(x_i)}$ denote the empirical error of h.

Theorem 6 ([11]). *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 (Theorem 5). 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 $\mathsf{marg}(S) \geq \varepsilon$, 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 6 to this selection scheme and by the assumption that $\mathsf{marg}(S) \geq \varepsilon$ for $1 - \delta/2$ of the space (note that $L_S(K(S)) = 0$ when $\mathsf{marg}(S) \geq \varepsilon$). $\qquad\square$

6   Robust Concepts

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. [4,6,15,16,21].

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.

Theorem 7 ([5]). *The number of ε-separated partitions of $\mathcal{X}$ is at most $(2(n+1))^{1/\varepsilon^2}$.*

Proof. 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 $(n+1)^{1/\varepsilon^2} \cdot 2^{1/\varepsilon^2}$. $\qquad\square$

The theorem is sharp in the following sense.

Example 1. 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 $Y(x) = \mathsf{sign}(w \cdot x)$ 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 $(N(0,1))$[5] 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 8 (implicit in [5]). *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 $\mathcal{X}$.*

The proof of the above theorem is a simple application of Theorem 7 together with the Johnson-Lindenstrauss lemma.

Lemma 2 ([19]). *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 $0 < \delta < 1/2$,*

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

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

References

1. Andoni, A., Panigrahy, R., Valiant, G., Zhang, L.: Learning polynomials with neural networks. PMLR **32**(2), 1908–1916 (2014)
2. Anlauf, J.K., Biehl, M.: The AdaTron: an adaptive perceptron algorithm. EPL **10**, 687 (1989)
3. Arriaga, R.I., Vempala, S.: An algorithmic theory of learning: robust concepts and random projection. Mach. Learn. **63**(2), 161–182 (2006)
4. Balcan, N., Blum, A., Vempala, S.: On kernels, margins and low-dimensional mappings. In: ALT (2004)
5. Ben-David, S., Eiron, N., Simon, H.U.: Limitations of learning via embeddings in Euclidean half spaces. JMLR **3**, 441–461 (2002)
6. Blum, A., Kannan, R.: Learning an intersection of k halfspaces over a uniform distribution. In: FOCS (1993)
7. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: COLT, pp. 144–152 (1992)

---

[5] Other distributions will work just as well.

8. Cesa-Bianchi, N., Conconi, A., Gentile, C.: On the generalization ability of on-line learning algorithms. IEEE Trans. Inf. Theory **50**(9), 2050–2057 (2004)
9. Collobert, R., Bengio, S.: Links between perceptrons, MLPs and SVMs. IDIAP (2004)
10. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. J. Mach. Learn. Res. **7**, 551–585 (2006)
11. David, O., Moran, S., Yehudayoff, A.: Supervised learning through the lens of compression. In: NIPS, pp. 2784–2792 (2016)
12. Farkas, G.: Über die Theorie der Einfachen Ungleichungen. Journal für die Reine und Angewandte Mathematik **124**(124), 1–27 (1902)
13. Freund, Y.: Boosting a weak learning algorithm by majority. Inf. Comput. **121**(2), 256–285 (1995)
14. Freund, Y., Schapire, R.E.: Large margin classification using the perceptron algorithm. Mach. Learn. **37**, 277–296 (1999). https://doi.org/10.1023/A:1007662407062
15. Garg, A., Har-Peled, S., Roth, D.: On generalization bounds, projection profile, and margin distribution. In: ICML, pp. 171–178 (2002)
16. Garg, A., Roth, D.: Margin distribution and learning. In: ICML, pp. 210–217 (2003)
17. Gentile, C.: A new approximate maximal margin classification algorithm. J. Mach. Learn. Res. **2**, 213–242 (2001)
18. Graepel, T., Herbrich, R., Shawe-Taylor, J.: PAC-Bayesian compression bounds on the prediction error of learning algorithms for classification. Mach. Learn. **59**, 55–76 (2005). https://doi.org/10.1007/s10994-005-0462-7
19. Johnson, W.B., Lindenstrauss, J.: Extensions of Lipschitz mappings into a Hilbert space. In: Conference in Modern Analysis and Probability (1982)
20. Khardon, R., Wachman, G.: Noise tolerant variants of the perceptron algorithm. J. Mach. Learn. Res. **8**, 227–248 (2007)
21. Klivans, A.R., Servedio, R.A.: Learning intersections of halfspaces with a margin. In: Shawe-Taylor, J., Singer, Y. (eds.) COLT 2004. LNCS (LNAI), vol. 3120, pp. 348–362. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27819-1_24
22. Korzeń, M., Klęsk, P.: Maximal margin estimation with perceptron-like algorithm. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 597–608. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69731-2_58
23. Krauth, W., Mézard, M.: Learning algorithms with optimal stablilty in neural networks. J. Phys. A: Math. Gen. **20**, L745–L752 (1987)
24. LeCun, Y., Cortes, C.: The MNIST database of handwritten digits (1998)
25. Littlestone, N., Warmuth, M.: Relating data compression and learnability (1986, unpublished)
26. Matoušek, J.: On variants of the Johnson-Lindenstrauss lemma. Random Struct. Algorithms **33**(2), 142–156 (2008)
27. Mohri, M., Rostamizadeh, A.: Perceptron Mistake Bounds. arXiv:1305.0208
28. Moran, S., Yehudayoff, A.: Sample compression schemes for VC classes. JACM **63**(3), 1–21 (2016)
29. von Neumann, J.: Zur Theorie der Gesellschaftsspiele. Math. Ann. **100**, 295–320 (1928)
30. Novikoff, A.B.J.: On convergence proofs for perceptrons. In: Proceedings of the Symposium on the Mathematical Theory of Automata, vol. 12, pp. 615–622 (1962)
31. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. **65**(6), 386–408 (1958)

32. Schapire, R.E., Freund, Y., Bartlett, P., Lee, W.S.: Boosting the margin: a new explanation for the effectiveness of voting methods. Ann. Stat. **26**(5), 1651–1686 (1998)
33. Shalev-Shwartz, S., Ben-David, S.: Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, Cambridge (2014)
34. Shalev-Shwartz, S., Singer, Y., Srebro, N., Cotter, A.: Pegasos: primal estimated sub-gradient solver for SVM. Math. Program. **127**(1), 3–30 (2011). https://doi.org/10.1007/s10107-010-0420-4
35. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. ICDAR **3**, 958–962 (2003)
36. Soudry, D., Hoffer, E., Srebro, N.: The implicit bias of gradient descent on separable data. arXiv:1710.10345 (2017)
37. Wendemuth, A.: Learning the unlearnable. J. Phys. A: Math. Gen. **28**, 5423 (1995)