Perceptron

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

This is the first of a series of lectures devoted to the topic of **linear classification**, which harbors a deep theory, and is arguably the form of classification best understood in machine learning.

Let P be a set of points in $\mathbb{R}^d$, where $\mathbb{R}$ denotes the real domain, and d is a positive integer. Each point in P is colored either **red** or **blue**.

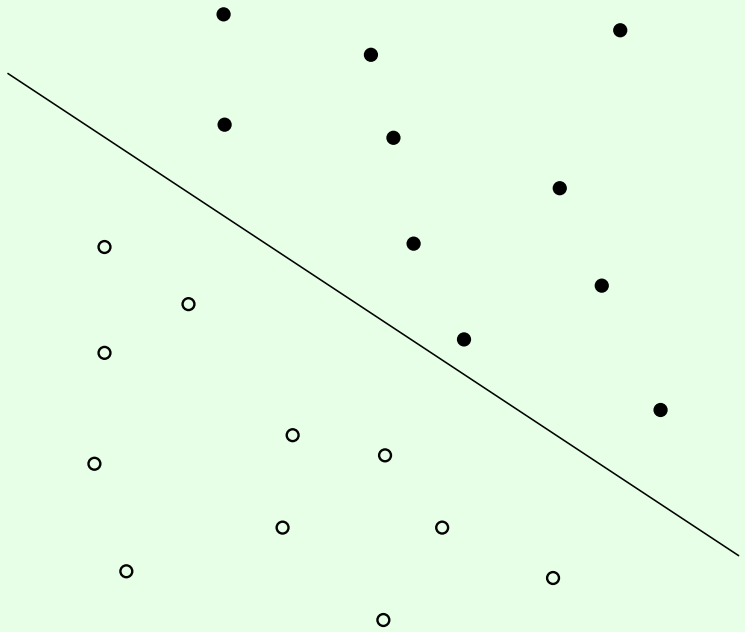
P is said to be **linearly separable** if there is a d-dimensional plane

$$x_1 \cdot c_1 + x_2 \cdot c_2 + ... + x_d \cdot c_d = \lambda$$

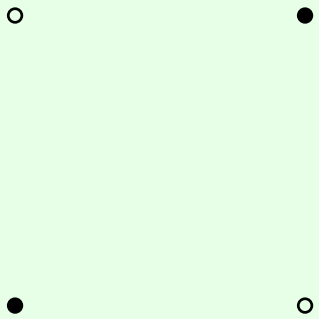
which separates the red points from the blue points in P. In other words, all the red points fall on the same side of the plane, while all the blue points fall on the other side; no point is allowed to fall on the plane. Such a plane is a **separation plane** of P.

P is **linearly non-separable** if it admits no separation planes.

Example:



Linearly separable



Linearly non-separable

In this lecture, we will study the following problem:

Problem 1 (Finding a Separation Plane): Given a d-dimensional set P of points that is linearly separable, find a separation plane of P.

A separation plane of P

$$x_1 \cdot c_1 + x_2 \cdot c_2 + ... + x_d \cdot c_d = \lambda$$

is said to be **origin passing** if $\lambda = 0$. Much of our efforts is dedicated to solving the following problem:

Problem 2 (Finding an Origin Passing Separation Plane): Given a d-dimensional set P of points that admits an origin passing separation plane, find such a plane.

Although Problem 2 may look different from Problem 1 at first glance, we will show at the end of this lecture that every instance of Problem 1 can be cast as an instance of Problem 2, and hence, solved by an algorithm for the latter problem.

Now let us focus on Problem 2. First, observe that no point in P can be at the origin of the data space. This allows us to introduce a new requirement. Suppose that $c_1, ..., c_d$ are the coefficients of the separation plane we return. We demand:

- if a point $p = (x_1, x_2, ..., x_d)$ is red, then it must hold that

$$x_1 \cdot c_1 + x_2 \cdot c_2 + ... + x_d \cdot c_d > 0;$$

- otherwise (i.e., p is blue), it must hold that

$$x_1 \cdot c_1 + x_2 \cdot c_2 + ... + x_d \cdot c_d < 0$$

Think: Why don't we have to worry about which color should be assigned to the "> 0" case?

Next, we will introduce a surprisingly simple algorithm called **perceptron** for solving Problem 2 with strong theoretical guarantees.

Henceforth, we will regard a point p in P as a vector $\vec{p} = (p[1], ..., p[d])$, where $p[i]$ is the i-th coordinate of p.

An origin passing plane

$$x_1 \cdot c_1 + x_2 \cdot c_2 + ... + x_d \cdot c_d = 0$$

is unambiguously specified by its normal vector $\vec{c} = (c_1, ..., c_d)$.

We want to look for a vector $\vec{c}$ such that:

- $\vec{p} \cdot \vec{c} > 0$ for every red point $p \in P$
- $\vec{p} \cdot \vec{c} < 0$ for every blue point $p \in P$.

Perceptron

The algorithm starts with $\vec{c} = (0, 0, ..., 0)$, and then runs in **iterations**.

In each iteration, it simply checks whether any point in $p \in P$ violates our requirement according to $\vec{c}$. If so, the algorithm adjusts $\vec{c}$ as follows:

- If p is red, then $\vec{c} \leftarrow \vec{c} + \vec{p}$.
- If p is blue, then $\vec{c} \leftarrow \vec{c} - \vec{p}$.

As soon as $\vec{c}$ has been adjusted, the current iteration finishes; and a new iteration starts.

The algorithm finishes if the iteration finds all points of P on the correct side of the plane.

Example: Suppose that P has four 2d points: $p_1 = (1, 0)$, $p_2 = (0, -1)$, $p_3 = (0, 1)$, and $p_4 = (-1, 0)$. Points p_1 and p_3 are red, and the other are blue.
The algorithm starts with $\vec{c} = (0, 0, ..., 0)$.

- Iteration 1: $\vec{p_1} \cdot \vec{c} = 0$ violating our requirement $\vec{p_1} \cdot \vec{c} > 0$. Hence, we update $\vec{c}$ to $\vec{c} + \vec{p_1} = (1, 0)$.
- Iteration 2: $\vec{p_2} \cdot \vec{c} = 0$ violating our requirement $\vec{p_2} \cdot \vec{c} < 0$. Hence, we update $\vec{c}$ to $\vec{c} - \vec{p_2} = (1, 0) - (0, -1) = (1, 1)$.
- Iteration 3: all points satisfy our requirements. Thus, the algorithm finishes with $\vec{c} = (1, 1)$.

We now analyze the number of iterations carried out by Perceptron.

Given a vector $\vec{v} = (v_1, ..., v_d)$, we define its **length** of $\vec{v}$ as

$$|\vec{v}| = \sqrt{\vec{v} \cdot \vec{v}} = \sqrt{\sum_{i=1}^{d} v_i^2}.$$

The following well-known property will be useful:

For any vectors $\vec{v}_1, \vec{v}_2$, it holds that $\vec{v}_1 \cdot \vec{v}_2 \leq |\vec{v}_1||\vec{v}_2|$.

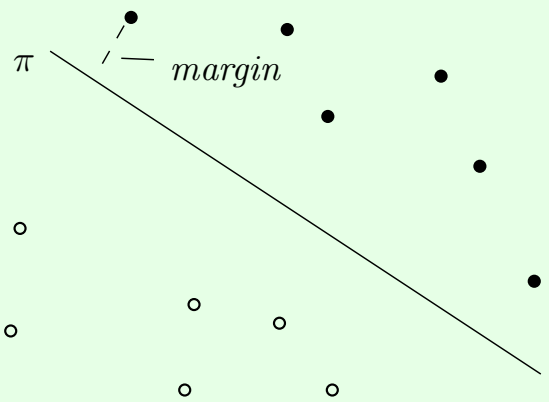
Define:

$$R = \max_{p \in P} \{|\vec{p}|\}.$$

In other words, all the points of P fall in a ball that centers at the origin and has radius R.

Given any origin-passing separation plane π, we define its **margin** as the smallest distance from the points of P to π.

Example:



biggest

Denote by γ the ~~smallest~~ margin of all the origin-passing separation planes. Let π^* be the origin-passing plane whose margin equals γ. If $\vec{u}$ is a **unit normal vector** (i.e., $|\vec{u}| = 1$) of π^*, then:

$$\gamma = \min_{p \in P} |\vec{u} \cdot \vec{p}|.$$

Theorem: Perceptron always terminates after $(R/\gamma)^2$ violations.

Proof: Let π^* be the origin-passing separation plane whose margin is γ. Denote by $\vec{u}$ the unit normal vector of π^* such that $\vec{u} \cdot \vec{p} > 0$ if p is red, and $\vec{u} \cdot \vec{p} < 0$ otherwise.

Recall that the perceptron algorithm maintains a normal vector $\vec{c}$. Let $\vec{c_i}$ $(i \geq 1)$ be the $\vec{c}$ after the i-th violation. Also, let $\vec{c_0} = (0, ..., 0)$ be the initial $\vec{c}$. Denote by k the total number of violations.

We first show that, for any $i \geq 0$, $\vec{c}_{i+1} \cdot \vec{u} \geq \vec{c}_i \cdot \vec{u} + \gamma$.

Recall that $\vec{c}_{i+1}$ was adjusted from $\vec{c}_i$ in one of the following cases:

- A red point $\vec{p}_r$ violates our requirement, namely, $\vec{p}_r \cdot \vec{c}_i < 0$. In this case, $\vec{c}_{i+1} = \vec{c}_i + \vec{p}_r$; and hence, $\vec{c}_{i+1} \cdot \vec{u} = \vec{c}_i \cdot \vec{u} + \vec{p}_r \cdot \vec{u}$. From the definition of γ, we know that $\vec{p}_r \cdot \vec{u} \geq \gamma$. Therefore, $\vec{c}_{i+1} \cdot \vec{u} \geq \vec{c}_i \cdot \vec{u} + \gamma$.
- A blue point $\vec{p}_b$ violates our requirement, namely, $\vec{p}_b \cdot \vec{c}_i > 0$. In this case, $\vec{c}_{i+1} = \vec{c}_i - \vec{p}_b$; and hence, $\vec{c}_{i+1} \cdot \vec{u} = \vec{c}_i \cdot \vec{u} - \vec{p}_b \cdot \vec{u}$. From the definition of γ, we know that $-\vec{p}_b \cdot \vec{u} \geq \gamma$. Therefore, $\vec{c}_{i+1} \cdot \vec{u} \geq \vec{c}_i \cdot \vec{u} + \gamma$.

It follows that

$$\begin{aligned} \vec{c}_k \cdot \vec{u} &\geq \vec{c}_{k-1} \cdot \vec{u} + \gamma \\ &\geq \vec{c}_{k-2} \cdot \vec{u} + 2\gamma \\ &\ldots \\ &\geq \vec{c}_0 \cdot \vec{u} + k\gamma = k\gamma. \end{aligned} \tag{1}$$

Next we will show that, for any $i \geq 0$, $|\vec{c}_{i+1}|^2 \leq |\vec{c}_i|^2 + R^2$.

Recall that $\vec{c}_{i+1}$ was adjusted from $\vec{c}_i$ in one of the following cases:

- A red point $\vec{p}_r$ violates our requirement, namely, $\vec{p}_r \cdot \vec{c}_i < 0$. In this case, $\vec{c}_{i+1} = \vec{c}_i + \vec{p}_r$. Thus:

$$\begin{aligned} |\vec{c}_{i+1}|^2 = \vec{c}_{i+1} \cdot \vec{c}_{i+1} &= (\vec{c}_i + \vec{p}_r) \cdot (\vec{c}_i + \vec{p}_r) \\ &= \vec{c}_i \cdot \vec{c}_i + 2\vec{c}_i \cdot \vec{p}_r + |\vec{p}_r| \\ (\text{by def. of } R) &\leq |\vec{c}_i|^2 + 2\vec{c}_i \cdot \vec{p}_r + R^2 \\ &\leq |\vec{c}_i|^2 + R^2 \end{aligned}$$

  where the last step used the fact that $\vec{p}_r \cdot \vec{c} < 0$.

- A blue point $\vec{p}_b$ violates our requirement, namely, $\vec{p}_b \cdot \vec{c}_i > 0$. The proof is similar, and omitted (a good exercise for you).

It follows that

$$|\vec{c}_k|^2 \leq |\vec{c}_{k-1}|^2 + R^2 \leq |\vec{c}_{k-2}|^2 + 2R^2 \ldots \leq |\vec{c}_0|^2 + kR^2 = kR^2. \quad (2)$$

Now we combine (1) and (2) to obtain an upper bound on k. From (1), we know that

$$|\vec{c}_k| = |\vec{c}_k||\vec{u}| \geq \vec{c}_k \cdot \vec{u} \geq k\gamma.$$

Therefore, $|\vec{c}_k|^2 \geq k^2\gamma^2$. Comparing this to (2) gives:

$$\begin{aligned} kR^2 &\geq k^2\gamma^2 \quad \Rightarrow \\ k &\leq \frac{R^2}{\gamma^2}. \end{aligned}$$

This completes the proof of the theorem. □

Remember that what we have solved is Problem 2, but our original goal was to solve Problem 1. Next, we show that any instance of Problem 1 can be converted to an instance of Problem 2 by introducing one more dimension.

Let P be a d-dimensional dataset which is the input to Problem 1. We create another dataset P' of dimensionality $d+1$ as follows. For each point $p = (x_1, ..., x_d)$ in P, create a point $p' = (x_1, ..., x_d, 1)$ in P'. The color of p' is the same as that of p.

Lemma: P is linearly separable if and only if P' admits an origin passing separation plane.

Proof: The $\Leftarrow$ direction:

Suppose that $x_1 \cdot c_1 + x_2 \cdot c_2 + ...x_d \cdot c_d + x_{d+1} \cdot c_{d+1} = 0$ is a separation plane on P'. This means that for every red point $p' = (x_1, x_2, ..., x_d, 1)$ in P', it holds that $x_1 \cdot c_1 + x_2 \cdot c_2 + ...x_d \cdot c_d + 1 \cdot c_{d+1} > 0$. Also, for every blue point $p' = (x_1, x_2, ..., x_d, 1)$ in P', it holds that $x_1 \cdot c_1 + x_2 \cdot c_2 + ...x_d \cdot c_d + 1 \cdot c_{d+1} < 0$. This means that $x_1 \cdot c_1 + x_2 \cdot c_2 + ...x_d \cdot c_d + c_{d+1} = 0$ is a separation plane on P.

The $\Rightarrow$ direction:

Suppose that $x_1 \cdot c_1 + x_2 \cdot c_2 + ...x_d \cdot c_d + c_{d+1} = 0$ is a separation plane on P. This means that for every red point $p = (x_1, x_2, ..., x_d)$ in P, it holds that $x_1 \cdot c_1 + x_2 \cdot c_2 + ...x_d \cdot c_d + c_{d+1} > 0$. Also, for every blue point $p = (x_1, x_2, ..., x_d)$ in P, it holds that $x_1 \cdot c_1 + x_2 \cdot c_2 + ...x_d \cdot c_d + c_{d+1} < 0$. Therefore, $x_1 \cdot c_1 + x_2 \cdot c_2 + ...x_d \cdot c_d + x_{d+1} \cdot c_{d+1} = 0$ is an origin-passing separation plane on P'. $\square$

There is a subtle issue: **How many bits are needed to encode $c_1, c_2, ..., c_{d+1}$?**

Let us assume that every coordinate of the points in P' can be represented as an s-bit integer (this does not lose generality from a practical point of view; **why**?), namely, each coordinate is in a range that contains 2^s integers.

Notice that each c_i $(1 \leq i \leq d+1)$ is a **weighted sum** of t coordinates, where t is the number of iterations performed by Perceptron. Specifically, convince yourself that $c_i = \sum_{j=1}^{t} \alpha_j \cdot p_j$ where each α_j equals either -1 or 1. Therefore, c_i is in a range that contains at most $t \cdot 2^s$ integers, and can be represented using $s + \log_2 t$ bits.

How is all of this related to classification?

The input P to Problem 1 corresponds to the training set. Suppose we **could** find $c_1, ..., c_{d+1}$ such that

- $x_1 \cdot c_1 + x_2 \cdot c_2 + ...x_d \cdot c_d + c_{d+1} < 0$ for every label-0 point
- $x_1 \cdot c_1 + x_2 \cdot c_2 + ...x_d \cdot c_d + c_{d+1} > 0$ for every label-1 point.

We have obtained a classifier $f(p) = c_{d+1} + \sum_{i=1}^{d} c_i \cdot p[i]$ that gives a label 0 to a point p if $f(p) < 0$ and 1 otherwise. This classifier achieves **training error 0**, and can often be encoded in a small number of bits. The generalization theorem then promises us that this classifier has a low generalization error as long as P is not too small.

> In fact, by resorting to the theory of VC dimensions, we can prove that $|P|$ only needs to be roughly $O(d)$ for this purpose, regardless of how $c_1, ..., c_{d+1}$ are encoded. This will be discussed further later in the course.