1 The Perceptron Algorithm

Today we discuss a classic online algorithm called the Perceptron Algorithm for learning a linear separator. We will give guarantees for the case that there is a perfect separator as well as guarantees in terms of the "hinge loss" of the best solution when there is no perfect separator. We then briefly discuss kernel functions.

1.1 Online linear classification

We are continuing our discussion of online learning. We have a sequence of examples (say, email messages) arriving one at a time $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots$, and for each example we have to predict if it is positive or negative (e.g., mark the email as "important" or "not important"). After making our prediction, we are then told (by our user) if we were correct or if we made a mistake.

Let's assume that our examples are represented as vectors in $\mathcal{R}^n$ (e.g., for email messages represented as a "bag of words", n could be the number of words in the dictionary, and the jth component of the vector could indicate the number of times word j appeared in the email). Let's furthermore suppose that there exists some unknown weight vector $\mathbf{w}^*$ such that $\mathbf{x}_i \cdot \mathbf{w}^* \geq 1$ for the the *positive* examples (the important emails) and $\mathbf{x}_i \cdot \mathbf{w}^* \leq -1$ for the *negative* examples (the unimportant emails). Our goal is to give an algorithm for performing this task that makes as few mistakes as possible.

The Perceptron Algorithm is a classic algorithm from the 1960s for this problem. It maintains a weight vector $\mathbf{w}$ that it uses for prediction, predicting positive on $\mathbf{x}_i$ if $\mathbf{x}_i \cdot \mathbf{w} > 0$ and predicting negative if $\mathbf{x}_i \cdot \mathbf{w} < 0$, and then updates $\mathbf{w}$ when it makes a mistake. Let's interpret $\mathbf{x}_i \cdot \mathbf{w} = 0$ as saying "I don't know" and making a mistake either way. What we will prove about the algorithm is the following theorem.

Theorem 1 *On any sequence of examples $\mathbf{x}_1, \mathbf{x}_2, \ldots$, if there exists a consistent $\mathbf{w}^*$, i.e., $\mathbf{x}_i \cdot \mathbf{w}^* \geq 1$ for the positive examples and $\mathbf{x}_i \cdot \mathbf{w}^* \leq -1$ for the negative examples, then the Perceptron algorithm makes at most $R^2||\mathbf{w}^*||^2$ mistakes, where $R = \max_i ||\mathbf{x}_i||$.*

Here, $||\mathbf{x}_i||$ means the length of $\mathbf{x}_i$ as a vector. To get a feel for this statement, notice that if we multiply all entries in all the $\mathbf{x}_i$ by 100, we can divide all entries in $\mathbf{w}^*$ by 100 and it will still be consistent. So the bound is invariant to this kind of scaling (i.e., what our "units" are).

Notice also that if we rewrite $\mathbf{x}_i \cdot \mathbf{w}^* \geq 1$ as $\mathbf{x}_i \cdot \mathbf{w}^*/||\mathbf{w}^*|| \geq 1/||\mathbf{w}^*||$, and we think of the $\mathbf{x}_i$ as points in $\mathcal{R}^n$, then we can think of the LHS as the distance of $\mathbf{x}_i$ to the hyperplane $\mathbf{x} \cdot \mathbf{w}^* = 0$. So, we can think of $\mathbf{w}^*$ as defining a *linear separator* that separates the positive examples from the negative examples, and what the theorem is saying is that if there exists a hyperplane that correctly separates the positive examples from the negative examples by a large *margin*, then the total number of mistakes will be small.

1.2 The algorithm

The Perceptron algorithm is simply the following:

1. Begin with $\mathbf{w} = \mathbf{0}$.
2. Given $\mathbf{x}_i$, predict positive if $\mathbf{x}_i \cdot \mathbf{w} > 0$ and predict negative if $\mathbf{x}_i \cdot \mathbf{w} < 0$, else say "I don't know".
3. If a mistake was made (or the prediction was "I don't know"):
   - If the correct answer was "positive", update: $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}_i$.
   - If the correct answer was "negative", update: $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}_i$.

1.3 The analysis

Proof (Theorem 1): Fix some consistent $\mathbf{w}^*$. We will keep track of two quantities, $\mathbf{w} \cdot \mathbf{w}^*$ and $||\mathbf{w}||^2$. First of all, each time we make a mistake, $\mathbf{w} \cdot \mathbf{w}^*$ increases by at least 1. That is because if $\mathbf{x}_i$ is a positive example, then

$$(\mathbf{w} + \mathbf{x}_i) \cdot \mathbf{w}^* = \mathbf{w} \cdot \mathbf{w}^* + \mathbf{x}_i \cdot \mathbf{w}^* \geq \mathbf{w} \cdot \mathbf{w}^* + 1,$$

by definition of $\mathbf{w}^*$. Similarly, if $\mathbf{x}_i$ is a negative example, then

$$(\mathbf{w} - \mathbf{x}_i) \cdot \mathbf{w}^* = \mathbf{w} \cdot \mathbf{w}^* - \mathbf{x}_i \cdot \mathbf{w}^* \geq \mathbf{w} \cdot \mathbf{w}^* + 1.$$

Next, on each mistake, we claim that $||\mathbf{w}||^2$ increases by at most R^2. Let us first consider mistakes on positive examples. If we make a mistake on a positive example $\mathbf{x}_i$ then we have

$$(\mathbf{w} + \mathbf{x}_i) \cdot (\mathbf{w} + \mathbf{x}_i) = ||\mathbf{w}||^2 + 2\mathbf{w} \cdot \mathbf{x}_i + ||\mathbf{x}_i||^2 \leq ||\mathbf{w}||^2 + ||\mathbf{x}_i||^2 \leq ||\mathbf{w}||^2 + R^2,$$

where the middle inequality comes from the fact that we made a mistake, which means that $\mathbf{w} \cdot \mathbf{x}_i \leq 0$. Similarly, if we make a mistake on a negative example $\mathbf{x}_i$ then we have

$$(\mathbf{w} - \mathbf{x}_i) \cdot (\mathbf{w} - \mathbf{x}_i) = ||\mathbf{w}||^2 - 2\mathbf{w} \cdot \mathbf{x}_i + ||\mathbf{x}_i||^2 \leq ||\mathbf{w}||^2 + ||\mathbf{x}_i||^2 \leq ||\mathbf{w}||^2 + R^2.$$

Note that it is important here that we only update on a mistake.

So, if we make M mistakes, then $\mathbf{w} \cdot \mathbf{w}^* \geq M$, and $||\mathbf{w}||^2 \leq MR^2$, or equivalently, $||\mathbf{w}|| \leq R\sqrt{M}$.

Finally, we use the fact that $\mathbf{w} \cdot \mathbf{w}^*/||\mathbf{w}^*|| \leq ||\mathbf{w}||$ which is just saying that the projection of $\mathbf{w}$ in the direction of $\mathbf{w}^*$ cannot be larger than the length of $\mathbf{w}$. This gives us:

$$\begin{aligned} M/||\mathbf{w}^*|| &\leq R\sqrt{M} \\ \sqrt{M} &\leq R||\mathbf{w}^*|| \\ M &\leq R^2||\mathbf{w}^*||^2 \end{aligned}$$

as desired. ■

1.4 Extensions and hinge-loss

We assumed above that there existed a perfect $\mathbf{w}^*$ that correctly classified all the examples, i.e., correctly classified all the emails into important versus non-important. This is rarely the case in real-life data. What if $\mathbf{w}^*$ isn't quite perfect? We can see what this does to the above proof: if there is an example that $\mathbf{w}^*$ doesn't correctly classify, then while the second part of the proof still holds, the first part (the dot product of $\mathbf{w}$ with $\mathbf{w}^*$ increasing) breaks down. However, if this doesn't happen too often, and also $\mathbf{x}_i \cdot \mathbf{w}^*$ is just a "little bit wrong" then this just means we will make a few more mistakes.

To make this formal, define the *hinge-loss* of $\mathbf{w}^*$ on a positive example $\mathbf{x}_i$ as $\max(0, 1 - \mathbf{x}_i \cdot \mathbf{w}^*)$. In other words, if $\mathbf{x}_i \cdot \mathbf{w}^* \geq 1$ as desired then the hinge-loss is zero; else, the hinge-loss is the amount the LHS is less than the RHS.[1] Similarly, the hinge-loss of $\mathbf{w}^*$ on a negative example $\mathbf{x}_i$ is $\max(0, 1+\mathbf{x}_i \cdot \mathbf{w}^*)$. Given a sequence of labeled examples S, define the total hinge-loss $L_{hinge}(\mathbf{w}^*, S)$ as the sum of hinge-losses of $\mathbf{w}^*$ on all examples in S. We now get the following extended theorem.

Theorem 2 *On any sequence of examples $S = \mathbf{x}_1, \mathbf{x}_2, \ldots$, the Perceptron algorithm makes at most*

$$\min_{\mathbf{w}^*} \left(R^2 ||\mathbf{w}^*||^2 + 2L_{hinge}(\mathbf{w}^*, S) \right)$$

mistakes, where $R = \max_i ||\mathbf{x}_i||$.

Proof: As before, each update of the Perceptron algorithm increases $||\mathbf{w}||^2$ by at most R^2, so if the algorithm makes M mistakes, we have $||\mathbf{w}||^2 \leq MR^2$.

What we can no longer say is that each update of the algorithm increases $\mathbf{w} \cdot \mathbf{w}^*$ by at least 1. Instead, on a positive example we are "increasing" $\mathbf{w} \cdot \mathbf{w}^*$ by $\mathbf{x}_i \cdot \mathbf{w}^*$ (it could be negative), which is at least $1 - L_{hinge}(\mathbf{w}^*, \mathbf{x}_i)$. Similarly, on a negative example we "increase" $\mathbf{w} \cdot \mathbf{w}^*$ by $-\mathbf{x}_i \cdot \mathbf{w}^*$, which is also at least $1 - L_{hinge}(\mathbf{w}^*, \mathbf{x}_i)$. If we sum this up over all mistakes, we get that at the end we have $\mathbf{w} \cdot \mathbf{w}^* \geq M - L_{hinge}(\mathbf{w}^*, S)$, where we are using here the fact that hinge-loss is never negative so summing over all of S is only larger than summing over the mistakes that $\mathbf{w}$ made.

Finally, we just do some algebra. Let $L = L_{hinge}(\mathbf{w}^*, S)$. So we have:

$$\begin{aligned}
\mathbf{w} \cdot \mathbf{w}^*/||\mathbf{w}^*|| &\leq ||\mathbf{w}|| \\
(\mathbf{w} \cdot \mathbf{w}^*)^2 &\leq ||\mathbf{w}||^2 ||\mathbf{w}^*||^2 \\
(M - L)^2 &\leq MR^2 ||\mathbf{w}^*||^2 \\
M^2 - 2ML + L^2 &\leq MR^2 ||\mathbf{w}^*||^2 \\
M - 2L + L^2/M &\leq R^2 ||\mathbf{w}^*||^2 \\
M &\leq R^2 ||\mathbf{w}^*||^2 + 2L - L^2/M \;\leq\; R^2 ||\mathbf{w}^*||^2 + 2L
\end{aligned}$$

as desired. ■

1.5 Kernel functions

What if even the best $\mathbf{w}^*$ has high hinge-loss? E.g., maybe instead of a linear separator decision boundary, the boundary between important emails and unimportant emails looks more like a circle?

[1] This is called "hinge-loss" because as a function of $\mathbf{x}_i \cdot \mathbf{w}^*$ it looks like a hinge.

A neat idea for addressing situations like this is to use what are called *kernel functions*, or sometimes the "kernel trick". Here is the idea. Suppose you have a function K (called a "kernel") over pairs of data points such that for some (doesn't even have to be known) function $\phi : \mathcal{R}^n \to \mathcal{R}^N$ (where perhaps $N \gg n$) we have $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$. In that case, if we can write the Perceptron algorithm so that it only interacts with the data via dot products, and then replace every dot-product with an invocation of K, then we can act as if we had performed the function ϕ explicitly without having to actually compute ϕ. For example, consider $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i \cdot \mathbf{x}_j)^d$. It turns out this corresponds to a mapping ϕ into a space of dimension $N \approx n^d$ (try doing it with $d = 2$), and perhaps in this higher-dimensional space there is a $\mathbf{w}^*$ such that the bound of Theorem 2 is small. But the nice thing is we didn't have to computationally perform the mapping ϕ!

So, how can we view the Perceptron algorithm as only interacting with data via dot-products? Notice that $\mathbf{w}$ is always a linear combination of data points, e.g., we might have $\mathbf{w} = \mathbf{x}_1 + \mathbf{x}_2 - \mathbf{x}_5$. So if we keep track of it this way, and need to predict on a new example $\mathbf{x}_6$, we can write $\mathbf{w} \cdot \mathbf{x}_6 = \mathbf{x}_1 \cdot \mathbf{x}_6 + \mathbf{x}_2 \cdot \mathbf{x}_6 - \mathbf{x}_5 \cdot \mathbf{x}_6$. So if we just replace each of these dot-products with "K", we are running the algorithm as if we had explicitly performed the ϕ mapping. This is called "kernelizing" the algorithm.