Perceptron

Perceptron linear threshold classifier

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n > 0.$$

We may add a bias term, or have a dummy input always set to 1

Perceptron Learning Algorithm

1. Start with the all-zeroes weight vector $\mathbf{w}_1 = \mathbf{0}$, and initialize t to 1. Also let's automatically scale all examples $\mathbf{x}$ to have (Euclidean) length 1, since this doesn't affect which side of the plane they are on.
2. Given example $\mathbf{x}$, predict positive iff $\mathbf{w}_t \cdot \mathbf{x} > 0$.
3. On a mistake, update as follows:
   - Mistake on positive: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{x}$.
   - Mistake on negative: $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \mathbf{x}$.

   $t \leftarrow t + 1$.

So, this seems reasonable. If we make a mistake on a positive $\mathbf{x}$ we get $\mathbf{w}_{t+1} \cdot \mathbf{x} = (\mathbf{w}_t + \mathbf{x}) \cdot \mathbf{x} = \mathbf{w}_t \cdot \mathbf{x} + 1$, and similarly if we make a mistake on a negative $\mathbf{x}$ we have $\mathbf{w}_{t+1} \cdot \mathbf{x} = (\mathbf{w}_t - \mathbf{x}) \cdot \mathbf{x} = \mathbf{w}_t \cdot \mathbf{x} - 1$. So, in both cases we move closer (by 1) to the value we wanted.

Mistake Bound of Perceptron Learning

Theorem 1 *Let $\mathcal{S}$ be a sequence of labeled examples consistent with a linear threshold function $\mathbf{w}^* \cdot \mathbf{x} > 0$, where $\mathbf{w}^*$ is a unit-length vector. Then the number of mistakes M on $\mathcal{S}$ made by the online Perceptron algorithm is at most $(1/\gamma)^2$, where*

$$\gamma = \min_{\mathbf{x} \in \mathcal{S}} \frac{|\mathbf{w}^* \cdot \mathbf{x}|}{||\mathbf{x}||}.$$

Margin of Classification

(I.e., if we scale examples to have Euclidean length 1, then γ is the minimum distance of any example to the plane $\mathbf{w}^* \cdot \mathbf{x} = 0$.)

The parameter "γ" is often called the "margin" of $\mathbf{w}^*$ (or more formally, the L_2 margin because we are scaling by the L_2 lengths of the target and examples). Another way to view the quantity $\mathbf{w}^* \cdot \mathbf{x}/||\mathbf{x}||$ is that it is the cosine of the angle between $\mathbf{x}$ and $\mathbf{w}^*$, so we will also use $\cos(\mathbf{w}^*, \mathbf{x})$ for it.

Proof of Theorem 1. We're going to look at the magic quantities $\mathbf{w}_t \cdot \mathbf{w}^*$ and $||\mathbf{w}_t||$.

Claim 1: $\mathbf{w}_{t+1} \cdot \mathbf{w}^* \geq \mathbf{w}_t \cdot \mathbf{w}^* + \gamma$. That is, every time we make a mistake, the dot-product of our weight vector with the target increases by at least γ.

> Proof: if $\mathbf{x}$ was a positive example, then we get $\mathbf{w}_{t+1} \cdot \mathbf{w}^* = (\mathbf{w}_t + \mathbf{x}) \cdot \mathbf{w}^* = \mathbf{w}_t \cdot \mathbf{w}^* + \mathbf{x} \cdot \mathbf{w}^* \geq \mathbf{w}_t \cdot \mathbf{w}^* + \gamma$ (by definition of γ). Similarly, if $\mathbf{x}$ was a negative example, we get $(\mathbf{w}_t - \mathbf{x}) \cdot \mathbf{w}^* = \mathbf{w}_t \cdot \mathbf{w}^* - \mathbf{x} \cdot \mathbf{w}^* \geq \mathbf{w}_t \cdot \mathbf{w}^* + \gamma$.

Claim 2: $||\mathbf{w}_{t+1}||^2 \leq ||\mathbf{w}_t||^2 + 1$. That is, every time we make a mistake, the length squared of our weight vector increases by at most 1.

> Proof: if $\mathbf{x}$ was a positive example, we get $||\mathbf{w}_t + \mathbf{x}||^2 = ||\mathbf{w}_t||^2 + 2\mathbf{w}_t \cdot \mathbf{x} + ||\mathbf{x}||^2$. This is less than $||\mathbf{w}_t||^2 + 1$ because $\mathbf{w}_t \cdot \mathbf{x}$ is negative (remember, we made a mistake on $\mathbf{x}$). Same thing (flipping signs) if $\mathbf{x}$ was negative but we predicted positive.

Claim 1 implies that after M mistakes, $\mathbf{w}_{M+1} \cdot \mathbf{w}^* \geq \gamma M$. On the other hand, Claim 2 implies that after M mistakes, $||\mathbf{w}_{M+1}|| \leq \sqrt{M}$. Now, all we need to do is use the fact that $\mathbf{w}_t \cdot \mathbf{w}^* \leq ||\mathbf{w}_t||$, since $\mathbf{w}^*$ is a unit vector. So, this means we must have $\gamma M \leq \sqrt{M}$, and thus $M \leq 1/\gamma^2$. ■

Discussion: In the worst case, γ can be exponentially small in n. On the other hand, if we're lucky and the data is well-separated, γ might even be large compared to $1/n$. This is called the "large margin" case. (In fact, the latter is the more modern spin on things: namely, that in many natural cases, we would hope that there exists a large-margin separator.) In fact, one nice thing here is that the mistake-bound depends on just a purely geometric quantity: the amount of "wiggle-room" available for a solution and doesn't depend in any direct way on the number of features in the space.

So, if data is separable by a large margin, then Perceptron is a good algorithm to use.

Winnow Algorithm for Learning Disjunctions

The Winnow Algorithm (a simple version)

1. Initialize the weights $w_1, \ldots, w_n$ of the variables to 1.
2. Given an example $x = \{x_1, \ldots, x_n\}$, output 1 if

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n \geq n$$

   and output 0 otherwise.
3. If the algorithm makes a mistake:
   (a) If the algorithm predicts negative on a positive example, then for each x_i equal to 1, double the value of w_i.
   (b) If the algorithm predicts positive on a negative example, then for each x_i equal to 1, cut the value of w_i in half.
4. Goto 2.

Theorem 5. *The Winnow Algorithm learns the class of disjunctions in the Mistake Bound model, making at most* $2 + 3r(1 + \lg n)$ *mistakes when the target concept is a disjunction of* r *variables.*