

## Perceptron

The perceptron is probably the oldest algorithm for binary classification and in a sense the easiest to understand. The idea is that we receive a sequence of example  $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)$ . We maintain a classifier at each point in time denoted by  $(\mathbf{w}, b)$  and have a classification rule based on  $f_{\mathbf{w}, b}(\mathbf{x}) = \text{sign}[\mathbf{w}^\top \mathbf{x} + b]$ .

The perceptron is an *error correcting* algorithm, which means that it updates itself only when it makes mistakes. The algorithm looks like:

1. For each data point  $n = 1, \dots, N$ :
  - (a) If  $f_{\mathbf{w}, b}(\mathbf{x}_n) \neq y_n$ , update:
    - i.  $\mathbf{w} \leftarrow \mathbf{w} + y_n \mathbf{x}_n$
    - ii.  $b \leftarrow b + y_n$

At every update, we move  $\mathbf{w}$  closer to  $y_n \mathbf{x}_n$ , which means that if we were to try to reclassify this example, we would be more likely to get it right. That is, if  $(\mathbf{w}, b)$  is the pre-update classifier and  $(\mathbf{w}', b')$  is the post-update classifier, then on example  $(\mathbf{x}_n, y_n)$ , we have:

$$\begin{aligned} f_{\mathbf{w}', b'}(\mathbf{x}_n) &= \text{sign}[\mathbf{w}'^\top \mathbf{x}_n + b'] \\ &= \text{sign}[\mathbf{w}^\top \mathbf{x}_n + y_n \mathbf{x}_n^\top \mathbf{x}_n + b' + y_n] \\ &= \text{sign}[\mathbf{w}^\top \mathbf{x}_n + b' + y_n \|\mathbf{x}_n\| + y_n] \\ &= \text{sign}[\mathbf{w}^\top \mathbf{x}_n + b' + y_n(\|\mathbf{x}_n\| + 1)] \end{aligned}$$

The first two terms are just the classification rule according to the old weights. We know that  $\|\mathbf{x}_n\| \geq 0$  so  $\|\mathbf{x}_n\| + 1 \geq 1$ , so we're adding a positive constant times  $y_n$  to the old prediction. Thus, if  $y_n = +1$ , we're increasing the prediction (making it more likely that the sign returns +1) and if  $y_n = -1$ , we're decreasing it.

This leads directly to the following theorem:

**Theorem (Block & Novikoff):** if the training data is linearly separable with margin  $\gamma$  with weights  $\mathbf{u}$ , then perceptron will terminate in  $\|\mathbf{u}\|^2 \leq R^2/\gamma^2$  iterations (assuming  $\|x\| \leq R$  for all  $x$ ).

*Proof sketch:* (Suppose  $R = 1$ .) Let  $\mathbf{w}_k$  be the weights before the  $k$ th update, so  $\mathbf{w}_1 = 0$  and if the  $k$ th error is on example  $n$ , then  $y_n(\mathbf{w}_k^\top \mathbf{x}_n) \leq 0$ . Then,  $\mathbf{w}_{k+1}^\top \mathbf{u} = \mathbf{w}_k^\top \mathbf{u} + y_n(\mathbf{u}^\top \mathbf{x}_n) \geq \mathbf{w}_k^\top \mathbf{u} + \gamma$ , so  $\mathbf{w}_{k+1}^\top \mathbf{u} \geq k\gamma$ . Also,  $\|\mathbf{w}_{k+1}\|^2 = \|\mathbf{w}_k\|^2 + 2y_n(\mathbf{w}_k^\top \mathbf{x}_n) + \|\mathbf{x}_n\|^2 \leq \|\mathbf{w}_k\|^2 + 1$  so  $\|\mathbf{w}_{k+1}\|^2 \leq k$ . Thus  $\sqrt{k} \geq \|\mathbf{w}_{k+1}\| \geq \mathbf{w}_{k+1}^\top \mathbf{u} \geq k\gamma$ ; then algebra.

For the inseparable case, just force it to be separable (add a feature).

Cool thing: you can actually understand the perceptron as doing a form of *stochastic sub-gradient descent* on a slightly modified version of hinge loss!!!