CS281B/Stat241B (Spring 2008) Statistical Learning Theory **Lecture: 2**

Linear threshold functions and the perceptron algorithm

Lecturer: Peter Bartlett *Scribe: Julia Palacios*

We'll consider a general family of approaches to the pattern classification problem, known as *empirical risk minimization* methods. We fix a class F of functions that map from $\mathcal{X}$ to $\mathcal{Y} = \{\pm 1\}$, and the method is such that, for all $(x_1, y_1), \ldots, (x_n, y_n)$, there is an $f \in F$ such that

$$f_n(x; x_1, y_1, \ldots, x_n, y_n) = f(x)$$

where $f \in F$ minimizes the empirical risk,

$$\hat{R}(f) = \hat{\mathbb{E}}\ell(f(X), Y) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i).$$

In the case of the 0-1 loss, the empirical risk is the proportion of training data that are misclassified.

1 Linear threshold functions

Consider the class of linear threshold functions on $\mathcal{X} = \mathbb{R}^d$,

$$F = \left\{ x \mapsto \text{sign}(\theta' x) : \theta \in \mathbb{R}^d \right\}.$$

The decision boundaries are hyperplanes through the origin ($d-1$-dimensional subspaces), and the decision regions are half-spaces.

Because of their simplicity, such functions are widely used. For instance, in an object recognition problem, the x vectors might be (hand-crafted) features of locations in an image. In a document classification problem, they might be counts of different words (called a bag-of-words representation).

Notice that we work with thresholded linear functions, so the decision boundaries are subspaces. If we wish to consider thresholded affine functions, so the decision boundaries are arbitrary hyperplanes, this involves a simple transformation: We can write

$$\begin{aligned} F &= \left\{ x \mapsto \text{sign}(\theta' x + c) : \theta \in \mathbb{R}^d,\, c \in \mathbb{R} \right\} \\ &= \left\{ x \mapsto \text{sign}(\theta' \tilde{x}) : \theta \in \mathbb{R}^{d+1} \right\}, \end{aligned}$$

where we define $\tilde{x}' = (x'1)$. For notational simplicity, we'll stick to the linear case.

Let's consider empirical risk minimization over the class of linear threshold functions. Looking at it through the lens of the three key issues of approximation, estimation, and computation also gives a taste of some of the later parts of the course.

Approximation There are probability distributions for which linear threshold functions are optimal (for instance, if the class-conditional distributions are isotropic Gaussians with the same variances). However, it is a very restricted class of decision functions on $\mathbb{R}^d$. We'll see later that we can overcome this difficulty, and retain many of the attractive properties of linearly parameterized functions, by first considering a nonlinear transformation $\phi : \mathbb{R}^d \to \mathbb{R}^D$ for some $D \gg d$. (This is the approach taken by kernel methods. There, we can avoid explicitly computing $\phi(x)$, which can be advantageous if D is very large.)

Estimation If we minimize empirical risk over the class, will that ensure that the risk is near minimal? It suffices if the size n of the data set is large compared to the dimension d (the number of parameters). This is not necessary: we'll see that we can achieve good estimation properties in high dimensional parameter spaces if we consider, for example, Euclidean balls in parameter space, rather than the whole space.

Computation It turns out that minimization of empirical risk over the class of linear threshold functions is easy if the best f has $\hat{R}(f) = 0$. In that case, it corresponds to solving a linear program. Otherwise, empirical risk minimization over this class is a difficult problem: NP-hard, in general. One strategy we'll investigate—it's a strategy used in support vector machines and AdaBoost—is to replace the 0-1 loss with a convex loss function, so that the minimization of this new empirical risk becomes a convex optimization.

2 The perceptron algorithm

The perceptron algorithm maintains a parameter vector, which it updates using a currently misclassified (x_i, y_i) pair. Figure 2 illustrates the intuition: each time the parameter vector is updated, the decision boundary is shifted so that the example used for the update becomes closer to being correctly classified.

The following theorem shows that whenever there is a linear threshold function that correctly classifies all of the training data, then the perceptron algorithm terminates after a finite number of updates and returns such a function. The number of iterations depends on how much slack the data allows in defining a suitable decision boundary.

```
     input  : training data, $(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n) \in \mathbb{R}^d \times \{\pm 1\}$
     output: linear threshold function, $f : \mathbb{R}^d \to \{\pm 1\}$
1.1  Set $\theta_0 := 0$, $t := 0$;
1.2  while some $i$ has $y_i \neq \text{sign}(\theta_t' x_i)$ do
1.3      pick some $i$ with $y_i \neq \text{sign}(\theta_t' x_i)$;
1.4      $\theta_{t+1} := \theta_t + y_i x_i$;
1.5      $t := t + 1$;
1.6  end
1.7  return $x \mapsto \text{sign}(\theta_t' x)$
```

Algorithm 1: perceptron

Theorem 2.1. Suppose that, for some $\theta \in \mathbb{R}^d$ and all i, $y_i = \text{sign}(\theta' x_i)$. Then for any sequence of choices made at step (1.3) of the algorithm, we have

(a) The perceptron algorithm terminates.

(b) If we define

$$r = \max_i \|x_i\| \qquad \delta = \min_i \left(\frac{\theta' x_i y_i}{\|\theta\|} \right),$$

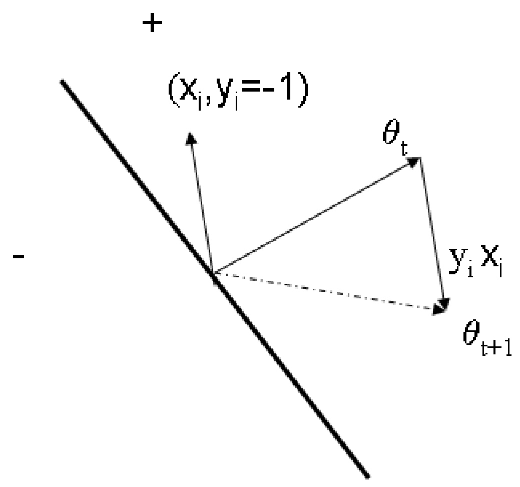

Figure 1: *Perceptron algorithm*

then the algorithm terminates after

$$t \leq \frac{r^2}{\delta^2}$$

updates.

Notice that r is the radius of a ball containing the data, and δ is the margin between the separating hyperplane and the closest point. (Also, we could rescale the data so the δ and r both get larger, but the bound on the number of iterations of the algorithm would be unchanged.)

Proof. Clearly $(b) \implies (a)$, so we'll prove (a).

Fix a parameter vector θ that satisfies the conditions of the theorem, that is, that separates the data with a large margin. The intuition behind the proof is that θ_t and θ become more aligned with each update, because $\theta_t'\theta$ increases linearly with t, whereas $\|\theta_t\|$ does not increase that rapidly.

First, $\theta_0'\theta = 0$ and

$$\begin{aligned}\theta_{t+1}'\theta &= \theta_t'\theta + y_i x_i'\theta \\ &\geq \theta_t'\theta + \delta\|\theta\|,\end{aligned}$$

where the inequality follows from the margin assumption. Clearly, $\theta_t'\theta \geq t\delta\|\theta\|$.

On the other hand, $\|\theta_0\| = 0$, and

$$\begin{aligned}\|\theta_{t+1}\|^2 &= \|\theta_t + y_i x_i\|^2 \\ &= \|\theta_t\|^2 + \|x_i\|^2 + 2y_i\theta_t'x_i \\ &\leq \|\theta_t\|^2 + r^2,\end{aligned}$$

where the inequality follows from the definition of r and the fact that the (x_i, y_i) pair chosen at step (1.3) is misclassified. Clearly, $\|\theta_t\|^2 \leq tr^2$.

Using Cauchy-Schwarz, we have

$$t\delta\|\theta\| \leq \theta_t'\theta \leq \|\theta_t\|\|\theta\| \leq \sqrt{t}r\|\theta\|.$$

Squaring and dividing both sides by $t\delta^2\|\theta\|^2$ gives the result. □

Note. If we let i_t denote the (x_i, y_i) pair chosen in the ith update, we can write

$$\theta_t = \sum_{s \le t} y_{i_s} x_{i_s} = \sum_{i=1}^{n} \alpha_i x_i,$$

where $\sum |\alpha_i| \le t$. That is, instead of representing the parameter vector θ_t directly, it can be expressed in terms of the coefficients α of this combination of the x_is. And this combination is sparse (has few non-zero components) if n is large compared to t.

The perceptron algorithm relies on $x_i's$ only via $\theta_t' x_i = \sum_{j=1}^{n} \alpha_j (x_j' x_i)$. Thus, we do not need to manipulate the points x_i directly, only their inner products with each other. This suggests the idea of mapping the data points to a higher-dimensional feature-space $\phi : \mathbb{R}^d \to \mathbb{R}^D$, with $D \gg d$. As long as we can efficiently compute the inner products between the $\phi(x_i)$, we need never explicitly compute the representation in $\mathbb{R}^D$.

3 A minimax lower bound on risk

Suppose we are in a situation where $n \le d$. In particular, if $x_1, \ldots, x_n$ are linearly independent, then for any $y_1, \ldots, y_n$, we can find $\theta \in \mathbb{R}^d$ such that

$$\theta' \left[x_1 | x_2 | \cdots | x_n\right] = [y_1, y_2, \ldots, y_n],$$

and hence $\mathrm{sign}(\theta' x_i) = y_i$. If we can fit *any* labels, we should not expect to predict the labels of subsequent points accurately.

The following theorem makes this precise, by showing that, under these conditions, any method will perform poorly. It shows a little more: when n/d is not too large, there is a probability distribution that makes the risk at least as large as $\Omega(n/d)$.

Theorem 3.1. For any $n \ge 1$ and any mapping $f_n : \mathbb{R}^d \times \left(\mathbb{R}^d \times \{\pm 1\}\right)^2 \to \{\pm 1\}$, there is a probability distribution on $\mathbb{R}^d \times \{\pm 1\}$ for which some linear threshold function $f \in F$ has $R(f) = 0$ but

$$\mathbb{E} R(f_n) \ge \min\left(\frac{n-1}{2n}, \frac{d-1}{2n}\right)\left(1 - \frac{1}{n}\right)^n.$$

This is an example of a minimax lower bound, since it gives a lower bound on

$$\min_{f_n} \max_{P} \mathbb{E} R(f_n),$$

where the max is over all P for which some $f \in F$ has zero risk, and the min is over all methods f_n—not just the perceptron algorithm, or other algorithms that predict with linear threshold functions, but any deterministic prediction rule.

One thing to notice about the result is that the probability distribution is allowed to depend on n. So it's not saying that, for every P the risk decreases to zero at a rate slower than $1/n$.