Lecture 11: Linear Classification and Support Vector Machines

Akshay Krishnamurthy
akshay@cs.umass.edu

October 12, 2017

1 Recap

Last time we talked about boosting and derived a new way to explain boosting's generalization behavior using the theory of margins. It turns out that margins play a fundamental role in linear prediction more generally, which we will investigate today. To build some intuition, we showed last time that

$$\mathbb{P}_{\mathcal{D}}[yf(x) \le 0] \le \mathbb{P}_S[yf(x) \le \theta] + O\left(\sqrt{\frac{\log(|\mathcal{H}|)}{n\theta^2} + \frac{\log(1/\delta)}{n}}\right) \tag{1}$$

for any function $f \in \text{conv}(\mathcal{H})$. Thus a sensible strategy is to find a function that has low training error but also has good "empirical margin distribution", so that

$$\mathbb{P}_S[yf(x) \le \theta]$$

is small for big θ. Adaboost does have this property:

Theorem 1. *Let $\gamma_t = 1/2 - \epsilon_t$ as in the usual AdaBoost. Then*

$$\mathbb{P}_S[yF(x) \le \theta] \le \prod_{t=1}^{T} \sqrt{(1+2\gamma_t)^{1+\theta}(1-2\gamma_t)^{1-\theta}}.$$

This generalizes the training error bound we proved for AdaBoost, which is just with $\theta = 0$.

Another way to achieve a good bound here is to "maximize the margin," which means make θ as large as possible while still having $\mathbb{P}_S[yf(x) \le \theta] = 0$. This maximum-margin intuition motivates support vector machines, although the setup is somewhat different.

2 Perceptron

Let's start out with something simpler and not worry too much about the statistical consequences of margins. Consider the realizable setting for binary classification where the hypothesis class is the set of linear separators. Here $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{-1, +1\}$ and $\mathcal{H} = \{x \to \text{sign}\{\langle w, x\rangle\}, w \in \mathbb{R}^d\}$. We are in the realizable case, so we have a dataset $(x_1, y_1), \ldots, (x_n, y_n)$ and we know that there exists a linear separator that has zero training error. As such, we can compute the ERM by setting up a linear feasibility problem

$$?\exists w \text{ s.t. } \forall i \in [n], y_i\langle w, x_i\rangle \ge 0$$

This is a linear feasibility problem with n constraints and can hence be solved efficiently using standard techniques (e.g., ellipsoid or simplex).

Another way to solve the same program is with the Perceptron algorithm. The algorithm is iterative and starts with $w_0 = 0$. Then on iteration t, if some example x_i is predicted incorrectly ($y_i \ne \text{sign}\{\langle w_t, x_i\rangle\}$) we update

$$w_{t+1} \leftarrow w_t + y_i x_i.$$

If no example is predicted incorrectly then we have an ERM and we are done.

Claim 2. *Suppose the data is linearly separable by $w^\star$ and define $r = \max_i \|x_i\|^2$ and $\gamma = \min_i(\langle w^\star, x_i\rangle y_i / \|w^\star\|_2)$. Then the perceptron algorithm is guaranteed to terminate in at most r^2/γ^2 updates.*

Proof. The basic idea here is to analyze the angle between w_{t+1} and $w^\star$ and understand how it is shrinking at every iteration. Specifically we will show that if the algorithm hasn't terminated for T rounds then,

$$\cos\angle(w_T, w^\star) = \frac{\langle w_T, w^\star\rangle}{\|w_T\|\|w^\star\|} \geq \frac{\gamma\sqrt{T}}{r}$$

This proves the result since the cosine can be at most one. Let us start by deriving a lower bound on the numerator.

$$\langle w_{t+1}, w^\star\rangle = \langle w_t, w^\star\rangle + \langle y_i x_i, w^\star\rangle \geq \langle w_t, w^\star\rangle + \gamma\|w^\star\|_2$$

And hence $\langle w_T, w^\star\rangle \geq T\gamma\|w^\star\|_2$. For the denominator, we know that $\|w_{t+1}\|_2^2 = \|w_t + y_i x_i\|_2^2 \leq \|w_t\|_2^2 + r^2$ since the cross term is negative. Thus after T iterations, we have

$$T\gamma\|w^\star\|_2 \leq \langle w_T, w^\star\rangle \leq \|w_T\|_2\|w^\star\|_2 \leq \sqrt{T}r\|w^\star\|_2$$

If $T > r^2/\gamma^2$ this is a contradiction, which means the algorithm must terminate. $\square$

The quantity γ is kind of a margin here and in Perceptron it controls the running time of the algorithm. To relate this margin to the one we saw in boosting, observe that $w^\star/\|w^\star\|$ has unit Euclidean norm, so for a unit-normed w, the margin here is

$$\gamma = \min_i f_w(x_i) y_i$$

where $f_w(x) = \langle w, x\rangle$ for pattern matching with the boosting result. The main differences are (a) we are using Euclidean geometry instead of the ℓ_1 geometry that we used in boosting, and (b) we are getting a "worst-case" margin, rather than something more refined about the margin distribution. Or in the boosting theorem, if we use this value then we get that the first term is zero, but recall that the boosting theorem applies for any value of γ.

Turning to statistics, if we just solve the linear program using say Ellipsoid, we have no guarantee about the margin of this function, so the best thing we can say is through the VC theorem.

Proposition 3. *The weight vector $\hat{w}$ obtained by solving the linear program satisfies with probability at least $1-\delta$*

$$\mathbb{P}_{x\sim\mathcal{D}}[\text{sign}\{\langle \hat{w}, x\rangle\} \neq \text{sign}\{\langle w^\star, x\rangle\}] \leq O\left(\frac{d\log(n/\delta)}{n}\right).$$

Note that this bound is dimension-dependent, which is really bad in high dimensions. On the other hand, the margin bound in Eq. (1) is *dimension independent* and can be much better when the margin is big, or when we're in infinite dimension (which we'll get to soon). The hope is we can also get a margin bound here.

For Perceptron it turns out to be challenging. First, another way to express the margin is in terms of the norm of the predictor w. Let

$$B = \min\{\|w\|_2 \text{ s.t. } \forall i \in [n], y_i\langle w, x_i\rangle \geq 1\}.$$

Then $B = 1/\gamma$ since you will always make one constraint tight, so the numerator in the definition of γ is exactly 1.

For Perceptron, we know that by the setting of T that $\|w_{T+1}\| \leq r^2/\gamma$, but unfortunately we do not know that it satisfies the constraints here. Instead all we know is that $y_i\langle w_{T+1}, x_i\rangle \geq 0$, which is not adequate to apply Eq. (1).

3 Hard-margin SVM

However, seeing the definition of B, directly motivates the Hard-Margin SVM. Since we know that when we have favorable margin γ, or equivalently low $\|w\|_2$, we have good generalization, why not try to minimize the norm directly?

$$\text{minimize}\|w\|_2^2 \text{ s.t. } \forall i \in [n], y_i\langle w, x_i\rangle \geq 1 \tag{2}$$

This is a quadratic optimization problem that again we can solve efficiently using a number of methods.

Generalization. Let us turn first to statistical issues and consider the realizable case. Observe that since $w^\star$ is feasible, we know that the solution we find must have norm at most $\|w^\star\|$. As a consequence, we immediately ensure the same margin and we can plug this into whatever margin-based generalization bound we can prove.

In particular, we get

Theorem 4. *Let $\mathcal{D}$ be a distribution over $B_2(R,d) \times \{-1,+1\}$ that is linearly separable with margin γ (e.g., there exists $w^\star$ with $\|w^\star\| \le 1/\gamma$ such that with probability 1 over the choice $(x,y) \sim \mathcal{D}$, we have $y\langle w^\star, x\rangle \ge 1$). Then the 0/1 error of Hard-Margin SVM is at most*

$$\sqrt{\frac{4R^2}{n\gamma^2}} + \sqrt{\frac{2\log(2/\delta)}{n}}.$$

1. The result is kind of like the margin theorem for boosting. The first term is zero since we'll use the margin value γ, for which we know that all of the points have larger margin value (by definition). The other main difference is that we replaced $\log(|\mathcal{H}|)$ with R^2, but this is a consequence of the Euclidean versus ℓ_1 geometry. However, note that we are getting a dimension independent bound! You'll prove a simpler version of this theorem in Homework 3.
2. But note that the dimension-dependent bound has the $O(1/n)$ convergence since we are in the realizable setting while the margin bounds have $O(\sqrt{1/n})$ convergence. Two points here are (1) both bounds hold simultaneously, for hard-margin SVM, so we get the best of both worlds, and (2) we're not likely to be in the realizable case anyway, so the $O(1/n)$ bound isn't really relevant. On the other hand, the Hard-margin SVM bound requires realizability. We will attempt to relax this in two different ways.
3. I am not aware of a $O(1/n)$-margin type bound. But it very well might exist. If anyone does run across such a thing, please do send me a pointer.

3.1 Kernel Trick

There are many deep insights about hard-margin SVM that are worth investigating. These typically follow from convex duality and we discuss some of these here. First, by introducing Lagrange parameters $\lambda_1, \ldots, \lambda_n$ we can re-write the problem as

$$\min_w \max_{\lambda \succeq 0} \|w\|^2 + \sum_{i=1}^n \lambda_i (1 - y_i \langle w, x_i \rangle)$$

This program is equivalent to Eq. (2), since if w is infeasible it violates some constraint, and we set that λ to infinity and drive the objective to infinity as well. On the other hand if w is feasible, we should just set all the λs to zero.

It will be helpful to multiply the norm by $1/2$, which again doesn't change the solution. Now, we always get

$$\min_w \max_{\lambda \succeq 0} \frac{1}{2}\|w\|^2 + \sum_{i=1}^n \lambda_i (1 - y_i \langle w, x_i \rangle) \ge \max_{\lambda \succeq 0} \min_w \frac{1}{2}\|w\|^2 + \sum_{i=1}^n \lambda_i (1 - y_i \langle w, x_i \rangle)$$

This is called *weak duality.* It turns out that this is actually an equality, which is known as *strong duality*, but we will not discuss this too much. The main point is, that working with this *dual problem*, we can actually analytically solve for w in terms of the λ by taking derivative. Specifically, fixing $\lambda_1, \ldots, \lambda_n$, setting the gradient with respect to w equal to zero implies

$$w - \sum_{i=1}^n \lambda_i y_i x_i = 0 \Rightarrow w = \sum_{i=1}^n \lambda_i y_i x_i$$

This reveals that the optimal w for any λ is in the linear span of the feature vectors $x_1, \ldots, x_n$. Then we can plug in this solution to re-write the dual problem as

$$\max_{\lambda \succeq 0} \frac{1}{2}\|\sum_{i=1}^n \lambda_i y_i x_i\|_2^2 + \sum_{i=1}^n \lambda_i (1 - y_i \langle \sum_{j=1}^n \lambda_j y_j x_j, x_i \rangle) = \max_{\lambda \succeq 0} \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i,j=1}^n \lambda_i \lambda_j y_i y_j \langle x_i, x_j \rangle.$$

Now we make a few observations, first the optimization problem only involves the feature vectors in terms of inner products $\langle x_i, x_j \rangle$. Second since we know that the optimal w is in the linear span of the features, and since we are doing linear classification, making a prediction also only requires inner products. Finally, we saw from the generalization analysis that the dimension of the space did not really matter from the statistical perspective. This discussion implies that it doesn't really matter from the computational perspective either, and this motivates the so-called *kernel trick*.

The idea here is to replace inner product with inner product in a much higher-dimensional space via a feature map $x \to \Phi(x)$. If inner products $\langle \Phi(x), \Phi(x') \rangle$ can be computed efficiently, then we can run the SVM and make predictions. Specifically, suppose there is a feature map Φ and a kernel function K such that $K(x, x') = \langle \Phi(x), \Phi(x') \rangle$, then we can re-write the dual problem as

$$\max_{\lambda \succeq 0} \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{n} \lambda_i \lambda_j y_i y_j K(x_i, x_j)$$

And computationally it doesn't matter what dimension Φ maps to. Of course, for the margin analysis it doesn't matter what dimension Φ maps to either, it just matters that we have a good margin in the feature space induced by Φ. Together, this means we can even map into infinite dimension, provided we can compute inner products efficiently.

Example 1 (RBF kernel)**.** *Let $K(x, y) = \exp(-\|x - y\|_2^2/2)$ in d dimensions with $\|x\| = \|y\| = 1$. Then by Taylor expansion*

$$K(x, y) = \exp(-1) \sum_{k=1}^{\infty} \frac{\langle x, y \rangle^k}{k!}$$

Observe that $\langle x, y \rangle^k$ is the inner product between the "tensorized" versions of x an y, which have all polynomials with k terms. Thus the Gaussian/RBF kernel lifts the problem to infinite dimensions by expressing x in terms of all possible polynomial terms.

The downside with the kernel trick is that you have to operate with an $n \times n$ matrix, which can be quite bad for large sample sizes. This motivates a lot of research on matrix approximation and specifically approximating kernel matrices.

4 Soft-margin SVM

If the training set is not linearly separable, the Hard-margin SVM can perform poorly, since there may not be any feasible solutions. Instead, the typical approach is to introduce slack parameters, one for each variable, that relax the constraint about classifying correctly with margin. Instead we solve

$$\min_{w,\xi} \lambda \|w\|_2^2 + \frac{1}{n} \sum_{i=1}^{n} \xi_i \text{ s.t. } \forall i \in [n], y_i \langle w, x_i \rangle \geq 1 - \xi_i, \xi_i \geq 0.$$

There are two things to keep in mind here. First, if the data is not linearly separable, then something is required to ensure feasibility, and introducing the slack variables does it. Second, this can be beneficial even when the problem is linearly separable, since we might be able to find a much larger margin if we don't care too much about a few points. For example, maybe there is just one point or a few that are hard to classify, then we can incur some slack penalty for them and push to a much bigger margin.

This form of program also has a connection to loss minimization. Observe that for any w, the minimization over ξ decomposes across the n samples, and the best thing to do is set $\xi = 0$ if $y \langle w, x_i \rangle \geq 1$ and otherwise we should set $\xi = 1 - y_i \langle w, x_i \rangle$. This is exactly the *hinge* loss $\ell(y, y') = \max(0, 1 - yy')$ for binary classification:

$$\min_{w} \lambda \|w\|_2^2 + \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i \langle w, x_i \rangle).$$

Thus soft-margin SVM is equivalent to solving a regularized ERM problem with the hinge loss.

Kernel trick. Soft-margin SVM can also be kernelized, as you will see on the homework.