Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.S064 Introduction to Machine Learning

Phase 2: classification (lecture 7)

Linear classifiers and regularization

We have so far discussed linear classifiers over examples $x \in \mathcal{R}^d$. These classifiers can be written in the form

$$h(x; \theta, \theta_0) = \text{sign}\big(\theta \cdot x + \theta_0 \big) \tag{1}$$

with parameters $\theta \in \mathcal{R}^d$ and $\theta_0 \in \mathcal{R}$. Given a training set of labeled examples, $S_n = \{(x^{(t)}, y^{(t)}), t = 1, \ldots, n\}$, we can estimate the classifier parameters by minimizing the empirical risk

$$R_n(\theta, \theta_0) = \frac{1}{n} \sum_{t=1}^{n} \text{Loss}\big(y^{(t)}(\theta \cdot x^{(t)} + \theta_0) \big) \tag{2}$$

The loss function could be the zero-one loss $\text{Loss}_{0/1}(z) = [\![\, z \leq 0 \,]\!]$ (resulting in classification error) or the hinge loss $\text{Loss}_h(z) = \max\{1 - z, 0\}$ (which we adopt here). One problem with this setup is that there are potentially many classifiers, i.e., many values for θ, θ_0, that result in the same empirical risk. For example, when the training examples are linearly separable, there are many classifiers that attain zero empirical risk. Which one of these should we choose?

The empirical risk does not sufficiently constrain the parameters. We ran into a similar (ill-posed) situation in the context of linear regression. We could remedy the problem here in the same way, i.e., by adding a regularization term that biases the parameters towards a default answer such as the all-zero parameters. In this case, we would find parameters that minimize

$$\frac{\lambda}{2} \|\theta\|^2 + \frac{1}{n} \sum_{t=1}^{n} \text{Loss}_h\big(y^{(t)}(\theta \cdot x^{(t)} + \theta_0) \big) \tag{3}$$

instead of the empirical risk alone. The regularization parameter $\lambda > 0$ balances the two terms, how much we favor zero parameters at the expense of the empirical risk. Note that the regularization term only applies to θ, not θ_0. The reason for this is that while θ specifies the orientation of the decision boundary, and how quickly the linear function $\theta \cdot x + \theta_0$ changes off the boundary, θ_0 is used to specify how far from the origin the boundary is located. Without prior information about where the points are, placing constraints on the location of the boundary may not be useful. The classification method that uses the regularized objective Eq.(3) is known as the *support vector machine* (there are many versions, this one in a primal form). We will develop here a general, geometric approach to such classifiers.

Maximum margin separators

Let's begin by assuming that the training examples are linearly separable. Clearly, in this case there are many linear separators that all perfectly classify the training examples. For example, if we run the perceptron algorithm with two different initializations, or cycle through the training examples in a slightly different order, we would potentially arrive at a different classifier (linear separator). Here we are trying to explicitly find a single linear separator that is "optimal" in some sense.

If we imagine that (yet unseen) test examples are noisy versions of the training examples, then it would seem sensible to draw the linear decision boundary in a manner that a) classifies all the training examples correctly, and b) is maximally removed from all the training examples. This is known as the *maximum margin separator* or the *optimal hyperplane*.

Suppose (θ, θ_0) are parameters of a linear classifier that is correct on all the training examples, i.e., $y^{(t)}(\theta \cdot x^{(t)} + \theta_0) > 0$, $t = 1, \ldots, n$. In this case, we can measure the distance of each training point to the decision boundary by

$$\gamma^t(\theta, \theta_0) = \frac{y^{(t)}(\theta \cdot x^{(t)} + \theta_0)}{\|\theta\|} \tag{4}$$

To see this, note that $\theta \cdot x + \theta_0$ is a linear function of x that increases at rate $\|\theta\|$ as we move x off the decision boundary $\theta \cdot x + \theta_0 = 0$ (towards the positive, θ direction). So, to get the distance traveled orthogonally to the boundary, we simply divide $\theta \cdot x + \theta_0$ by $\|\theta\|$. Note that $\gamma^t(\theta, \theta_0)$ is (clearly) a function of the parameters as well as the point itself. Now, to find the maximum margin separator, we should search for parameters (θ, θ_0) that maximize

$$\min_{t=1,\ldots,n} \gamma^t(\theta, \theta_0) \tag{5}$$

In other words, we are trying to maximize the minimum distance to the boundary. While this looks like a challenging problem to optimize, it can be formulated more simply as a *quadratic program* (quadratic objective, linear constraints)

$$\text{(primal)} \qquad \min \frac{1}{2}\|\theta\|^2 \text{ subject to } y^{(t)}(\theta \cdot x^{(t)} + \theta_0) \geq 1, \;\; t = 1, \ldots, n \tag{6}$$

Note that the *classification constraints* $y^{(t)}(\theta \cdot x^{(t)} + \theta_0) \geq 1$ ensure that we consider only θ, θ_0 for which all the training examples are correctly classified. The choice of 1 on the right hand side is arbitrary (any positive number would do). Subject to these constraints, we minimize the norm $\|\theta\|$.

Let's understand this quadratic programming problem geometrically. Figure 1a) shows a "valid" setting of θ, θ_0. In addition to the decision boundary $\theta \cdot x + \theta_0 = 0$, we have drawn the associated *margin boundaries*, $\theta \cdot x + \theta_0 = 1$ and $\theta \cdot x + \theta_0 = -1$. In order for θ, θ_0 to satisfy the classification constraints, all the training examples must appear on or outside the two margin boundaries. For example, all the positive training examples must be on the side of the margin boundary $\theta \cdot x + \theta_0 = 1$ that θ points to. As displayed, the two margin boundaries are parallel to the decision boundary. If they were not, they would have to cross at some point, and thus have a point x in common. But there's no x for which $\theta \cdot x + \theta_0 = 1$ and $\theta \cdot x + \theta_0 = 0$ can hold at the same time. In addition, the two margin boundaries lie exactly $1/\|\theta\|$ away from the decision boundary, on the opposite sides. To see this, let $x^{(0)}$ be a point that is exactly on the decision boundary, i.e., $\theta \cdot x^{(0)} + \theta_0 = 0$, and let $u = \theta/\|\theta\|$ be a unit vector in the direction of θ. If we move from $x^{(0)}$ exactly length $1/\|\theta\|$ in the direction of u, we get

$$\theta \cdot (x^{(0)} + u/\|\theta\|) + \theta_0 = \theta \cdot x^{(0)} + \theta \cdot u/\|\theta\| + \theta_0 = \underbrace{\theta \cdot x^{(0)} + \theta_0}_{=0} + \underbrace{\theta \cdot \theta/\|\theta\|^2}_{=1} = 1 \qquad (7)$$

Since the margin boundaries are $1/\|\theta\|$ away from the decision boundary, by minimizing $\|\theta\|$, we push the margin boundaries apart. At some point, they cannot be pushed further without violating the classification constraints. At this point, the boundary "locks" into the unique maximum margin solution. This is shown in Figure 1b). Some of the points will necessarily lie exactly on the margin boundaries. They are called *support vectors* (circled in the figure).

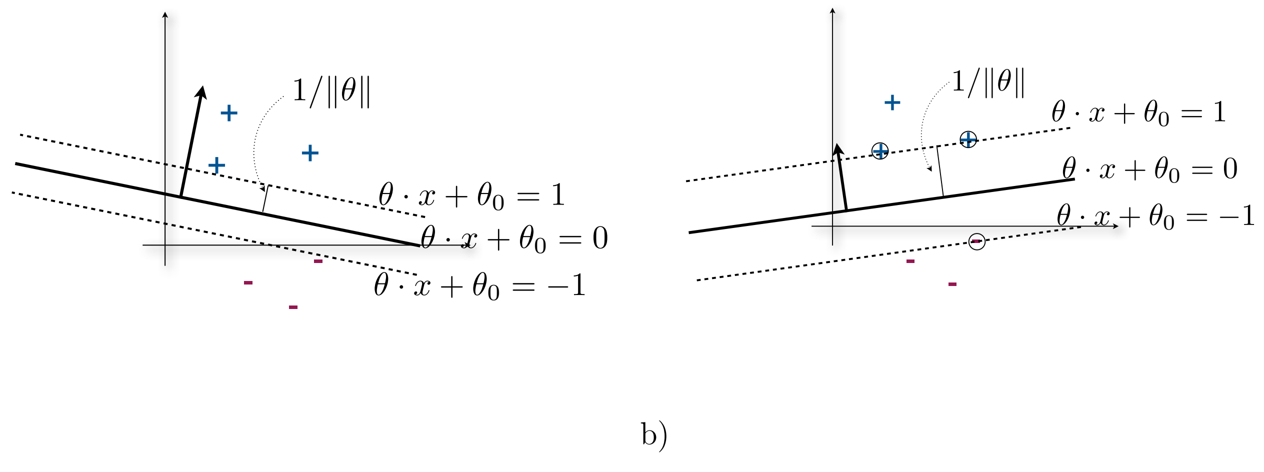

Figure 1: a) a linear separator with margin boundaries that satisfy the classification constraints. b) the maximum margin separator

In many cases the dimension d of the parameters (and examples) is quite large. This makes the quadratic programming problem expressed in Eq.(6) a bit challenging to solve. We can, however, solve its *dual problem*. This change from primal to dual form highlights two ways of representing the parameters θ: directly in terms of coordinates as in the vector θ (primal) or in terms of examples $\theta = \sum_{t=1}^{n} \alpha_t y^{(t)} x^{(t)}$ (dual). We have already seen the "dual form" in the context of the perceptron algorithm. Specifically, we could write the parameters after k updates as

$$\theta^{(k)} = \sum_{t=1}^{n} \alpha_t^{(k)} y^{(t)} x^{(t)} \tag{8}$$

where $\alpha_t^{(k)} \geq 0$ specifies the number of times we had made a mistake on $(x^{(t)}, y^{(t)})$. Each mistake resulted in adding exactly $y^{(t)}x^{(t)}$ to the parameter vector, and the above expression simply adds those updates together. In the context of maximum margin separators, we can also write $\theta = \sum_{t=1}^{n} \alpha_t y^{(t)} x^{(t)}$, where $\alpha_t \geq 0$. However, in this case, α_t are no longer integer valued. Instead, as real numbers, they smoothly change how much we rely on a particular training example. Note that since the maximum margin solution appears to depend only on a few points – the support vectors – most of the α_t coefficients will be exactly zero (as if the corresponding point was not included in the training set). The larger the value α_t is, the harder it was to satisfy the associated classification constraint. Recall from the perceptron analysis that adding $y^{(t)}x^{(t)}$ increases the agreement, thus getting us closer to satisfying the specific classification constraint.

The non-negative coefficients $\alpha_1, \ldots, \alpha_n$ are actually so called Lagrange multipliers associated with the classification constraints. Let's derive this a bit more formally. For simplicity, we drop the bias parameter θ_0, and try to find the dual problem for

$$\text{(primal)} \qquad \min \frac{1}{2}\|\theta\|^2 \;\text{ subject to }\; y^{(t)}(\theta \cdot x^{(t)}) \geq 1, \;\; t = 1, \ldots, n \tag{9}$$

As the first step, we construct so called Lagrangian. This is an *unconstrained* minimization problem

$$L(\theta, \alpha) = \frac{1}{2}\|\theta\|^2 + \sum_{t=1}^{n} \alpha_t [1 - y^{(t)}(\theta \cdot x^{(t)})] \tag{10}$$

where we have introduced non-negative coefficients $\alpha_1, \ldots, \alpha_n$ (Lagrange multipliers) that serve to enforce the classification constraints. They behave like adversaries. Our goal is to minimize $L(\theta, \alpha)$ while the adversaries (the α's) are trying to maximize it. If for a given choice of θ, we have $[1 - y^{(t)}(\theta \cdot x^{(t)})] > 0$, i.e., the classification constraint is violated, then the adversary can choose a large value for α_t and increase $L(\theta, \alpha)$ arbitrarily, effectively prohibiting us from choosing θ values that violate the constraint. If, on the other hand, the constraint is satisfied for our choice of θ, the best the adversary can do is to set $\alpha_t = 0$. If we now solve θ as a function of $\alpha's$, we get

$$\frac{\partial}{\partial \theta} L(\theta, \alpha) = \theta - \sum_{t=1}^{n} \alpha_t y^{(t)} x^{(t)} = 0 \tag{11}$$

Thus, indeed, θ has the form discussed above. Plugging this θ back into the Lagrangian, we obtain the dual problem

$$\text{(dual)} \qquad \max \sum_{t=1}^{n} \alpha_t - \frac{1}{2}\sum_{t=1}^{n}\sum_{t'=1}^{n} \alpha_t \alpha_{t'} y^{(t)} y^{(t')} (x^{(t)} \cdot x^{(t')}) \tag{12}$$

$$\text{subject to } \; \alpha_t \geq 0, \;\; t = 1, \ldots, n \tag{13}$$

which is again a quadratic program but with simpler "box" constraints. By maximizing this dual objective we find the setting of α_t's – the adversaries – that ensure that the resulting $\hat{\theta} = \sum_{t=1}^{n} \hat{\alpha}_t y^{(t)} x^{(t)}$ satisfies the classification constraints. The solution satisfies the following *complementary slackness* constraints:

$$\hat{\alpha}_t > 0 \quad : \quad y^{(t)} \left(\sum_{t'=1}^{n} \hat{\alpha}_{t'} y^{(t')} x^{(t')} \right) \cdot x^{(t)} = 1 \;\; \text{(support vector)} \tag{14}$$

$$\hat{\alpha}_t = 0 \quad : \quad y^{(t)} \left(\sum_{t'=1}^{n} \hat{\alpha}_{t'} y^{(t')} x^{(t')} \right) \cdot x^{(t)} \geq 1 \;\; \text{(non-support vector)} \tag{15}$$

The classification constraints are satisfied with equality for support vectors as these points lie exactly on the margin boundaries. $\hat{\alpha}_t = 0$ for all the points that are beyond the margin boundaries.

The main reason for consider the dual problem is that now the optimization problem we have to solve depends on the examples only via their inner products, i.e., via $(x^{(t)} \cdot x^{(t')})$. If someone gave us these without the examples, we could still solve the learning problem. It turns out that in many cases the inner products can be evaluated efficiently even for very high dimensional vectors even though they couldn't be used *as is* in the primal problem. We will discuss such inner products or *kernels* later on.

Sparsity and generalization

The fact that the resulting α's are *sparse*, i.e., most of them are exactly zero, may help generalization. Consider, for example, leave-one-out cross-validation error as a surrogate measure for how well the maximum margin classifier might generalize:

$$\text{LOOCV} \leq \frac{\#\text{ of support vectors}}{n} \tag{16}$$

In other words, the fewer support vectors we get, potentially the better we generalize. This bound is easy to understand from figure 1b). If we hold out one point from the training set, and this point is not a support vector, then our solution doesn't change. It was classified correctly when included, and will continue to be classified correctly when excluded from the training set (as the solution remains the same). We can only obtain an error if we hold out a support vector (but not necessarily even then). The above bound is conservative in this sense.