# COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Michael Y. Li

Lecture #13
March 25, 2019

## 1 Introduction

In last lecture, we did a thorough empirical and theoretical analysis of Adaboost's performance. We saw that Adaboost tends to maximize the *margins*, which capture the classifier's confidence in its classification decisions. We then proved that large margins lead to low generalization error, which helps explain Adaboost's classification accuracy. In this lecture, we will study a learning algorithm, the *support vector machine*, that explicitly maximizes the margins.

## 2 Background

Consider the problem of classifying a sample of points in Euclidean space labeled as positive and negative. One intuitive approach to this classification problem is to find a line (or hyperplane in higher dimensions) that perfectly separates the two classes of points. All points above this line will be classified as positive and all points below this line will be classified as negative.

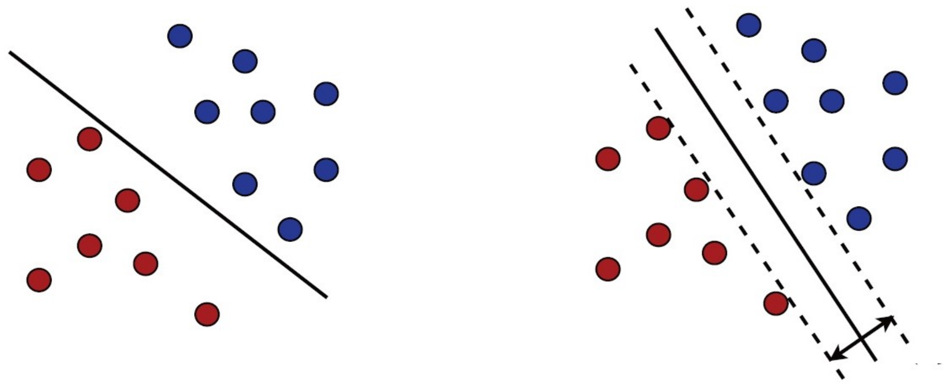

Image from Chapter 5 of Foundations of Machine Learning Mohri et al 2nd edition

However, there are many lines that can perfectly separate these two classes of points. So intuitively, we not only want to find a line that can separate these points but also a line that is robust or insensitive to small perturbations of the data. Consider the lines above. While both lines are consistent with the data, the line on the right has a larger margin with respect to points on either side of the boundary while the line on the left is fairly close to classifying a negative point as positive instead. This example leads to the intuition that small, local movements around the sampled points should not change the classification decisions of a good classifier.

To try to formalize this intuition, we want all points within a $\delta$ sized ball around a given point $\mathbf{x}_i$ to be classified with the same label $y_i$. We now want to find a hyperplane given this additional condition. Finding a hyperplane that satisfies this condition turns out to be equivalent to finding a hyperplane that correctly classifies all of the training points, and that is also at least distance $\delta$ from all of the training points, where we want $\delta$ to be as large as possible. This hyperplane can be, in a sense, purely defined by the points that are exactly a distance $\delta$ away. We call this distance $\delta$ the *margin* of our classifier, and the points that lie exactly a distance $\delta$ away from the hyperplane are called the *support vectors*.

### 2.1 Why does maximization of margins make for a good classifier? Looking into VC-Dimension

We have seen that there are three properties that tend to be important for a good classifier. The classifier must have sufficient data, fit the data well, and be relatively simple.

We will only consider complexity since we assume that the amount of data is fixed and we are considering hyperplanes with zero training error. From a previous class, we learned that the VC-dimension of $n$-dimensional linear threshold functions is $n$. This is not desirable because $n$ can become large in practice. However, the VC-dimension of linear threshold functions with margin $\delta$ is $\leq 1/\delta^2$, assuming all points lie within a unit ball so that $\|\mathbf{x}\|_2 \leq 1$. Observe that this term **does not** depend on the dimensionality of our linear threshold function! Furthermore, increasing $\delta$ will decrease the complexity of our hypothesis. This argument motivates analysis of a max-margin hyperplane.

## 3 Finding the Max-Margin Hyperplane

We'll begin by formalizing the learning problem, discuss some simplifications, and then finally study the properties of the solution to the learning problem.

### 3.1 Linear Algebra

Recall the following facts from linear algebra: $\|\mathbf{x}\|_2$ denotes the $\ell_2$-norm of the vector $\mathbf{x}$. A hyperplane is defined by a vector $\mathbf{v}$ that is normal to its surface. The distance of a point $\mathbf{x}$ to $\mathbf{v}$ is given by the absolute value of the dot product $|\mathbf{v} \cdot \mathbf{x}|$. Note that $\mathbf{v} \cdot \mathbf{x}$ gives a signed distance. If $\mathbf{v} \cdot \mathbf{x} > 0$, then $\mathbf{x}$ lies above the hyperplane, relative to the direction of $\mathbf{v}$. If $\mathbf{v} \cdot \mathbf{x} = 0$, then $\mathbf{x}$ lies on the hyperplane. If $\mathbf{v} \cdot \mathbf{x} < 0$, then $\mathbf{x}$ lies below the hyperplane.

### 3.2 Formalization of Learning Problem

#### 3.2.1 Form 1

We'll begin with a vanilla formalization of the learning problem and iteratively simplify. Let $\mathbf{v}$ be the unit vector defining our hyperplane. Given a sample of $m$ training examples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ with $\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in \{-1, +1\}$ we want to maximize the margin $\delta$ subject to the following constraints:
$\|\mathbf{v}\| = 1$
$\mathbf{v} \cdot \mathbf{x}_i > \delta$ if $y_i = +1$
$\mathbf{v} \cdot \mathbf{x}_i < -\delta$ if $y_i = -1$
We can turn the last two constraints into a single constraint $y_i(\mathbf{v} \cdot \mathbf{x}_i) \geq 1$ that should hold for all $i$; this constraint basically forces the true label and the classifier's label to be the same but also to have margin at least $\delta$. Our output classifier will be $h(\mathbf{x}) = sign(\mathbf{v} \cdot \mathbf{x})$.

### 3.2.2 Form 2

We can next divide through by $\delta$ to obtain $y_i(\mathbf{w} \cdot \mathbf{x}_i) \geq \delta$ where $\mathbf{w} = \frac{\mathbf{v}}{\delta}$. We can now reformulate the optimization problem in terms of $\mathbf{w}$ because $\|\mathbf{w}\| = \frac{1}{\delta}$ since $\|\mathbf{v}\| = 1$. We wanted to maximize the margin $\delta$ in the previous formalization which is equivalent to minimizing $\|\mathbf{w}\|$. We multiply by $\frac{1}{2}$ and square for convenience. This yields the following optimization problem.
$\min \frac{1}{2}\|\mathbf{w}\|^2$
subject to:
$y_i(\mathbf{w} \cdot \mathbf{x}_i) \geq 1 \ \forall i$
In this new formulation, we removed $\delta$ and also removed the constraint that $\|\mathbf{v}\| = 1$. If $\mathbf{w}$ is the solution to this optimization problem, then our classifier will be $h(\mathbf{x}) = sign(\mathbf{w} \cdot \mathbf{x})$.

## 3.3 Lagrangian

A standard technique to solve constrained optimization problems of this form is to construct the Lagrangian function. To construct the Lagrangian function, we move all terms in the constraints to the lefthand side of the inequality, isolating zero on the right to get $b_i(\mathbf{w}) \geq 0$ where $b_i(\mathbf{w}) = y_i(\mathbf{w} \cdot \mathbf{x}_i) - 1$. We then subtract this from our objective function.
The Lagrangian function is thus:

$$L(\mathbf{w}, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i^m \alpha_i b_i(\mathbf{w})$$

The new variables $\alpha_i$ are the Lagrange multipliers where $\alpha_i \geq 0$.

We will next see that the optimization problem in the previous section is identical to solving:

$$\min_{\mathbf{w}} \max_{\boldsymbol{\alpha}} L(\mathbf{w}, \boldsymbol{\alpha})$$

Here and in all similar expressions that follow below, it is understood that the minimization is over all vectors $\mathbf{w} \in \mathbb{R}^n$ and the maximization is over all vectors $\boldsymbol{\alpha}$ with $\alpha_i \geq 0 \ \forall i$.

## 3.4 Min-max games

The last min-max optimization problem can be viewed as two players Mindy and Max who are playing the following game. Mindy wants to choose $\mathbf{w}$ to minimize $L(\mathbf{w}, \boldsymbol{\alpha})$ while Max wants to choose $\boldsymbol{\alpha}$ to maximize $L(\mathbf{w}, \boldsymbol{\alpha})$. Mindy plays first and Max plays second knowing what Mindy chose. Let's consider what strategies Mindy would play if she's best responding. Mindy clearly does not want to choose some $\mathbf{w}$ such that $b_i(\mathbf{w}) < 0$. Otherwise, Max could choose infinitely large $\alpha_i$ to place on $b_i(\mathbf{w})$ which would cause the objective function to be infinitely large. If Mindy chooses $\mathbf{w}$ so that $b_i(\mathbf{w}) = 0$, then Max's choice of $\alpha_i$ is irrelevant since it will be multiplied by zero. If Mindy chooses $b_i(\mathbf{w}) > 0$, then it's now in Max's best interest to set $\alpha_i = 0$ since otherwise he is decreasing the objective function. So Max's best response is to set $\alpha_i = 0$. Observe in either case that $\alpha_i b_i(\mathbf{w}) = 0$. Thus the right term of the Lagrangian disappears and our optimization problem reduces to the previous formulation.

Why is the new formulation useful? If we study it more carefully, we can characterize the solution more precisely. First observe that it's certainly better for Mindy to play second, because she gets to know exactly what Max played instead of "guessing". Thus, it follows that:

$$\max_{\boldsymbol{\alpha}} \min_{\mathbf{w}} L(\mathbf{w}, \boldsymbol{\alpha}) \ \leq \min_{\mathbf{w}} \max_{\boldsymbol{\alpha}} L(\mathbf{w}, \boldsymbol{\alpha})$$

Again, this is because if Mindy has the additional knowledge of Max's strategy she does at least as well as if she played without that knowledge. For functions $L$ that satisfy certain convexity properties, this turns out to be an equality.

$$\max_{\boldsymbol{\alpha}} \min_{\mathbf{w}} L(\mathbf{w}, \boldsymbol{\alpha}) \ = \min_{\mathbf{w}} \max_{\boldsymbol{\alpha}} L(\mathbf{w}, \boldsymbol{\alpha})$$

Our current optimization problem turns out to have this property. What does this equality say about our solution? Define $\mathbf{w}^*$ as:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \max_{\boldsymbol{\alpha}} L(\mathbf{w}, \boldsymbol{\alpha})$$

Define $\boldsymbol{\alpha}^*$ as:

$$\boldsymbol{\alpha}^* = \arg \max_{\boldsymbol{\alpha}} \max_{\mathbf{w}} L(\mathbf{w}, \boldsymbol{\alpha}).$$

Then we can say the following about the Lagrangian:

$$\max_{\boldsymbol{\alpha}} \min_{\mathbf{w}} L(\mathbf{w}, \boldsymbol{\alpha}) = \min_{\mathbf{w}} L(\mathbf{w}, \boldsymbol{\alpha}^*) \leq L(\mathbf{w}^*, \boldsymbol{\alpha}^*) \leq \max_{\boldsymbol{\alpha}} L(\mathbf{w}^*, \boldsymbol{\alpha}) = \min_{\mathbf{w}} \max_{\boldsymbol{\alpha}} L(\mathbf{w}, \boldsymbol{\alpha})$$

The first inequality holds because $\boldsymbol{\alpha^*}$ is the maximizer. The second holds because $\mathbf{w}^*$ is no less than the minimum value over all $\mathbf{w}$. The third inequality holds because $L$ evaluated at $\boldsymbol{\alpha^*}$ is at most the maximum value of $L$ over all possible $\boldsymbol{\alpha}$. The last follows because $\mathbf{w}^*$ is the minimizer. Since the leftmost and rightmost expressions are equal, this shows that all of the expressions are equal, which implies that $\boldsymbol{\alpha^*}$ is the maximizer of $L(\mathbf{w}^*, \boldsymbol{\alpha})$ and $\mathbf{w}^*$ is the minimizer of $L(\mathbf{w}, \boldsymbol{\alpha^*})$. This means $(\mathbf{w}^*, \boldsymbol{\alpha^*})$ is a saddle point.

## 3.5 KKT

We saw that the following conditions must hold when Max and Mindy are playing optimally:
$b_i(\mathbf{w}) \geq 0$
$\alpha_i b_i(\mathbf{w}) = 0$
$\alpha_{\mathbf{i}} \geq 0$
This must hold $\forall i$. Since $\mathbf{w}^*$ maximizes $L(\mathbf{w}, \boldsymbol{\alpha}^*)$, and since $\mathbf{w}^*$ is unconstrained, this also means that:

$$\frac{\partial L(\mathbf{w}^*, \boldsymbol{\alpha^*})}{\partial w_j} = 0$$

These four conditions are called the Karush-Kuhn-Tucker (KKT) conditions and the first three are called complementary slackness.

Computing the partial derivative we get that

$$\frac{\partial L(\boldsymbol{w^*}, \boldsymbol{\alpha^*})}{\partial w_j} = w_j - \sum_{i=1}^{m} \alpha_i y_i x_{ij}$$

This implies that $\mathbf{w}^* = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i$. So our solution is a linear combination of the inputs! We can solve for $\boldsymbol{\alpha^*}$ by plugging back into the original Lagrangian. We then get an optimization problem where we want to find $\boldsymbol{\alpha}$ that maximizes:

$$\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x_i} \cdot \boldsymbol{x_j}$$

$$\alpha_i \geq 0 \, \forall i$$

This can be solved using techniques like gradient descent. By our complementary slackness conditions, we know that $\alpha_i b_i(\mathbf{w}) = 0$. If $\alpha_i \neq 0$ then this means $b_i(\mathbf{w}) = y_i(w \cdot x_i) - 1 = 0$. So $y_i(\mathbf{w} \cdot \mathbf{x}_i) = 1$. If we substitute $\mathbf{v}$ for $\mathbf{w}$, we can see that $\mathbf{x}_i$ is a support vector since it lies exactly distance $\delta$ away from the hyperplane. So the only non-zero coefficients on any training examples are the coefficients on the support vectors! So our hypothesis depends only a small subset of the training examples! In a previous homework, we showed that when we have these properties we have essentially compressed our training set. If there are $k$ support vectors, then (from Homework 2) we have the following bound on the generalization error $err(h) \leq \tilde{O}(\frac{k+\ln(\frac{1}{\delta})}{m})$. Crucially, this bound does not depend on the dimensionality of the problem.