COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Matt Myers

Lecture # 13
March 26, 2018

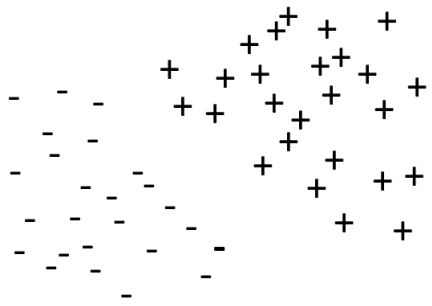
1 Motivation

1.1 Introduction

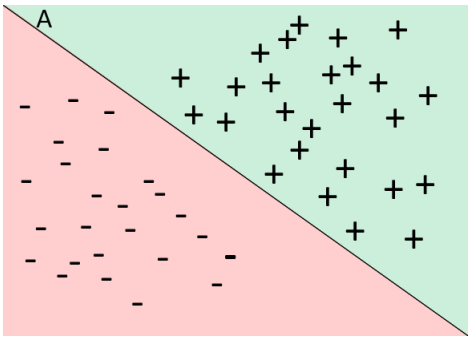
Last week, we talked about the theoretical question behind boosting (increasing the effectiveness of a weak learning algorithm), and analyzed an algorithm for boosting in the form of AdaBoost. We found that it ultimately maximized the *margins* of the training examples — intuitively, the classifier's "confidence" in its predicted label — which translated to better generalization error. Today, we will discuss a classification algorithm that seeks to directly maximize these margins, in the form of support vector machines (SVMs).

1.2 Classifier margins

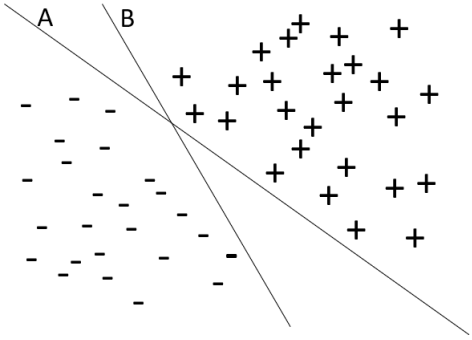
The algorithm that we are about to study assumes that examples are points in Euclidean space. This is a reasonable assumption, since generally most of the data that we are interested in has features that can be encoded as real-valued dimensions in Euclidean space (note that we are still in the context of classification, where each example is associated with a binary label).



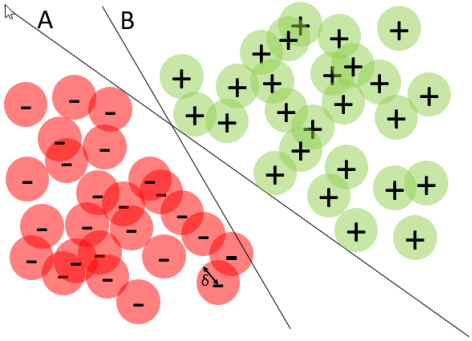
Consider this sample of positive and negative points. One intuitive method for classifying these points is to divide them using a line, where all of the points above the line are predicted to be positive examples (green), and all the points below the line are predicted negative (red). This type of classifier is known as a linear threshold function (LTF).



Previously in this class, we have proven some results that hold for *any* hypothesis that is consistent with the training examples. However, some lines seem intuitively better fitting to the data than others. Consider lines A and B below. Both of them are consistent with the data, thus equally"good" in terms of the previous analysis. However, while line A appears to give each example a healthy margin, line B is very close to calling the negative point towards the bottom left positive.

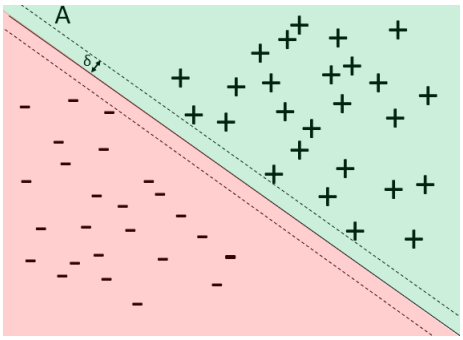


Intuitively, it seems small movements of an example shouldn't change how the example is classified. More formally, any point within distance δ from an example x_i should have the same label y_i, as shown below.



The problem of finding a linear separator (a hyperplane in higher dimensions) consistent with the sample such that this δ is maximized is equivalent to finding a consistent linear separator with a distance of at least δ from any example in the training set.

Intuitively, this separating line (generally a hyperplane in higher dimensions) is held in place by the examples that are exactly at distance δ away. This distance δ from the



hyperplane to the closest point(s) is referred to as the *margin*, and the set of points at a distance of exactly δ are the *support vectors* for this linear separator. We will see that this set of points is sufficient to define the linear separator.

1.3 Justifying maximization of margins using VC-dimension

In order to formally justify selecting the separating hyperplane with the largest margin, we turn to the three ingredients for effective learning that we have explored so far in this class: having enough data to train on, finding a hypothesis that fits the sample well, and the complexity of the hypothesis space. Because we are currently comparing between classifiers that are a) trained on the same sample and b) are consistent with this sample, the remaining tool we have is analyzing the complexity of the hypothesis space. Using VC-dimension (only considering planes passing through the origin for simplicity):

- The VC-dimension of linear threshold functions in n dimensions is n. While this may seem manageable, in practice n can be very large in applications like machine learning.
- If all of our points fit inside a ball of radius R (which is true for some R for any finite set of points), then the VC-dimension of linear threshold functions with margin δ is $\leq (R/\delta)^2$. This expression doesn't (directly) depend on the dimensionality of the data, and we also have that the VC-dimension shrinks as δ grows, motivating the idea of choosing a linear threshold function to make δ as large as possible.

2 Finding the hyperplane with the largest margin

2.1 Required linear algebra

For this analysis, we will use the following facts from linear algebra:

- $||\mathbf{v}||_2$ represents the ℓ_2 or Euclidean norm of the vector $\mathbf{v}$ (abbreviated as $||\mathbf{v}||$ for the remainder of the notes).
- A hyperplane passing through the origin is defined by a vector $\mathbf{v}$ normal to it. The distance from a point $\mathbf{x}$ to this hyperplane is the absolute value of the dot product, or inner product, $|\mathbf{v} \cdot \mathbf{x}|$. Importantly, this distance $\mathbf{v} \cdot \mathbf{x}$ is signed: the direction of $\mathbf{v}$ indicates the "above" direction of the hyperplane, and $\mathbf{v} \cdot \mathbf{x} > 0$ indicates that $\mathbf{x}$ is above the hyperplane, while $\mathbf{v} \cdot \mathbf{x} < 0$ indicates that $\mathbf{x}$ is below the hyperplane (naturally, $\mathbf{v} \cdot \mathbf{x} = 0$ implies that $\mathbf{x}$ is on the hyperplane).

2.2 Optimization problem

Given a sample of m examples $S = \langle(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\rangle$, $\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in \{+1, -1\}$:

Problem: Find $\mathbf{v}$ to maximize δ, subject to $||\mathbf{v}|| = 1$ and

$$\text{for all } i, \begin{cases} \mathbf{v} \cdot \mathbf{x}_i \geq \delta & \text{if } y_i = +1 \\ \mathbf{v} \cdot \mathbf{x}_i \leq -\delta & \text{if } y_i = -1 \end{cases}$$

Immediately, we can combine these into a single case using the definition of y_i, to obtain $y_i(\mathbf{v} \cdot \mathbf{x}_i) \geq \delta$ for all i. Then, we can divide both sides of this inequality by δ to obtain $y_i(\mathbf{w} \cdot \mathbf{x}_i) \geq 1$ where $\mathbf{w} = \mathbf{v}/\delta$, thus getting rid of the constraint on $\mathbf{v}$. Because $||\mathbf{w}|| = 1/\delta$, we now have a new form for the optimization problem:

Problem: Find $\mathbf{w}$ to minimize $\frac{1}{2}||\mathbf{w}||^2$ subject to $y_i(\mathbf{w} \cdot \mathbf{x}_i) \geq 1$ for all i.

(The 1/2 and square make it easier to take the derivative in subsequent steps). We now have a convex program (a convex program with linear constraints), one of the first applications of convex programming in machine learning. Note that in order to classify a test example $\mathbf{x}$ using the output of these programs, simply predict $\hat{y} = \text{sign}(\mathbf{v} \cdot \mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x})$.

2.3 Forming the Lagrangian

To form the Lagrangian function, we first solve for 0 on the right-hand side of the constraints and introduce a function $b_i(\mathbf{w}) = y_i(\mathbf{w} \cdot \mathbf{x}_i) - 1$ to consolidate the left-hand side. The Lagrangian is then the difference between the objective and the constraints:

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

where α_i are the Lagrange multipliers. The claim is that the previous optimization problem can be written as the following min-max of the Lagrangian:

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

where the minimum is over all $\mathbf{w} \in \mathbb{R}^n$ and the maximum is over all $\boldsymbol{\alpha} \in \mathbb{R}^m$ with $\alpha_i \geq 0$ for all i.

2.4 Manipulating min-max

Intuitively, min-max is like a game played by 2 players: Mindy, who has control over the $\mathbf{w}$ variable and wants to minimize the function, and Max, who has control over the $\boldsymbol{\alpha}$ variable and wants to maximize the function. Mindy first chooses $\mathbf{w}$ to minimize the function, then Max chooses $\boldsymbol{\alpha}$ to maximize the function knowing $\mathbf{w}$. If they were to actually play, Mindy might choose $\mathbf{w}$ such that $b_i(\mathbf{w}) < 0$ for some i, but then Max could set the corresponding α_i to an arbitrarily large value (or infinity) so that L will also be arbitrarily large or infinite. Therefore, Mindy will not choose $\mathbf{w}$ to make any $b_i(\mathbf{w}) < 0$. If $b_i(\mathbf{w}) = 0$ then the quantity $\alpha_i b_i(\mathbf{w}_i) = 0$ and α_i is irrelevant, and if $b_i(\mathbf{w}) > 0$ then Max will maximize the function by setting $\alpha_i = 0$. Since $\alpha_i b_i(\mathbf{w}) = 0$ for all i, the Lagrangian will simply be equal to the

original objective $\frac{1}{2}||\mathbf{w}||^2$. So in summary, Mindy will choose $\mathbf{w}$ so that $b_i(\mathbf{w}) \geq 0$ for all i, and among all such $\mathbf{w}$, she will select the one for which the original objective is minimized. Thus, the resulting min-max problem is equivalent to the original optimization problem.

One might wonder why this formulation is any more useful than the original optimization problem. To this end, suppose that Mindy and Max play the game in the opposite order. Intuitively,

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

because playing second is at least as good as playing first. And for some special functions L (roughly those convex in $\mathbf{w}$ and concave in $\boldsymbol{\alpha}$),

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

Fortunately, it turns out that this is one of those functions. So solving min-max also solves max-min, and vice versa. But what does this tell us about solutions?

Let $\mathbf{w}^* = \arg\min_{\mathbf{w}} \max_{\boldsymbol{\alpha}} L(\mathbf{w}, \boldsymbol{\alpha})$, and let $\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}} \min_{\mathbf{w}} L(\mathbf{w}, \boldsymbol{\alpha})$. Then

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

The first relationship holds because $L(\mathbf{w}^*, \cdot)$ evaluated at $\boldsymbol{\alpha}^*$ is at most the maximum value of the function over all possible $\boldsymbol{\alpha}$; the second by the definition of $\mathbf{w}^*$; the third by the equality we just presented; the fourth by the definition of $\boldsymbol{\alpha}^*$; and the last because $L(\cdot, \boldsymbol{\alpha}^*)$ evaluated at $\mathbf{w}^*$ is at least the minimum value of the function over all possible $\mathbf{w}$. Because the beginning and end of this chain are equal, all of the inner inequalities must be equalities. We now know that $\boldsymbol{\alpha}^*$ maximizes $L(\mathbf{w}^*, \cdot)$ and that $\mathbf{w}^*$ minimizes $L(\cdot, \boldsymbol{\alpha}^*)$. As a result, $L(\mathbf{w}^*, \boldsymbol{\alpha}^*)$ must be a saddle point of L.

2.5 Characterizing solutions to the Lagrangian optimization problem

Knowing that the solution to the Lagrangian optimization problem must be a saddle point, the following conditions must hold for all $\mathbf{w}^*$ and $\boldsymbol{\alpha}^*$:

$$\forall i, j: \ \frac{\partial L(\mathbf{w}^*, \boldsymbol{\alpha}^*)}{\partial(w_j)} = 0, \ b_i(\mathbf{w}^*) \geq 0, \ \alpha_i^* \geq 0 \ , \ \alpha_i b_i(\mathbf{w}^*) = 0$$

These conditions are the Karush-Kuhn-Tucker (KKT) conditions, and the last three conditions are referred to as complementary slackness. For our Lagrangian function,

$$\frac{\partial L(\mathbf{w}^*, \boldsymbol{\alpha}^*)}{\partial(w_j)} = w_j - \sum_{i=1}^{m} \alpha_i y_i x_{ij} = 0$$

We can then write the solution (the variable that we care about) as a linear combination of the examples: $\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$.

How can we find the α values? Simply plug this $\mathbf{w}$ back into the original Lagrangian to obtain the dual to the original optimization problem, which is an easily-optimized paraboloid:

$$\max_{\boldsymbol{\alpha}} \sum_i \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j), \text{ s.t. } \forall i, \alpha_i \geq 0$$

Additionally, complementary slackness tells us that $\alpha_i b_i(\mathbf{w}) = \alpha_i (y_i(\mathbf{w} \cdot \mathbf{x}_i) - 1) = 0$. This implies that if $\alpha_i \neq 0$, then $y_i(\mathbf{w} \cdot \mathbf{x}_i) - 1 = 0$ so $y_i(\mathbf{w} \cdot \mathbf{x}_i) = 1$, i.e. $\mathbf{x}_i$ has a margin of exactly 1 and is a support vector. This means that $\mathbf{w}$ can be expressed as a linear combination of only those examples that are support vectors. This is similar to problem 4 on homework 2, which explored the representation of hypotheses using subsets of the sample. Using this information, and the bound proved on that homework problem, we can obtain a bound on the generalization error of

$$\tilde{O}\left(\frac{k + \ln(1/\delta)}{m}\right)$$

with probability at least $1 - \delta$, where k is the number of support vectors. This alternative method of analyzing support vector machines, in terms of the number of support vectors rather than the margin, again yields a bound on the generalization error that is independent of the number of dimensions n.