COS 511: Theoretical Machine Learning

Lecturer: Rob Schapire
Scribe: Yingfei Wang

Lecture #14
March 28, 2013

1 More on SVM

1.1 A quick review

Given m labeled examples $(\vec{x}_1, y_1), ..., (\vec{x}_m, y_m)$, where $\vec{x}_i \in \mathbb{R}^n$ and $y_i \in \{-1, +1\}$. We are first assuming that the data is linearly separable. Our goal is to find the hyperplane passing through the origin which maximizes the margin. We end up with the following optimization problem:

$$\begin{aligned} &\text{find } \vec{v}, \delta \\ &\max \delta \\ &\text{s.t. } \|\vec{v}\|_2 = 1 \\ &\forall i, y_i(\vec{v} \cdot \vec{x}_i) \geq \delta. \end{aligned} \tag{1}$$

If we let $\vec{w} = \frac{\vec{v}}{\delta}$, we have $\|\vec{w}\|_2 = \frac{1}{\delta}$ and we can rewrite the above optimization problem as:

$$\begin{aligned} &\min \frac{1}{2}\|\vec{w}\|^2 \\ &\text{s.t. } \forall i, y_i(\vec{w} \cdot \vec{x_i}) \geq 1. \end{aligned} \tag{2}$$

In the previous lecture, we claim that it can be converted to its dual problem:

$$\begin{aligned} &\max \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j \\ &\text{s.t. } \forall i, \alpha_i \geq 0. \end{aligned} \tag{3}$$

After we find out optimal α_i's, we have $\vec{w} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i$. And $\alpha_i \neq 0 \Rightarrow (\vec{x}_i, y_i)$ is a support vector, meaning: $y_i(\vec{w} \cdot \vec{x}_i) = 1$. In the next section we'll show how to convert the above optimization problem to its dual problem.

1.2 Convert to dual problem

Basically, we use Lagrangian method. We first notice that if we don't have constraints in eq.(2), we could find the minimum value by simply taking first derivative over $\vec{w}$. But here, we do have constraints, so we first convert each constraint into the standard form $y_i(\vec{w} \cdot \vec{x}_i) - 1 \geq 0$. We denote $y_i(\vec{w} \cdot \vec{x}_i) - 1$ as $b_i(\vec{w})$. Then we form Lagrangian by subtracting each constraint :

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

where α_i's are called Lagrange multipliers. This optimization problem can be written in this form:

$$\min_{\vec{w}} \max_{\vec{\alpha}:\forall i,\alpha_i \geq 0} L(\vec{w}, \vec{\alpha}). \tag{4}$$

This can be treated as a two-player game, where Min chooses $\vec{w}$ first and then Max chooses $\vec{\alpha}$. The goal of Min is to minimize $L(\vec{w}, \vec{\alpha})$ and the goal of Max is to maximize it. If Min chooses $\vec{w}$ s.t. $b_i(\vec{w}) < 0$, then Max chooses $\alpha_i = +\infty$, and the value of L will be infinity. But since Min's goal is to minimize L, Min will never choose $\vec{w}$ s.t. $b_i(\vec{w}) < 0$.

$$b_i(\vec{w}) = 0 \Rightarrow \alpha_i \text{ is irrelevant},$$
$$b_i(\vec{w}) > 0 \Rightarrow \alpha_i = 0.$$

In the above two cases, we have $\alpha_i b_i(\vec{w}) = 0$ for all i. So Min will choose $\vec{w}$ such that $b_i(\vec{w}) \geq 0$ for all i and Max will choose α_i such that $\sum_{i=1}^m \alpha_i b_i(\vec{w}) = 0$ for all i. Recall that Min's goal is to minimize $L(\vec{w}, \vec{\alpha}) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^m \alpha_i b_i(\vec{w})$, so $\vec{w}$ chosen by Min is the one that minimizes $\frac{1}{2}\|\vec{w}\|^2$. So the solution here is the same as the original optimization problem.

Now consider another two-player game:

$$\max_{\vec{\alpha}:\forall i,\alpha_i \geq 0} \min_{\vec{w}} L(\vec{w}, \vec{\alpha}), \tag{5}$$

where Max chooses $\vec{\alpha}$ first and then Min chooses $\vec{w}$. This is called the dual.

In the two-player game $\min_{\vec{w}} \max_{\vec{\alpha}:\forall i,\alpha_i \geq 0} L(\vec{w}, \vec{\alpha})$, Min goes first and then Max. While in the other game, Max goes first. We claim that the condition for Min in the second game will not be worse than that in the first game, since if Min goes second, he can choose $\vec{w}$ based on the value of $\vec{\alpha}$ that is chosen by Max. More knowledge is always better! The fact that the second game is better (or not worse) than the first game for Min means the value of L in the second game is less than or equal to that in the first game. And this argument holds for Max at the same time. So we have

$$\max_{\vec{\alpha}:\forall i,\alpha_i \geq 0} \min_{\vec{w}} L(\vec{w}, \vec{\alpha}) \leq \min_{\vec{w}} \max_{\vec{\alpha}:\forall i,\alpha_i \geq 0} L(\vec{w}, \vec{\alpha}),$$

where the equality holds if L is convex in $\vec{w}$ and concave in $\vec{\alpha}$ in addition to some other conditions that usually hold.

Let

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

We have

$$\begin{aligned}
L(\vec{w}^*, \vec{\alpha}^*) &\leq \max_{\vec{\alpha}} L(\vec{w}^*, \vec{\alpha}) && (6)\\
&= \min_{\vec{w}} \max_{\vec{\alpha}} L(\vec{w}, \vec{\alpha}) && (7)\\
&= \max_{\vec{\alpha}} \min_{\vec{w}} L(\vec{w}, \vec{\alpha}) && (8)\\
&= \min_{\vec{w}} L(\vec{w}, \vec{\alpha}^*) && (9)\\
&\leq L(\vec{w}^*, \vec{\alpha}^*) && (10)
\end{aligned}$$

Inequality (6) holds because $L(\vec{w}^*, \vec{\alpha}^*)$ is the value of $L(\vec{w}^*, \vec{\alpha})$ at a particular $\vec{\alpha}^*$, while $\max_{\vec{\alpha}} L(\vec{w}^*, \vec{\alpha})$ is taking the maximum over all choices of $\vec{\alpha}$. So $L(\vec{w}^*, \vec{\alpha}^*) \leq \max_{\vec{\alpha}} L(\vec{w}^*, \vec{\alpha})$. The equality (7) holds since $\vec{w}^*$ minimizes the expression $\max_{\vec{\alpha}} L(\vec{w}, \vec{\alpha})$ by definition. So $\max_{\vec{\alpha}} L(\vec{w}^*, \vec{\alpha}) = \min_{\vec{w}} \max_{\vec{\alpha}} L(\vec{w}, \vec{\alpha})$. Reversing these arguments proves Eq.(9) and Eq.(10). So we end up with the above chain of equalities and inequalities. Since the first and the last term are the same, everything between these two all have to be equal. In particular, we have $L(\vec{w}^*, \vec{\alpha}^*) = \max_{\vec{\alpha}} L(\vec{w}^*, \vec{\alpha})$. In other words, $\vec{\alpha}^*$ maximizes $L(\vec{w}^*, \cdot)$. We can also show that $\vec{w}^*$ minimizes $L(\cdot, \vec{\alpha}^*)$ in a similar way. So $(\vec{w}^*, \vec{\alpha}^*)$ is the saddle point of $L(\vec{w}, \vec{\alpha})$. What does this means? Imagine you are at some point $(\vec{w}^*, \vec{\alpha}^*)$ in a high dimensional space. Now start to moving along the function L in the $\vec{\alpha}$ direction. It says that the value at this point is the maximum of L in this direction. On the other hand, if you move in $\vec{w}$ direction which happens to be perpendicular to $\vec{\alpha}$ direction, the value at this point is the minimum of L in this direction. $(\vec{w}^*, \vec{\alpha}^*)$ is said to be the saddle point of L.

Up to now, we have all the conditions we need to find the optimal $\vec{w}$:

$$\frac{\partial L}{\partial w_j}\Big|_{\vec{w}^*, \vec{\alpha}^*} = 0, \tag{11}$$

$$\forall i : b_i(\vec{w}^*) \geq 0, \tag{12}$$

$$\forall i : \alpha_i^* \geq 0, \tag{13}$$

$$\forall i : \alpha_i^* b_i(\vec{w}^*) = 0. \tag{14}$$

These four together are called Karush-Kuhn-Tucker (KKT) conditions, and the last three together are called complementary slackness condition.

By equation (11), we have $0 = \frac{\partial L}{\partial w_j} = w_j - \sum_i \alpha_i y_i x_{ij}$, or $\vec{w} = \sum_i \alpha_i y_i \vec{x_i}$. Plugging into L, we get

$$\max \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j$$
$$\text{s.t. } \forall i, \alpha_i \geq 0. \tag{15}$$

Also, by condition(14): $\forall i, \alpha_i^* b_i(\vec{w}^*) = 0$, we get $\alpha_i[y_i(\vec{w} \cdot \vec{x}_i) - 1] = 0$. Therefore if α_i is not equal to 0, then $y_i(\vec{w} \cdot \vec{x}_i) - 1$ has to be 0 and hence $(\vec{x}_i, y_i)$ is a support vector.

1.3 Compare SVM with Boosting

We now compare SVM with Boosting algorithm. In SVM, we get examples $\vec{x}$ which are points in $\mathbb{R}^N$. It is natural to assume that $\|\vec{x}\|_2 \leq 1$. In boosting we get examples x from some space. Boosting algorithm never directly touches training examples. Instead, it gets weak hypotheses and gives a prediction. Suppose weak hypothesis space $\mathcal{H}$ is finite, $\mathcal{H} = \{g_1, ..., g_N\}$ and $\vec{h}(x) = \langle g_1(x), ..., g_N(x) \rangle$. In a sense, the input of boosting algorithm is this huge vector $\vec{h}(x)$. Since $g_i(x) \in \{-1, +1\}$, $\|\vec{h}(x)\|_\infty = \max_j |g_j(x)| = 1$.

The next thing is that SVM finds a vector $\vec{v}$ with $\|v\|_2 = 1$, and given an example $\vec{x}$, it predicts $\text{sign}(\vec{v} \cdot \vec{x})$. In boosting, instead, the algorithm finds the weight $\vec{\alpha_t}$ for each weak hypothesis h_t, and the output of the algorithm is $\text{sign}(\frac{\sum_t \alpha_t h_t(x)}{\sum_t \alpha_t})$. This expression is a convex combination of weak hypotheses, and each one of these comes from the weak hypothesis space. That's to say $h_t \in \{g_1, ...g_N\}$. So we can rewrite this as

the convex combination of g_j's rather than h_t's. So we can choose weights a_j's such that $\text{sign}(\frac{\sum_t \alpha_t h_t(x)}{\sum_t \alpha_t}) = \text{sign}(\sum_{j=1}^N a_j g_j(x))$. And $\sum_{j=1}^N a_j g_j(x) = \vec{a} \cdot \vec{h}(x)$, where $\vec{a} = \langle a_1, ..., a_N \rangle$. What boosting is doing is finding those weights $\vec{a}$. The sum of these weights is 1 and each of them is non-negative. So $\vec{a}$: $\|\vec{a}\|_1 = \sum_j |a_j| = 1$. And the prediction of boosting is $\text{sign}(\vec{a} \cdot \vec{h}(x))$.

Finally we'll talk about the margin. The margin of SVM is $y(\vec{v} \cdot \vec{x})$. In boosting, the margin of (x, y) is $yf(x) = y\frac{\sum \alpha_t h_t(x)}{\sum \alpha_t} = y(\vec{a} \cdot \vec{h}(x))$. We point out here that SVM and Boosting both maximize margin, but the definitions of margin are different, the norms used here are different.

They both can work over high dimensions, but SVM uses kernel trick to deal with high dimensional data. And in boosting, if we view $\vec{h}(x)$ as input, the dimension is the number of weak hypotheses which is typically gigantic and we use a weak learning algorithm as a oracle to give us those weak hypotheses to work over high dimensional space.

We sum up the comparison in the following table:

SVM	Boosting
input $\vec{x}$: $\|\vec{x}\|_2 \leq 1$	input $\vec{h}(x)$: $\|\vec{h}(x)\|_\infty = 1$
$\vec{v}$: $\|\vec{v}\|_2 = 1$	$\vec{a}$: $\|\vec{a}\|_1 = \sum_j \|a_j\| = 1$
predict $\text{sign}(\vec{v} \cdot \vec{x})$	predict $\text{sign}(\vec{a} \cdot \vec{h}(x))$
margin $y(\vec{v} \cdot \vec{x})$	margin $y(\vec{a} \cdot \vec{h}(x))$

2 Online learning

2.1 Introduction

The learning framework for on-line learning is quite different from the PAC learning model we have seen so for. First, instead of learning from a training set and then testing on a test set, training and testing happen all at the same time in the online learning scenario. It gets one example at a time, predicts the label and then gets the truth. Second, the PAC learning model follows the key assumption that the training data and test data are i.i.d. from some fixed distribution, while in online learning, no statistical assumptions are made. We no longer assume that examples are random, and in fact examples can be generated by an adversary in online learning.

One example of the online learning problem is to predict the stock market. In the morning, the learner predicts whether the price will go up or down, and then in the late afternoon it can find out whether it actually went up or down.

2.2 Learning with expert advice

We still use the stock market example. Suppose there are four experts who will give you their predictions of the price every morning, and based on their predictions, you make your own prediction. A specific example might look like this:

	Experts					
	1	2	3	4	Learner(Master)	Outcome
Day 1	↑	↓	↑	↑	↑	↑
Day 2	↓	↑	↑	↓	↓	↑
⋮						
# mistakes	37	12	63	88	18	

The goal of the learner is to be not much worse than the best expert in terms of the number of mistakes that it makes..

General form of learning with expert advice is:

N=# experts
for $t = 1, 2, ...T$,
    each expert i predicts $\xi_i \in \{0, 1\}$,
    learner predicts $\hat{y} \in \{0, 1\}$,
    observe actual outcome $y \in \{0, 1\}$.
        (mistake if $\hat{y} \neq y$)

Suppose some expert is perfect, but the learner doesn't know which one is perfect. We have the following "Halving Algorithm", in which on each round we kill an expert immediately after he makes a mistake and take the majority vote of surviving experts as the prediction:

on each round:
    $\hat{y}$ = majority vote of experts with no mistakes so far

We now prove that the halving algorithm makes at most $\lg N$ mistakes, where N is the number of experts. Let W be the number of surviving experts. Initially $W = N$. If the learner made a mistake on some round, then at least $\frac{1}{2}$ survivors made a mistake on that round. So at least $\frac{1}{2}$ experts were eliminated on that round. In other words, if the learner made 1 mistake on some round, the number of survivors is at most $\frac{1}{2}N$. If the learner made another mistake, the number of survivors after the second mistake is at most $(\frac{1}{2})^2 N$. So after the learner made the mth mistake, the number of survivors is at most $(\frac{1}{2})^m N$. Since we know that at least one expert is perfect, we have $W \geq 1$. By $1 \leq W \leq 2^{-m}N$ we have $m \leq \lg N$.

At last, we want to point out that experts really can be anything, for example, fixed functions, other learning algorithms and actual human experts.