CS 475 Machine Learning: Lecture 10

Notes for SVMs[*]

Prof. Mark Dredze

1 SVM Derivation

Let's start by writing the quadratic program

$$\begin{aligned} \arg\min_{\mathbf{w}} \quad & \tfrac{1}{2}||\mathbf{w}||^2 \\ s.t. \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1 = 0, \forall i \end{aligned} \tag{1}$$

The purpose of the $\frac{1}{2}$ will be apparent momentarily. We will ignore the bias term b for some of the analysis since it isn't necessary for the derivation.

This form of the objective can be solved directly using quadratic programming, which minimizes (or maximizes) a quadratic function with linear constraints. This is a convex function so we will find a single global optimum. There is a single hyperplane that satisfies all of the constraints. The single result is from the normalization of $\mathbf{w}$.

1.1 Dual

In these types of optimization problems, we call this the objective the primary problem. However, we can also define a dual problem. The dual problem and the primary problem have the same solution. If the primary problem minimizes an objective, then we can rewrite it as a problem that maximizes the objective.

Let's rewrite this objective using Lagrange multipliers. We introduce a multiplier for each constraint: α_i such that $\alpha_i \geq 0$. We have n such variables, so you can guess these will become the dual variables.

Using these multipliers we write the Lagrange function:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{N} \alpha_i \{y_i(\mathbf{w}\cdot\mathbf{x}_i + b) - 1\} \tag{2}$$

Notice the $-$ in front of the multipliers since this is minimization.
Sanity check: This equation is minimized when the norm of $\mathbf{w}$ is 0 and all margins are 1.

Let's start by taking the derivative of L with respect to b since that is easy.

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i \tag{3}$$

Setting that equal to 0 gives the optimal solution for b, which is just a constraint on the values of α.

$$0 = \sum_{i=1}^{N} \alpha_i y_i \tag{4}$$

[*]You will notice that the notation I use differs from the Bishop book. I have chosen this alternate notation to remain consistent with the majority of the literature on machine learning. This means that when you read papers in the literature you will be familiar with their notation.

Good-bye b ☺

Let's take the derivative of L with respect to $\mathbf{w}$ for one of the positions in $\mathbf{w}$.

$$\frac{\partial L}{\partial w_j} = w_j - \sum_{i=1}^{N} \alpha_i y_i x_{i,j} \tag{5}$$

We see that the derivative for a value in $\mathbf{w}$ depends on a combination of values from $\mathbf{x}$ weighted by α. The derivative for the full vector $\mathbf{w}$ is then just:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \tag{6}$$

Setting the derivative equal to 0 and solving for $\mathbf{w}$:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \tag{7}$$

Here comes the magic.

We now can use these as constraints and obtain a new objective. This takes some slick algebra, which we'll skip.

$$\tilde{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j \tag{8}$$

with constraints

$$\alpha_i \geq 0 \quad \forall i$$
$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

We can then maximize this dual objective, which minimizes our primal objective by finding the values for α that satisfy the constraints.
Sanity check: when is this maximized? Let's go back to the slides and look at how predictions are made.

2 Dual Perceptron

Let's look at $\mathbf{w}$ in the perceptron. If we've seen lots of examples of a perceptron then many of them have contributed to $\mathbf{w}$.

What is $\mathbf{w}$ after no examples? $\mathbf{w} = 0$

What is $\mathbf{w}$ after the ith examples?

$$\mathbf{w}^{i+1} = \mathbf{w}^{i} + y_i \mathbf{x}_i \tag{9}$$

What about $\mathbf{w}$ after $i-1$ examples?

$$\mathbf{w}^{i} = \mathbf{w}^{i-1} + y_{i-1} \mathbf{x}_{i-1} \tag{10}$$

So we can write $\mathbf{w}^{i+1}$ as:

$$\mathbf{w}^{i+1} = \mathbf{w}^{i-1} + y_{i-1} \mathbf{x}_{i-1} + y_i \mathbf{x}_i \tag{11}$$

If we continue until $i = 0$ we get

$$\mathbf{w}^{i+1} = \mathbf{0} + \sum_{i=1}^{n} y_i \mathbf{x}_i \tag{12}$$

Of course, we only make an update if we make a mistake. Let's write $\mathcal{M}$ as the set of examples for which we made a mistake.

$$\mathbf{w}^{i+1} = \mathbf{0} + \sum_{i \in \mathcal{M}} y_i \mathbf{x}_i \tag{13}$$

So we see that $\mathbf{w}$ is a linear combination of the inputs $\mathbf{x}_i$. In this case, α_i is just an indicator if we made a mistake $\alpha_i = 1$ or if we got the example correct $\alpha_i = 0$.

What is the prediction rule for the perceptron?

$$\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x}) \tag{14}$$

But now we can express $\mathbf{w}$ in terms of $\mathbf{x}$ and our indicator α.

$$\hat{y} = \text{sign}(\{\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i\} \cdot \mathbf{x}) \tag{15}$$

and moving $\mathbf{x}$ inside the sum we see.

$$\hat{y} = \text{sign}(\sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \mathbf{x}) \tag{16}$$

This looks very similar to an SVM. The difference is really in how we set α_i. For the perceptron, we set them at each point based a local decision. In SVM, we set them globally by looking at all of the points.

Notice that since we no longer have $\mathbf{w}$, this is a *dual perceptron.* We have n variables α to solve instead of m variables for w.