1 A Little Bit on Support Vector Machines (SVMs)

We saw that the Perceptron algorithm achieves a mistake bound, on non-separable data, that is a function of both the margin and hinge-loss of the best $\mathbf{w}^*$ (where "best" means the one that produces the best bound).

Support vector machines, given a sample S of datapoints perform convex optimization to essentially find that $\mathbf{w}^*$. For the discussion below, assume that the examples $\mathbf{x}_i$ have been scaled to all have length at most 1: this will make the interpretation of our equations cleaner. Also, we will use $\ell_i \in \{-1, 1\}$ to denote the label of datapoint $\mathbf{x}_i$.

Let's first consider the easier "realizable case" where data *is* linearly separable and we want to find the linear separator of largest margin. In this case, we can write our problem as:

$$\begin{aligned} &\text{minimize:} && ||\mathbf{w}||^2 \\ &\text{subject to:} && \ell_i(\mathbf{w} \cdot \mathbf{x}_i) \geq 1 \quad \text{for all } (\mathbf{x}_i, \ell_i) \in S. \end{aligned}$$

This is a convex optimization problem, so we can solve it efficiently. Equivalently we could write a constraint $||\mathbf{w}||^2 \leq 1$ and maximize the margin γ such that $\ell_i(\mathbf{w} \cdot \mathbf{x}_i) \geq \gamma$ for all i, but people prefer to write it the former way, for reasons that will make more sense in a minute. Note that if we write it the former way, then $1/||\mathbf{w}||$ is the margin and so $||\mathbf{w}||^2 = 1/\gamma^2$.

More generally, there might not be a perfect separator, or even if there is, we may want a tradeoff betwen margin and hinge-loss. So, what SVMs really do is (C is a given constant):

$$\begin{aligned} &\text{minimize:} && ||\mathbf{w}||^2 + C \sum_{\mathbf{x}_i \in S} \xi_i \\ &\text{subject to:} && \ell_i(\mathbf{w} \cdot \mathbf{x}_i) \geq 1 - \xi_i \quad \text{for all } (\mathbf{x}_i, \ell_i) \in S, \\ & && \xi_i \geq 0 \quad \text{for all } i. \end{aligned}$$

These ξ_i variables are called "slack variables" because they represent the extent to which we violate our original constraints, and their sum is the total hinge loss. Let's rewrite the same thing by dividing the objective function by $|S|$ (which doesn't change anything since $|S|$ is fixed but will help with our motivation):

$$\begin{aligned} &\text{minimize:} && ||\mathbf{w}||^2/|S| + C \sum_{\mathbf{x}_i \in S} \xi_i/|S| \\ &\text{subject to:} && \ell_i(\mathbf{w} \cdot \mathbf{x}_i) \geq 1 - \xi_i \quad \text{for all } (\mathbf{x}_i, \ell_i) \in S, \\ & && \xi_i \geq 0 \quad \text{for all } i. \end{aligned}$$

Here is the motivation. What we *really* want is to minimize the true error $err_D(h)$, but what we observe is our empirical error $err_S(h)$. So, let's split $err_D(h)$ into two parts: (1) $err_D(h) - err_S(h)$, which is the amount we're overfitting, and (2) $err_S(h)$. One bound on part (1) is approximately $(1/\gamma^2)/|S|$, if $err_S(h)$ is small (we will need more machinery to argue this, but if you think about the Perceptron algorithm, the "mistake-bound to PAC" conversion would give a sample complexity

bound of approximately $|S| = M/\epsilon$ for $M = 1/\gamma^2$, so ϵ is approximately $(1/\gamma^2)/|S|)$. So this is the first part of the objective function. The second part of the optimization (for C=1) is our average hinge-loss, which is an upper bound on $err_S(h)$. So, the second part of the objective function is an upper bound on (2). So, the two parts of the objective function are upper bounds on the two quantities we care about: overfitting and empirical error. And if you feel your upper bound on (1) is too loose, you might want to set C to be larger.

1.1 The Lagrangian Dual

Think of game between a corporation and the government. The corporation wants to make as much money as possible and the government doesn't want it to break any laws. Let's say the convex objective represents costs incurred by the corporation, so the lower the objective, the better for the company. The constraints correspond to laws: violating a constraint is like illegally "cutting corners". We can imagine what the government will do is charge a fine for breaking laws. Let's say the fine has to be linear in the amount by which the laws are broken. E.g., if there is a fine of \$100 on the constraint $\xi_1 \geq 0$ and $\xi_1 = -0.5$ then the corporation has to pay the government \$50.

At this point you might ask: why doesn't the government just set the fines to be infinitely high? So, there's one more catch: every fine has to be fully linear, which means that if there is a fine of \$100 on the constraint $\xi_1 \geq 0$ and $\xi_1 = +0.5$ then the government has to *pay* the corporation \$50.

Now we've defined the game, and can ask: what are the optimal strategies for the two players? Technically, we haven't said what the government is trying to optimize. We can either (a) say that the government would like the corporation not to break any laws and not to pay the company anything, or (b) have a zero-sum game where the government wants to maximize the total cost spent by the corporation. We will see that these are actually equivalent.

Let $\mathbf{w}$ denote the strategy for the corporation and let $\boldsymbol{\alpha}$ denote the strategy for the government. In particular, $\alpha_i \geq 0$ is the fine for violating the ith constraint, so if there are m constraints then $\boldsymbol{\alpha}$ is a vector with m non-negative components. Let $L(\mathbf{w}, \boldsymbol{\alpha})$ denote the total amount spent by the corporation. This is called the "Lagrangian". If we think of the company as playing first and then the government, then the company wants to solve:

$$\min_{\mathbf{w}} \max_{\boldsymbol{\alpha}} L(\mathbf{w}, \boldsymbol{\alpha}) \quad \text{subject to: } \boldsymbol{\alpha} \geq \mathbf{0}. \tag{1}$$

Note that this is the same as our original optimization problem since the company better not break any laws if it is playing first: the government could make the associated penalties arbitrarily high, and set penalties to 0 for the laws not violated. So, the optimal strategy for the company is to minimize its objective subject to not violating any constraints. But what if the government has to go first: defining the penalties $\boldsymbol{\alpha}$ before the company selects $\mathbf{w}$? Let's assume we are using the zero-sum definition of the game. For any given strategy of fines $\boldsymbol{\alpha}$, the company might be able to make more money than in (1), either by violating some laws or by making some constraints with positive fines not tight, so it can collect money from the government. In such a case, the government would want to modify its fines by increasing the fines on laws being violated and decreasing the fines on the non-tight constraints. It turns out the minimax theorem applies to this game, and so we can find the optimal also by flipping the order of "min" and "max" in (1), giving the dual problem.

In the end, the corresponding optimization problem for SVM is the following dual form:

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j \ell_i \ell_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ \text{subject to:} \quad & \textstyle\sum_i \alpha_i \ell_i = 0 \\ & C \geq \alpha_i \geq 0 \quad \text{for all } i. \end{aligned}$$

Note that this dual form allows the algorithm to be kernelized, just like Perceptron. In essence, here we are explicitly writing the weight vector $\mathbf{w}$ as a weighted sum of examples $\mathbf{w} = \sum_i \alpha_i \ell_i \mathbf{x}_i$. The examples $\mathbf{x}_i$ with non-zero α_i are called "support vectors".