Statistical Techniques in Robotics (16-831, F10) Lecture #14 (Thursday October 7th)

Support Vector Machines

Lecturer: Drew Bagnell *Scribe: Robert Fisher* [1]

1 Introduction

We begin by considering binary, linear classification. In this problem, we are trying to map elements from our feature space into the set $\{-1, 1\}$. When we have two features, $F_1, F_2 \in \mathbb{R}$, we can think of the problem graphically. In the following figure, we represent the points with a label of 1 as O's, we the points with a label of -1 as X's. We wish to find a linear decision boundary (a straight line) that separates the points from each class. The decision boundary is shown as the dotted line.

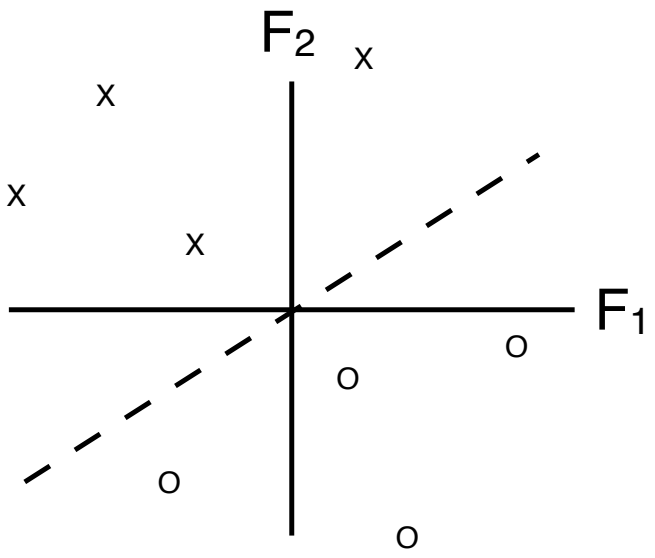

Figure 1: Linear binary classification

To represent this notion mathematically, we denote the i^{th} training point as f_i, and the class of this point as y_i. The linear classification problem requires us to find a set of weights w such that:

$$\forall i \; y_i w^T f_i \geq 0 \tag{1}$$

Note that the decision boundary need not pass through the origin. We often use a dummy variable $F_0 = c$ for constant $c \neq 0$, which allows us to use w_0 as an offset.

One issue with this formulation of the problem is that it has a trivial solution. Specifically, if we set all of our weights to 0, we will have $y_i w^T f_i = 0$ for all points. To address this issue, we modify

[1] Some content adapted from previous scribe: Alan Kraut

our constraints to be:

$$\forall i \; y_i w^T f_i \geq margin \tag{2}$$

This can have the added benefit of improving generalization. Once again, we can represent this graphically. In the following picture, the space between the dotted line and the decision boundary represents the margin.

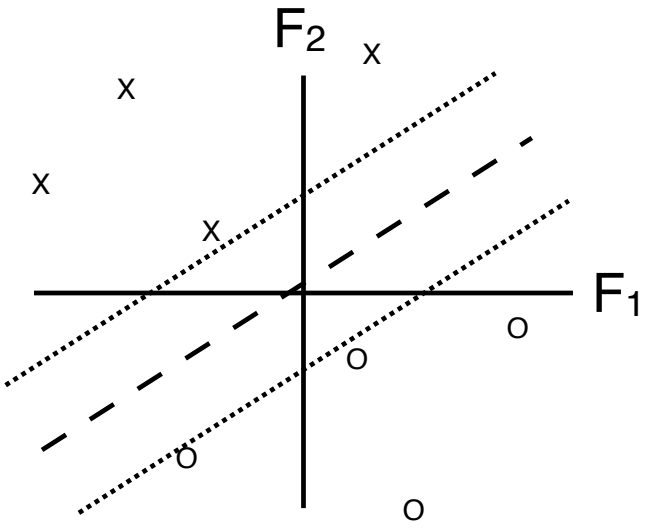

Figure 2: Linear binary classification with margins

We further see that the magnitude of our weight vector does not matter, only the orientation affects classification. Therefore we can restrict w to $||w||^2 = 1$. Taken together, this yields the following formulation:

$$\begin{array}{ll} \textbf{Maximize} & margin \\ \textbf{Such that} & \forall i \; y_i w^T f_i \geq margin \\ & ||w||^2 \leq 1 \end{array}$$

We use $||w||^2 \leq 1$ instead of $||w||^2 = 1$ to make sure that our problem remain convex.

This problem is generally solved in the dual, where we hold the margin fixed and minimize the magnitude of the weights. Therefore the final formulation of the hard margin SVM problem is:

$$\begin{array}{ll} \textbf{Minimize} & ||w||^2 \\ \textbf{Such that} & \forall i \; y_i w^T f_i \geq 1 \\ & margin > 0 \end{array}$$

This last formulation is an example of a quadratic problem, which we are able to solve efficiently. Unfortunately, the hard margin SVM requires that the data be linearly separable, which is almost

never the case. To address this issue, we introduce a slack variable, ξ. The problem now becomes:

$$\begin{aligned} \textbf{Minimize} \quad & \lambda||w||^2 + \sum_i \xi_i \\ \textbf{Such that} \quad & y_i w^T f_i \geq 1 - \xi_i \\ & \xi_i \geq 0 \\ & margin > 0 \end{aligned}$$

There will be one slack variable ξ_i to correspond to each of the T data-points that we are considering.

To make this online, we observe that $\xi = \max(0, 1 - y_i w^T f_i)$, because the slack variable is 0 if this point is labeled correctly. This allows us to generate the loss function

$$l_t = \lambda||w||^2 + \max(0, 1 - y_t w^T f_t) \tag{3}$$

Our update for w is now as follows.

$$w \leftarrow w - 2\alpha_t \lambda w \tag{4}$$

And if the output for this time step was incorrect,

$$w \leftarrow w + \alpha_t y_t f_t \tag{5}$$

We note that this loss function will **not** allow us to minimize the number of mistakes made by the algorithm. In fact, solving this problem with a 0-1 loss function (which would minimize the number of mistakes) is known to be NP-hard. We can visualize the difference between these loss functions with the following figure, in which $Z = w^T f_i$:

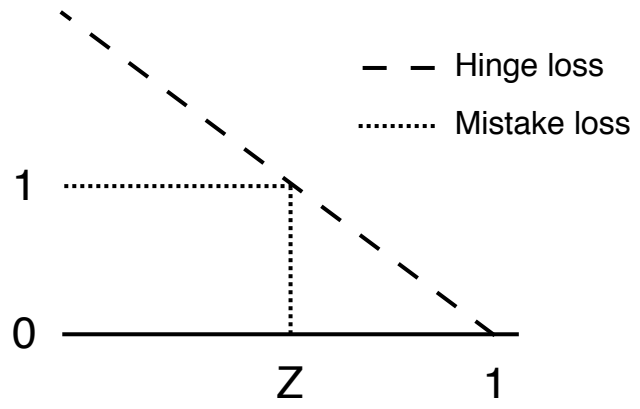

Figure 3: Loss functions

Our loss function, the hinge loss function, is convex, while the mistake loss (0-1 loss) function is not. However, we do see intuitively that the loss function we are using is the best "convexification" of the NP-hard problem.

2 Implementing Online SVMs

2.1 Selecting α_t

- Stock algorithm would be to set α_t proportional to $\frac{1}{\sqrt{t}}$.
- If we have T elements, each with a maximum value of F, the maximum gradient, G, is $\sqrt{TF}$. This results in a regret that is $R \leq \sqrt{FGT}$.
- This is not as good as we could do.
- Notice that l_t is an extremely good convex function. It is a quadratic plus a convex function. In the same way all convex functions lie above a line (a subgradient) from every point, l_t lies above a quadratic from every point.
- Specifically, if it is always the case that

$$f(y) \geq f(x) + \frac{H}{2}(y-x)^2 + \nabla f_x^T (y-x) \tag{6}$$

  then $f(x)$ is said to be H-strongly convex.
- In this case l_t is λ-strongly convex.
- If $\alpha_t = \frac{G}{Ht}$, then regret$\leq \frac{G^2}{H}(1+\log t)$. $\log t$ is *really* good, and this learning rate and algorithm is essentially the current best for this class of problem.

2.2 SVMs with Multiple Classes

We can represent problems with more than two classes by having a weight vector, w_i for each class.

- When we get a classification of a particular example (for example, example i is of class 1), we generate a set of constraints that can be expressed as either

$$\begin{array}{rcl} w_1^T f_i & \geq & w_2^T f_i + 1 \\ w_1^T f_i & \geq & w_3^T f_i + 1 \\ w_1^T f_i & \geq & w_4^T f_i + 1 \\ & \cdots & \end{array} \tag{7}$$

  or

$$w_1^T f_i \geq \max_{c \neq i}(w_c^T f_i + 1) \tag{8}$$

- By the same argument as before

$$\xi = \max(0, \max(w_c^T f + 1) - w_1^T f) \tag{9}$$

- We want to update each w by gradient descent on the partial of the cost with respect to that particular w.

- Remember the cost is $l_t = \lambda||w||^2 + \xi$. We want the update step to be

$$w_c \leftarrow w_c - \partial_{w_c} l_t \tag{10}$$

- In the case that the example was classified correctly, $\partial_{w_c} l_t = 0$. If it was misclassified, there are three cases with different partials: the correct class, the class we incorrectly decided this was an example of, and all others.

$$\begin{aligned} \partial_{w_c} &= -f_i, \quad y_i = c & (11)\\ \partial_{w_c} &= f_i, \quad c = \underset{c}{\text{argmax}}(w_c^T f_i + 1) & (12)\\ \partial_{w_c} &= 0, \quad \text{otherwise} & (13) \end{aligned}$$

- That update is for the max representation. If we use the multiple constraints representation it is similar, except we update both w_1 and w_c for all c which violate the constraint.