Statistical Techniques in Robotics (16-831, F12) Lecture#14 (Wednesday October 17)

Learning by constraints and SVMs (2)

Lecturer: Drew Bagnell *Scribe: Albert Wu* [1]

1 Support Vector Ranking Machine

Opening comment: ten years ago, SVM's were still a fairly new concept, and solving them as constrained quadratic programs was difficult and often required custom code. Now, they are essentially taken for granted and solving them is much simpler using gradient descent,

In the previous lecture, we discussed example ranking problems like selecting foot placement locations for walking in Little Dog or predicting winners between football teams. Each candidate location/team i is described with a feature vector f_i (for example, height difference or concavity of terrain, or average points per game or number of star players), and the desirability of the candidate is computed using the weight vector w as $w^T f_i$. The ranked training examples provide constraints as pairwise comparisons between candidates, and the ranking problem is solved by finding weight vector w such that the constraints

$$w^T f_1 \leq w^T f_2,$$
$$\vdots$$
$$w^T f_{100} \leq w^T f_{101}$$

are satisfied.

This formulation leaves three major gaps:

1. There may not exist a w that can satisfy all of the constraints; the training data may be inconsistent. For example, non-transitive rankings like $w^T f_1 \leq w^T f_2,\ w^T f_2 \leq w^T f_3,\ w^T f_3 \leq w^T f_1$.
2. There can be multiple solutions. At the very least, if w^T is a solution, any λw^T for $\lambda \geq 0$ is a solution.
3. There is always the trivial solution of $w^T = 0$. Note: if we were to constrain $||w|| \geq r$, this would make the problem non-convex.

1.1 Maximum Margin Approach

To address the latter two problems, we can use a maximum margin formulation. Instead of simply trying the meet the constraints, we use a margin describing by how much the inequalities are exceeded. The goal is to maximize this margin. In implementation, this can be equivalently be

[1] Some content adapted from previous scribes: Andrew Chambers, Carl Doersch, and those before them.

described by setting the margins (of each inequality) to a constant (we use 1) and then minimizing the magnitude of $||w||$:

$$\begin{aligned} \min_w ||w||^2 \text{ subject to:} \\ w^T f_1 \leq w^T f_2 - 1, \\ \vdots \\ w^T f_{100} \leq w^T f_{101} - 1. \end{aligned}$$

The zero vector is no longer a solution, and the minimization problem picks the out the best from multiple w that satisfy the constraints. Intuitively, minimizing the magnitude of the weights increases the relative role of the constant 1 in satisfying the inequality constraints, thus "maximizing the margin." This problem formulation is called the **hard margin support vector ranking machine**.

1.2 Soft constraints

The first issue of potentially no solutions has not yet been addressed. One realistic goal would be choosing w to minimize the number of constraints are not met. However, this is an NP hard combinatorial problem. Instead, we will use non-negative slack variables ξ_j to describe by how much violated constraints are missed, and we will include these penalties in our cost function:

$$\begin{aligned} \min_{w,\xi} \frac{\lambda}{2}||w||^2 + \sum_{j=1}^{T} \xi_j \text{ subject to:} \\ \xi_j \geq 0, \\ w^T f_1 \leq w^T f_2 - 1 + \xi_1, \\ \vdots \\ w^T f_{100} \leq w^T f_{101} - 1 + \xi_T. \end{aligned}$$

There is one slack variable for each of T constraints, and the factor λ describes the relative importance of the margin (for the correctly evaluated comparisons) versus the penalty paid for the incorrect decisions. A smaller λ allows for larger weights, violates fewer constraints, and takes longer to learn. The slack variables ξ_j are constrained to be non-negative such that the total cost cannot be improved by "making the correct decisions even better."

This **soft margin support vector ranking machine** is still a quadratic program and thus tedious to solve numerically (cubic in number of constraints T). However, it can be re-expressed as a convex programming problem that can be tackled much more efficiently using online gradient descent.

1.3 Online Support Vector Ranking

First, we re-write the optimal values of the slack variables as functions of the weight vector and the features:

$$\xi_j = \max(0, w^T \Delta f_j + 1)$$

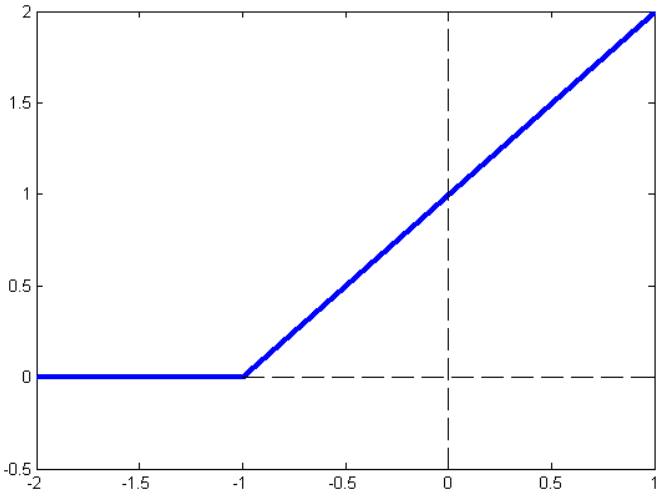

Figure 1: hinge-loss function. The x-axis is $w^T \Delta f_j$, and the y axis is ξ_j.

where

$$\Delta f_1 = f_1 - f_2, \ldots \tag{1}$$

This comes from the fact that if the constraint is satisfied, i.e., $w^T \Delta f_j + 1 \leq 0$, no slack is required and the best penalty would be the minimum value of 0. Otherwise, the minimum ξ required to satisfy the constraint is exactly $w^T \Delta f_j + 1$. This gives us the **hinge-loss function** (Figure 1) as the penalty component of our total cost.

Now, we can write the total cost function and optimization problem as

$$\min_w \left(\frac{\lambda}{2} ||w||^2 + \sum_j^T \max(0, w^T \Delta f_j + 1) \right),$$

or, re-arranging to within the summation,

$$\min_w \left(\sum_j^T \frac{\lambda}{2T} ||w||^2 + \max(0, w^T \Delta f_j + 1) \right).$$

This loss function is convex (as a function of w, since $w^T \Delta f + 1$ is linear (thus convex) w^2 is quadratic (thus convex), and the maximums and sums of 2 convex functions is convex. Though it is not differentiable (sharp corner in the hinge loss), it has sub-gradients everywhere due to the convexity. Thus, we now have a **unconstrained, convex** optimization problem that can be easily solved using sub-gradient descent. The sub-gradient G_j is

$$G_j = \frac{\lambda}{T} w \text{ if } w^T \Delta f_j + 1 \leq 0,$$
$$G_j = \frac{\lambda}{T} w + \Delta f_j \text{ otherwise.}$$

The update rule is then

$$w_{t+1} = w_t - \alpha_t G_t.$$

Intuitively, whenever the constraint is not satisfied, the $-\alpha_t \Delta f_t$ term makes w look more like f_2 and less like f_1 (see equation 1), and the $-\alpha_t \frac{\lambda}{T} w$ term always tends to shrink w in all directions in order to maximize the margin. The magnitude of λ scales this effect, and smaller values shrinks w more slowly in a stronger attempt to get all the labels right.

Similar to the derivations in no-regret learning, we can set the learning rate $\alpha_t \sim \frac{1}{\sqrt{t}}$ to achieve no regret. However, because this problem is strongly convex, we can actually use a faster rate of $\alpha_t \sim \frac{1}{t}$ (not proved or derived here).

1.4 Algorithm pseudo-code

```
w = zeros(size(features_i)) % initialize weights
for t = 1:T
  Δf = features_i − features_j;
  grad = lambda/T*w;
  if dot(w,Δf) > −1 % violated constraint
    grad = grad + Δf;
  end
  w = w − alpha(t)*grad;
end
```

Remarks:

- The problem is no unconstrained, and therefore projection is not needed in the sub-gradient descent.
- The training comparisons can be processed one at a time, but iteration and multiple passes through the data may be needed until convergence. If so, the order of the data should be randomized. It is also possible for very large datasets (i.e., Google) that convergence occurs well before all of the data is seen.
- Upon convergence, w gives exactly the maximum margin classifier.

2 General (not ranking) Support Vector Machines

The binary classification problem can be solved in a very similar way. Here, instead of ranking examples and providing pairwise comparison constraints, examples are classified with labels y_i as either positive or negative (see Figure 2) such that:

$$w^T f_i > 0 \text{ if } y_i = +1,$$
$$w^T f_i < 0 \text{ if } y_i = -1,$$

or simply

$$y_i w^T f_i \geq 0.$$

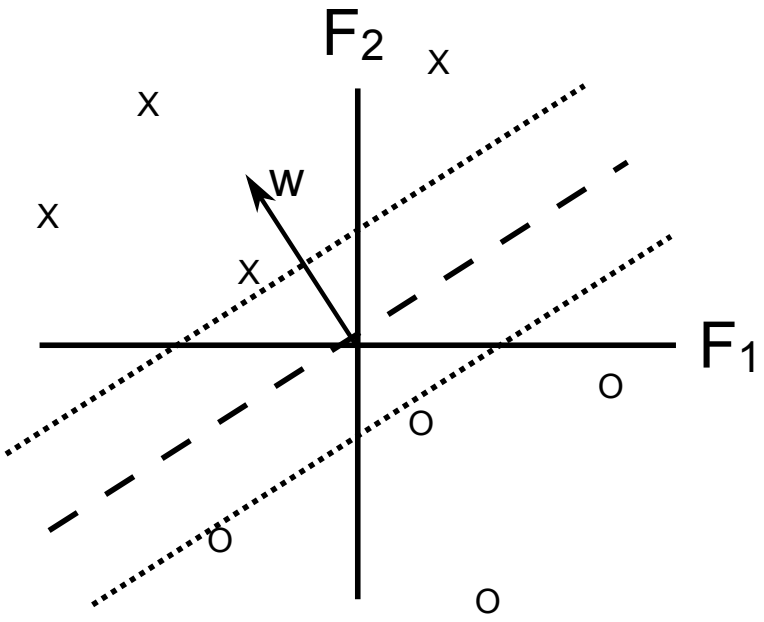

Figure 2: Classification problem. Let the x's be positive labels and the o's be negative.

Just as before, we have the same three problems of potentially no solutions, repeated solutions, and trivial solutions using this base formulation, so we again introduce margins and slack.

$$\min_{w,\xi} \frac{\lambda}{2}||w||^2 + \sum \xi_i \text{ subject to} \tag{2}$$

$$\xi_i \geq 0,$$

$$y_i w^T f_i \geq 1 - \xi_i. \tag{3}$$

The slack variables can be written as

$$\xi_i = \max(0, 1 - y_i w^T f_i),$$

and the constrained problem is then re-expressed as

$$\min_{w} \frac{\lambda}{2}||w||^2 + \sum \max(0, 1 - y_i w^T f_i)$$

This leads to a very similar update rule as the ranking SVM:

```
w = zeros(size(f,1),1) % initialize weights
for t = 1:T
  grad = lambda/T*w;
  if -y(t)*dot(w,f(t)) > -1 % violated constraint
    grad = grad -y(t)*f(t);
  end
  w = w - alpha(t)*grad;
end
```

2.1 Extension: weighted data points

In practice, some data points will be more critical to classify correctly than others. There are three ways of scaling the data such that it is not handled uniformly:

1. Replicate important data points. This works if the desired scaling is by integer factors.
2. Scale the margin of individual data points, i.e., use a number greater than 1 on the R.H.S of Equation 3 for important data.
3. Scale the slack variables, i.e., change the sum in Equation 2 to $\sum s_i \xi_i$ with scalar $s_i > 1$ for important data. This is equivalent to the first method, but allows for non-integer distributions of weights.

2.2 Extension: multi-class SVM

Applications are not limited to binary decisions. For example, candidate pictures of fruit can be classified as apples, oranges, or grapes. In this case, the problem can't be approached with positive versus negative weighted feature sums. Instead, we would set up the inequalities

$$\begin{aligned} w_a^T f_i &\geq w_o^T f_i, \\ w_a^T f_i &\geq w_g^T f_i. \\ &\vdots \end{aligned}$$

using separate weight vectors $w_{a,o,g}$ for each label apple, orange, and grape, assuming training example i was an apple. That is, each example provides two constraints, and the classification of a test example t would be made as $\text{argmax}(w_{\text{label}}^T f_t)$. Naturally, this formulation is expanded in the same way using margins and slack variables, and the three weight vectors can be updated independently using on-line gradient descent.