Statistical Techniques in Robotics (16-831, F08) Lecture #11 (Thursday October 1st)

Support Vector Machines and Learning by Constraints

Lecturer: Drew Bagnell *Scribe: Lindsey Hines*

1 Support Vector Applications

1.1 Little Dog walking robot

One example of a support vector application is crossing terrain with the robot Little Dog. At a high level, we wish to find a set of footsteps that are likely to be successful, which can be accomplished by optimizing body and leg trajectory to minimum cost while maintaining kinematic, dynamic, and collision constraints. In this problem, we can choose a lowest cost sequence of footsteps in a greedy way, each subsequent footstep minimizing the cost. In can be difficult, however, to determine the cost function for footstep planning.

One method for determining the cost function is to utilize human expertise. Since guessing the cost function is hard, a human user is given two pieces of terrain for comparison with the option of choosing which would be better for robot foot placement. An area that is steep would be considered expensive, while concave regions, that allow more stability, would be cheap. A cost function can then be generated from the learned data using regression. Possible features in this particular example include terrain filter response, triangle of support, stepping foot, etc.

1.2 Sports Examples

This technique for learning cost functions can be extended to other problems. For example, there are continuous or discrete rankings of teams in sports. In football and basketball rankings are created to decide which team might be 'better', or more likely to win. Cost features can be generated for two teams through their pairwise comparison.

2 Implementation

For each human compared terrain (i, j) where $cost(i) \leq cost(j)$, a constraint can be formed as $w^T x_i \leq w^T x_j$ where w is a vector of weights and x is a feature set.

Multiple constraints can then be compiled in the form:
$$\begin{aligned} &w^T x_1 \leq w^T x_2 \\ &w^T x_2 \leq w^T x_3 \\ &w^T x_4 \leq w^T x_5 \ \ldots \end{aligned}$$

There are several problems, however, with the constraints as they are now written.

1. There may not be a set of weights that satisfy all of the constraints. The human trainer may not have been consistent.

2. There can be multiple solutions; weights can be scaled and still satisfy the constraints.
3. There is a trivial solution of $w = 0$.

Maximize Margin Approach
Both number 2 and number 3 can be solved with the maximize margin approach. By adding a constant to the inequalities, one must not only satisfy them, but do so by a margin. One can then attempt to maximize the size of the margin as suggested, but is equivalent, and simpler, to fix the margins to a constant value and minimize the weights. The constraints can then be reformatted as follows:

Objective function: $min||w||^2$, subject to:

$w^T x_i^1 \leq w^T x_i^2 - 1$
$w^T x_{i+1}^1 \leq w^T x_{i+1}^2 - 1$
$\ldots$
$w^T x_T^1 \leq w^T x_T^2 - 1$

where the subscripts denote the constraint number and the superscripts distinguish the preferred / not preferred feature set in each particular constraint.

Constraint Softening
Problem number 1 can be solved with the addition of a slack variable ξ_i, where $\xi_i \geq 0$. If a judgment is wrong, the slack variable can be increased until the constraint is satisfied, though a price is paid with an increase of the total cost. The objective function and constraints are now:

Objective function: $\min\lambda||w||^2 + \sum_{i=1}^{n} \xi_i$, subject to:

$w^T x_i^1 \leq w^T x_i^2 - 1 + \xi_i$
$w^T x_{i+1}^1 \leq w^T x_{i+1}^2 - 1 + \xi_{i+1}$
$\ldots$
$w^T x_T^1 \leq w^T x_T^2 - 1 + \xi_T$

where lambda:
$\lambda \leftarrow$ smaller = larger (growing) set of weights, longer to learn but do better.
$\lambda \leftarrow$ larger = smaller (constrained) set of weights, learn initially faster.
Bigger weight = tiny margin, and smaller weight = bigger margin.

Creating Online SVM
In practice, solving the objective function with the series of constraints can take a considerable amount of time. We can, instead, turn it into an online problem, eliminating the constraints.

If we consider the case of conflicting assumptions, we can define a bound on our slack variables ξ_i. If a constraint is met within the margin, ξ_i is not needed and is equal to zero. If $w^T(x_i^2 - x_i^1) + 1 > 0$, where the assumption is wrong or not within the margin, ξ_i is required, but will never need to be greater than $w^T(x_i^2 - x_i^1) + 1$. Our objective function can now be written as:

$\min\lambda||w||^2 + \sum_{i=1}^{n} max(0, w^T(x_i^2 - x_i^1) + 1)$

At each time step, our loss function is now:

$l_t = ||w||^2 + \lambda^i max(0, w^T(x_i^2 - x_i^1) + 1)$

This is, unfortunately undifferentiable, but it is not as scary as it sounds. As can be seen in Figure **??**, the subgradient of the loss function can fall within one of three cases. When $w^T \nabla f < -1$, the subgradient is $2w$, slope equal to zero. When $w^T \nabla f >= -1$ the subgradient is $2w + (x_{pref} - x_{notpref})\lambda$. The third case, $w^T \nabla f = -1$ can be placed in either of the two previous cases, since there are multiple valid subgradients.

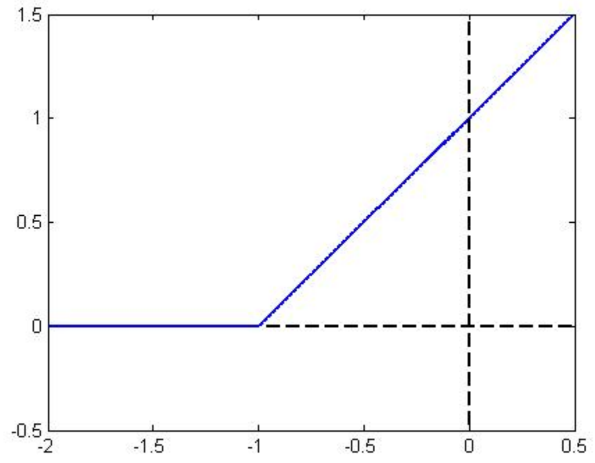

Figure 1: Loss function with discontinuity at -1

The function is convex, which is usually the case when one takes the max of a convex function.

3 Online Algorithm Outline

- Step 1: $w- = 2\alpha_t w$
- Step 2: If a misranking occurred, or it the constraint wasn't satisfied by a margin, $w- = \alpha_t \lambda(x_{pref} - x_{notpref})$
  Else, nothing
- Iterate - May need to pass through data multiple times and data order should be randomized.