Statistical Techniques in Robotics (16-831, F08) Lecture #11 (Tuesday September 30th)

Support Vector Machines and Learning by Constraints

Lecturer: Drew Bagnell *Scribe: Breelyn Kane*

1 Support Vector Applications

1.1 Little dog walking robot

One example of a Support Vector Application is shown through the video of the 'little dog walking robot'. There are methods which can be applied for choosing where to next place the robot's feet.

1.1.1 Goal

Find the next footstep that is likely to be successful, given where the foot is put and how all the joints move. In this case, the goal is to choose a lowest cost sequence of footsteps in a greedy way. Each subsequent footstep would minimize the cost.

The simplest case is a feedback algorithm with coaching. There are two places to go, right or left, and it isn't clear which way is better. A coach would determine which terrain patch is better than another. It would be best to never pick a position where the robot might fall.

Determining the cost of where to put each foot next turns into a regression problem. Cost functions may represent any set of features. Possible features in this particular example: terrain, or the triangle of support. Two feature vectors represent where to place a foot next. They are given a rank of which is the 'best'.

1.1.2 Example Cost function

A learned cost function related to terrain: Red areas may be expensive, and blue areas are cheap. An area that is steep = expensive, and concave regions, that allow more stability, = cheap. A human 'coach' would pick areas of stability more.

The terrain could also be a series of steps where a person may determine it's less expensive to step close to the bottom of a step than close to the top of a next step. The flat areas away from steps are most ideal.

1.2 Sports Examples

There are continuous or discrete rankings of teams in sports. In football/basketball rankings are determined to decide which team might be 'better', or more likely to win. Cost features can be generated for two teams that affect their pairwise comparison.

2 Support Vector Framework

- Feedback Mechanism - instead of random choices (best of two), human picks the next best place (best of all)
- Greedy Version - (one-step inverse optimal control) how do you get the planner to match the humans next best prediction.
- Converging - make minimum of the cost function agree with what human said, the next best step is more informative since it places a constraint on the next possible footstep.

3 Implementation

For a simple example footsteps are ranked linearly. Think in terms of reward instead of cost. Note: this is picking the highest value item causing a sign flip.

- Features of terrain path: vector x_i, ith example.
- Reward $= w^T x_i$, weight on vector of features.
- Series of data points $w^T x_1 \leq w^T x_2$
  $w^T x_2 \leq w^T x_3$
  $w^T x_4 \leq w^T x_4$
  $\ldots$

  There is something wrong with this constraint when $w = 0$, since this solution has triviality.
- To offset this add marginality c
  $w^T x_1 \leq w^T x_2 + c$
- Maximum Margin Approach
  Since the scale of the reward function is arbitrary, the weight vector can be constrained to unit norm: $||w||^2 \leq 1$
  The maximum margin approach is where you are trying to find the largest c that makes the constraint true. Instead equivalently it's possible to keep c fixed and try to minimize w.
  objective function = min$||w||^2$ where constraint is now $w^T x_i^1 \leq w^T x_i^2 + 1$

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

- Noise is something that can happen when a judgement may be wrong. A limitation of linearity is that the feature sets can't interact or 'affect' each other. The problem goes from constrained to unconstrained when the notion of error is added.
  Slack variable $\xi \geq 0$

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

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

  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.
- If the constraint is met within margin then $\xi_i = 0$
  otherwise noise becomes $\xi_i \geq w^T(x_i^2 - x_i^1) + 1$
- Revised objective function:
  $\min \lambda ||w||^2 + \sum_{i=1}^n max(0, w^T(x_i^2 - x_i^1) + 1)$

3.1 Explanation Of Loss

In the case of conflicting assumptions the slack variable is either needed or not. If the constraint is met ξ shouldn't affect it so the slack variable equals zero. Similarly if $w^T(x_i^2 - x_i^1) + 1 > 0$ then the assumption is wrong within the margin and a penalty is paid.

At every time step a loss is paid where the loss function is:

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

Since this value is always positive it represents a convex function, and the max of a convex function is convex.

3.2 Online Gradient Descent

At every time step get loss $l^t = \lambda \; ||w||^2 + \sum_{i=1}^n max(0, w^T(x_i^2 - x_i^1) + 1)$

Then the gradient is taken of this, where if the function is differentiable it agrees with the gradient.

- When
  $w^T(x_i^2 - x_i^1) + 1 \leq 0$, means correctly ranked by a margin of one, the gradient=0
- else
  $(x_t^2 - x_t^1)$ is the gradient with respect to w.

4 Summary

- At each time step initialize support vector ranker
  $w_i^0 = \vec{0}$
- for each sample i predict
  $w_T x_1$, $w_T x_2$, next two
- If ranked correctly and makes the margin
  $w_{t+1} \leftarrow (1 - \eta_t \lambda)\vec{w_t}$, where η = learning rate.
- Else (ranking wrong or they were within one of each other)
  $w_{t+1} \leftarrow (1 - \eta_t \lambda)\vec{w_t} - \eta_t (x_t^{wrongprediction} - x_t^{shouldpredicted})$

  Value on the right is applied to shrink the weight vector.

A Batch algorithm is where the subset of data is randomized. Above was a generalization of the standard support vector machine by using binary classification. Data points are ranked as positive or negative, and progress is observed linearly one data point at a time.