Statistical Techniques in Robotics (16-831, F08) Lecture #12 (Thursday October 2^{nd})

Support Vector Machines Continued

Lecturer: Drew Bagnell *Scribe: Don Burnette*

1 Support Vector Machine Tricks

- Phrase learning problems as constraints, or patches (e.g. 'A' is better than 'B')
- Soften Contraints - Account for mistakes (could be wrong), and removes constraints from problem
- Add margin as a method to fix degeneracies and improve generalization
- Can use a unified batch method (randomization is key) or an online approach
- Take advantage of the fact that the max of a convex functions is convex

2 Binary Support Vector Machine

We consider the case of support vector machines where the result is binary. Suppose we can given a set $\{(x_i, y_i)\}$, where x_i is the input vector and y_i is the true value.

- If $y_i = 1$, we want $w^T x_i \geq 1$, otherwise we want $w^T x_i \leq -1$. This can be written more compactly as
$$y_i w^T x_i \geq 1 \tag{1}$$
- As in the general case, we add a softening term ξ_i, where $\xi_i \geq 0$. Our contraint then becomes
$$\begin{aligned} y_i w^T x_i &\geq 1 - \xi_i && (2) \\ \xi_i &\geq 1 - y_i w^T x_i && (3) \end{aligned}$$
- We wish to minimize $\frac{\lambda||w||^2}{2} + \sum_i max(0, 1 - y_i w^T x_i)$
- The loss for each data point is then $l_i = \frac{\lambda||w||^2}{2} + max(0, 1 - y_i w^T x_i)$
- If $max(0, 1 - y_i w^T x_i) = 0$, then the gradient becomes
$$\nabla l_i = \lambda \vec{w} \tag{4}$$
- If $max(0, 1 - y_i w^T x_i) = 1 - y_i w^T x_i$, then the gradient becomes
$$\nabla l_i = \lambda \vec{w} - y_i x_i \tag{5}$$
- This loss function is often called the 'Hinge Loss'
- Hinge loss is a convex upper bound on the '0/1' Loss

3 More Applications of Online Learning

Projected gradient descent applied to range images. Two neighbooring pixels in an image are likely to be the same depth if they are the same color.

- Use a random field, one node per pixel
- Penalize depth changes
- We calculate the probability desity by taking the product over maximal cliques

$$p(\vec{d}) = \prod_{i \in O} e^{-\gamma |z_i - d_i|^2} \prod_{i,j \in N} e^{-c_{ij} |d_i - d_j|^2} \tag{6}$$

  where O are the observed nodes, are N are neighbooring nodes. If $Pixel_i$ and $Pixel_j$ are 'close' then $c_{ij} = 1$, else $c_{ij} = \epsilon \ll 1$

- We wish to compute $argmax\left(p(\vec{d})\right)$

$$\begin{aligned} &= argmax\left(\log p(\vec{d})\right) && (7) \\ &= argmax\left(\sum_i -\gamma |z_i - d_i|^2 + \sum_{i,j} -c_{ij} |d_i - d_j|^2\right) && (8) \\ &= argmin\left(\sum_i \gamma |z_i - d_i|^2 + \sum_{i,j} c_{ij} |d_i - d_j|^2\right) && (9) \\ & && (10) \end{aligned}$$

- Calculate gradient

$$\frac{\partial}{\partial d_i} l = -2\gamma (z_i - d_i) + \sum_{j \in N} 2c_{ij} (d_i - d_j) \tag{11}$$

- Apply update rule