Statistical Techniques in Robotics (16-831, F10) Lecture #25 (Nov 16, 2010)

Kernel Machines

Lecturer: Drew Bagnell *Scribe: Dave Rollinson*[1]

1 References

There are a number of books and papers on kernels and *Reproducing Kernel Hilbert Spaces.*

- Aronszajn and Bergman - *Theory of Reproducing Kernels* 1950
- Wahba - *Spline Models for Observational Data* 1990
- Cortes and Vapnik - *Support-Vector Networks* 1995

The online algorithm in Section 3 is also described in this paper is a flavor of NORMA (Naive Online Regularized Risk Minimization Algorithm).

- Kivinen, Smola, Williamson - *Online Learning with Kernels* 2003

2 Review

- Ultimately, we wish to learn a function $f : \mathbb{R}^n \to \mathbb{R}$ that assigns a meaningful score given a data point. For example, in binary classification, we would like an $f(\cdot)$ to return positive and negative values, given positive and negative samples, respectively.
- A kernel $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ intuitively measures the *correlation* between $f(\mathbf{x_i})$ and $f(\mathbf{x_j})$. Considering a matrix $\mathbf{K}$ with entries $K_{ij} = K(\mathbf{x_i}, \mathbf{x_j})$, then matrix $\mathbf{K}$ must satisfy the properties:
  - $\mathbf{K}$ is symmetric ($K_{ij} = K_{ji}$)
  - $\mathbf{K}$ is positive-definite ($\forall \mathbf{x} \in \mathbb{R}^n : \mathbf{x} \neq \mathbf{0}, \mathbf{x^T K x} > 0$)

  Hence, a valid kernel is the inner product: $K_{ij} = \langle \mathbf{x_i}, \mathbf{x_j} \rangle$.
- A function can be considered that is a weighted composition of many kernels centered at various locations $\mathbf{x_i}$:
$$f(\cdot) = \sum_{i=1}^{Q} \alpha_i K(\mathbf{x_i}, \cdot), \tag{1}$$
  where Q is the number of kernels that compose $f(\cdot)$ and $\alpha_i \in \mathbb{R}$ is each kernel's associated weight. All functions $f(\cdot)$ with kernel K that satisfy the above properties and can be written in the form of Equation 1 are said to lie in a *Reproducing Kernel Hilbert Space* (RKHS) $\mathcal{H}_K$: $f \in \mathcal{H}_K$

[1]Based on the scribe work of Daniel Munoz, Jack Singleton, and Sergio Valcarcel

3 Online Kernel Machine - Algorithm

- Initialize the function $f = 0$.
- For t = 1 to T:
  1. Observe some measurement over some set of features
  $$x_t$$
  2. Predict the class using $f(x_t)$
  $$f(x_t) = \sum_{i=1}^{n} \alpha_i K(x_i, x_t)$$
  3. Receive loss based on the prediction from $f(x_t)$ and the true class y_t
  $$L(f(x_t), y_t)$$
  4. Update f based on the gradient of the loss function L and learning rate η_t. Shrink the weights on previous kernel functions by $(1 - \eta_t \lambda)$.
  $$f_{t+1} \leftarrow (1 - \eta_t \lambda) f_t + \eta_t \nabla L(f(x_t), y_t)$$
  For Kernel SVMs the loss function is the hinge loss. $f(x_t)$ is the prediction from the sum of weighted kernel functions and y_t is a binary indicator of the true class, either -1 or 1.
  $$L(f(x_t), y_t) = \max(0, 1 - f(x_t) y_t) \tag{2}$$
  Like the loss function L, the gradient ∇L has two cases. One where the prediction is correct by margin = 1, and the other where is not correct by margin = 1.
  $$\nabla L((x_t), y_t) = \begin{cases} 0 & \text{if } (1 - y_i f(x_i)) \leq 0 \\ L'(f(x_t), y_t) f'(x_t) = -y_t K(x_t, \cdot) & \text{otherwise} \end{cases} \tag{3}$$
- The regret is computed as:
$$Regret = \sum_t (C_t(f_t(x_t)) - C_t(f^*(x_t))) | f^* \in H_k$$

The regret bound:
$$Regret = ||\nabla C_t(f)||_k \cdot ||f^*||_k \sqrt{T}$$
$||f^*||$ is the size of the function. $||\nabla C_t(f)||$ can get as big as $\alpha^T K \alpha$.

4 Online Kernel Machines - Discussion

- As mentioned in the previous lecture, this algorithm qualitatively corresponds to adding weighted 'bumps' that predicts some value based on the kernel function in each new observation's neighborhood of the feature space in x.
- As time progresses and the data set grows, the prediction step will take longer and longer to compute. To shorten this computation time you may want to throw out old data points by weight or age. If interested, there are some papers on that use tricks to find sparse solutions to large-scale problems:
  - Rahimi and Recht - *Random Features for Large-Scale Kernel Machines* 2007
  - Dekel, Shalev-Shwartz and Singer *The Forgetron: A Kernel-Based Perceptron on a Budget* 2007
- The choice and tuning of the kernel and their corresponding bandwidth parameters are what affect the bias-variance tradeoff. These are parameters that need to be tuned in addition to the learning rate η and decay rate λ from the update equations.
  - Often simple kernels work quite well. When approaching a new problem it is usually a good idea start with linear or polynomial kernels. Radial basis functions are another good kernel to try early on. Note that any kernels K_1 and K_2 that satisfy the conditions mentioned in Section 2 can be summed to form a new valid kernel.