Statistical Techniques in Robotics (16-831, F09) Lecture #22 (Nov 5, 2009)

Kernel Machines / Functional Gradient Descent

Lecturer: Drew Bagnell *Scribe: Tomas Simon*[1]

1 Motivation

We have seen how to use online convex programming to learn linear functions by optimizing costs of the following form:

$$\underbrace{|y_t - \mathbf{w}^T\mathbf{x}_t|}_{loss} + \underbrace{\lambda\mathbf{w}^T\mathbf{w}^2}_{regularization/prior}$$

We want generalize this to learn over a space of more general functions $f : \mathbb{R}^n \to \mathbb{R}$. The high-level idea is to learn non-linear models using the same gradient-based approach used to learn linear models.

$$|y_t - f(\mathbf{x}_t)| + \lambda||f||^2$$

Up till now we have only considered functions of the form $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, but we will now extend this to a more general space of functions, the Reproducing Kernel Hilbert Space.

2 Reproducing Kernel Hilbert Space

The Reproducing Kernel Hilbert Space (RKHS), denoted by $\mathcal{H}_k$, is the space of functions $f(\cdot)$ that can be written as $\sum_i \alpha_i k(\mathbf{x}_i, \cdot)$, where $k(\mathbf{x}_i, \mathbf{x}_j)$ satisfies certain properties described below.

To be able to manipulate objects in this space of functions, we will look at some key properties:

The *inner product* of $f, g \in \mathcal{H}_k$ is defined as

$$\langle f, g\rangle \triangleq \sum_i \sum_j \alpha_i \beta_j k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\beta}$$

where $f(\cdot) = \sum_i \alpha_i k(\mathbf{x}_i, \cdot)$, $g(\cdot) = \sum_j \beta_j k(\mathbf{x}_j, \cdot)$, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are vectors comprising, respectively, α_i and β_i components, and $\mathbf{K}$ is n by m (where n is the number of $\mathbf{x}_i$ in f, and m those in g) with $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

Note that this will satisfy linearity (in both arguments):

- $\langle \lambda f, g\rangle = \lambda\langle f, g\rangle$
- $\langle f_1 + f_2, g\rangle = \langle f_1, g\rangle + \langle f_2, g\rangle$

[1]Based on the scribe work of Daniel Munoz and Bryan Low

With this inner product, the *norm* will be: $||f||^2 = \langle f, f \rangle = \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}$.

For these definitions to work, we can see that the *kernel* $k(\cdot, \cdot)$ will have to satisfy the following conditions:

- $\mathbf{K}$ is symmetric ($K_{ij} = K_{ji}$), so $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i)$
- $\mathbf{K}$ is positive-definite ($\forall \mathbf{x} \in \mathbb{R}^n : \mathbf{x} \neq \mathbf{0}, \mathbf{x}^T \mathbf{K} \mathbf{x} > 0$)

An example of a valid kernel for $\mathbf{x} \in \mathbb{R}^n$ is the inner product: $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$. Intuitively, the kernel measures the *correlation* between $\mathbf{x}_i$ and $\mathbf{x}_j$.

A very commonly used kernel is the RBF or Radial Basis Function kernel, which takes the form $k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{1}{\gamma}||\mathbf{x}_i - \mathbf{x}_j||^2}$. With this kernel in mind, a function can be considered as a weighted (by α_i) composition of bumps (the kernels) centered at the Q locations $\mathbf{x_i}$:

$$f(\cdot) = \sum_{i=1}^{Q} \alpha_i K(\mathbf{x}_i, \cdot),$$

See figure 4 for an illustration.

Note that as long as you can define a valid kernel $k(\mathbf{x}_i, \mathbf{x}_j)$, the samples $\mathbf{x}$ themselves can in principle be any object, eg. text strings. The kernel function might in this case measure the similarity between samples using some sort of edit distance. Each type of data will require kernel functions that are appropriate for that particular data.

3 Loss Minimization

Let us consider again our cost function defined over all functions f in our RKHS (but using the hinge loss this time):

$$L[f] = max(0, 1 - y_t f(\mathbf{x}_t)) + \lambda \langle f, f \rangle \tag{1}$$

The purpose of $\langle f, f \rangle$ is to penalize the complexity of the solution f. Here it acts like the log of a gaussian prior over functions. Intuitively, the probability can be thought of as being distributed according to $P(f) = \frac{1}{Z} e^{-\frac{1}{2}\langle f, f \rangle}$ (in practice this expression doesn't work because Z becomes infinite).

We want to find the best function f in our RKHS so as to minimize this cost, and we will do this by moving in the direction of the negative gradient: $f - \alpha \nabla L$. To do this, we will first have to be able to express the gradient of a function of functions (ie. a *functional* such as $L[f]$).

3.1 Functional gradient

A gradient can be thought of as:

- Vector of partial derivatives
- Direction of steepest ascent

- Linear approximation of the function (or functional), ie. $f(x_0+\epsilon) = f(x_0)+\epsilon\cdot\underbrace{\nabla f(x_0)}_{gradient}+O(\epsilon^2)$.

We will use the third definition. A *functional* $F : f \to \mathbb{R}$ is a function of functions $f \in \mathcal{H}_K$. Examples:

- $F[f] = ||f||^2$
- $F[f] = (f(x) - y)^2$
- $F[f] = \frac{\lambda}{2}||f||^2 + \sum_i (f(x_i) - y_i)^2$

A functional gradient $\nabla F[f]$ is defined implictly as the linear term of the change in a function due to a small perturbation ϵ in its input: $F[f + \epsilon g] = F[f] + \epsilon\langle \nabla F[f], g\rangle + O(\epsilon^2)$

- Example: $\nabla F[f] = \nabla||f||^2 = 2f$

$$\begin{aligned} F[f+\epsilon g] &= \langle f+\epsilon g, f+\epsilon g\rangle \\ &= ||f|| + 2\langle f, \epsilon g\rangle + \epsilon^2||g|| \\ &= ||f|| + \epsilon\langle 2f, g\rangle + O(\epsilon^2) \end{aligned}$$

3.2 More functional gradients

- The *evaluation functional* evaluates f at the specified x: $F_x[f] = f(x)$
  - Gradient is $\nabla F_x = K(x, \cdot)$

$$\begin{aligned} F_x[f+\epsilon g] &= f(x) + \epsilon g(x) + 0 \\ &= f(x) + \epsilon\langle K(x,\cdot), g\rangle + 0 \\ &= F_x[f] + \epsilon\langle \nabla F_x, g\rangle + O(\epsilon^2) \end{aligned}$$

  - It is called a *linear functional* due to the lack of a multiplier on perturbation ϵ.
- Consider *differentiable* functions $C : \mathbb{R} \to \mathbb{R}$ that are functions of functionals G, $C(G[f])$. Our cost function $L[f]$ from before was such a function, these are precisely the functions that we are interested in minimizing.
- The derivative of these functions follows the chain rule: $\nabla C(G[f]) = C'(G[f])\nabla G[f]$
  - Example: If $C = (||f||^2)^3$, then $\nabla C = 3(||f||^2)^2(2f)$

4 Functional gradient descent

- Consider the regularized least squares loss function $L[f]$

$$\begin{aligned} L[f] &= (f(x_i) - y_i)^2 + \lambda||f||^2 \\ L[f] &= (F_{x_i}[f] - y_i)^2 + \lambda||f||^2 \\ \nabla L[f] &= 2(f(x_i) - y_i)K(x_i, \cdot) + 2\lambda f \end{aligned}$$

- Update rule:

$$\begin{aligned} f^{t+1} &\leftarrow f^t - \eta_t \nabla L \\ &\leftarrow f^t - \eta_t(2(f^t(x_i) - y_i)K(x_i, \cdot) + 2\lambda f^t) \\ &\leftarrow f^t(1 - 2\eta_t\lambda) - \eta_t(2(f^t(x_i) - y_i)K(x_i, \cdot)) \end{aligned}$$

- Need to perform $O(T)$ work at each time step
- Example: Figure 4 shows an update over 3 points $\{(x_1, +), (x_2, -), (x_3, +)\}$. The individual kernels centered at the points are **independently** drawn with colored lines. After 3 updates, the function f looks like the solid black line.

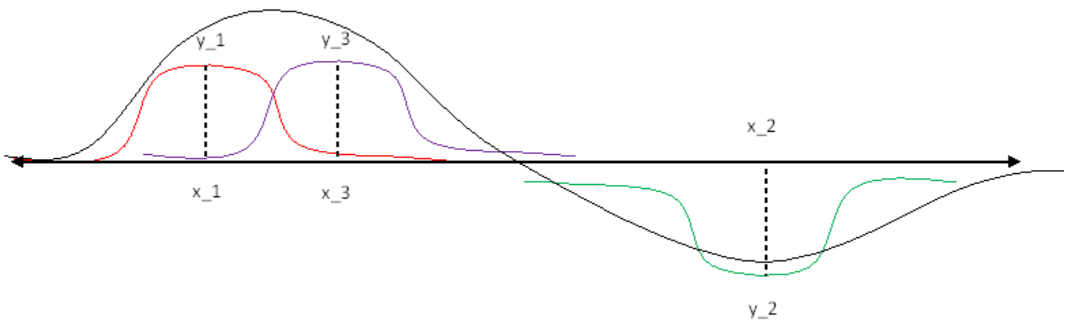

Figure 1: Illustration of function after 3 updates

- **Representer Theorem** (informally): Given a loss function and regularizer objective with many data points $\{x_i\}$, the minimizing solution f^* can be represented as

$$f^*(\cdot) = \sum_i \alpha_i K(x_i, \cdot) \tag{2}$$

- Alternate idea from class: perform gradient descent in the space of α coefficients: $\nabla_\alpha L$
  - Takes n^2 iterations to get same performance (n = number of iterations of functional gradient descent)
  - Every iteration is $O(T^2)$

5 Kernel SVM

- General loss function: $L[f] = \frac{\lambda}{2}||f||^2 + C_t(F_{x_i}[f])$
- General update rule: $f_{t+1} \leftarrow (1 - \lambda\eta_t)f_t - \eta_t C'_t(F_{x_i}[f])K(x_i, \cdot)$
- SVM cost function: $C_t(F_{x_i}) = \max(0, 1 - f(x_i)y_i)$

$$\nabla C_t = \begin{cases} 0 & , 1 - y_i f(x_i) \leq 0 \\ (C'(F_{x_i}[f]))(\nabla F_{x_i}[f]) = -y_i K(x_i, \cdot) & , \text{otherwise} \end{cases} \tag{3}$$