Statistical Techniques in Robotics (16-831, F010) Lecture #22 (Nov 11, 2010)

Kernel Machines / Functional Gradient Descent

Lecturer: Drew Bagnell *Scribe: Varun Ramakrishna*[1]

1 The Basic Idea

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.

2 Review of Kernels

We observed in Gaussian Processes that the mean function had the form $\mu_{f(\mathbf{x})|data} = K^T K_{data}^{-1} y$ which can be interpreted as a sum of Kernel functions on the observed data points:

$$f = \sum_i \alpha_i K(x_i, \cdot) \tag{1}$$

Kernels functions are a natural choice of non-linear functions for our task i.e to learn a function $f : \mathbb{R}^n \to \mathbb{R}$ that assigns a meaningful score given a data point.

E.g. in binary classification, we would like $f(\cdot)$ 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 ($\forall \mathbf{x} \in \mathbb{R}^n : \mathbf{x} \neq \mathbf{0}, \mathbf{x^T K x} > 0$)

[1] Based on the scribe work of Daniel Munoz and Tomas Simon

However to do gradient descent on the space of such functions, we need the notion of a distance, norm and an inner product. We formalize this by introducing the Reproducing Kernel Hilbert Space.

3 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

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

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):

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

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

The *reproducing property* is observed by taking the inner-product of a function with a kernel

$$\langle f, K(\mathbf{x_j}, \cdot) \rangle = \langle \sum_{i=1}^{Q} \alpha_i K(\mathbf{x_i}, \cdot), K(\cdot, \mathbf{x_j}) \rangle = \sum_{i=1}^{Q} \alpha_i \langle K(\mathbf{x_i}, \cdot), K(\cdot, \mathbf{x_j}) \rangle = \sum_{i=1}^{Q} \alpha_i K(\mathbf{x_i}, \mathbf{x_j}) = f(\mathbf{x_j}) \quad (2)$$

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 5 for an illustration.

4 Loss Minimization

Let us consider again our cost function defined over all functions f in our RKHS, as before our loss is :

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

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 $L[f]$).

4.1 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 $f \in \mathcal{H}_K$. As an example let us write the terms of our loss function from abov as functionals:

- $F_1[f] = ||f||^2$
- $F_2[f] = (f(x) - y)^2$
- $F[f] = \frac{\lambda}{2}||f||^2 + \sum_i (f(x_i) - y_i)^2$

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: $F[f + \epsilon g] = F[f] + \epsilon \langle \nabla F[f], g \rangle + O(\epsilon^2)$.

Before computing the gradients for these functionals, let us look at a few tools that will help us derive the gradient of the loss functional

4.2 Chain rule for functional gradients

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:

$$\nabla C(G[f]) = \frac{\partial C(G[f])}{\partial \lambda}|_{G(f)} \nabla G[f] \tag{3}$$

Example: If $C = (||f||^2)^3$, then $\nabla C = 3(||f||^2)^2(2f)$

4.3 Another useful functional gradient

Another useful functio nthat we come across often is the evaluation functional. 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.

4.4 Functional gradient of the regularized least squares loss function

- Let's look at the functional gradient of the first term of the loss function:

$$\nabla F[f] = \nabla ||f||^2 \tag{4}$$

  Expanding it out using a Taylor's series type expansion

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

  We observe that

$$\nabla F[f] = \nabla ||f||^2 = 2f \tag{5}$$

- Now for the second term of the loss function

$$F[f] = \sum_i (f(x_i) - y_i)^2 \tag{6}$$

  Using the chain rule we have

$$\nabla F[f] = 2(f(x_i) - y_i) \nabla (f(x_i)) \tag{7}$$

  We observe that $\nabla(f(x_i))$ is the functional gradient of the evaluation functional. Substituting in the gradient of the evaluaton functional as computed in the previous section we have :

$$\nabla F[f] = 2(f(x_i) - y_i) K(x_i, \cdot) \tag{8}$$

5 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 5 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.

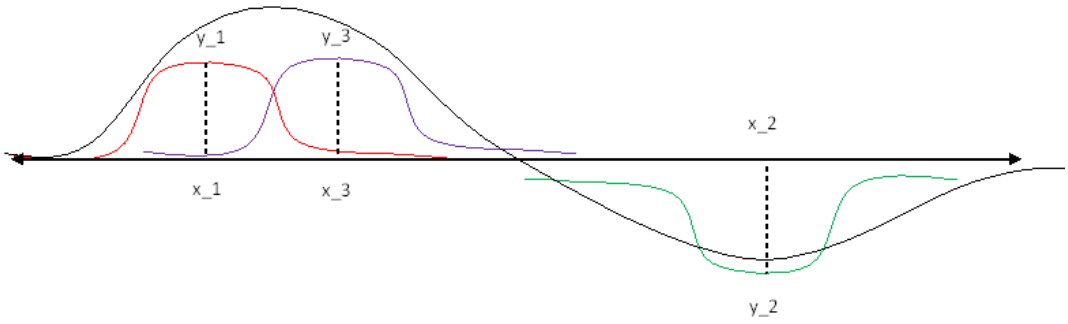

Figure 1: Illustration of function after 3 updates

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

$$f^*(\cdot) = \sum_i \alpha_i K(x_i, \cdot) \tag{9}$$

- 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 $O(T^2)$