1 Goal

The high-level idea is to learn non-linear models using the same gradient-based approach used to learn linear models. Hopefully this will result in better models that improve classification.

2 Review

• Ultimately, we wish to learn a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ 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 $K : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ intuitively measures the correlation between $f(x_i)$ and $f(x_j)$. Considering a matrix $K$ with entries $K_{ij} = K(x_i, x_j)$, then matrix $K$ must satisfy the properties:
  - $K$ is symmetric ($K_{ij} = K_{ji}$)
  - $K$ is positive-definite ($\forall x \in \mathbb{R}^n : x \neq 0, x^T K x > 0$)

Hence, a valid kernel is the inner product: $K_{ij} = \langle x_i, x_j \rangle$.

• A function can be considered as a weighted composition of many kernels centered at various locations $x_i$:

$$f(\cdot) = \sum_{i=1}^{Q} \alpha_i K(x_i, \cdot), \quad (1)$$

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) $\mathcal{H}_K$: $f \in \mathcal{H}_K$

  - The inner-product of two functions $f$ and $g$ is defined as

$$\langle f, g \rangle = \sum_{i=1}^{Q} \sum_{j=1}^{P} \alpha_i \beta_j K(x_i, x_j) = \alpha^T K \beta, \quad (2)$$

where $\alpha \in \mathbb{R}^Q$ and $\beta \in \mathbb{R}^P$ are the kernel coefficients for $f$ and $g$, respectively.

  * By definition, the following property holds: $\langle K(x_i, \cdot), K(\cdot, x_j) \rangle = K(x_i, x_j)$
The reproducing property is observed by taking the inner-product of a function with a kernel:
\[ \langle f, K(x_j, \cdot) \rangle = \langle \sum_{i=1}^Q \alpha_i K(x_i, \cdot), K(\cdot, x_j) \rangle = \sum_{i=1}^Q \alpha_i \langle K(x_i, \cdot), K(\cdot, x_j) \rangle = \sum_{i=1}^Q \alpha_i K(x_i, x_j) = f(x_j) \]

Note that due to positive-definite constraint, the squared norm of a function \( f \) is always positive when \( \alpha \neq 0 \).
\( (||f||^2 = \langle f, f \rangle = \alpha^T K \alpha > 0) \)

A functional \( F : f \rightarrow \mathbb{R} \) is a function of functions \( f \in \mathcal{H}_K \). Examples:

- \( F[f] = ||f||^2 \)
- \( F[f] = (f(x) - y)^2 \)
- \( F[f] = \frac{1}{2}||f||^2 + \sum_i (f(x_i) - y_i)^2 \)

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

Example:
\[ F[f + \epsilon g] = \langle f + \epsilon g, f + \epsilon g \rangle \\
= ||f|| + 2 \langle f, \epsilon g \rangle + \epsilon^2 ||g|| \\
= ||f|| + \epsilon \langle 2f, g \rangle + O(\epsilon^2) \]

3 More functional gradients

- Consider differentiable functions \( C : \mathbb{R} \rightarrow \mathbb{R} \) that are functions of functionals \( G, C(G[f]) \). We will be minimizing these (cost) functions in the near future.

- The derivative of these functions follows the chain rule: \( \nabla C(G[f]) = C'(G[f]) \nabla G[f] \)

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

- The evaluation functional evaluates \( f \) at the specified \( x \): \( F_x[f] = f(x) = e_x[f] \)

- Its gradient is \( \nabla e_x = K(x, \cdot) \)

\[ e_x[f + \epsilon g] = f(x) + \epsilon g(x) + 0 \]
\[ = f(x) + \epsilon \langle K(x, \cdot), g \rangle + 0 \]
\[ = e_x[f] + \epsilon \langle \nabla e_x, g \rangle + O(\epsilon^2) \]

- Called a linear functional due to lack of multiplier on perturbation \( \epsilon \)

4 Functional gradient descent

- Consider the regularized least squares loss function \( L[f] \)

\[ L[f] = (f(x_i) - y_i)^2 + \lambda ||f||^2 \]
\[ \nabla L[f] = 2(f(x_i) - y_i)K(x_i, \cdot) + 2\lambda f \]
• Update rule:

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

• 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.

![Illustration of function after 3 updates](image)

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

\[
f^*(\cdot) = \sum_i \alpha_i K(x_i, \cdot)
\]

• Alternate idea from class: perform gradient descent in the space of \(\alpha\) 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: \(L[f] = \frac{1}{2}||f||^2 + C_t(F_x[f])\)

• General update rule: \(f_{t+1} \leftarrow f_t(1 - \lambda\eta_t) - \eta_t C'_t(F_x[f])K(x_i, \cdot)\)

• SVM cost function: \(C_t(F_x) = \max(0, 1 - f(x_i)y_i)\)

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

(4)