Machine Learning, Spring 2019
EN 2105
MWF 9-9:50am

Chao Lan
EN 4084A
clan@uwyo.edu

Kernel Methods

Kernel methods deal with non-linearly-separable data. It is observed that these data often become (more) linearly separable when mapped to a higher dimensional feature space. A kernel method maps data to a high dimensional space and builds a linear model there.

[*Discussion*] What is the geometric interpretation of kernel methods?

Example 1: Kernel Ridge Regression

Let ϕ be a function mapping x from its raw feature space to a higher dimensional space. To learn a ridge regression model β, one can solve the following objective function

$$J(\beta) = \sum_{i=1}^{n} (\phi(x_i)^T \beta - y_i)^2 + \lambda \beta^T \beta. \tag{1}$$

After that, one can make prediction on a testing instance $\phi(x_t)$ by

$$\hat{y}_t = \phi(x_t)^T \beta. \tag{2}$$

Here is a typical way to make prediction on $\phi(x_t)$: first designs an explicit mapping function ϕ, optimizes (1) using standard ridge regression, and then applies (2) to obtain $\hat{y}_t$. The limitation of this process is that it is computationally expensive, especially when ϕ is mapping data to a very high (possibly infinite) dimensional space.

Kernel methods bypass this limitation by making predictions without computing the explicit $\phi(x)$ and β. Applying the critical point method, we have

$$J'(\beta) = \sum_{i=1}^{n} 2(\phi(x_i)^T \beta - y_i)\phi(x_i) + 2\lambda\beta. \tag{3}$$

Solving $J'(\beta) = 0$ for β, we have

$$\beta = \sum_{i=1}^{n} -\frac{1}{\lambda}(\phi(x_i)^T \beta - y_i) \cdot \phi(x_i) = \sum_{i=1}^{n} \alpha_i \, \phi(x_i), \tag{4}$$

where $\alpha_i = -\frac{1}{\lambda}(\phi(x_i)^T \beta - y_i)$. This suggests the optimal β is a linear combination of training instances.[1] Plugging this back to $J(\beta)$, we have

$$\begin{aligned} J(\beta) &= \sum_{i=1}^{n} \left(\phi(x_i)^T \left(\sum_{i'=1}^{n} \alpha_{i'} \, \phi(x_{i'})\right) - y_i\right)^2 + \lambda \left(\sum_{i=1}^{n} \alpha_i \, \phi(x_i)\right)^T \left(\sum_{i'=1}^{n} \alpha_{i'} \, \phi(x_{i'})\right) \\ &= \sum_{i=1}^{n} \left(\sum_{i'=1}^{n} \alpha_{i'} \phi(x_{i'})^T \phi(x_i) - y_i\right)^2 + \lambda \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'} \phi(x_i)^T \phi(x_{i'}) \\ &= \sum_{i=1}^{n} \left(\sum_{i'=1}^{n} \alpha_{i'} \kappa(x_{i'}, x_i) - y_i\right)^2 + \lambda \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'} \kappa(x_i, x_{i'}), \end{aligned} \tag{5}$$

[1] Recall we have a similar insight in LSVM.

where

$$\kappa(x_{i'}, x_i) = \phi(x_{i'})^T \phi(x_i) \tag{6}$$

is the inner product function – it is also called the kernel function.

(5) shows $J(\beta)$ can be expressed by just inner products of $\phi(x)$'s but not any $\phi(x)$, so we can optimize $J(\beta)$ without knowing the explicit form of $\phi(x)$. This is known as the kernel trick.

Of course, we still need an explicit form of $\kappa(x_{i'}, x_i)$, but its computation is way more efficient than first computing $\phi(x_i)$'s and then computing their inner product. There are many well-studied kernel functions [2] such as Gaussian kernel

$$\kappa(x_i, x_{i'}) = \exp\left(-\frac{||x_i - x_{i'}||^2}{2\sigma^2}\right), \tag{7}$$

where σ is a hyper-parameter, or polynomial kernel

$$\kappa(x_i, x_{i'}) = (x_i^T x_{i'} + a)^d, \tag{8}$$

where a, d are hyper-parameters.

Each kernel function corresponds to an explicit mapping function ϕ. For example, the Gaussian kernel corresponds to a ϕ that maps data to an infinitely high dimensional feature space

$$\phi(x) = \exp\left(-\frac{x^2}{2\sigma^2}\right)\left[1, \frac{x}{\sigma\sqrt{1!}}, \frac{x^2}{\sigma^2\sqrt{2!}}, \frac{x^3}{\sigma^3\sqrt{3!}}, \dots\right]^T, \tag{9}$$

which is impossible to compute.

[*Exercise*] Let $x = [x_{\cdot 1}, x_{\cdot 2}]^T$ and $\phi(x) = [x_{\cdot 1}^2,\ \sqrt{2}x_{\cdot 1}x_{\cdot 2},\ x_{\cdot 2}^2]^T$. Verify that ϕ corresponds to the polynomial kernel $\kappa(x_i, x_{i'}) = \phi(x_i)^T \phi(x_{i'}) = (x_i^T x_{i'})^2$.

In addition, we can construct new kernel functions based on existing ones.[3]

[*Exercise*] If $\kappa_a(x, x')$ is a kernel, verify that $\kappa(x, x') := \lambda \cdot \kappa_a(x, x')$ is also a kernel. (Tip: if you can find an explicit ϕ for κ, then it is a valid kernel.)

[*Exercise*] If $\kappa_a(x, x')$, $\kappa_b(x, x')$ are two kernels, verify that $\kappa(x, x') := \kappa_a(x, x') + \kappa_b(x, x')$ is also a kernel. (Tip: if you can find an explicit ϕ for κ, then it is a valid kernel.)

From (5) we can write $J(\alpha)$ in a matrix form

$$\begin{aligned} J(\alpha) &= \sum_{i=1}^n \left(\sum_{i'=1}^n \alpha_{i'} \kappa(x_{i'}, x_i) - y_i\right)^2 + \lambda \sum_{i=1}^n \sum_{i'=1}^n \alpha_i \alpha_{i'} \kappa(x_i, x_{i'}) \\ &= (K\alpha - Y)^T (K\alpha - Y) + \lambda \alpha^T K \alpha, \end{aligned} \tag{10}$$

where $\alpha = [\alpha_1, \dots, \alpha_n]^T$, $Y = [y_1, \dots, y_n]^T$, and $K \in \mathcal{R}^{n \times n}$ is the Gram matrix where

$$K_{ii'} = \kappa(x_i, x_{i'}). \tag{11}$$

[*Exercise*] Verify (10).

[2] See e.g., [PRML, Chapter 6.2].
[3] See e.g., [PRML, Ch 6.2].

Clearly $J(\alpha)$ is a quadratic function of α. Applying the critical point method, we have

$$\alpha = (K + \lambda I)^{-1} Y. \tag{12}$$

[*Exercise*] Verify (12).

Once α is obtained, we can make prediction on any instance $\phi(x_t)$ by

$$\phi(x_t)^T \beta = \phi(x_t)^T \sum_{i=1}^n \alpha_i \phi(x_i) = \sum_{i=1}^n \alpha_i \phi(x_t)^T \phi(x_i) = \sum_{i=1}^n \alpha_i \kappa(x_t, x_i). \tag{13}$$

Note that we do not need explicit form of $\phi(x)$ or α to make prediction.

Also note all training instances are used to make predictions in (13). This somewhat increases the computational and memory costs, and is a limitation of the kernel methods. During training, computing the inverse of n-by-n Gram matrix is also as expensive as $O(n^3)$.

In kernel ridge regression, we see the optimal β can be linearly expressed by training instances, which is a key that enables $J(\beta)$ to be re-represented using only kernels. In fact, this observation holds broadly on many machine learning methods as long as they have a certain form of objective function. This form is specified by the Representer Theorem.

Theorem 1. *Let κ be a kernel on X and $\mathcal{F}$ be its associated Reproducing Kernel Hilbert Space (RKHS). Fix instances $x_1, \ldots, x_n \in X$ and consider the following optimization problem*

$$\min_{f \in \mathcal{F}} L(f(x_1), \ldots, f(x_n)) + \Omega(||f||), \tag{14}$$

where L depends on x_i only through f and Ω is a non-decreasing function. If (14) has a minimizer, then one minimizer f_* has the form

$$f_* = \sum_{i=1}^n \alpha_i \, \kappa(\cdot, x_i), \tag{15}$$

where $\alpha_i \in \mathbb{R}$. And if Ω is strictly increasing, then every minimizer has the form (15).

[*Discussion*] What machine learning methods can be kernelize? Can we kernelize least square?

Kernel methods can deal with non-linearly separable because its mapping ϕ is often non-linear.

Example 2: Kernel Linear SVM (a.k.a. SVM)

Recall the Wolfe dual problem of hard-margin LSVM is

$$\begin{aligned} &\min_{\alpha} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j \\ &\text{s.t.} \sum_{i=1}^n \alpha_i y_i = 0, \text{ and } \alpha_i \geq 0, \forall i, \end{aligned} \tag{16}$$

whic only involves inner products of instances. Replacing x with $\phi(x)$ and applying the kernel trick, we have the optimization problem of kernel LSVM (a.k.a. SVM)

$$\begin{aligned} &\min_{\alpha} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \kappa(x_i, x_j) \\ &\text{s.t.} \sum_{i=1}^n \alpha_i y_i = 0, \text{ and } \alpha_i \geq 0, \forall i. \end{aligned} \tag{17}$$

Similar discussion applies to soft-margin LSVM.