ELEG/CISC 867: Advanced Machine Learning **Spring 2019**

Lecture 13: Kernel Method

Lecturer: Xiugang Wu *05/14/2019*

Last time we described the SVM paradigm for learning halfspaces and we mentioned that if the training set is not linearly separable in the input space $\mathcal{X}$ then one can map the examples to a higher dimensional feature space $\mathcal{F}$ and run SVM on $\mathcal{F}$. While this approach enriches the expressive power of halfspaces by mapping the data to a high dimensional space, it raises both sample complexity and computational complexity challenges — learning halfspaces in the high dimensional space requires a sample complexity that grows with the dimension, and computations over high dimensional spaces can be expensive. In the last lecture, we tackled the sample complexity issue using the concept of margin; today we will address the computational complexity challenge using the method of kernels.

1 Embedding into Feature Spaces

To illustrate the idea of embedding data in $\mathcal{X}$ into feature space $\mathcal{F}$, consider the following example. Let $\mathcal{X} = \mathbb{R}$ and consider a training dataset where

$$x \in \{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$$

and $y = +1$ if $|x| \geq 3$ and $y = -1$ if $|x| < 3$. A halfspace over $\mathcal{X} = \mathbb{R}$ is a threshold function and it cannot be used to separate the training set. Instead, one can take the mapping $\psi(x) = (x, x^2)$ and learn a halfspace over the feature space $\mathcal{F} = \mathbb{R}^2$. After embedding the data in $\mathcal{X}$ into $\mathcal{F}$, the training set can now be separated by the halfspace

$$\text{sgn}(\mathbf{w}^T\mathbf{f} + b)$$

where $\mathbf{w} = (0, 1)$ and $b = -5$. Equivalently, the training set in the original space $\mathcal{X}$ can be separated by the interval

$$\text{sgn}(x^2 - 5)$$

which is obtained by replacing $\mathbf{f}$ with $\psi(x)$ in the halfspace classifer $\text{sgn}(\mathbf{w}^T\mathbf{f} + b)$ over $\mathcal{F}$.

In general, one can take the following steps to enrich the expressive power of linear predictors:

- Given the domain set $\mathcal{X}$ and the learning task, choose a mapping $\psi : \mathcal{X} \to \mathcal{F}$, where $\mathcal{F}$ is the feature space and usually takes the form of $\mathbb{R}^d$ for some d.
- Learn a linear predictor $h(\mathbf{f})$ over the feature space $\mathcal{F}$ using the training data $\{(\psi(\mathbf{x}_i), y_i)\}_{i=1}^n$.
- When given a new instance $\mathbf{x}$ in $\mathcal{X}$, make the prediction $\hat{y} = h(\psi(\mathbf{x}))$.

The success of the above learning paradigm depends on choosing a good ψ for a given learning task: you want to pick ψ such that the image of the data distribution is (or close to being) linearly separable in the feature space, thus making the resulting algorithm a good learner for the given task. This requires prior knowledge about the task; for example, if we believe that positive examples can be distinguished by some ellipse, we can define ψ to include all the monomials up to order 2.

One can also rely on some generic ψ mappings to enrich the class of halfspaces and extend its expressive power. One notable example is polynomial mappings. Recall that the prediction of a standard halfspace

classifier is based on the linear function $\mathbf{x} \mapsto \mathbf{w}^T\mathbf{x}$. We can generalize it to the prediction based on polynomial function $\mathbf{x} \mapsto p_k(\mathbf{x})$, where $p_k(\mathbf{x})$ is a multivariate polynomial of degree k. We now describe the ψ mapping that enables such a generalization.

- First consider the one-dimensional case when $\mathcal{X} = \mathbb{R}$. In this case, the polynomial function $p_k(x)$ is given by

$$p_k(x) = \sum_{j=0}^{k} w_j x^j,$$

  where $\mathbf{w} \in \mathbb{R}^{k+1}$ is the vector of coefficients of the polynomial that we need to learn. Equivalently, we can write $p_k(x)$ as $p_k(x) = \mathbf{w}^T\psi(x)$, where $\psi : \mathbb{R} \to \mathbb{R}^{k+1}$ is defined to be the mapping

$$\psi(x) = (1, x, x^2, \ldots, x^k).$$

  Therefore, learning a k-degree polynomial predictor over $\mathcal{X} = \mathbb{R}$ can be done by learning a linear predictor over the feature space $\mathbb{R}^{k+1}$.

- Generally, when $\mathcal{X} = \mathbb{R}^m$ the polynomial function $p_k(\mathbf{x})$ is given by

$$p_k(\mathbf{x}) = \sum_{r=0}^{k} \sum_{J \in [1:m]^r} w_J \prod_{i=1}^{r} x_{J_i}.$$

  As before, we can rewrite $p_k(\mathbf{x})$ as $p_k(\mathbf{x}) = \mathbf{w}^T\psi(\mathbf{x})$, where $\psi : \mathbb{R}^m \to \mathbb{R}^d$ is defined to be the mapping

$$\psi(\mathbf{x}) = \left\{ \prod_{i=1}^{r} x_{J_i} \right\}_{J \in [1:m]^r : r \leq k}$$

  and the dimension d of the feature space is given by

$$d = \sum_{r=0}^{k} m^r.$$

  Note that here compared to the original space $\mathcal{X} = \mathbb{R}^m$, the feature space $\mathcal{F} = \mathbb{R}^d$ could have a much higher dimension and learning over the feature space might be too costly computationally. To address this issue, we will introduce the kernel trick, which allows us to learn a predictor over $\mathbb{R}^d$ solely based on calculations over the original space $\mathbb{R}^m$.

2 The Kernel Trick

A kernel function for a mapping ψ is a function that implements inner product in the feature space, namely,

$$K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle.$$

We will soon demonstrate the advantage of using kernels, that is, they allow to calculate $\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$ in the feature space efficiently, without having to applying ψ and expressing points in the feature space explicitly. But before that, let's first ask ourselves whether it is enough to just use inner products $\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$ in order to learn a predictor in the feature space. The following theorem implies that this is indeed true for a class of learning problems which encompass SVM.

Theorem 2.1 (Representer Theorem) *Consider any learning rule of the following form:*

$$\mathbf{w}^* = \underset{\mathbf{w}}{argmin}\, f(\{\langle \mathbf{w}, \psi(\mathbf{x}_i)\rangle\}_{i=1}^n) + R(\|\mathbf{w}\|), \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is an arbitrary function and $R : \mathbb{R}_+ \to \mathbb{R}$ is a monotonically nondecreasing function. There exists some vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ such that $\mathbf{w}^* = \sum_{i=1}^n \alpha_i \psi(\mathbf{x}_i)$.

Note that the learning problem (1) includes SVM as special case. In particular, consider SVM for homogenous halfspaces. Soft-SVM can be derived from (1) by setting $R(a) = \lambda a^2$ and $f(a^n) = \frac{1}{n}\sum_{i=1}^n \max\{0, 1 - y_i a_i\}$; Hard-SVM can be derived from (1) by letting $R(a) = a^2$ and letting $f(a^n)$ be zero if there $y_i a_i \geq 1$ for all i and be infinity otherwise.

2.1 Learning using Kernels

An immediate implication of the representer theorem is that we can now optimize (1) with respect to $\boldsymbol{\alpha} \in \mathbb{R}^n$ instead of $\mathbf{w} \in \mathbb{R}^{\dim(\mathcal{F})}$. In order to do this, all we need to know is the Gram matrix G, which is an $n \times n$ matrix with

$$G_{ij} = K(\mathbf{x}_i, \mathbf{x}_j).$$

Using the Gram matrix, the learning problem (1) can be rewritten as

$$\min_{\boldsymbol{\alpha}} f(G\boldsymbol{\alpha}) + R(\sqrt{\boldsymbol{\alpha}^T G \boldsymbol{\alpha}}).$$

This is because

$$\begin{aligned}
\langle \mathbf{w}, \psi(\mathbf{x}_i)\rangle &= \left\langle \sum_{j=1}^n \alpha_j \psi(\mathbf{x}_j), \psi(\mathbf{x}_i) \right\rangle \\
&= \sum_{j=1}^n \alpha_j \langle \psi(\mathbf{x}_j), \psi(\mathbf{x}_i)\rangle \\
&= (G\boldsymbol{\alpha})_i
\end{aligned}$$

and

$$\begin{aligned}
\|\mathbf{w}\|^2 &= \left\langle \sum_{i=1}^n \alpha_i \psi(\mathbf{x}_i), \sum_{j=1}^n \alpha_j \psi(\mathbf{x}_j) \right\rangle \\
&= \sum_{i,j} \alpha_i \alpha_j \langle \psi(\mathbf{x}_i), \psi(\mathbf{x}_j)\rangle \\
&= \boldsymbol{\alpha}^T G \boldsymbol{\alpha}.
\end{aligned}$$

Once we have learned $\boldsymbol{\alpha}$, we can make the prediction on a new instance $\mathbf{x}$ based on the linear function over the feature space, i.e. the inner product between $\mathbf{w}$ and $\psi(\mathbf{x})$. This inner product can be again calculated using the kernel function:

$$\begin{aligned}
\langle \mathbf{w}, \psi(\mathbf{x})\rangle &= \left\langle \sum_{j=1}^n \alpha_j \psi(\mathbf{x}_j), \psi(\mathbf{x}) \right\rangle \\
&= \sum_{j=1}^n \alpha_j \langle \psi(\mathbf{x}_j), \psi(\mathbf{x})\rangle \\
&= \sum_{j=1}^n \alpha_j K(\mathbf{x}_j, \mathbf{x}).
\end{aligned}$$

Therefore, we conclude that one can do both training and prediction solely using kernels.

2.2 Examples of Kernels

As already mentioned, the advantage of working with kernels rather than directly optimizing $\mathbf{w}$ in the feature space is that in some situations the dimension of the feature space is extremely large while implementing the kernel function is very simple. We illustrate this using two examples: polynomial kernels and Gaussian kernels.

Polynomial Kernel. The k-degree polynomial kernel is defined to be

$$K(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^k.$$

To see that this is indeed a kernel function, i.e. there exists mapping ψ for which $K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$, denote $x_0 = x_0' = 1$ and expand $K(\mathbf{x}, \mathbf{x}')$ as

$$\begin{aligned} K(\mathbf{x}, \mathbf{x}') &= (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^k \\ &= \left(\sum_{j=0}^{m} x_j x_j' \right) \cdot \left(\sum_{j=0}^{m} x_j x_j' \right) \cdot \cdots \cdot \left(\sum_{j=0}^{m} x_j x_j' \right) \\ &= \sum_{J \in [0:m]^k} \prod_{i=1}^{k} x_{J_i} x'_{J_i} \\ &= \sum_{J \in [0:m]^k} \prod_{i=1}^{k} x_{J_i} \cdot \prod_{i=1}^{k} x'_{J_i}. \end{aligned}$$

This is precisely the inner product $\langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$ in the feature space if we define ψ to be

$$\psi(\mathbf{x}) = \left\{ \prod_{i=1}^{k} x_{J_i} \right\}_{J \in [0:m]^k}.$$

Note that here the complexity of implementing K is $O(m)$ while the dimension of the feature space is $(m+1)^k$.

Gaussian Kernel. Let the original space $\mathcal{X} = \mathbb{R}$ and consider the mapping ψ given by

$$\psi(x) = \left\{ \frac{1}{\sqrt{m!}} e^{-\frac{x^2}{2}} x^m \right\}_{m=0}^{\infty}.$$

Then we have

$$\begin{aligned} \langle \psi(x), \psi(x') \rangle &= \sum_{m=0}^{\infty} \left(\frac{1}{\sqrt{m!}} e^{-\frac{x^2}{2}} x^m \right) \left(\frac{1}{\sqrt{m!}} e^{-\frac{(x')^2}{2}} (x')^m \right) \\ &= e^{-\frac{x^2 + (x')^2}{2}} \sum_{m=0}^{\infty} \left(\frac{(xx')^m}{m!} \right) \\ &= e^{-\frac{(x - x')^2}{2}}. \end{aligned}$$

Define the Gaussian kernel $K(x, x') = e^{-\frac{(x-x')^2}{2}}$. Obviously, evaluating the Gaussian kernel is very simple while in sharp contrast the feature space is of infinite dimension. Note that since $\psi(x)$ includes all the monomial terms, using the Gaussian kernel we can learn polynomial predictor of any degree over the original space.