Kernel Methods

Kernel methods deal with non-linearly-separable data. It is observed that these data often become (more) linearly separable when mapped to higher dimensional feature spaces. Therefore, kernel methods map data to high dimensional spaces and build linear models there.

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

Example 1: Kernel Ridge Regression

Let $\phi$ be a function mapping $x$ from its raw feature space to a higher dimensional space. The objective function of a ridge regression model in that space is

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

The spirit of kernel methods is learning $\beta$ and using it to make prediction without knowing $\phi$, as $\phi$ can be hard to design and inefficient to compute. Kernel ridge regression realizes this spirit by the following arguments. Applying the critical point method, we have

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

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

$$\beta = \sum_{i=1}^{n} -\frac{1}{\lambda}(\phi(x_i)^T \beta - y_i) \cdot \phi(x_i) = \sum_{i=1}^{n} \alpha_i \phi(x_i),$$  \hspace{1cm} (3)

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

$$J(\beta) = \sum_{i=1}^{n} \left( \phi(x_i)^T \left( \sum_{i' = 1}^{n} \alpha_i' \phi(x_{i'}) \right) - y_i \right)^2 + \lambda \left( \sum_{i=1}^{n} \alpha_i \phi(x_i) \right)^T \left( \sum_{i' = 1}^{n} \alpha_{i'} \phi(x_{i'}) \right)$$

$$= \sum_{i=1}^{n} \left( \sum_{i' = 1}^{n} \alpha_i' \phi(x_{i'})^T \phi(x_i) - y_i \right)^2 + \lambda \sum_{i=1}^{n} \sum_{i' = 1}^{n} \alpha_i \alpha_{i'} \phi(x_i)^T \phi(x_{i'})$$

$$= \sum_{i=1}^{n} \left( \sum_{i' = 1}^{n} \alpha_i' \kappa(x_{i'}, x_i) - y_i \right)^2 + \lambda \sum_{i=1}^{n} \sum_{i' = 1}^{n} \alpha_i \alpha_{i'} \kappa(x_i, x_{i'}),$$

where

$$\kappa(x_{i'}, x_i) = \phi(x_{i'})^T \phi(x_i)$$  \hspace{1cm} (5)

is the inner product function – often called the kernel function.

\(^1\)We have seen the same insight in LSVM.
In (14), $J(\beta)$ is expressed only by inner products of instances but not any individual instance. This is known as the kernel trick. This implies one can optimize $J(\beta)$ without knowing $\phi$ but only $\kappa$. There are many well-developed kernel functions \(^2\). A common one is Gaussian kernel
\begin{equation}
\kappa(x_i, x_{i'}) = \exp \left(-\frac{||x_i - x_{i'}||^2}{2\sigma^2}\right),
\end{equation}
where $\sigma$ is a hyper-parameter. This kernel corresponds to a $\phi$ that maps data to an infinitely high dimensional feature space
\begin{equation}
\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!}}, \ldots \right]^T.
\end{equation}
Another common is polynomial kernel
\begin{equation}
\kappa(x_i, x_{i'}) = (x_i^T x_{i'} + a)^d,
\end{equation}
where $a, d$ are hyper-parameters.

[Exercise] Let $x = [x_1, x_2]^T$ and $\phi(x) = [x_1^2, \sqrt{2}x_1x_2, x_2^2]^T$. Verify that $\phi$ 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] Verify if $\kappa_a(x, x')$ and $\kappa_b(x, x')$ are two kernels, then $\kappa(x, x') := \kappa_a(x, x') + \kappa_b(x, x')$ is also a kernel.

Formula (13) shows the problem of learning $\beta$ is converted to a problem of learning $\alpha$. Write
\begin{equation}
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'})
\end{equation}
\begin{equation}
= (K\alpha - Y)^T (K\alpha - Y) + \lambda \alpha^T K\alpha,
\end{equation}
where $\alpha = [\alpha_1, \ldots, \alpha_n]^T$, $Y = [y_1, \ldots, y_n]^T$, and $K \in \mathcal{R}^{n \times n}$ is the Gram matrix where
\begin{equation}
K_{ii'} = \kappa(x_i, x_{i'}).
\end{equation}
Clearly $J(\alpha)$ is a quadratic function of $\alpha$. Applying the critical point method, we have
\begin{equation}
\alpha = (K + \lambda I)^{-1} Y.
\end{equation}

[Exercise] Verify (9) and (11).

Once $\alpha$ is obtained, we can make prediction on any instance $\phi(x_t)$ by
\begin{equation}
\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).
\end{equation}

Note all training instances are stored to make predictions. This will increase the computational and memory costs, and is a potential limitation of the kernel methods. During training, computing the inverse of $n$-by-$n$ Gram matrix is also as expensive as $O(n^3)$.

---

\(^2\)See e.g., [PRML, Chapter 6.2].

\(^3\)See e.g., [PRML, Ch 6.2].
In kernel ridge regression, we see the optimal $\beta$ 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 $\kappa$ 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

$$
\min_{f \in \mathcal{F}} L(f(x_1), \ldots, f(x_n)) + \Omega(||f||), \quad (13)
$$

where $L$ depends on $x_i$ only through $f$ and $\Omega$ is a non-decreasing function. If (13) has a minimizer, then one minimizer $f_*$ has the form

$$
f_* = \sum_{i=1}^{n} \alpha_i \kappa(\cdot, x_i), \quad (14)
$$

where $\alpha_i \in \mathbb{R}$. And if $\Omega$ is strictly increasing, then every minimizer has the form (14).

We can kernelize many linear machine learning methods, such as linear SVM and PCA. The kernelized LSVM is known as Support Vector Machine (SVM).

**Discussion** Can we kernelize least square?

Because the (implicit) mapping function $\phi$ is often non-linear, kernel methods can model non-linear relations between features and label.

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

Note the Wolfe dual problem of hard-margin LSVM is

$$
\min_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \\
s.t. \sum_{i=1}^{n} \alpha_i y_i = 0, \text{ and } \alpha_i \geq 0, \forall i, \quad (15)
$$

which 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)

$$
\min_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \kappa(x_i, x_j) \\
s.t. \sum_{i=1}^{n} \alpha_i y_i = 0, \text{ and } \alpha_i \geq 0, \forall i. \quad (16)
$$

Similar discussion applies to soft-margin LSVM.