CSCI 5525 Machine Learning Fall 2019

Lecture 8: Kernels

Feb 17th 2020

Lecturer: Steven Wu *Scribe: Steven Wu*

Feature Expansion

We have seen some examples of feature expansions that enrich the feature space and provde more flexible predictor. To motivate the idea of kernel, we will consider several examples of feature expansion mappings ϕ such that there is essentially no additional computational cost in computing products $\phi(x)^\intercal\phi(x')$. Later we will show that for many methods that use linear models, the algorithm does not need to explicitly compute $\phi(x)$, as long as it can compute these products.

Quadratic expansion For $x \in \mathbb{R}^d$, consider the following quadratic expansion:

$$\phi(x) = (1, \sqrt{2}x_1, \dots, \sqrt{2}x_d, x_1^2, \dots, x_d^2, \sqrt{2}x_1x_2, \dots, \sqrt{2}x_{d-1}x_d)$$

Under this expansion, the product

$$\phi(x)^\intercal\phi(x') = (1 + x^\intercal x')^2$$

which can be computed in $O(d)$ time, as opposed to $O(d^2)$.

Products of all subsets. Now let us blow up the dimension even more. Consider the following feature expansion mapping

$$\phi(x) = \left(\prod_{i\in S} x_i\right)_{S\subseteq[d]}$$

Then we still compute the product in time $O(d)$ (instead of 2^d):

$$\phi(x)^\intercal\phi(x) = \prod_{i=1}^{d}(1 + x_i x_i')$$

Gaussian kernel. Now, let's be more ambitious and push the dimension to infinity. For any parameter $\sigma > 0$, consider a feature expansion ϕ such that

$$\phi(x)^\intercal\phi(x') = \exp\left(-\frac{\|x - x'\|_2^2}{2\sigma^2}\right)$$

This product can be computed in $O(d)$ time. So what is ϕ? Let us try the simple case of $x \in \mathbb{R}$. Then

$$\begin{aligned}\phi(x)\phi(y) &= \exp(-(x-y)^2/(2\sigma^2)) \\ &= \exp(-x^2/(2\sigma^2))\exp(-y^2/(2\sigma^2))\exp(xy/\sigma^2) \\ &= \exp(-x^2/(2\sigma^2))\exp(-y^2/(2\sigma^2))\sum_{j=0}^{\infty}\frac{1}{j!}\left(xy/\sigma^2\right)^j\end{aligned}$$

where the last step comes from Taylor expansion. This gives

$$\phi(x) = \exp(-x^2/(2\sigma^2))\left(1, \frac{x}{\sigma}, \frac{1}{2!}\left(\frac{x}{\sigma}\right)^2, \frac{1}{3!}\left(\frac{x}{\sigma}\right)^3 \dots\right)$$

which is in $\mathbb{R}^\infty$. The similarity measure $K(x, x') = \phi(x)^\intercal\phi(x')$ defined above is called RBF kernel or Gaussian kernel. Now we will formally define kernel.

Kernel

Definition 0.1 (Kernel)**.** A *kernel function* $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a symmetric function such that for any $x_1, x_2, \dots, x_n \in \mathcal{X}$, the $n \times n$ *Gram matrix* G with each (i, j)-th entry $G_{ij} = K(x_i, x_j)$, is positive semidefinite (p.s.d.).

Recall that a matrix $G \in \mathbb{R}^{n\times n}$ is positive semi-definite if and only if $\forall q \in \mathbb{R}^n$, $q^\intercal G q \geq 0$.

How to show K is a kernel? A simple way to show a function K is a kernel is to find a feature expansion mapping ϕ such that[1]

$$\phi(x)^\intercal\phi(x') = K(x, x')$$

Now consider the Gram matrix defined above, where each entry $G_{ij} = \phi(x_i)^\intercal\phi(x_j)$. This means the Gram $G = \Phi^\intercal\Phi$, where $\Phi = [\phi(x_1), \dots, \phi(x_n)]$ (that is each $\phi(x_i)$ is a column vector). It follows that K is p.s.d., because

$$q^\intercal G q = q^\intercal\Phi^\intercal\Phi q = \|\Phi q\|_2^2 \geq 0$$

Inversely, if K is a valid kernel, then there exists a feature mapping ϕ such that $\phi(x)^\intercal\phi(x') = K(x, x')$. Relatedly, any p.s.d. matrix G can be factorized as $G = \Phi^\intercal\Phi$ for some realization of Φ.

[1] The function ϕ maps $\mathcal{X}$ to $\mathbb{H}$, where $\mathbb{H}$ is a Hilbert space called the *Reproducing Kernel Hilbert Space (RKHS)* corresponding to K.You could read more about it here.

Examples of kernel functions:

- Linear: $K(x, x') = x^{\intercal} x'$.

  (Althoug this does not modify the features, it can be faster to pre-compute the Gram matrix when the dimensionality d of the data is high.)

- Polynomial:
$$K(x, x') = (1 + x^{\intercal} x')^d$$

- Radial Basis Function (RBF) (aka Gaussian Kernel):
$$K(x, x') = \exp\left(\frac{-\|x - x'\|^2}{2\sigma^2}\right)$$

- Exponential Kernel:
$$K(x, x') = \exp\left(\frac{-\|x - x'\|_2}{2\sigma^2}\right)$$

Building new kernels from existing kernels. Suppose K_1, K_2 are valid kernels, $c \geq 0$, g is a polynomial function with positive coefficients (that is $\sum_{j=1}^{m} \alpha_j x^j$ for some $m \in \mathbb{N}$, $\alpha_1, \ldots, \alpha_m \in \mathbb{R}^+$), f is any function and matrix $A \succeq 0$ is positive semi-definite. Then following functions are also valid kernels:

- $K(x, x') = cK_1(x, x')$
- $K(x, x') = K_1(x, x') + K_2(x, x')$
- $K(x, x') = g(K(x, x'))$
- $K(x, x') = K_1(x, x')K_2(x, x')$
- $K(x, x') = f(x)K_1(x, x')f(x')$
- $K(x, x') = \exp\left(K_1(x, x')\right)$
- $K(x, x') = x^{\top} A x'$

In the homework, you will show some of them formally. Now let's apply kernel to different linear methods.

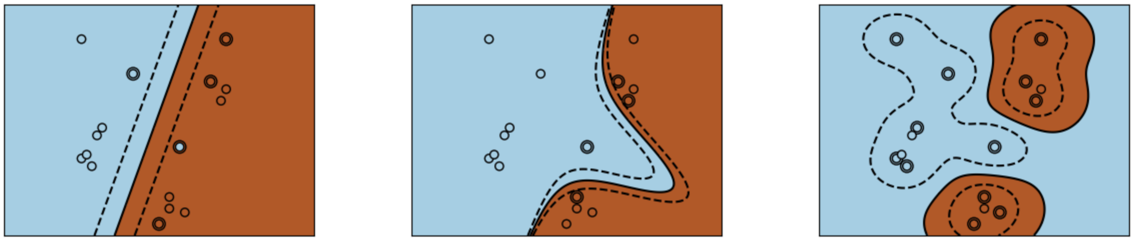

Figure 1: From left to right: decision boundaries of kernel SVM with linear, polynomial (of degree 3), and RBF kernels. Image source.

Kernel SVM

Note that all of our derivation for SVM holds if we replace each feature vector x_i by some feature expansion $\phi(x_i)$. The dual SVM problem then becomes:

$$\max_{\alpha,\lambda} \sum_i \lambda_i - \frac{1}{2} \sum_{i,j\in[n]} \lambda_i\lambda_j y_i y_j \phi(x_i)^\intercal \phi(x_j)$$

$$\text{such that for all } i: \qquad 0 \le \lambda_i \le C$$

We can replace the product $\phi(x_i)^\intercal \phi(x_j)$ with $K(x_i, x_j)$ and rewrite the objective:

$$\max_{\alpha,\lambda} \sum_i \lambda_i - \frac{1}{2} \sum_{i,j\in[n]} \lambda_i\lambda_j y_i y_j K(x_i, x_j)$$

How do we represent the underlying linear predictor now? If we are forced to write down $\hat{\mathbf{w}}$, this will be an infinite-dimensional object. Well, it turns out that we only need to remember the support vectors (with $\lambda_i^* > 0$), since the prediction for any example x is:

$$\phi(x)^\intercal \hat{\mathbf{w}} = \sum_{i=1}^n y_i \lambda_i^* \phi(x)^\intercal \phi(x_i) = \sum_{i=1}^n y_i \lambda_i^* K(x, x_i)$$

This requires iterating the support vector examples (x_i, y_i) along with their associated dual variables λ_i^* to actually get a prediction. Figure 1 shows kernel SVM with different choices of kernel functions.

Kernelized ridge regression

We can also apply the idea of kernels to ridge regression. This is sometimes called kernelized ridge regression. Recall the notations of design matrix and response vector in linear regression:

$$A = \begin{bmatrix} \leftarrow x_1^\intercal \rightarrow \\ \vdots \\ \leftarrow x_n^\intercal \rightarrow \end{bmatrix} \qquad \mathbf{b} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

The ridge regression solution is given by

$$\hat{\mathbf{w}} = (A^\intercal A + \lambda I)^{-1} A^\intercal \mathbf{b}. \tag{1}$$

Here is a neat trick in linear algebra. Let P be an $N \times M$ matrix while Q be an $M \times N$ matrix:

$$(PQ + I_N)^{-1} P = P(QP + I_M)^{-1}$$

Now if we set $P = (1/\lambda)A^\intercal$ and $Q = A$, then

$$(A^\intercal A + \lambda I_d)^{-1} A^\intercal = A^\intercal (AA^\intercal + \lambda I_n)^{-1} = A^\intercal (G + \lambda I_n)^{-1}$$

where G is the $n \times n$ Gram matrix with $G_{ij} = x_i^\intercal x_j$. This gives an alternative form of the ridge regression solution in terms of the Gram matrix

$$\hat{w} = A^\intercal \underbrace{(G + \lambda I_n)^{-1} \mathbf{b}}_{\mathbf{v}} = A^\intercal \mathbf{v} = \sum_{i=1}^{n} \mathbf{v}_i x_i$$

Thus, for any new feature vector x, the prediction will be

$$x^\intercal \hat{\mathbf{w}} = \sum_{i=1}^{n} \mathbf{v}_i x^\intercal x_i$$

Now if we replace each x_i with $\phi(x_i)$, then G is the Gram matrix, each entry $G_{ij} = K(x_i, x_j)$, which will allow us to compute $\mathbf{v}$. Then during prediction time

$$\phi(x)^\intercal \hat{\mathbf{w}} = \sum_{i=1}^{n} \mathbf{v}_i \phi(x)^\intercal \phi(x_i)$$

While we can potentially obtain richer representation of the feature space, the downside is that we need to store a lot of information for making predictions.