Discussion 8
CS 189
02/21/2017

Alex Francis
Email: afrancis@berkeley.edu
OH: Wednesday 10:00am - 12:00pm (283E Soda)

1 Feedback

Please give me feedback so that I can get better. Google form: `https://tinyurl.com/alex189-feedback`.

2 Lecture on Kernels

2.1 Motivation

Kernels are complex and elegant mathematical objects that allow us to use duality to improve the computational feasibility of various machine learning problems.

Recall a simple algorithm, like linear regression on polynomial features. Say we have a data point of the form,

$$x = \begin{bmatrix} x_1 & x_2 & 1 \end{bmatrix}$$

In linear regression, we fit a line/plane/hyperplane to the data by the function,

$$w^\top x = w_1 x_1 + w_2 x_2 + b = 0$$

Say, instead, we want to fit some quadratic surface to the data. This requires some additional features, since the standard form of the quadratic is,

$$ax_1^2 + bx_2^2 + cx_1x_2 + dx_1 + ex_2 + f = 0$$

We realized in an earlier lecture that we could simply use the linear model to fit a quadratic surface by using an "expanded" feature vector,

$$\Phi(x) = \begin{bmatrix} x_1^2 & x_2^2 & x_1x_2 & x_1 & x_2 & 1 \end{bmatrix}$$

Then,

$$w'^\top \Phi(x) = w_1x_1^2 + w_2x_2^2 + w_3x_1x_2 + w_4x_1 + w_5x_2 + b = 0$$

Note that w' is now a 6×1 vector instead of a 3×1 vector. In general, a polynomial feature expansion results in a vector with $O(d^p)$ features. This is a problem because "polynomial time" algorithms in the dimension of the feature space quickly blow up.

Secondly, recall an important result from calculus, the *Taylor Series*: *any function satisfying certain conditions may be represented by a Taylor series,*

$$f(x) = \sum_{i=0}^{\infty} f^{(i)}(a)(x-a)^i$$

A Taylor series is useful for approximating any real function that is differentiable an arbitrary number of times at the point a. Then, any decision boundary we could possible imagine could be represented by a Taylor approximation of the real decision function. However, the feature vector the the problem becomes infinite dimensional. Consider, for $d = 1$,

$$\Phi(x) = \begin{bmatrix} 1 & x_1 & \frac{x_1^2}{2!} & \dots \end{bmatrix}$$

Just stating this problem is intractable. Writing out the matrix is impossible. How do we overcome these steep obstacles in computational complexity?

2.2 The Dual Problem

One non-trivial fact is that in many learning algorithms, the weights can be written as a linear combination of sample points. That is, we can write

$$w = X^\top a = \sum_{i=1}^{n} a_i X_i$$

For some n *dual weights.* Then, we can rewrite the objective function for the learning algorithms by plugging in the formulation of w as a linear combination of the data points, in essence recasting the problem in the *sample space* instead of the *feature space.* This is just the *dual* of the original problem - we've added no features yet. But while we're at it, let's define the kernel function as,

$$k(x, z) = x^\top z$$

For some vectors x and z. Then, for linear least squares we,

```
1. Use the normal equations to derive the optimal dual weight vector a.
We will see that,
```

$$a = (X^\top X)^{-1} y = K^{-1} y$$

For the kernel matrix $K_{ij} = k(X_i, X_j)$. Then,

```
2. For the classification of some z in the test set,
```

$$h(z) = w^\top z = (X^\top a)^\top z = \sum_{i=1}^{n} a_i k(X_i, z)$$

Now, consider expanding the feature vector to have polynomial features.

Claim. The only modification to the algorithm above is letting the kernel function be,

$$k(x, z) = \Phi(x)^\top \Phi(z)$$

Do you see why? But we're still stuck at this point. The computation time is a function of the number of features, which is problematic in our exponential and infinite-dimensional feature expansions in the motivation. We need the kernel function to have computational complexity that grows with smaller quantities!

2.3 The Kernel Trick and Kernel Functions

The Polynomial Kernel. The polynomial kernel is defined as, $k(x, z) = (x^\top z + 1)^p$.

Claim. The identity holds,

$$(x^\top z + 1)^p = \Phi(x)^\top \Phi(z)$$

Proof. See problem 2. □

The first computation is $O(d)$, while the second is $O(d^p)$.

The Gaussian (Radial Basis Function) Kernel. Frequently used in SVM contexts. The kernel function is,

$$k(x, z) = \exp\left(-\frac{|x - z|^2}{2\sigma^2}\right)$$

Claim. For $d = 1$, let $\Phi(x) = \exp\left(-\frac{x^2}{2\sigma^2}\right)\begin{bmatrix} 1 & \frac{x}{\sigma\sqrt{1!}} & \frac{x^2}{\sigma^2\sqrt{2!}} & \ldots \end{bmatrix}^\top$. The identity holds,

$$\exp\left(-\frac{|x - z|^2}{2\sigma^2}\right) = \Phi(x)^\top \Phi(z)$$

Proof. Recall the common Taylor approximation,

$$e^x \approx \sum_{i=1}^{\infty} \frac{x^i}{i!}$$

Starting from the right-hand-side,

$$\begin{aligned}
\Phi(x)^\top \Phi(z) &= \exp\left(-\frac{x^2}{2\sigma^2} - \frac{z^2}{2\sigma^2}\right) \sum_{i=1}^{\infty} \left(\frac{x^2 z^2}{i!\sigma^2}\right)^i \\
&= \exp\left(-\frac{x^2}{2\sigma^2} - \frac{z^2}{2\sigma^2}\right) \exp\left(\frac{x^2 z^2}{\sigma^2}\right) \\
&= \exp\left(-\frac{|x - z|^2}{2\sigma^2}\right)
\end{aligned}$$

□

Allows us to get smooth decision boundaries that mimic nearest neighbors approaches. Notice that the hypothesis function is a linear combination of Gaussian's centered at the sample point at that index of the linear combination. The RBF kernel is a measure of "similarity" between two vectors. The closer the test point is to the sample point, the more heavily weighted that Gaussian.

The computation of the kernel is $O(d)$. The un-kernelized problem is too large to even state.