CS 189 Introduction to Machine Learning
Spring 2019 Jonathan Shewchuk

DIS10

1 Kernelizing Nearest Neighbors

In this question, we will be looking at how we can kernelize k-nearest neighbors (k-NN). k-NN is a simple classifier that relies on nearby sample points to decide what a new point's class should be.

Given a query point q, there are 2 steps to decide what class to predict.

1. Find the k sample points nearest q.
2. Return the class with the most votes from the k sample points.

For the following parts, assuming that our sample points $x \in \mathbb{R}^d$, and we have n sample points.

(a) What is the runtime to classify a newly given query point q, using euclidean distance?

Solution: To classify a point q, we first need to calculate the distances to every other point. Euclidean distance is $\sqrt{(x_1 - q_1)^2 + (x_2 - q_2)^2 + \ldots + (x_d - q_d)^2}$, and so for one point, takes $O(d)$ time to compute. For n points, we get $O(nd)$ time.

As we compute the distances, we can use a max-heap to keep track of the k shortest distances seen so far. In the worst case, we have n insertions, taking $O(log(k))$ time each.

This gives us a total run time of $0(nd + nlogk)$.

(b) What if instead of looking at the distance between the points in $\mathbb{R}^d$, we wanted to consider the distance in p polynomial space? What dimension would this space be? What would the runtime be to classify a newly given query point q, in terms of n, d, and k, and p?

Solution: To classify a point q, we first need to calculate the distances to every other point. To do this, we would first need to raise q into p polynomial feature space.

In quadratic feature space, our sample points now have d^p features, and so for one point, takes $O(d^2)$ time to compute. For n points, we get $O(nd^p)$ time.

As we compute the distances, we can use a max-heap to keep track of the k shortest distances seen so far. In the worst case, we have n insertions, taking $O(log(k))$ time each.

This gives us a total run time of $0(nd^p + nlogk)$.

(c) Instead, we can use the polynomial kernel to compute the distance between 2 points in p polynomial space without having to move all of the points into the higher dimensional space. Using the polynomial kernel, $k(x, y) = (x^T y + \alpha)^p$ instead of Euclidean distance, what is the runtime for k-NN to classify a new point q?

Solution: Now, instead of calculating Euclidean distance between between q and each sample point, we calculate $k(x, q)$, which takes $O(d)$ time to compute! For n points, we get $O(nd)$ time. Following the same logic as before, we get a total run time of $0(nd + nlogk)$.

Note: You might point out that there is an exponentiation in our calculation (to the p power), which should factor into our runtime. However, for practical purposes, exponentiation with two floating-point numbers takes $O(1)$ time. It can be done with numerical algorithms; repeated squaring and whatnot is not necessary (and not used for large or non-integer p).

It turns out that we can actually use any valid kernel function to compute some notion of "distance" or dissimilarity. The Euclidean distance is equal to the square root of the dot product. Squaring our distances will still yield the same k nearest neighbors, since all distances will be positive. You'll also notice that Euclidean distance itself is a valid kernel function.

Therefore, the "polynomial" kernel gives us a valid distance metric in polynomial space. The greater $k_p(x, q)$ is, the farther x and q are from each other in p polynomial space.

There are many other kernels we can choose to use to compute distances, including the Gaussian kernel and the sigmoid kernel.

2 Gaussian Kernels

In this question, we will look at training a binary classifier with a Gaussian kernel. Specifically given a labelled dataset $S = \{(x_i, y_i)\}_{i=1}^n \subseteq \mathbb{R}^d \times \{\pm 1\}$ and a kernel function $k(x_1, x_2)$, we consider classifiers of the form:

$$\widehat{f}(x) = \text{sign}\left(\sum_{i=1}^{n} a_i k(x_i, x)\right),$$

where we define sign(u) to be 1 if $u \geq 0$ or -1 if $u < 0$. In order to choose the weights $a_i, i = 1, ..., n$, we will consider the least-squares problem:

$$a \in \arg\min_{a \in \mathbb{R}^n} \|Ka - y\|_2^2, \tag{1}$$

where $K = (k(x_i, x_j))_{i=1,j=1}^n$ is the kernel matrix and $y = (y_1, ..., y_n)$ is the vector of labels. We will work with the Gaussian kernel. Recall that the Gaussian kernel with bandwidth $\sigma > 0$ is defined as:

$$k(x_i, x_j) := \exp\left(-\frac{\|x_i - x_j\|_2^2}{2\sigma^2}\right).$$

(a) When the bandwidth parameter $\sigma \to 0$, observe that the off-diagonal entries of the kernel matrix K tend also to zero. Consider a two sample dataset S (i.e. $n = 2$) with $(x_1, y_1) = (1, 1)$ and $(x_2, y_2) = (-1, -1)$. Assuming that as $\sigma \to 0$ the off-diagonal entries of K are equal to zero (and the diagonal entries unmodified), what is the optimal solution of a for the optimization problem (1) and what is the resulting classifier $\widehat{f}(x)$?

Solution: With our assumptions, the kernel matrix K is identity, so $a_\star = (+1, -1)$. By symmetry, the resulting $\widehat{f}(x) = \text{sign}(x)$.

(b) Now we consider the regime when the bandwidth parameter $\sigma \to +\infty$. Observe in this regime, the off-diagonal entries of the kernel matrix K tend to one. Given a dataset S, suppose we solve the optimization problem (1) with all the off-diagonal entries of K equal to one (and the diagonal entries unmodified). Prove that if the number of $+1$ labels in S equals the number of -1 labels in S, then $a = \mathbf{0}$ is an optimal solution of (1). What is the resulting classifier $\widehat{f}(x)$?

Solution: With our assumptions, the kernel matrix $K = \mathbf{1}\mathbf{1}^\top$, where $\mathbf{1}$ is the vector of all 1s.

The normal equations for (1) are given by:

$$(\mathbf{1}\mathbf{1}^\top)^\top(\mathbf{1}\mathbf{1}^\top)a = (\mathbf{1}\mathbf{1}^\top)\mathbf{y} \iff n\mathbf{1}\mathbf{1}^\top a = \mathbf{1}(\mathbf{1}^\top\mathbf{y})\,.$$

Since we assumed the number of $+1$ labels equals the number of -1 labels, then $\mathbf{1}^\top\mathbf{y} = 0$. Hence $a = \mathbf{0}$ satisfies the normal equations. The resulting classifier is a constant function (depending on what the value of sign(0) is).

(c) Now we consider the regime when the bandwidth parameter is large but finite. Consider again the two sample dataset S with $(x_1, y_1) = (1, 1)$ and $(x_2, y_2) = (-1, -1)$. When $\sigma \gg 1$, we can approximate $k(x_1, x_2) \approx 1 + \frac{x_1 x_2}{2\sigma^2}$. Show that the solution of the optimization problem (1) with the kernel $k_a(x_1, x_2) = 1 + \frac{x_1 x_2}{2\sigma^2}$ is given by $a = (\sigma^2, -\sigma^2)$. What is the resulting classifier $\widehat{f}(x)$?

Hint: The inverse of a 2×2 *matrix is given by the formula* $\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$.

Solution: The kernel matrix $K = \begin{bmatrix} 1 + \frac{1}{2\sigma^2} & 1 - \frac{1}{2\sigma^2} \\ 1 - \frac{1}{2\sigma^2} & 1 + \frac{1}{2\sigma^2} \end{bmatrix}$. The inverse matrix is:

$$K^{-1} = \frac{1}{4}\begin{bmatrix} 1 + 2\sigma^2 & 1 - 2\sigma^2 \\ 1 - 2\sigma^2 & 1 + 2\sigma^2 \end{bmatrix}.$$

Hence the optimal $a = (\sigma^2, -\sigma^2)$. Plugging this a into $\widehat{f}(x)$ we see that $\widehat{f}(x) = \text{sign}(x)$ (once again by symmetry).

3 Kernel Validity

For a function $k(x_i, x_j)$ to be a valid kernel, it suffices to show either of the following conditions is true:

1. k has an inner product representation: $\exists\, \Phi : \mathbb{R}^d \to \mathcal{H}$, where $\mathcal{H}$ is some (possibly infinite-dimensional) inner product space such that $\forall x_i, x_j \in \mathbb{R}^d,\ k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$.
2. For every sample $x_1, x_2, \ldots, x_n \in R^d$, the kernel matrix

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & k(x_i, x_j) & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix}$$

is positive semidefinite. For the following parts you can use either condition (1) or (2) in your proofs.

(a) Show that the first condition implies the second one, i.e. if $\forall x_i, x_j \in \mathbb{R}^d,\ k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$ then the kernel matrix K is PSD.

Solution: $\forall a \in \mathbb{R}^n, a^T K a = \sum_{i,j} a_i a_j k(x_i, x_j) = \sum_{i,j} a_i a_j \langle \Phi(x_i), \Phi(x_j) \rangle = \| \sum_i a_i \Phi(x_i) \|_2^2 \geq 0$

(b) Given a positive semidefinite matrix A, show that $k(x_i, x_j) = x_i^\top A x_j$ is a valid kernel.

Solution: We can show k admits a valid inner product representation:

$$k(x_i, x_j) = x_i^\top A x_j = x_i^\top P D^{1/2} D^{1/2} P^\top x_j = \langle D^{1/2} P^\top x_i, D^{1/2} P^\top x_j \rangle = \langle \Phi(x_i), \Phi(x_j) \rangle$$

where $\Phi(x) = D^{1/2} P^\top x$

(c) Show why $k(x_i, x_j) = x_i^\top (\text{rev}(x_j))$ (where rev(x) reverses the order of the components in x) is *not* a valid kernel.

Solution: A counterexample: We have that $k((-1, 1), (-1, 1)) = -2$, but this is invalid since if k is a valid kernel then $\forall x,\ k(x, x) = \langle \Phi(x), \Phi(x) \rangle \geq 0$.

(d) When solving Kernel ridge regression, one can show that the key intermediate step is solving the following optimization problem:

$$\text{argmin}_{\alpha \in \mathbb{R}^n} \left[\frac{1}{2} \alpha^T (K + \lambda I) \alpha - \lambda \langle \alpha, y \rangle \right]$$

where $y \in \mathbb{R}^n$, $\lambda \geq 0$, and $K \in \mathbb{R}^{n \times n}$ is the kernel matrix computed by applying a kernel function k on every sample pair: $k(x_i, x_j)$. When λ is close to 0, why is it important that K is a valid kernel?

Solution: Since K is a symmetric matrix, it has a spectral of the form: $K = Q \Lambda Q^\top$. If K is not a valid kernel function, then there must be a negative eigenvalue. Without the loss of generality, assume that it is $\Lambda_{11} < 0$. Let $\alpha = cQe_1$, where e_1 is the first cannonical basis vector, and $c \in \mathbb{R}$. Then the argmin is at most

$$\begin{aligned}
\frac{1}{2} \alpha^\top ((\mathbf{K} + \lambda I)) \alpha - \lambda \langle \alpha, y \rangle &= \frac{1}{2} (cQe_1)^\top (Q \Lambda Q^\top + \lambda I)(cQe_1) - \lambda \langle cQe_1, y \rangle \\
&= \frac{1}{2} c^2 e_1^\top (Q^\top Q \Lambda Q^\top Q + \lambda Q^\top Q) e_1 - c\lambda \langle Qe_1, y \rangle \\
&= \frac{1}{2} c^2 e_1^\top (\Lambda + \lambda I) e_1 - c\lambda \langle Qe_1, y \rangle \\
&= c^2 \cdot \frac{\Lambda_{11} + \lambda}{2} - c \langle Qe_1, y \rangle.
\end{aligned}$$

When $\lambda < -\Lambda_{11}$, then as $c \to \infty$, the term $c^2 \cdot \frac{\Lambda_{11} + \lambda}{2}$ dominates $c\langle Qe_1, y \rangle$, and tends to $-\infty$. Therefore, the arg min does not exist if K has a strictly negative eigenvalue (and $\lambda < -\Lambda_{11}$).

4 Polynomial Kernel

An alternative formulation of the polynomial kernel is

$$k(x, y) = (x^T y + \alpha)^p$$

where $x, y \in \mathbb{R}^n$, and $\alpha \geq 0$. When we take $p = 2$, this kernel is called the quadratic kernel. Find the feature mapping $\Phi(z)$ that corresponds to the quadratic kernel.

Solution: First we expand the dot product inside, and square the entire sum. We will get a sum of the squares of the components and a sum of the cross products.

$$(x^T y + \alpha)^2 = (\alpha + \sum_{i=1}^{n} x_i y_i)^2$$

$$= \alpha^2 + \sum_{i=1}^{n} x_i^2 y_i^2 + \sum_{i=2}^{n} \sum_{j=1}^{i-1} 2 x_i y_i x_j y_j + \sum_{i=1}^{n} 2 x_i y_i \alpha$$

Pulling this sum into a dot product of x components and y components, we have

$$\Phi(x) = \left[\alpha,\ x_1^2, \ldots, x_n^2,\ \sqrt{2}x_1x_2, \ldots,\ \sqrt{2}x_1x_n,\ \sqrt{2}x_2x_3, \ldots,\ \sqrt{2}x_{n-1}x_n,\ \sqrt{2\alpha}x_1,\ \ldots,\ \sqrt{2\alpha}x_n\right]$$

In this feature mapping, we have α, the squared components of the vector x, $\sqrt{2}$ multiplied by all of the cross terms, and $\sqrt{2\alpha}$ multiplied by all of the components.