DIS10

1 Kernelizing Nearest Neighbors

Part(a)

Computing the euclidean distance between q and all the other n sample points takes $O(nd)$ time. To keep track of the k nearest points, we can use a max-heap. In the worst case, we have n insertions, taking $O(\log k)$ time each.

This gives us a total run time of $O(nd + n \log k)$.

Part(b)

Similarly, the total run time is $O(nd^p + n \log k)$

Part(c)

Now it only takes $O(d)$ to compute the "distance" of two points in the p polynomial space. The total run time goes back to $O(nd + n \log k)$.

Note that exponentiation with two floating-point numbers takes $O(1)$ time, which can be done with numerical algorithms (repeated squaring is not necessary). Besides, 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.

2 Gaussian Kernels

Part(a)

With our assumptions, K is the identity matrix, so $a^* = (1, -1)$. By symmetry, the resulting $\hat{f}(x) = sign(x)$.

Part (b)

With our assumptions, the kernel matrix $K = \mathbf{1}\mathbf{1}^\intercal$, where **1** is the vector of n 1s.

The normal equation for $|Ka - y|^2$ is given by:

$$K^\intercal K a = K^\intercal y \Leftrightarrow n\mathbf{1}\mathbf{1}^\intercal a = \mathbf{1}(\mathbf{1}^\intercal y) = \mathbf{0}$$

Since we assumed the number of +1 labels equals the number of −1 labels, then $\mathbf{1}^\intercal y$ = 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).

Part (c)

The kernel matrix

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

and

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

Hence the optimal $a = (\sigma^2, -\sigma^2)$, and $f(x) = \text{sign}(x)$ (once again by symmetry).

3 Kernel Validity

Part(a)

$\forall c \in \mathbb{R}^n$, $c^\intercal K c = \sum_{i=1}^n \sum_{j=1}^n c_i c_j \Phi(x_i)^\intercal \Phi(x_j)$.

By linearity of a dot product, we have $c^\intercal K c = (\sum_{i=1}^n c_i \Phi(x_i)^\intercal) \left(\sum_{j=1}^n c_j \Phi(x_j)\right) = || \sum_{i=1}^n c_i \Phi(x_i)||^2 \geq 0$

Therefore, K is PSD.

Part(b)

Since A is SPSD, we can write $A = Q\Lambda^{1/2}\Lambda^{1/2}Q^\intercal$. Then $k(x_i, x_j) = x_i^\intercal A x_j = (\Lambda^{1/2}Q^\intercal x_i)^\intercal(\Lambda^{1/2}Q^\intercal x_j)$, which is a form of dot product.

Part(c)

Counter example: for $x_i = [1 \ \ -1]^\intercal$, $k(x_i, x_i) = -2 < 0$.

Part(d)

If K is not a valid kernel, it must have a strictly negative eigenvalue, which will make $\arg\min_\alpha J(\alpha)$ not exist as $\lambda \to 0$.

4 Polynomial Kernel

$$(x^\intercal y + \alpha)^2 = (\alpha + \sum_{i=1}^{n} x_i y_i)^2 = \alpha^2 + \sum_{i=1}^{n} x_i^2 y_i^2 + 2\alpha \sum_{i=1}^{n} x_i y_i + 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_i y_i x_j y_j = \Phi(x)^\intercal \Phi(y)$$

Therefore,

$$\Phi(x) = [\alpha \quad x_1^2 \quad \cdots \quad x_n^2 \quad \sqrt{2\alpha} x_1 \quad \cdots \quad \sqrt{2\alpha} x_n \quad \sqrt{2} x_1 x_2 \quad \cdots \quad \sqrt{2} x_1 x_n \quad \sqrt{2} x_2 x_3 \quad \cdots \quad \sqrt{2} x_{n-1} x_n]$$