Machine Learning - Exercise 5 - Theoretical Part

Anat Balzam, Niv Shani

May 26, 2019

Question 1

(a) $\boxed{f(x,y) = e^{xy} \text{ where } 2x^2 + y^2 - 72 = 0}$

Using Lagrange multipliers we get:

$$\begin{aligned} L =& e^{xy} - \lambda(2x^2 + y^2 - 72) \\ \nabla L =& \nabla \left[e^{xy} - \lambda(2x^2 + y^2 - 72) \right] \end{aligned}$$

Computing the derivatives by x, y and λ:

$$\nabla x : \quad y \cdot e^{xy} - 4\lambda x = 0 \qquad \nabla y : \quad x \cdot e^{xy} - 4\lambda y = 0 \qquad \nabla \lambda : \quad -2x^2 - y^2 + 72 = 0$$

From equations I and II we get:

$$\frac{4\lambda x}{y} = \frac{4\lambda y}{x} \quad \rightarrow \quad x^2 = \frac{y^2}{2}$$

Thus from equation III:

$$-(2 \cdot \frac{y^2}{2} + y^2 - 72) = 0 \quad \rightarrow \quad y = \pm 6 \quad \rightarrow \quad x = \pm\sqrt{18}$$

$$(\sqrt{18}, 6), \quad (-\sqrt{18}, 6), \quad (\sqrt{18}, -6), \quad (-\sqrt{18}, -6)$$

Computing $f(x, y)$ for each point, we can conclude:

$$\begin{aligned} \mathbf{f(\sqrt{18}, 6) = f(-\sqrt{18}, -6) = e^{6\sqrt{18}} \Longrightarrow \text{Maximum}: (\sqrt{18}, 6), \quad (-\sqrt{18}, -6)} \\ \mathbf{f(\sqrt{18}, -6) = f(-\sqrt{18}, 6) = e^{-6\sqrt{18}} \Longrightarrow \text{Minimum}: (\sqrt{18}, -6), \quad (-\sqrt{18}, 6)} \end{aligned}$$

(b) $\boxed{f(x,y) = x^2 + y^2 \text{ where } y - cos2x = 0}$

Using Lagrange multipliers we get:

$$L = x^2 + y^2 - \lambda(y - cos2x)$$
$$\nabla L = \nabla \left[x^2 + y^2 - \lambda(y - cos2x)\right]$$

Computing the derivatives by x, y and λ:

$$\nabla x : \quad 2x - 2\lambda sin2x = 0 \quad \rightarrow \quad \lambda = \frac{x}{sin2x}$$
$$\nabla y : \quad 2y - \lambda = 0 \quad \rightarrow \quad = \lambda = 2y$$
$$\nabla \lambda : \quad -y + cos2x = 0$$

From equations I and II we get:

$$y = \frac{x}{2sin2x} \quad \longrightarrow \quad eq.III : \quad cos2x = \frac{x}{2sin2x}$$
$$= \quad \cdots$$
$$\implies \quad sin4x = x$$

After plotting the function we find the equation solutions:

$$x = 0,\ \pm 0.619 \quad \rightarrow \quad y = 1,\ 0.327,\ 0.327$$

$$(0,1), \quad (0.619, 0.327), \quad (-0.619, 0.327)$$

Computing $f(x,y)$ for each point, we can conclude:

$$\mathbf{f(0,1) = 1 \implies Maximum: (0,1)}$$
$$\mathbf{f(\pm 0.619, 0.327) = 0.490 \implies Minimum: (0.619, 0.327), \quad (-0.619, 0.327)}$$

Question 2

(a) Let x, y be two vectors in dimensions m_1, m_2 respectively, and assume ϕ_1,ϕ_2 are mappings to dimensions n_1, n_2 respectively.

Hence we got:

$$K_1(x,y) = \phi_1(x) \cdot \phi_1(y) = \sum_{i=1}^{n_1} \phi_1(x)_i \cdot \phi_1(y)_i$$

$$K_2(x,y) = \phi_2(x) \cdot \phi_2(y) = \sum_{i=1}^{n_2} \phi_2(x)_i \cdot \phi_2(y)_i$$

Observe $K(x,y)$

$$\begin{aligned}
&= 7K_1(x,y) + 3K_2(x,y) \\
&= 7\left[\sum_{i=1}^{n_1} \phi_1(x)_i \cdot \phi_1(y)_i\right] + 3\left[\sum_{i=1}^{n_2} \phi_2(x)_i \cdot \phi_2(y)_i\right] \\
&= 7\left[\phi_1(x)_1 \cdot \phi_1(y)_1 + \ldots + \phi_1(x)_{n_1} \cdot \phi_1(y)_{n_1}\right] + 3\left[\phi_2(x)_1 \cdot \phi_2(y)_1 + \ldots + \phi_2(x)_{n_2} \cdot \phi_2(y)_{n_2}\right]
\end{aligned}$$

Setting $\phi(x) = (\sqrt{7}\phi_1(x)_1, \ldots, \sqrt{7}\phi_1(x)_{n_1}, \sqrt{3}\phi_2(x)_1, \ldots, \sqrt{3}\phi_2(x)_{n_2})$ we get that the above expression is exactly:

$$= \phi(x) \cdot \phi(y)$$

Meaning, $K(x,y)$ is an inner product, hence a kernel function.

(b) We know there is a linear classifier with w as its weights vector in $\mathbb{R}^m$. From the definition of a linear classifier we get:

$$C(x) = sgn(\sum_{i}^{m} w_i \cdot \phi_1(x)_i) = sgn(w \cdot \phi_1(x))$$

We define w', the weights vector in the higher dimension, to be:

$$w' = (\frac{w_1}{\sqrt{7}}, \ldots, \frac{w_m}{\sqrt{7}}, 0, \ldots, 0) \in \mathbb{R}^{n+m}$$

Using the linear classifier definition we show that w' is a linear classifer in the higher dimension $(m+n)$:

$$sgn(w' \cdot \phi(x)) = sgn(\frac{w_1}{\sqrt{7}} \cdot \sqrt{7}\phi_1(x)_1, \ldots, \frac{w_m}{\sqrt{7}} \cdot \sqrt{7}\phi_1(x)_m, 0 \cdot \sqrt{3}\phi_2(x)_1, \ldots, 0 \cdot \sqrt{3}\phi_2(x)_n)$$
$$= sgn(w_1\phi_1(x)_1, \ldots, w_m\phi_1(x)_m, 0, \ldots, 0)$$
$$= sgn(\sum_i^m w_i \cdot \phi_1(x)_i) = C(x)$$

Since we know w is a linear classifer, we found the linear classifier in the higher dimension.

(c) Given the lower dimension is n, and the kernel function is $K(x, y) = (\alpha x \cdot y + \beta)^d$, we can look at the rational varieties of order r:

$$\phi_i(x) = 1^{r_0} x_1^{r_1} ... x_n^{r_n} \qquad where \quad \sum_{i=0}^{n} r_i = r$$

Since the kernel degree is d, in our case $r = d$. Concluding from the above, the number of different monomer terms is $\frac{(n+d)!}{n! \cdot d!} = \binom{\mathbf{n+d}}{\mathbf{d}}$

(d) Given: $S = \{1, 2, ..., N\}$ and $f(x, y) = min(x, y)$. We define:

$$\phi(x) = (\sqrt{5}, \sqrt{5}, ..., 0, ..., 0)$$

<u>Explanation:</u>

We map each 1-dimensional vector $v = (x) \in S$ to a N-dimensional vector $v' \in \mathbb{R}^N$ such that the first x entries in v' are $\sqrt{5}$,and the $N - x$ left entries are 0s.

Assuming w.l.o.g that $f(x, y) = x$, meaning $x \leq y$:

$$\phi(x) \cdot \phi(y) = (\sqrt{5}_1, ..., \sqrt{5}_x, 0, ..., 0) \cdot (\sqrt{5}_1, ..., \sqrt{5}_y, 0, ..., 0)$$
$$= \sum_{i=0}^{x} \sqrt{5} \cdot \sqrt{5} = \sum_{i=0}^{x} 5 = 5x = 5min(x, y)$$

(e) First, a matrix $A_{n \times n}$ is positive-definite if $x^T Ax > 0$ for all $x \neq 0 \in \mathbb{R}^n$. From that we can conclude:

$$x^T A x \iff x^T \lambda x \iff \lambda x^T x \iff \lambda ||x||^2$$

Since $||x||^2 \geq 0$, we must have $\lambda > 0$. We show that there is an eigenvalue λ that does not satisfy this condition, for the S Gram-Matrix.

Assume towards contradiction that $f(x, y) = max(x, y)$ is a valid kernel function, and let $x = 1 \in S, \quad y = 2 \in S$.

Computing the Gram-Matrix using f we get:

$$A = \begin{bmatrix} f(1,1) & f(1,2) \\ f(1,2) & f(2,2) \end{bmatrix} = \begin{bmatrix} max(1,1) & max(1,2) \\ max(1,2) & max(2,2) \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix}$$

We find the eigenvalues of A:

$$A - \lambda I = \begin{bmatrix} 1-\lambda & 2 \\ 2 & 2-\lambda \end{bmatrix} \iff det(A - \lambda I) = det\left(\begin{bmatrix} 1-\lambda & 2 \\ 2 & 2-\lambda \end{bmatrix}\right) = (1-\lambda)(2-\lambda) - (2 \cdot 2)$$

$$det(A) = \lambda^2 - 3\lambda - 2 = 0$$

$$\lambda_1 = 3.562 \qquad \lambda_2 = -0.562$$

We can see that $\lambda_2 < 0 \implies$ **contradiction.**

Thus A is not a positive-definite matrix, and from Mercer's theorem, $f(x, y) = max(x, y)$ is not a valid kernel function.

Question 3

(a) Let $x, y \in \mathbb{R}^2$. From the given mapping function we get:

$$\phi(x) = (x_1^3, x_2^3, \sqrt{3}x_1^2x_2, \sqrt{3}x_1x_2^2, 2\sqrt{3}x_1^2, 2\sqrt{3}x_2^2, 2\sqrt{6}x_1x_2, 4\sqrt{3}x_1, 4\sqrt{3}x_2, 8)$$

$$\phi(y) = (y_1^3, y_2^3, \sqrt{3}y_1^2y_2, \sqrt{3}y_1y_2^2, 2\sqrt{3}y_1^2, 2\sqrt{3}y_2^2, 2\sqrt{6}y_1y_2, 4\sqrt{3}y_1, 4\sqrt{3}y_2, 8)$$

$$K(x, y) = \phi(x) \cdot \phi(y)$$

$$= x_1^3y_1^3 + x_2^3y_2^3 + 3x_1^2x_2y_1^2y_2 + 3x_1x_2^2y_1y_2^2 + 12x_1^2y_1^2 + 12x_2^2y_2^2 + 24x_1x_2y_1y_2 + 48x_1y_1 + 48x_2y_2 + 64$$

$$= (x_1y_1 + x_2y_2)^3 + 12(x_1^2y_1^2 + 2x_1x_2y_1y_2 + y_1^2y_2^2) + 48(x_1y_1x_2y_2) + 64$$

$$= (x \cdot y)^3 + 12(x \cdot y)^2 + 48(x \cdot y) + 64$$

$$= [x \cdot y + 4]^3$$

Defining $K_1(x, y) = (x \cdot y + 4)^3$, $\alpha = 1$, $\beta = 0$, we get

$$K(x, y) = \phi(x) \cdot \phi(y) = 1K_1 + 0K_2 = \mathbf{K_1}$$

(b) Let $x, y \in \mathbb{R}^2$. From the given mapping function we get:

$$\phi(x) = (\sqrt{10}x_1^2, \sqrt{10}x_2^2, \sqrt{20}x_1x_2, \sqrt{8}x_1, \sqrt{8}x_2, \sqrt{2})$$

$$\phi(x) = (\sqrt{10}y_1^2, \sqrt{10}y_2^2, \sqrt{20}y_1y_2, \sqrt{8}y_1, \sqrt{8}y_2, \sqrt{2})$$

$$K(x, y) = \phi(x) \cdot \phi(y) = 10x_1^2y_1^2 + 10x_2^2y_2^2 + 20x_1x_2y_1y_2 + 8x_1y_1 + 8x_2y_2 + 2$$

$$= 10(x_1^2y_1^2 + 2x_1x_2y_1y_2 + x_2^2y_2^2) + 8(x_1y_1 + x_2y_2 + \frac{1}{4})$$

$$= 10(x \cdot y)^2 + 8(x \cdot y + \frac{1}{4})$$

Defining $K_1(x, y) = 10(x \cdot y)^2$, $K_2(x, y) = (x \cdot y + \frac{1}{4})$, $\alpha = 10$, $\beta = 8$ we get:

$$K(x, y) = \phi(x) \cdot \phi(y) = \mathbf{10K_1 + 8K_2}$$

Question 4

The script itself is on the next page, and in a separate Python file in the submission folder:

kernel_vs_phi.py

Computing the mapping dimension:

Similarly to what we computed in a previous recitation, we can look at a general polynomial kernel function:

$$K(x, y) = (x \cdot y + c)^d$$

Since its a valid kernel function, it is an inner product. Assuming $x, y \in \mathbb{R}^n$, from using the multinormial formula we get:

$$K(x, y) = \phi(x) \cdot \phi(y) = c^2 + \sum_{i=1}^{n} \sqrt{2c} x_i y_i + \sum_{i=1}^{n} x_i^2 y_i^2 + \sum_{i=2}^{n} \sum_{j=1}^{n-1} 2 x_i x_j y_i y_j$$

From the rational varieties of order d we can conclude that the higher dimension is $\binom{n+d}{d}$.

In our specific case, with the lower dimension $n = 20$ we can conclude that the higher dimension $m = \binom{20+2}{2} = 231$.
Meaning, $\phi(x)$ is mapping each vector to $\mathbb{R}^m$:

$$\phi(\mathbf{x}) = (\mathbf{x_1^2} \ldots, \mathbf{x_n^2}, \ \sqrt{\mathbf{2}}\mathbf{x_1}\mathbf{x_2}, \ldots, \sqrt{\mathbf{2}}\mathbf{x_i}\mathbf{x_j}, \ldots, \sqrt{\mathbf{2}}\mathbf{x_{n-1}}\mathbf{x_n}, \ \sqrt{\mathbf{2}}\mathbf{x_1} \ldots, \sqrt{\mathbf{2}}\mathbf{x_n}, \mathbf{1}) \qquad \forall \mathbf{i} \neq \mathbf{j} \in [\mathbf{1}, \mathbf{n}]$$

We can observe better performance when calculating the Gram-Matrix using the kernel trick, in comparison to calculating the inner-product of each two vectors i, j.
The calculation of the inner-product of two 20-dimension vectors is faster than the inner-product of two 231-dimension vectors - thus the Kernel trick is a significant improvement.

```
import time
import sklearn.metrics.pairwise as sk_kernel
import numpy as np from sklearn.preprocessing
import PolynomialFeatures

# draw 20,000 random vectors with 20 dimensions
num_of_vectors = 20000
n = 20
vectors = np.random.rand(num_of_vectors, n)

# calculating the gram matrix (M[i][j] = K(Xi, Xj))
start_time = time.time()
gram_matrix = np.square(np.matmul(vectors, vectors.T) + 1)
end_time = time.time()

# mapping the vectors from the lower dimension (20) to the higher dimension (231)
phi = PolynomialFeatures(degree=2)
mapped_vectors = phi.fit_transform(vectors)
coef_list = []
i = 0
while i <= n:
      j = i
      while j <= n:
            if i == j:
                  coef_list.append(1)
        else:
                  coef_list.append(np.sqrt(2))
        j += 1
      i += 1
coef_vector = np.array(coef_list)
mapped_vectors = np.multiply(mapped_vectors, coef_vector)

# calculating the mapping matrix (M[i][j] = phi(x)phi(y))
start_time = time.time()
phi_matrix = np.matmul(mapped_vectors, mapped_vectors.T)
end_time = time.time()

# comparing the matrices
np.allclose(gram_matrix, phi_matrix)
```