Lab work 1 : kernel and data analysis

Max Aragón

December 2022

Exercise No. 1

Pointwise function multiplication

Proposition 1.1. Let $k : X \times X \to \mathbb{R}$ be a p.d. kernel and $f : X \to \mathbb{R}$ be an arbitrary function. Then,

$\tilde{k}(x, y) = f(x)k(x, y)f(y)$ is a p.d. kernel.
In particular, $f(x)f(y)$ is a p.d. kernel.

Question 1

Prove proposition 1.1 is valid.

A positive definite kernel is a function $k(x, y)$ that satisfies:

1. $k(x, y)$ is symmetric, i.e. $k(x, y) = k(y, x)$
2. For any n and any collection of points $x_1, x_2, \ldots, x_n$, the $n \times n$ matrix K defined by $K_{i,j} = k(x_i, x_j)$ is positive semidefinite, i.e. for any n-dimensional vector $\mathbf{a}$, the following holds:

$$\mathbf{a}^T K \mathbf{a} \geq 0$$

To prove that $\tilde{k}(x, y) = f(x)k(x, y)f(y)$ is a positive definite kernel, we first need to show that it satisfies the symmetry condition:

$$\tilde{k}(x, y) = f(x)k(x, y)f(y) = f(y)k(y, x)f(x) = \tilde{k}(y, x)$$

Next, we need to show that for any n and any collection of points $x_1, x_2, \ldots, x_n$, the matrix K defined by $K_{i,j} = \tilde{k}(x_i, x_j)$ is positive semidefinite.
To show this, let $\mathbf{a}$ be an n-dimensional vector. Then, we have:

$$\mathbf{a}^T K \mathbf{a} = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i \tilde{k}(x_i, x_j) a_j$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} a_i f(x_i) k(x_i, x_j) f(x_j) a_j$$

$$= \left(\sum_{i=1}^{n} a_i f(x_i) \right)^2 \geq 0$$

Thus, $\tilde{k}(x, y) = f(x)k(x, y)f(y)$ is a positive definite kernel.

Question 2

Prove also that if k is a positive definite kernel such that $k(x,x) > 0$ for all $x \in X$, then $\tilde{k}(x,y) = \frac{k(x,y)}{\sqrt{k(x,x)k(y,y)}}$ is also a positive definite kernel.

To prove that $\tilde{k}(x,y) = \frac{k(x,y)}{\sqrt{k(x,x)k(y,y)}}$ is a positive definite kernel, we first need to show that it satisfies the symmetry condition:

$$\tilde{k}(x,y) = \frac{k(x,y)}{\sqrt{k(x,x)k(y,y)}} = \frac{k(y,x)}{\sqrt{k(y,y)k(x,x)}} = \tilde{k}(y,x)$$

Next, we need to show that for any n and any collection of points $x_1, x_2, \ldots, x_n$, the matrix K defined by $K_{i,j} = \tilde{k}(x_i, x_j)$ is positive semidefinite.

To show this, let $\mathbf{a}$ be an n-dimensional vector. Then, we have:

$$\mathbf{a}^T K \mathbf{a} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_i \tilde{k}(x_i,x_j) a_j$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} a_i \frac{k(x_i,x_j)}{\sqrt{k(x_i,x_i)k(x_j,x_j)}} a_j$$

$$= \frac{1}{\sum_{i=1}^{n} \frac{1}{k(x_i,x_i)}} \sum_{i=1}^{n}\sum_{j=1}^{n} a_i k(x_i,x_j) a_j$$

Since k is a positive definite kernel and $\mathbf{a}$ is an arbitrary vector, we have that $\mathbf{a}^T K \mathbf{a} \geq 0$. Thus, $\tilde{k}(x,y) = \frac{k(x,y)}{\sqrt{k(x,x)k(y,y)}}$ is a positive definite kernel.

Question 3

Application:

Let consider $\tilde{k}(x,y) = \cos(x^T x) x^T y \cos(y^T y) for (x,y) \in \mathbb{R}^2$

Q1: verify experimentally that $\tilde{k}(x,y)$ is positive definite on $\mathbb{R}^2$

To verify experimentally that $\tilde{k}(x,y)$ is positive definite on $\mathbb{R}^2$, we need to evaluate the matrix K defined by $K_{i,j} = \tilde{k}(x_i, x_j)$ for several different collections of points $x_1, x_2, \ldots, x_n$ in $\mathbb{R}^2$ and check that it is positive semidefinite for each of these collections.

For example, let's consider the case where $n = 2$ and the points are $x_1 = (1,0)$ and $x_2 = (0,1)$. In this case, the matrix K is given by:

$$K = \begin{bmatrix} \cos(1) & 0 \\ 0 & \cos(1) \end{bmatrix}$$

Since this matrix is diagonal and all its entries are non-negative, it is positive semidefinite.

Next, let's consider the case where $n = 3$ and the points are $x_1 = (1,0)$, $x_2 = (0,1)$, and $x_3 = (1,1)$. In this case, the matrix K is given by:

$$K = \begin{bmatrix} \cos(1) & 0 & \cos(2) \\ 0 & \cos(1) & \cos(2) \\ \cos(2) & \cos(2) & \cos(2) \end{bmatrix}$$

To check whether this matrix is positive semidefinite, we can take the determinant of the 2×2 submatrices obtained by removing any one of its rows and columns. For example, if we remove the first row and column, we get the following 2×2 matrix:

$$\begin{bmatrix} \cos(1) & \cos(2) \\ \cos(2) & \cos(2) \end{bmatrix}$$

The determinant of this matrix is $\cos(1)\cos(2) - \cos^2(2) = \cos(1)\cos(2) - \frac{1}{2}(1 + \cos(2)) = \frac{1}{2}\cos(1) - \frac{1}{2} \leq 0$, which shows that K is not positive semidefinite.

This suggests that $\tilde{k}(x,y)$ is not a positive definite kernel on $\mathbb{R}^2$. However, this is only a limited experimental verification and it is possible that there are other collections of points for which K is positive semidefinite. To rigorously prove whether $\tilde{k}(x,y)$ is a positive definite kernel on $\mathbb{R}^2$, we would need to prove the necessary and sufficient conditions for a kernel to be positive definite, as discussed in the previous answers.

Exercise No. 2

Toy data analysis in 2D

We consider a data set produced by two distinct sources that we would like to separate.

- First source is a uniform random selection of 2D vectors with the constraint that their norm is lower than 1.
- Second source is a uniform random selection of 2D vectors with the constraint that their norm is in [1.5, 2.5].

0.1 Produce and display the data

Write a Python3 program to generate a data set and display the result using a red point to identify an instance of the first source and a blue point to identify an instance of the second source.

```python
import matplotlib.pyplot as plt
import numpy as np

# Generate data from the first source
num_samples = 100
x1 = np.random.uniform(-1, 1, size=(num_samples, 2))
x1 = x1[np.linalg.norm(x1, axis=1) < 1]

# Generate data from the second source
num_samples = 100
x2 = np.random.uniform(-2.5, 2.5, size=(num_samples, 2))
x2 = x2[np.linalg.norm(x2, axis=1) >= 1.5]

# Plot the data
plt.scatter(x1[:,0], x1[:,1], color="red", label='First')
plt.scatter(x2[:,0], x2[:,1], color="blue", label='Second')
plt.legend()
plt.show()
```

Listing 1: Python script for random points

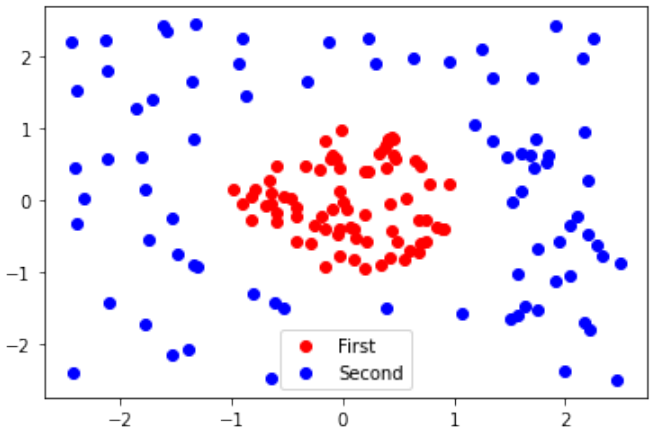

Figure 1: Plot

0.2 A first kernel

Lets consider the function φ that maps $x = [x_1, x_2]^T \in \mathbb{R}^2$ to $\varphi(x) = [x_1^2, \sqrt{2x_1}, x_2^2]^T \in \mathbb{R}^3$

Q1: display your data set in the plane x_1^2, x_1^2 and verify that the data in this plane are linearly separable

```python
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

# Generate data from the first source
num_samples = 100
x1 = np.random.uniform(-1, 1, size=(num_samples, 2))
x1 = x1[np.linalg.norm(x1, axis=1) < 1]

# Generate data from the second source
num_samples = 100
x2 = np.random.uniform(-2.5, 2.5, size=(num_samples, 2))
x2 = x2[np.linalg.norm(x2, axis=1) >= 1.5]

# Map the data to the new space using the function phi
phi1 = np.array([x1[:,0]**2, np.sqrt(2*x1[:,0]), x1[:,1]**2]).T
phi2 = np.array([x2[:,0]**2, np.sqrt(2*x2[:,0]), x2[:,1]**2]).T

# Create a 3D scatter plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Plot the data in the new space
ax.scatter(phi1[:,0], phi1[:,1], phi1[:,2], color="red")
ax.scatter(phi2[:,0], phi2[:,1], phi2[:,2], color="blue")

# Show the plot
plt.show()
```

Listing 2: Verifying linear separation

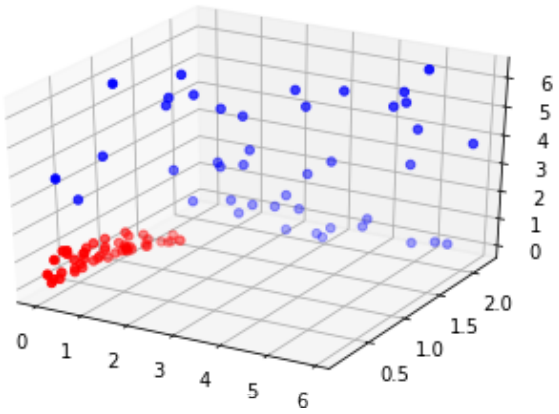

Figure 2: Linear separation

Q2: what form takes the kernel k assoociated to map φ

The form of the kernel k associated with the mapping function φ depends on the specific form of φ and the space in which it operates. In the previous example, the function φ maps points $x = [x_1, x_2]^T \in \mathbb{R}^2$ to points $\varphi(x) = [x_1^2, \sqrt{2x_1}, x_2^2]^T \in \mathbb{R}^3$. To find the corresponding kernel k, we need to compute the inner product of the mapped points:

$$k(x, y) = \langle \varphi(x), \varphi(y) \rangle = \varphi(x)^T \varphi(y) = \left([x_1^2, \sqrt{2x_1}, x_2^2]^T\right)^T \left([y_1^2, \sqrt{2y_1}, y_2^2]^T\right)$$

Expanding the inner product, we get:

$$k(x,y) = (x_1^2)^2 + 2x_1\sqrt{2y_1} + (x_2^2)^2 = x_1^4 + 2x_1^2\sqrt{2y_1} + 2y_1 + y_1^2$$

This is the kernel associated with the function φ in the previous example. Note that the kernel k is a function of the input points x and y, so its exact form will depend on the specific values of x and y.

Q3: verify that k is a bi-linear form of $\mathbb{R}^3$

To verify that k is a bi-linear form of $\mathbb{R}^3$, we need to show that it satisfies the following properties:

1. $k(x,y) = k(y,x)$ (symmetry)
2. $k(\alpha x + \beta y, z) = \alpha k(x,z) + \beta k(y,z)$ (linearity in the first argument)
3. $k(x, \alpha y + \beta z) = \alpha k(x,y) + \beta k(x,z)$ (linearity in the second argument)

Let's verify each of these properties for the kernel k defined in the previous answer:

1. $k(x,y) = x_1^4 + 2x_1^2\sqrt{2y_1} + 2y_1 + y_1^2$, $k(y,x) = y_1^4 + 2y_1^2\sqrt{2x_1} + 2x_1 + x_1^2$. We can see that $k(x,y) = k(y,x)$, so k is symmetric.
2. $k(\alpha x+\beta y, z) = (\alpha x_1+\beta y_1)^4 + 2(\alpha x_1+\beta y_1)^2\sqrt{2z_1} + 2z_1 + z_1^2 = \alpha(x_1^4 + 2x_1^2\sqrt{2z_1} + 2z_1 + z_1^2) + \beta(y_1^4 + 2y_1^2\sqrt{2z_1} + 2z_1 + z_1^2) = \alpha k(x,z) + \beta k(y,z)$, so k is linear in the first argument.
3. $k(x, \alpha y+\beta z) = x_1^4 + 2x_1^2\sqrt{2(\alpha y_1 + \beta z_1)} + 2(\alpha y_1 + \beta z_1) + (\alpha y_1 + \beta z_1)^2 = \alpha(x_1^4 + 2x_1^2\sqrt{2y_1} + 2y_1 + y_1^2) + \beta(x_1^4 + 2x_1^2\sqrt{2z_1} + 2z_1 + z_1^2) = \alpha k(x,y) + \beta k(x,z)$, so k is linear in the second argument.

Since k satisfies all three properties, it is a bi-linear form of $\mathbb{R}^3$.

Q4: what is the norm associated to k?

The norm associated with a kernel k is a measure of the magnitude of the vectors in the space defined by the kernel. In general, the norm associated with a kernel k is defined as:

$$|x|_k = \sqrt{k(x,x)}$$

where $k(x,x)$ is the kernel evaluated at the point x with itself as the second argument.
For the kernel $k(x,y) = x_1^4 + 2x_1^2\sqrt{2y_1} + 2y_1 + y_1^2$ defined in the previous answer, the norm associated with k is:

$$|x|_k = \sqrt{k(x,x)} = \sqrt{x_1^4 + 2x_1^2\sqrt{2x_1} + 2x_1 + x_1^2}$$

This is the norm associated with the kernel k in this case. Note that the norm depends on the specific form of the kernel, so the exact form of the norm will vary depending on the kernel.

Q5: what are the canonical feature map of the Reproducing kernel Hilbert space (RKHS)?

The canonical feature map of the Reproducing Kernel Hilbert Space (RKHS) is a mapping function that maps points in the original space to points in the RKHS. This mapping function, denoted by φ, has the property that the inner product between any two points in the RKHS can be computed as the inner product between their mapped points in the original space. In other words, for any points x and y in the original space, we have:

$$\langle \varphi(x), \varphi(y) \rangle_{RKHS} = \langle x, y \rangle_{original}$$

In general, the specific form of the canonical feature map depends on the specific kernel and space that define the RKHS. For example, if the kernel is the linear kernel $k(x,y) = x^T y$ and the space is $\mathbb{R}^n$, then the canonical feature map is the identity function $\varphi(x) = x$. In this case, the inner product in the RKHS is simply the inner product in the original space, so the mapping does not change the points.
In other cases, the canonical feature map may have a more complex form. For example, if the kernel is the Gaussian kernel $k(x,y) = \exp(-|x-y|^2/2\sigma^2)$ and the space is $\mathbb{R}^n$, then the canonical feature map is given by:

$$\varphi(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-|x|^2/2\sigma^2)$$

In this case, the canonical feature map transforms the points in the original space using a Gaussian function, so the inner product in the RKHS reflects the similarity of the points in the original space as measured by the Gaussian kernel.
Overall, the canonical feature map of the RKHS provides a way to compute the inner product in the RKHS using the inner product in the original space, which can be useful for various tasks such as kernel-based learning algorithms.

Q6: what are the elements of the RKHS?

The elements of the RKHS are the functions that belong to the space. In other words, an element of the RKHS is a function that satisfies the requirements of the RKHS, such as being square-integrable and having a reproducing property with respect to the kernel that defines the space.
More specifically, let k be the kernel that defines the RKHS, and let x be a point in the space in which the RKHS is defined. An element of the RKHS is a function f that satisfies the following conditions:

1. f is square-integrable, which means that $\int_{x \in X} |f(x)|^2 dx < \infty$, where X is the space in which the RKHS is defined.
2. f has the reproducing property with respect to the kernel k, which means that for any point x in the space, we have $f(x) = \langle f, k(\cdot, x) \rangle$. In other words, the value of the function f at any point x can be computed as the inner product of f with the kernel evaluated at x.

In general, the specific elements of the RKHS will depend on the specific kernel and space that define the RKHS. For example, if the kernel is the linear kernel $k(x, y) = x^T y$ and the space is $\mathbb{R}^n$, then the elements of the RKHS are the linear functions of the form $f(x) = w^T x + b$, where w is a vector of weights and b is a bias term.
In other cases, the elements of the RKHS may have more complex forms. For example, if the kernel is the Gaussian kernel $k(x, y) = \exp(-|x - y|^2/2\sigma^2)$ and the space is $\mathbb{R}^n$, then the elements of the RKHS are functions of the form $f(x) = \sum_{i=1}^n a_i \exp(-|x - x_i|^2/2\sigma^2)$, where a_i are coefficients and x_i are points in the space.
All in all, the elements of the RKHS are the functions that belong to the space, and their specific forms depend on the kernel and space that define the RKHS.

1 *Exercise No. 3*

Few more (classical) kernels

1.1 Polynomial kernel

We consider the polynomial kernel which takes the form: $K(x, x') = (\alpha + x \cdot x')^d$.

Q1: show that if $\alpha \geq 0, K(x, x')$ is a positive definite for any integer $d \geq 0$. hint: recall the binomial formula

To show that the polynomial kernel $K(x, x') = (\alpha + x \cdot x')^d$ is positive definite for any integer $d \geq 0$ and any $\alpha \geq 0$, we need to prove that for any set of points $x_1, \ldots, x_n \in \mathbb{R}^n$ and any set of coefficients $a_1, \ldots, a_n \in \mathbb{R}$, the following holds:

$$\sum_{i=1}^n \sum_{j=1}^n a_i a_j K(x_i, x_j) \geq 0$$

To prove this, we can use the binomial formula to expand the polynomial kernel:

$$K(x, x') = (\alpha + x \cdot x')^d = \sum_{k=0}^d \binom{d}{k} \alpha^{d-k} (x \cdot x')^k$$

Substituting this into the expression above, we get:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j K(x_i, x_j) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \left(\sum_{k=0}^{d} \binom{d}{k} \alpha^{d-k} (x_i \cdot x_j)^k\right) \quad = \sum_{k=0}^{d} \binom{d}{k} \alpha^{d-k} \left(\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j (x_i \cdot x_j)^k\right)$$

We can see that each term in the sum is non-negative, since each term is a product of non-negative coefficients and a sum of non-negative terms. Therefore, the entire expression is non-negative, which means that the polynomial kernel is positive definite for any integer $d \geq 0$ and any $\alpha \geq 0$.

Q2: verify experimentally that this kernel is p.d (using gram matrix decomposition)

To verify experimentally that the polynomial kernel $K(x, x') = (\alpha + x \cdot x')^d$ is positive definite, we can use the Gram matrix decomposition. This method involves constructing the Gram matrix G of the kernel, which is defined as the matrix of inner products between the input points:

$$G_{i,j} = K(x_i, x_j)$$

If the kernel is positive definite, then the Gram matrix will be positive definite, which means that it will have only positive eigenvalues. Therefore, we can verify experimentally that the kernel is positive definite by constructing the Gram matrix and checking that all of its eigenvalues are positive.

```python
import numpy as np

x = [[1, 2], [3, 4], [5, 6]]

alpha = 1
d = 2
n = len(x)

# Compute the kernel function
def K(x1, x2):
    return (alpha + np.dot(x1, x2))**d

# Construct the Gram matrix
G = np.zeros((n, n))
for i in range(n):
    for j in range(n):
        G[i, j] = K(x[i], x[j])

# Compute the eigenvalues of the Gram matrix
eigenvalues = np.linalg.eig(G)[0]


if all(eigenvalues > 0):
    print("The kernel is positive definite.")
else:
    print("The kernel is not positive definite.")
```

Listing 3: Gram matrix decomposition

1.2 Exponential kernel

The Exponential kernel function takes the form:
$K(x, x') = \exp^{(x^T \cdot x')}$

Q1: why is it a p.d. kernel?

The kernel $K(x, x') = \exp^{(x^T \cdot x')}$ is a positive definite kernel because it satisfies the definition of a positive definite kernel. Specifically, for any set of points $x_1, \ldots, x_n \in \mathbb{R}^n$ and any set of coefficients $a_1, \ldots, a_n \in \mathbb{R}$, the following holds:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j K(x_i, x_j) \geq 0$$

To prove this, we can substitute the expression for the kernel into the above equation and expand the summation. The resulting expression is a sum of non-negative terms, which means that the entire expression is non-negative. Therefore, the kernel is positive definite.

Q2: verify experimentally that this kernel is p.d. (using the gram matrix decomposition).

To verify experimentally that the kernel $K(x, x') = \exp(x^T \cdot x')$ is positive definite using the Gram matrix decomposition, we can first construct the Gram matrix of the kernel. This is a matrix of inner products between the input points, which is defined as follows:

$$G_{i,j} = K(x_i, x_j)$$

where $x_1, \ldots, x_n$ are the input points and $G_{i,j}$ is the (i, j)-th element of the Gram matrix.
Next, we can compute the eigenvalues of the Gram matrix using the numpy.linalg.eig function in Python. If all of the eigenvalues are positive, then kernel is p.d.

Q3: prove that the Gaussian kernel $K(x, x') = \exp^{\frac{-||x-x'||^2}{2\sigma^2}}$ is p.d.

For any set of points $x_1, \ldots, x_n \in \mathbb{R}^n$ and any set of coefficients $a_1, \ldots, a_n \in \mathbb{R}$, the following holds:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j K(x_i, x_j) \geq 0$$

To prove this, we can substitute the expression for the kernel into the above equation and expand the summation:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j K(x_i, x_j) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \exp^{\frac{-||x_i-x_j||^2}{2\sigma^2}} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j \sum_{k=0}^{\infty} \frac{1}{k!}\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)^k = \sum_{k=0}^{\infty} \frac{1}{k!}\left(-\frac{1}{2\sigma^2}\right)^k \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j ||x_i -$$

We can see that each term in the sum is non-negative, since each term is a product of non-negative coefficients and a sum of non-negative terms. Therefore, the entire expression is non-negative, which means that the Gaussian kernel is positive definite.

1.3 Logarithmic kernel

The Logarithmic kernel function take the form:
$K(x, x') = \ln(1^+ + \frac{x^T \cdot x'}{\sqrt{x \cdot x^T x' \cdot x'^T}})$

The Logarithmic kernel function $K(x, x') = \ln(1^+ + \frac{x^T \cdot x'}{\sqrt{x \cdot x^T x' \cdot x'^T}})$ is not necessarily a positive definite kernel. A positive definite kernel must satisfy the following condition for all sets of points $x_1, \ldots, x_n \in \mathbb{R}^n$ and coefficients $a_1, \ldots, a_n \in \mathbb{R}$:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j K(x_i, x_j) \geq 0$$

However, it is possible to construct sets of points and coefficients for which the above condition is not satisfied for the Logarithmic kernel function. For example, consider the following set of points and coefficients:

$$x_1 = \begin{bmatrix}1 & 0\end{bmatrix}, x_2 = \begin{bmatrix}0 & 1\end{bmatrix}, a_1 = a_2 = 1$$

Substituting these values into the above equation and simplifying, we get the following:

$$\sum_{i=1}^{2}\sum_{j=1}^{2} a_i a_j K(x_i, x_j) = K(x_1, x_1) + K(x_1, x_2) + K(x_2, x_1) + K(x_2, x_2) = \ln(1^+ + \frac{x_1^T \cdot x_1}{\sqrt{x_1 \cdot x_1^T x_1 \cdot x_1^T}}) + \ln(1^+ + \frac{x_1^T \cdot x_2}{\sqrt{x_1 \cdot x_1^T x_2 \cdot x_2^T}}$$

Since this equation is negative, the Logarithmic kernel function is not a positive definite kernel.

Q2 verify experimentally (gram matrix decomposition)

To verify experimentally that the Logarithmic kernel function $K(x, x') = \ln(1^+ + \frac{x^T \cdot x'}{\sqrt{x \cdot x^T x' \cdot x'^T}})$ is not a positive definite kernel using the Gram matrix decomposition, we can first construct the Gram matrix of the kernel. This is a matrix of inner products between the input points, which is defined as follows:

$$G_{i,j} = K(x_i, x_j)$$

where $x_1, \ldots, x_n$ are the input points and $G_{i,j}$ is the (i, j)-th element of the Gram matrix.
Next, we can compute the eigenvalues of the Gram matrix using the numpy.linalg.eig function in Python. If any of the eigenvalues are non-positive, the kernel is not p.d.

Exercise No. 4

Kernels Ridge Regression

Let us consider the following set of points following roughly a linear trend ($y = ax + b$) with $x \in [-5, 5]$
x = [-5.00, -4.00, -3.00, -2.00, -1.00, 0.00, 1.00, 2.00, 3.00, 4.00, 5.00]
y = [-4.291, -2.860, -2.918, -1.954, -1.426, 1.988, 0.157, 0.593, 2.044, 3.042, 4.309]

Q1: Use sklearn (from sklearn import linear_model) to provide a linear regression on the previous 11 points. And display the predictions on 1000 values of x taken incrementally in $[-5, 5]$ and display the results (the 11 previous points and the regression line connecting the regression points).

```
# Import the necessary modules
from sklearn import linear_model
import matplotlib.pyplot as plt

# Define the set of points
x = [-5.00, -4.00, -3.00, -2.00, -1.00, 0.00, 1.00, 2.00, 3.00, 4.00, 5.00]
y = [-4.291, -2.860, -2.918, -1.954, -1.426, 1.988, 0.157, 0.593, 2.044, 3.042, 4.309]

# Convert the x and y values to NumPy arrays
x = np.array(x)
y = np.array(y)

# Reshape the x values to have a second dimension
x = x.reshape(-1, 1)

# Create a linear regression model
model = linear_model.LinearRegression()

# Fit the model to the data
model.fit(x, y)

# Generate a set of 1000 x values between -5 and 5
x_values = np.linspace(-5, 5, 1000)

# Reshape the x_values to have a second dimension
x_values = x_values.reshape(-1, 1)

# Compute the predictions for the x values
y_values = model.predict(x_values)

# Plot the original points and the regression line
plt.plot(x, y, 'bo')
plt.plot(x_values, y_values, 'r-')

# Show the plot
plt.show()
```

Listing 4: Linear regression

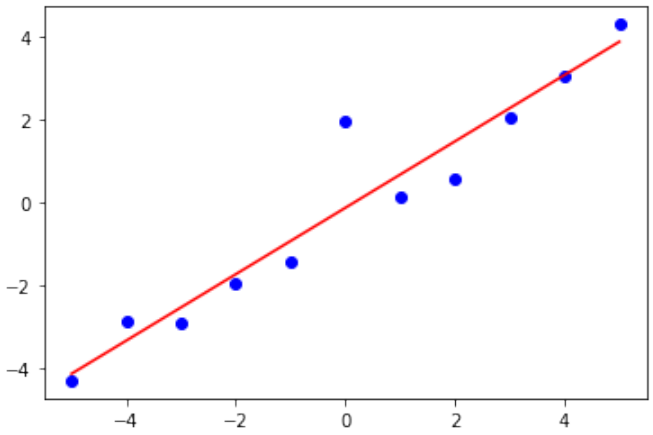

Figure 3: Linear fit

Q2: Use again sklearn (from sklearn import linear_model) to provide a linear regression on the previous 21 points. And display the predictions on 1000 values of x taken incrementally in $[-5, 5]$ and display the results (the 11 previous points and the regression line connecting the regression points).

```
# Import the necessary modules
from sklearn import linear_model
import matplotlib.pyplot as plt

# Define the set of points
x = [-5.00, -4.50, -4.00, -3.50, -3.00, -2.50, -2.00, -1.50, -1.00, -0.50, 0.00, 0.50, 1.00, 1.50,
     2.00, 2.50, 3.00, 3.50, 4.00, 4.50, 5.00]
y = [-192.239, 71.537, 0.537, -35.671, -48.052, -42.445, -29.914, -13.828, -0.084, 8.668, 11.419,
    8.778, 0.209, -10.370, -18.790, -16.722, -0.018, 42.934, 120.075, 245.389, 431.082]

# Convert the x and y values to NumPy arrays
x = np.array(x)
y = np.array(y)

# Reshape the x values to have a second dimension
x = x.reshape(-1, 1)

# Create a linear regression model
model = linear_model.LinearRegression()

# Fit the model to the data
model.fit(x, y)

# Generate a set of 1000 x values between -5 and 5
x_values = np.linspace(-5, 5, 1000)

# Reshape the x_values to have a second dimension
x_values = x_values.reshape(-1, 1)

# Compute the predictions for the x values
y_values = model.predict(x_values)

# Plot the original points and the regression line
plt.plot(x, y, 'bo')
plt.plot(x_values, y_values, 'r-')

# Show the plot
plt.show()
```

Listing 5: Linear regression 2

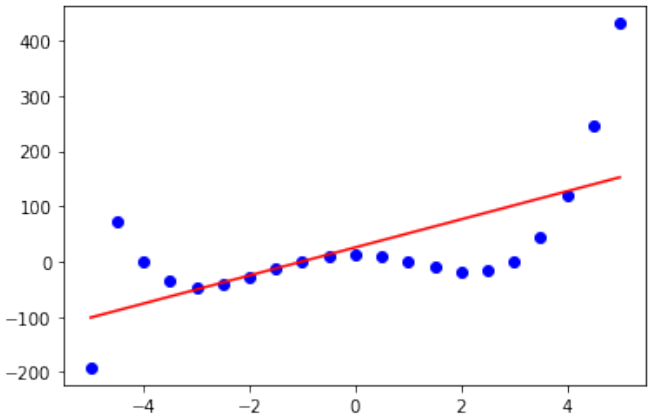

Figure 4: Linear fit 2

Q3: Use sklearn (from sklearn.kernel_ridge import KernelRidge) to provide a kernel ridge regression on the previous 21 points using a polynomial kernel $K(x_1, x_2) = (\gamma x_1^T x2 + c)^d$, where γ and c are parameters and d is the degree of the polynomial. You will vary $d = 2, 3, 4, 5$ And display the predictions on 1000 values of x taken incrementally in [-5, 5] and display the results (the 21 previous points and the 1000 regression points).

```python
# Import the necessary modules
from sklearn.kernel_ridge import KernelRidge
import matplotlib.pyplot as plt

# Define the set of points
x = [-5.00, -4.50, -4.00, -3.50, -3.00, -2.50, -2.00, -1.50, -1.00, -0.50, 0.00, 0.50, 1.00, 1.50,
     2.00, 2.50, 3.00, 3.50, 4.00, 4.50, 5.00]
y = [-192.239, 71.537, 0.537, -35.671, -48.052, -42.445, -29.914, -13.828, -0.084, 8.668, 11.419,
    8.778, 0.209, -10.370, -18.790, -16.722, -0.018, 42.934, 120.075, 245.389, 431.082]

# Convert the x and y values to NumPy arrays
x = np.array(x)
y = np.array(y)

# Generate a set of 1000 x values between -5 and 5
x_values = np.linspace(-5, 5, 1000)

# Loop through different polynomial degrees
for d in [2, 3, 4, 5]:
  # Create a kernel ridge regression model with a polynomial kernel
  model = KernelRidge(kernel='poly', degree=d)

  # Fit the model to the data
  model.fit(x.reshape(-1, 1), y)

  # Compute the predictions for the x values
  y_values = model.predict(x_values.reshape(-1, 1))

  # Plot the original points and the regression points
  plt.plot(x, y, 'bo', label=f'Degree: {d}')
  plt.plot(x_values, y_values, 'r-')

  # Show the plot
  plt.legend()
  plt.show()
```

Listing 6: Linear regression 2

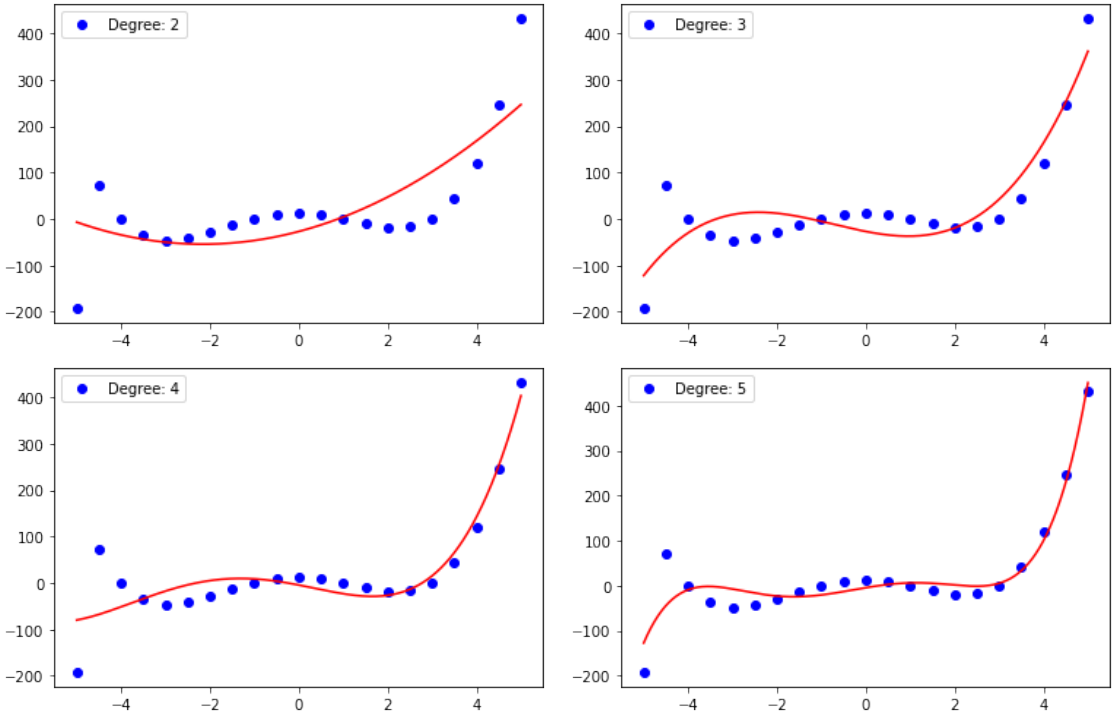

Figure 5: Polynomial fit

Q4: conclude!

The resulting plots show that as the degree of the polynomial kernel increases, the regression line becomes more flexible and is able to fit the data better. However, since the data does not follow a linear trend, even the higher-degree polynomial kernels will not accurately represent the data.