Homework 5: Kernels

December 10, 2022

Exercise 1

Exercise 1a

$$k(x,x') = (< x, x' > +1)^2 = (x_1x_1' + x_2x_2' + 1)^2 = (x_1^2x_1'^2 + x_2^2x_2'^2 + 2x_1x_1'x_2x_2' + 2x_1x_1' + 2x_2x_2' + 1) \quad (1)$$

Overall we have 6 monomials, so feature space dimensionality is 6.

The function e^t can be shown as an indefinite sum according to Taylor's formula: $e^t = 1 + t/1 + t^2/2 + ...$ In our case, $t = -||x - x'||^2/2$ and if we apply Taylor's formula to the exponential function we get the indefinite number of monomials, so the feature space dimensionality is indefinite.

Exercise 1b

First of all, we have an RBF kernel, that can be applied despite producing an indefinite number of features. So there is a workaround for that case.

Exercise 2

Exercise 2a

It is enough to prove that the Gram matrix is positive semi-definite, or K satisfies the following inequality: $\sum_{i,j} c_i c_j K_{ij} \geq 0$

$$\sum_{i,j} c_i c_j K_{ij} = \sum_{i,j} c_i c_j k(x_i, x_j) = \sum_{i,j} c_i c_j < x_i, x_j >= \sum_{i,j} < c_i x_i, c_j x_j >= || \sum_i c_i x_i ||^2 \geq 0 \quad (2)$$

Exercise 2b

A similar approach is applied here:

$$\sum_{i,j} c_i c_j K_{ij} = \sum_{i,j} c_i c_j k(\phi(x_i), \phi(x_j)) = \sum_{i,j} c_i c_j < \phi(x_i), \phi(x_j) >= \sum_{i,j} < c_i \phi(x_i), c_j \phi(x_j) >= || \sum_i c_i \phi(x_i) ||^2 \geq 0 \quad (3)$$

Exercise 2c

Since $k_1(x,x)$ and $k_2(x,x)$ are kernels, it satisfies inequalities: $\sum_{i,j} c_i c_j k_1(x_i, x_j) \geq 0$ and $\sum_{i,j} c_i c_j k_2(x_i, x_j) \geq 0$ We need to prove the similar inequalities for $k_3(x,x')$ and $k_4(x,x')$:

$$\sum_{i,j} c_i c_j k_3(x_i, x_j) = \sum_{i,j} c_i c_j (k_1(x_i, x_j) + k_2(x_i, x_j)) = \sum_{i,j} c_i c_j k_1(x_i, x_j) + \sum_{i,j} c_i c_j k_2(x_i, x_j)) \geq 0 \quad (4)$$

$$\sum_{i,j} c_i c_j k_4(x_i, x_j) = \sum_{i,j} c_i c_{j1}(x_i, x_j) = \lambda \sum_{i,j} c_i c_j k_1(x_i, x_j) \geq 0 \quad (5)$$

Exercise 3

Exercise 3a

Since $<x,x>^4$ is a linear kernel with $c=0, p=4$, 1 is a linear kernel with $c=0, p=0$ $x^Tx = <x,x>$ is a linear kernel with $c=0, p=1$ $exp(-||x-x||^2/2\sigma^2)$ is a RBF kernel. So, by applying closure properties, we might get that: $6<x,x>^4, 3, 6<x,x>^4+3, 6<x,x>^4+3+x^Tx, 6<x,x>^4+3+x^Tx+exp(-||x-x||^2/2\sigma^2)$ are kernels.

Exercise 3b

$$k_{GXY}(X, X') = \sum_{s\in S, s'\in S'} k_{base}(s,s) \tag{6}$$

where S and S' are sets of all 3-mers of input strings X and X' respectively. And

$$k_{base}(s,s) = \sum_{s\in S, s'\in S'} k_{base}(s,s) \begin{cases} 0, if s[0] \neq' G' or s'[0] \neq' G' \\ 1 + I(s[1], s'[1]) + I(s[2], s'[2]) \end{cases} \tag{7}$$

where $I(c, c')$ is a delta function, it is equal to 1 if $c = c'$ and 0 otherwise. And s[i] is the i-th symbol of string s (numeration starts with 0).

Exercise 3c

According to the definition, $k_{GXY}(X_1, X_2) = 0$ since all 3-mers of X_2 do not start with 'G'. For calculation $k_{GXY}(X_1, X_3)$ the following code was used:

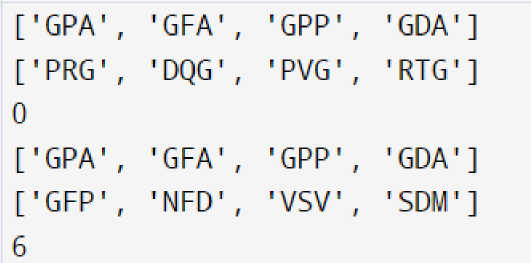
```python
def KXY(x1, x2):
    M1 = mers(x1)
    print(M1)
    M2 = mers(x2)
    print(M2)
    k = 0
    for m1 in M1:
        for m2 in M2:
            k += k_base(m1, m2)
    return k

def mers(X):
    m = []
    i = 0
    while i < len(X) - 2:
        m.append(X[i:(i + 3)])
        i += 3
    return m

def k_base(mer1, mer2):
    if(mer1[0] != 'G' or mer2[0] != 'G'):
        return 0
    return 1 + (mer1[1] == mer2[1]) + (mer1[2] == mer2[2])

X1 = 'GPAGFAGPPGDA'
X2 = 'PRGDQGPVGRTG'
X3 = 'GFPNFDVSVSDM'
print(KXY(X1, X2))
print(KXY(X1, X3))
```

It extracts all 3-mers of both input strings and summarizes k_{base} values for all pairs of 3-mers. In our case, X_3 has only one mer that starts with 'G', so it is enough to calculate k_{base} for this 3-mer ('GFP') and other 3-mers of X_1.



```
['GPA', 'GFA', 'GPP', 'GDA']
['PRG', 'DQG', 'PVG', 'RTG']
0
['GPA', 'GFA', 'GPP', 'GDA']
['GFP', 'NFD', 'VSV', 'SDM']
6
```

Exercise 3d

If we have sequences of different lengths, it does not affect calculations a lot since we use all pairs of 3-mers of both input strings. But we have issues if one of the input strings is not multiples of 3 because we have a problem by extracting complete 3-mers. We might ignore the last characters to avoid such an issue.