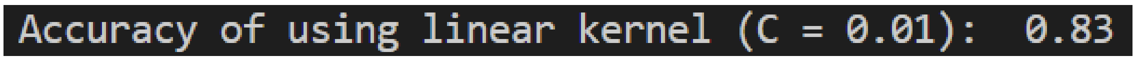
NYCU Introduction to Machine Learning, Homework 4

110550108 施柏江

Part. 1, Coding (50%):

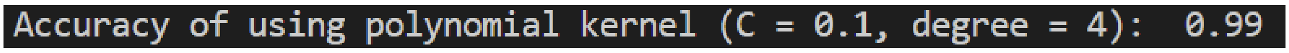
(50%) Support Vector Machine

1. (10%) Show the accuracy score of the testing data using linear_kernel. Your accuracy score should be higher than 0.8.



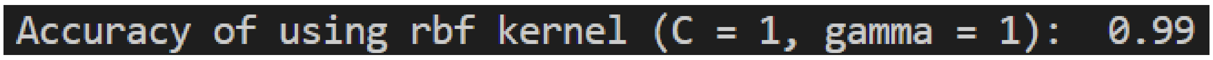
```
Accuracy of using linear kernel (C = 0.01):  0.83
```

2. (20%) Tune the hyperparameters of the polynomial_kernel. Show the accuracy score of the testing data using polynomial_kernel and the hyperparameters you used.



```
Accuracy of using polynomial kernel (C = 0.1, degree = 4):  0.99
```

3. (20%) Tune the hyperparameters of the rbf_kernel. Show the accuracy score of the testing data using rbf_kernel and the hyperparameters you used.



```
Accuracy of using rbf kernel (C = 1, gamma = 1):  0.99
```

Part. 2, Questions (50%):

1. (20%) Given a valid kernel $k_1(x, x')$, prove that the following proposed functions are or are not valid kernels. If one is not a valid kernel, give an example of $k(x, x')$ that the corresponding K is not positive semidefinite and shows its eigenvalues.
   a. $k(x, x') = k_1(x, x') + exp(x^T x')$
   b. $k(x, x') = k_1(x, x') - 1$
   c. $k(x, x') = exp(\|x - x'\|^2)$
   d. $k(x, x') = exp(k_1(x, x')) - k_1(x, x')$

a. This function is a valid kernel. To prove this, let’s consider the kernel matrix K with elements Kij = k(xi, xj).If K1 is the kernel matrix corresponding to k1, then we have: K = K1 + E, where E is the matrix with elements exp(xi^T *xj). Now, consider any vector v and calculate v^T*K*v: v^T*K*v = v^T*K1*v + v^T*E*v. The first term is non-negative because K1 is PSD. The second term is also non-negative because exp(x^T *x’) is always positive. Therefore, v^T*K*v >= 0 for any v, and K is PSD.

b. This function is not necessarily a valid kernel. To see why, consider a specific example where k1(x, x’) = x^T * x’. Now, let's choose x = [1, 0]^T and x’ = [0,1]^T. The proposed kernel becomes: k(x, x’) = x^T * x’ – 1 = 0 – 1 = −1. Now, let’s create a kernel matrix K for these points: K = [k(x, x)   k(x, x’)] = [0 -1]
[k(x’, x)   k(x’, x’)]   [-1 0]. To find the eigenvalues of K, solve the characteristic equation |K – λI| = 0: det(K – λI) = λ^2 – 1 = 0. This equation has eigenvalues λ = 1 and λ = -1. The kernel matrix for this example is not positive semidefinite, as it has at least one negative eigenvalue.

c. This function is a valid kernel. The exponentiation of the squared Euclidean distance is a valid kernel because it corresponds to the Gaussian (radial basis function) kernel. The corresponding kernel matrix K is defined as Kij = exp(||xi – xj||^2), and for any set of input points x1, x2, ..., xn, all principal minors of K are non-negative due to the properties of the exponential function. Therefore, the proposed function satisfies the positive semidefinite condition for kernel matrices, making it a valid kernel.

d. This function is not necessarily a valid kernel. Suppose k1(x, x’) is a valid kernel. However, the proposed function: k(x, x’) = exp(k(x, x’)) – k1(x, x’)may not preserve positive semidefiniteness. For instance, consider the case when x = [1, 1]^T and x’ = [1, -1]^T. In this case, the resulting kernel matrix may not be positive semidefinite, violating the positive semidefinite property. Therefore, the proposed function may not be a valid kernel in general.

2. (15%) One way to construct kernels is to build them from simpler ones. Given three possible "construction rules": assuming $K_1(x,\ x')$ and $K_2(x,\ x')$are kernels then so are
    a. (scaling) $f(x)K_1(x,\ x')f(x'),\ f(x)\ \in\ R$
    b. (sum) $K_1(x,\ x')\ +\ K_2(x,\ x')$
    c. (product) $K_1(x,\ x')K_2(x,\ x')$

    Use the construction rules to build a normalized cubic polynomial kernel:

$$K(x,\ x')\ =\left(1\ +\left(\frac{x}{||x||}\right)^T\left(\frac{x'}{||x'||}\right)\right)^3$$

    You can assume that you already have a constant kernel $K_0(x,\ x') = 1$ and a linear kernel $K_1(x,\ x')\ = x^Tx'$. Identify which rules you are employing at each step.

Step 1:

K2(x, x') = (1 / ||x||) * K1(x, x') * (1 / ||x'||)

This corresponds to the scaling rule (a) with f(x) = 1 / ||x||.

Step 2:

K3(x, x') = K0(x, x') + K2(x, x')

This corresponds to the sum rule (b).

Step 3:

K(x, x') = (K3(x, x'))^3 = (1 + x^T * x' / ||x|| / ||x'||)^3

This corresponds to the product rule (c).

So, the normalized cubic polynomial kernel K(x, x') is constructed using the scaling, sum and product rules.

3. (15%) A social media platform has posts with text and images spanning multiple topics like news, entertainment, tech, etc. They want to categorize posts into these topics using SVMs. Discuss two multi-class SVM formulations: \`One-versus-one\` and \`One-versus-the-rest\` for this task.
   a. The formulation of the method [how many classifiers are required]
   b. Key trade offs involved (such as complexity and robustness).
   c. If the platform has limited computing resources for the application in the inference phase and requires a faster method for the service, which method is better.

a.

One-versus-one:

For N classes, N(N-1)/2 binary classifiers are trained. Each classifier distinguishes between two classes. So, for three classes (A, B, C), they would need three classifiers: AB, BC, and AC.

One-versus-the-rest:

N binary classifiers are trained, each distinguishing one class from the rest. For three classes (A, B, C), they would have three classifiers: A vs. not A, B vs. not B, and C vs. not C.

b.

One-versus-one:

Requires N(N-1)/2 classifiers, so the training complexity is higher. Generally robust as each classifier is trained on a smaller subset of the data, but it might be sensitive to noisy data.

One-versus-the-rest:

Requires N classifiers, which is computationally less expensive. Can be less robust than One-versus-one, especially when there is significant class imbalance.

c.

If the platform has limited computing resources for the inference phase and requires a faster method, One-versus-the-rest might be a better choice. This is because it involves fewer classifiers, making the inference phase computationally less expensive. It is often preferred in practical scenarios where efficiency is a significant concern. However, the choice between them can also depend on other factors, such as the size and balance of the dataset, and the specific characteristics of the classes. It's better to experiment with both approaches and evaluate their performance on the specific data to make an informed decision.