NYCU Introduction to Machine Learning, Homework 4

109550085, 陳欣妤

The screenshot and the figures we provided below are just examples. **The results below are not guaranteed to be correct.** Please make sure your answers are clear and readable, or no points will be given. Please also remember to convert it to a pdf file before submission. You should use English to answer the questions. After reading this paragraph, you can delete this paragraph.

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 = 10):  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 = 1, degree = 3):  0.98
```

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 = 1000, gamma = 0.001):  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. $k(x, x') = k_1(x, x') + exp(x^T x')$

      This is a valid kernel. The sum of two valid kernels is also a valid kernel. The exponential function is always non-negative, and adding a non-negative function to a positive semidefinite matrix preserves positive semidefiniteness.

   b. $k(x, x') = k_1(x, x') - 1$

      This is not necessarily a valid kernel. Subtracting 1 from a valid kernel may result in a function that is not positive semidefinite. For example, if $k_1(x, x') = x^T x'$, then $k(x, x') = x^T x' - 1$ is not positive semidefinite.

Consider $x = [1, 0]$ and $x' = [0, 1]$, then

$$K = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

has eigenvalue -1 and 1, which are not all non-negative.

c. $k(x, x') = exp\left(\|x - x'\|^2\right)$

This is a valid kernel. The exponential function of a valid kernel is also a valid kernel. The squared Euclidean distance is a valid kernel, and the exponential of a valid kernel is still positive semidefinite.

d. $k(x, x') = exp\left(k_1(x, x')\right) - k_1(x, x')$

This is not necessarily a valid kernel. The subtraction of $k_1(x, x')$ from the exponential may lead to a function that is not positive semidefinite. Consider $k_1(x, x') = x^T x'$, then $k(x, x') = exp(x^T x') - x^T x'$ is not positive semidefinite. For example, with $x = [1, 0]$ and $x' = [0, 1]$,

$$K = \begin{bmatrix} 0 & e \\ e & 0 \end{bmatrix}$$

has eigenvalue e and $-e$, which are not all non-negative.

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 $K_1(x, x') = x^T x'$. Identify which rules you are employing at each step.

Step 1: Scaling

$$K_2(x, x') = f(x)K_1(x, x')f(x')$$
$$\text{Let's choose } f(x) = \frac{1}{\|x\|}$$
$$K_2(x, x') = \frac{1}{\|x\|}x^T x' \frac{1}{\|x'\|}$$

Step 2: Sum

$$K_3(x, x') = K_0(x, x') + K_2(x, x')$$
$$K_3(x, x') = 1 + \frac{1}{\|x\|}x^T x' \frac{1}{x'}$$

Step 3: Product

$$K(x, x') = K_3(x, x') \cdot K_3(x, x') \cdot K_3(x, x')$$
$$K(x, x') = (1 + \frac{1}{\|x\|}x^T x' \frac{1}{\|x'\|})^3$$

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.

- One-versus-one

  Formulations:

  In the One-versus-One (OvO) approach, we build a binary classifier for every pair of classes. If there are N classes, we need $\frac{N \cdot (N-1)}{2}$ classifiers. Each classifier is trained on the data from only

two classes and predicts whether a given instance belongs to one class or the other. During inference, each classifier makes a prediction, and the class that receives the most "votes" is the final predicted class.

Trade-offs:

- Complexity: Requires a large number of classifiers, making it computationally expensive during training.
- Robustness: More classifiers can potentially improve performance, especially in scenarios where classes are not well-separated

- One-versus-the-rest

Formulations:

In the One-versus-the-Rest (OvR) approach, we build *N* binary classifiers, each trained to distinguish one class from the rest. For *N* classes, we need *N* classifiers. Each classifier is trained on the data for one class against the data for all other classes. During inference, the classifier with the highest confidence or probability is selected as the predicted class.

Trade-offs:

- Complexity: Requires fewer classifiers compared to OvO, making it computationally more efficient during training.
- Robustness: Can be less sensitive to noisy data as each classifier focuses on distinguishing one class from all others.

OvR is generally more efficient during the inference phase because it requires evaluating only N classifiers, compared to OvO, which requires evaluating $\frac{N \cdot (N-1)}{2}$ classifiers. Therefore, if computational resources are limited during inference, OvR may be a better choice.