NYCU Introduction to Machine Learning, Homework 4

109350008, 張詠哲

Deadline: Dec. 19, 23:59

Part. 1, Coding (50%):

For this coding assignment, you are required to implement some fundamental parts of the Support Vector Machine Classifier using only NumPy. After that, train your model and tune the hyperparameter on the provided dataset and evaluate the performance on the testing data.

(50%) Support Vector Machine

Requirements:

- Implement the *gram_matrix* function to compute the Gram matrix of the given data with an argument **kernel_function** to specify which kernel function to use.
- Implement the *linear_kernel* function to compute the value of the linear kernel between two vectors.
- Implement the *polynomial_kernel* function to compute the value of the polynomial kernel between two vectors with an argument **degree**.
- Implement the *rbf_kernel* function to compute the value of the rbf kernel between two vectors with an argument **gamma**.

Tips:

- Your functions will be used in the SVM classifier from scikit-learn like the code below.

```
svc = SVC(kernel='precomputed')
svc.fit(gram_matrix(X_train, X_train, your_kernel), y_train)
y_pred = svc.predict(gram_matrix(X_test, X_train, your_kernel))
```

- For hyperparameter tuning, you can use any third party library's algorithm to automatically find the best hyperparameter, such as GridSearch. In your submission, just give the best hyperparameter you used and do not import any additional libraries/packages.

Criteria:

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 = 5):  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 = 128, gamma = 0.1):  0.99
```

The following table is the grading criteria for question 2 and 3:

Points	Testing Accuracy
20 points	**0.98 <= acc**
15 points	**0.90 <= acc < 98**
10 points	**0.85 <= acc < 0.90**
5 points	**0.8 <= acc < 0.85**
0 points	**acc < 0.8**

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. $k(x, x') = exp\left(\|x - x'\|^2\right)$
   d. $k(x, x') = exp\left(k_1(x, x')\right) - k_1(x, x')$

Q1.

a). $x^T x'$ is linear kernel (is valid)

and by $K_1(x,x') + K_2(x,x')$ (6.17)

$\exp(K_1(x,x'))$ (6.16)

therefore we can proof that $K(x,x') = K_1(x,x') + \exp(x^T x')$ is valid

b). Let $x = (0,1)^T$, $x' = (1,0)^T$, $K_1(x,x') = x^T x'$ (linear kernel)

then the kernel matrix is

$$K = \begin{bmatrix} k(x,x) & k(x,x') \\ k(x',x) & k(x',x') \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

$\det\left(\begin{bmatrix} -\lambda & -1 \\ -1 & -\lambda \end{bmatrix}\right) = \lambda^2 - 1 = 0 \Rightarrow \lambda = \pm 1$ $(-1 < 0)$

Therefore $K(x,x') = K_1(x,x') - 1$ is not a valid kernel.

c) Take $x = (0,1)^T$, $x' = (1,0)^T$

We have $K = \begin{bmatrix} K(x,x) & K(x,x') \\ K(x',x) & K(x',x') \end{bmatrix} = \begin{bmatrix} e^{\|(0,1)^T\|^2} & e^{\|(1,-1)^T\|^2} \\ e^{\|(1,-1)^T\|^2} & e^{\|(0,1)^T\|^2} \end{bmatrix} = \begin{bmatrix} 1 & e^2 \\ e^2 & 1 \end{bmatrix}$

$\Rightarrow \det\left(\begin{bmatrix} 1-\lambda & e^2 \\ e^2 & 1-\lambda \end{bmatrix}\right) = 0 \Rightarrow (1-\lambda)^2 - e^4 = 0 \Rightarrow (1-e^2-\lambda)(1+e^2-\lambda) = 0$

$\Rightarrow \lambda = (1+e^2)$ or $(1-e^2)$ $(1-e^2 < 0)$

Therefore $K(x,x') = \exp(\|x-x'\|^2)$ is not a valid kernel

d). $\exp(K_1(x,x')) = \lim_{i\to\infty} \left(1 + K_1(x,x') + \frac{1}{2!}(K_1(x,x'))^2 + \cdots + \frac{1}{i!}(K_1(x,x'))^i\right)$

$\therefore \exp(K_1(x,x')) - K_1(x,x') = \lim_{i\to\infty} \left(1 + \frac{1}{2!}(K_1(x,x'))^2 + \cdots + \frac{1}{i!}(K_1(x,x'))^i\right)$

by $c\,k_1(x,x')$ where $c > 0$ is a constant (6.13)

$g(k_1(x,x'))$ where $g(\cdot)$ is a polynomial with nonnegative coefficient (6.15)

$K_1(x,x') + K_2(x,x')$ (6.17)

we can proof that $K(x,x') = \exp(K_1(x,x')) - K_1(x,x')$ is a valid kernel

Summary: a & d is valid kernel

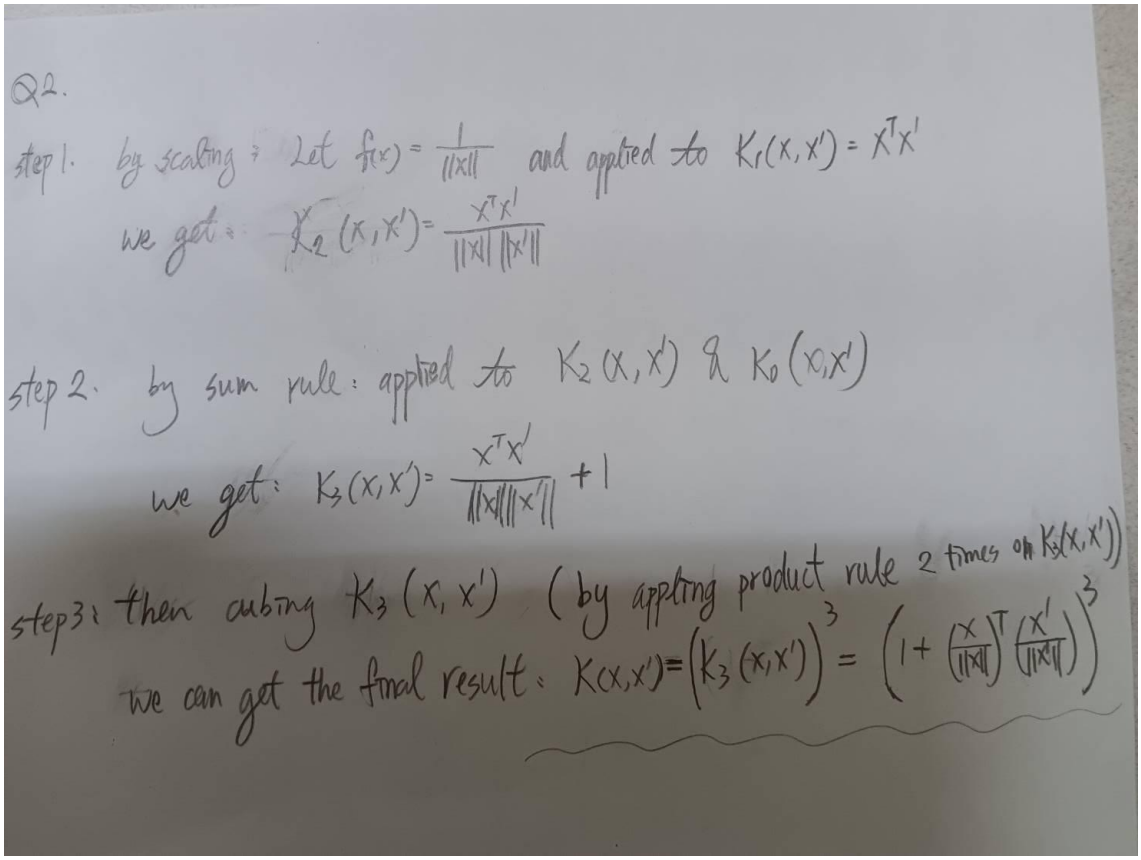
b & c is not valid kernel

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.



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]**

- **One-versus-One (OvO):**
  - **Number of classifiers:**

    For OvO, a binary classifier is trained for each pair of classes. If there are K classes, $\frac{K(K-1)}{2}$ classifiers are required.
  - **Training:**

    For each pair of classes, a binary SVM is trained to distinguish between instances of one class and instances of the other. During classification, each binary classifier "votes" for a class, and the class with the most votes is selected.

- **One-versus-the-Rest (OvR):**
  - **Number of classifiers:**

    For OvR, K binary classifiers are trained. (K is the number of classes)
  - **Training:**

    Each binary classifier is trained to distinguish instances of one class from the instances of all other classes. During classification, the class associated with the classifier that gives the highest decision values is selected as the final prediction.

b. **Key trade offs involved (such as complexity and robustness).**

- **One-versus-One (OvO):**
  - **Complexity:**

    OvO requires more classifiers ($\frac{K(K-1)}{2}$), making it computationally expensive for a large number of classes.
  - **Training Time:**

    Training time scales with the number of classifiers, which can be a significant factor for large datasets.
  - **Robustness:**

    OvO maybe can be more robust in situations where the binary classifiers need to be tailored to specific class pairs.

- **One-versus-the-Rest (OvR):**
  - **Complexity:**
    OvR requires fewer classifiers (K), making it computationally more efficient than OvO.
  - **Training Time:**
    Training time is generally faster compared to OvO due to fewer classifiers (K).
  - **Robustness:**
    OvR can be less sensitive to imbalances in class sizes as each classifier is focused on distinguishing one class from the rest.

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.**

If the platform has limited computing resources for the application in the inference phase and requires a faster method, One-versus-the-Rest (OvR) may be a better choice. Because OvR typically requires fewer classifiers (K, which K is the number of classes) than OvO ($\frac{K(K-1)}{2}$), leading to faster inference times. And the training time of it is generally faster compared to OvO, which can be crucial in scenarios where resources are limited. OvR tends to scale better with a large number of classes, making it more suitable for more classes scenarios.