NYCU Introduction to Machine Learning, Homework 4

110652021, 龔大承

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%):

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:

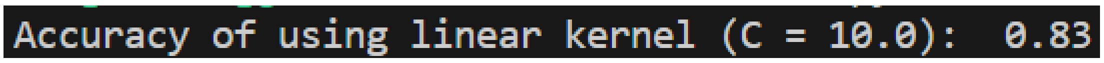
- Implement the *gram_matrix* function to compute the Gram matrix of the given data with an argument **kernel_function** to specify which kernel function to use.
- Implement the *linear_kernel* function to compute the value of the linear kernel between two vectors.
- Implement the *polynomial_kernel*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

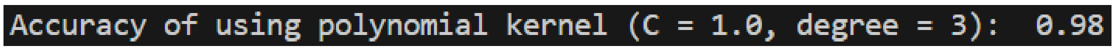
Criteria:

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



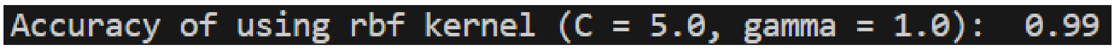
```
Accuracy of using linear kernel (C = 10.0):  0.83
```

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 = 1.0, 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 = 5.0, gamma = 1.0):  0.99
```

Part. 2, Questions (50%):

(20%) Given a valid kernel k1(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.

(15%) One way to construct kernels is to build them from simpler ones. Given three possible "construction rules": assuming K1(x, x') and K2(x, x')are kernels then so are

a. (scaling) f(x)K1(x, x')f(x'), f(x)R
b. (sum) K1(x, x')+K2(x, x')
c. (product) K1(x, x')K2(x, x')

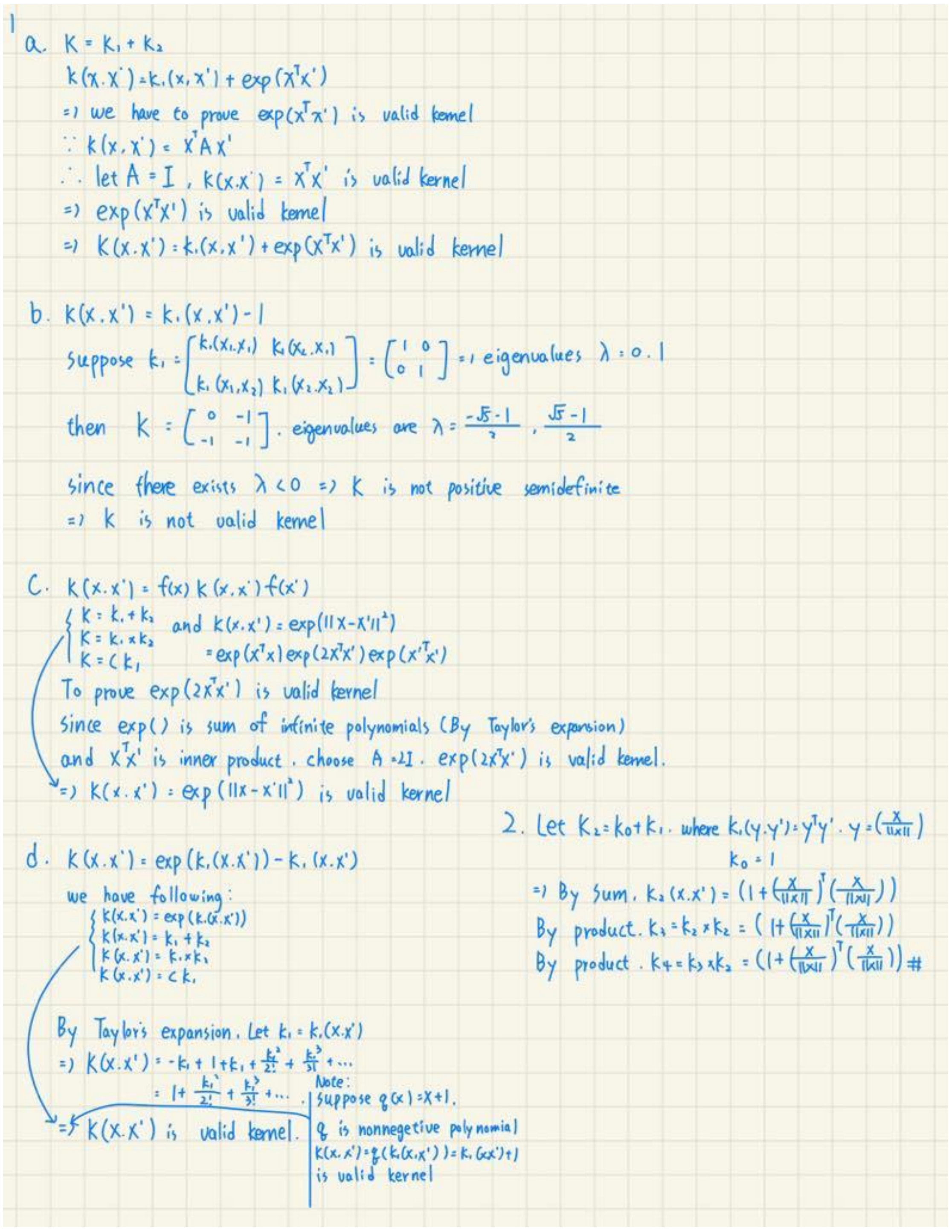
Use the construction rules to build a normalized cubic polynomial kernel:

K(x, x')=(1+x / ||x||Tx' / ||x'||)^3

You can assume that you already have a constant kernel K0(x, x') = 1 and a linear kernel K1(x, x')=xTx'. Identify which rules you are employing at each step.

//

This is answer for question 1 and 2



1

a. $K = K_1 + K_2$

$k(x.x') = k_1(x,x') + \exp(x^T x')$

$\Rightarrow$ we have to prove $\exp(x^T x')$ is valid kernel

$\because k(x,x') = x^T A x'$

$\therefore$ let $A = I$, $k(x.x') = x^T x'$ is valid kernel

$\Rightarrow \exp(x^T x')$ is valid kernel

$\Rightarrow K(x.x') = k_1(x,x') + \exp(x^T x')$ is valid kernel

b. $K(x.x') = k_1(x.x') - 1$

suppose $k_1 = \begin{bmatrix} k_1(x_1.x_1) & k_1(x_2.x_1) \\ k_1(x_1,x_2) & k_1(x_2.x_2) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \Rightarrow$ eigenvalues $\lambda = 0.1$

then $K = \begin{bmatrix} 0 & -1 \\ -1 & -1 \end{bmatrix}$. eigenvalues are $\lambda = \frac{-\sqrt{5}-1}{2}, \frac{\sqrt{5}-1}{2}$

since there exists $\lambda < 0 \Rightarrow K$ is not positive semidefinite

$\Rightarrow K$ is not valid kernel

c. $K(x.x') = f(x)\, k(x.x')\, f(x')$

$$\begin{cases} K = k_1 + k_2 \\ K = k_1 \times k_2 \\ K = c k_1 \end{cases}$$

and $k(x.x') = \exp(\|x - x'\|^2)$

$= \exp(x^T x) \exp(2x^T x') \exp(x'^T x')$

To prove $\exp(2x^T x')$ is valid kernel

Since $\exp()$ is sum of infinite polynomials (By Taylor's expansion)

and $x^T x'$ is inner product. choose $A = 2I$. $\exp(2x^T x')$ is valid kernel.

$\Rightarrow K(x.x') = \exp(\|x - x'\|^2)$ is valid kernel

d. $K(x.x') = \exp(k_1(x.x')) - k_1(x.x')$

we have following:

$$\begin{cases} K(x.x') = \exp(k_1(x.x')) \\ K(x.x') = k_1 + k_2 \\ K(x.x') = k_1 \times k_1 \\ K(x.x') = c k_1 \end{cases}$$

By Taylor's expansion. Let $k_1 = k_1(x.x')$

$\Rightarrow K(x.x') = -k_1 + 1 + k_1 + \frac{k_1^2}{2!} + \frac{k_1^3}{3!} + \dots$

$= 1 + \frac{k_1^2}{2!} + \frac{k_1^3}{3!} + \dots$

$\Rightarrow K(x.x')$ is valid kernel.

Note:

suppose $g(x) = x+1$.

g is nonnegetive polynomial

$K(x,x') = g(k_1(x,x')) = k_1(x,x') + 1$

is valid kernel

2. Let $K_2 = k_0 + k_1$. where $k_1(y.y') = y^T y'$. $y = \left(\frac{x}{\|x\|}\right)$

$k_0 = 1$

$\Rightarrow$ By sum. $k_2(x.x') = \left(1 + \left(\frac{x}{\|x\|}\right)^T \left(\frac{x}{\|x\|}\right)\right)$

By product. $k_3 = k_2 \times k_2 = \left(1 + \left(\frac{x}{\|x\|}\right)^T \left(\frac{x}{\|x\|}\right)\right)$

By product. $k_4 = k_3 \times k_2 = \left(1 + \left(\frac{x}{\|x\|}\right)^T \left(\frac{x}{\|x\|}\right)\right)$ #

//

(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 (OvO) Formulation:

a. Formulation:

For k classes, the one-versus-one approach trains $C=k(k-1)/2$ binary classifiers, where each classifier is trained to distinguish between a pair of classes. During inference, a vote is taken from each classifier, and the class with the most votes is chosen as the final prediction.

b. Key Trade-offs:

- **Classifiers Required:**$C=k(k-1)/2$ classifiers are needed, which can be computationally expensive for a large number of classes.
- **Complexity:** Training C classifiers can be computationally intensive and may require more memory, especially as the number of classes increases.
- **Robustness:** OvO is less sensitive to imbalanced datasets since each classifier is trained on a balanced subset of the data (two classes).

c. Limited Computing Resources:

- **Suitability:** OvO may become computationally expensive as the number of classes increases due to the quadratic increase in the number of classifiers. If computing resources are limited, this method might be less efficient for inference.

One-versus-the-Rest (OvR) Formulation:

a. Formulation:

In the one-versus-the-rest approach, k classifiers are trained, each treating one class as the positive class and the rest as the negative class. During inference, the class associated with the classifier yielding the highest confidence score is chosen as the predicted class.

b. Key Trade-offs:

- **Classifiers Required:** k classifiers are needed, which is computationally more efficient than OvO.
- **Complexity:** Training k classifiers is generally less computationally intensive

than OvO.

- **Robustness:** OvR can be sensitive to imbalanced datasets, especially if one class significantly outnumbers the others.

c. Limited Computing Resources:

- **Efficiency:** OvR is computationally more efficient for inference, as only *k* classifiers need to be evaluated. If computing resources are limited, this method might be more suitable for quick predictions.

Conclusion:

- **OvO vs. OvR:** The choice between OvO and OvR depends on factors such as the number of classes, dataset balance, and available computational resources.
- **Limited Resources:** If the platform has limited computing resources for the application in the inference phase and requires a faster method, OvR may be preferred due to its linear dependence on the number of classes, making it more computationally efficient during inference.