NYCU Introduction to Machine Learning, Homework 4

110550071, 田松翰

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:

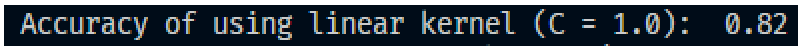
- Implement the *gram_matrix* function to compute the Gram matrix of the given data with an argument **kernel_function** to specify which kernel function to use.
- Implement the *linear_kernel* function to compute the value of the linear kernel between two vectors.
- Implement the *polynomial_kernel*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

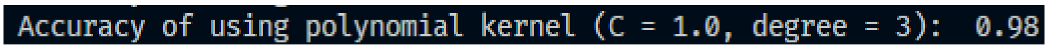
Criteria:

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 = 1.0):  0.82
```

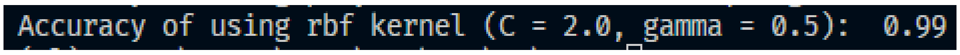
2. (20%) Tune the hyperparameters of the *polynomial_kernel*. Show the accuracy score of the testing data using *polynomial_kernel* and the hyperparameters you used.



```
Accuracy of using polynomial kernel (C = 1.0, degree = 3):  0.98
```

(next page)

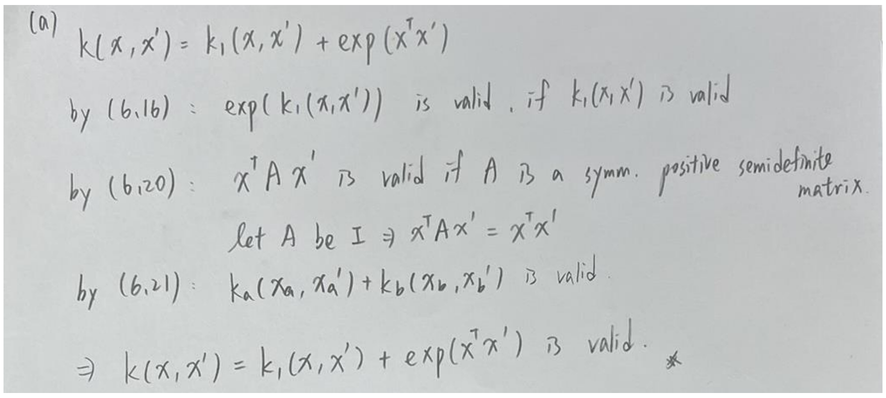
3. (20%) Tune the hyperparameters of the *rbf_kernel*. Show the accuracy score of the testing data using *rbf_kernel* and the hyperparameters you used.



```
Accuracy of using rbf kernel (C = 2.0, gamma = 0.5):  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. $k(x, x') = k_1(x, x') + \exp(x^T x')$



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

by (6.16) : $\exp(k_1(x,x'))$ is valid , if $k_1(x,x')$ is valid

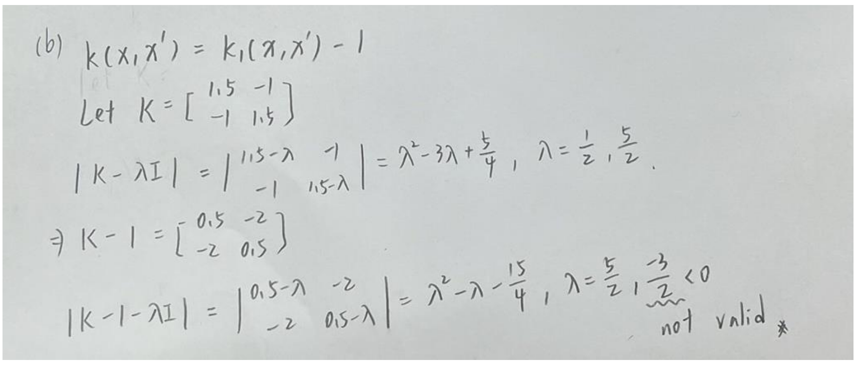
by (6.20) : $x^T A x'$ is valid if A is a symm. positive semidefinite matrix.

let A be $I \Rightarrow x^T A x' = x^T x'$

by (6.21) : $k_a(x_a, x_a') + k_b(x_b, x_b')$ is valid.

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

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



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

Let $K = \begin{bmatrix} 1.5 & -1 \\ -1 & 1.5 \end{bmatrix}$

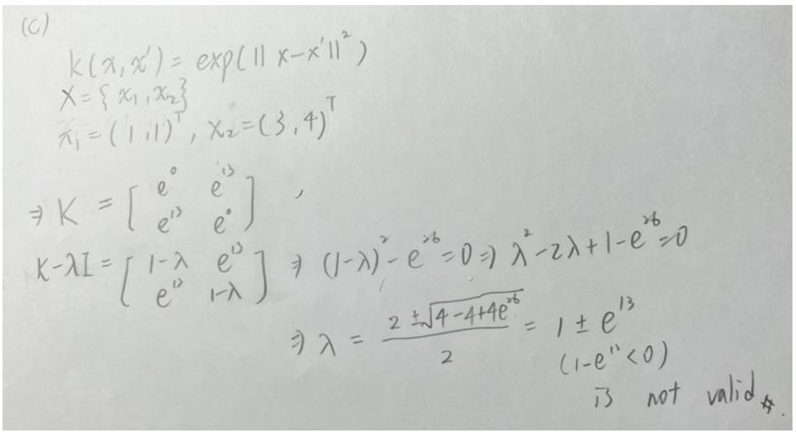
$|K - \lambda I| = \begin{vmatrix} 1.5-\lambda & -1 \\ -1 & 1.5-\lambda \end{vmatrix} = \lambda^2 - 3\lambda + \frac{5}{4}, \quad \lambda = \frac{1}{2}, \frac{5}{2}.$

$\Rightarrow K - 1 = \begin{bmatrix} 0.5 & -2 \\ -2 & 0.5 \end{bmatrix}$

$|K - 1 - \lambda I| = \begin{vmatrix} 0.5-\lambda & -2 \\ -2 & 0.5-\lambda \end{vmatrix} = \lambda^2 - \lambda - \frac{15}{4}, \quad \lambda = \frac{5}{2}, \frac{-3}{2} < 0$

not valid *

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



(c)

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

$X = \{x_1, x_2\}$

$x_1 = (1,1)^T, \; x_2 = (3,4)^T$

$\Rightarrow K = \begin{bmatrix} e^0 & e^{13} \\ e^{13} & e^0 \end{bmatrix}$,

$K - \lambda I = \begin{bmatrix} 1-\lambda & e^{13} \\ e^{13} & 1-\lambda \end{bmatrix} \Rightarrow (1-\lambda)^2 - e^{26} = 0 \Rightarrow \lambda^2 - 2\lambda + 1 - e^{26} = 0$

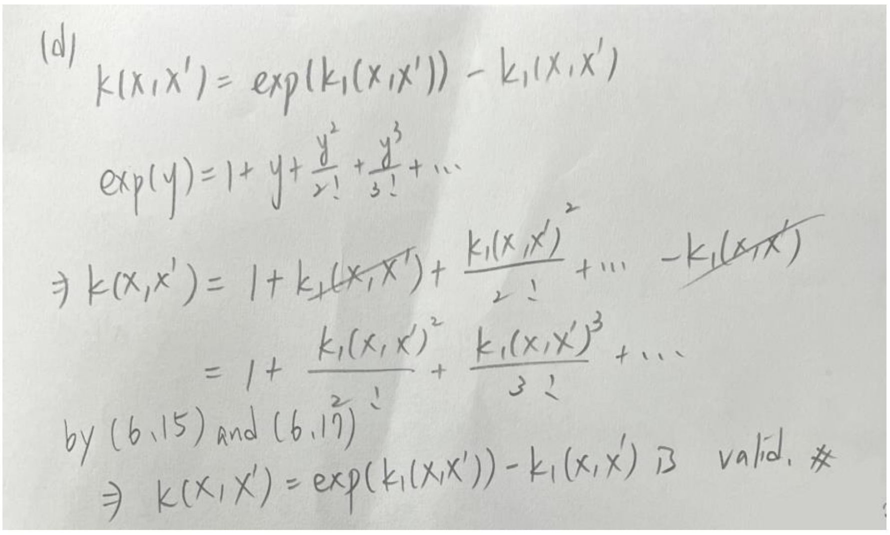
$\Rightarrow \lambda = \frac{2 \pm \sqrt{4 - 4 + 4e^{26}}}{2} = 1 \pm e^{13}$

$(1 - e^{13} < 0)$

is not valid \#.

(next page)

d. k (x, x') = exp (k1(x, x')) – k1(x, x')



(d)

$k(x,x') = \exp(k_1(x,x')) - k_1(x,x')$

$\exp(y) = 1 + y + \frac{y^2}{2!} + \frac{y^3}{3!} + \cdots$

$\Rightarrow k(x,x') = 1 + \cancel{k_1(x,x')} + \frac{k_1(x,x')^2}{2!} + \cdots \cancel{-k_1(x,x')}$

$= 1 + \frac{k_1(x,x')^2}{2!} + \frac{k_1(x,x')^3}{3!} + \cdots$

by (6.15) and (6.17)

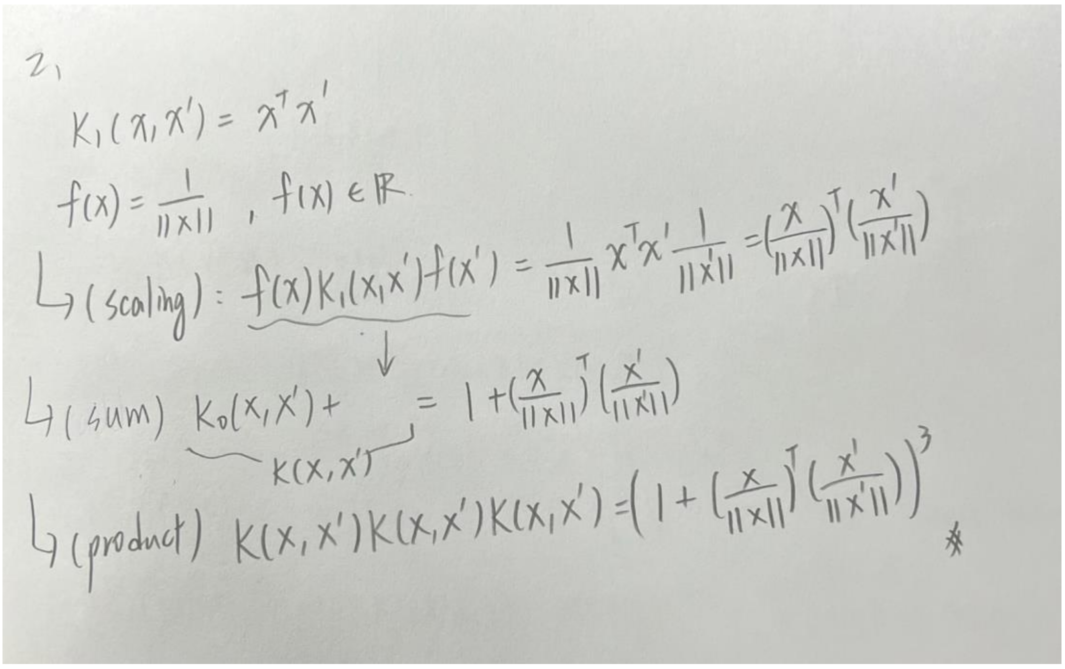
$\Rightarrow k(x,x') = \exp(k_1(x,x')) - k_1(x,x')$ is valid. ※

2. (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:

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

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



2,

$K_1(x,x') = x^T x'$

$f(x) = \frac{1}{\|x\|}$, $f(x) \in \mathbb{R}$

↳ (scaling): $\underbrace{f(x)K_1(x,x')f(x')} = \frac{1}{\|x\|} x^T x' \frac{1}{\|x'\|} = \left(\frac{x}{\|x\|}\right)^T \left(\frac{x'}{\|x'\|}\right)$

↓

↳ (sum) $\underbrace{K_0(x,x') + \quad}_{K(x,x')} = 1 + \left(\frac{x}{\|x\|}\right)^T \left(\frac{x'}{\|x'\|}\right)$

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

※

(next page)

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: For n classes, it needs $C^n{}_2$=N(N-1)/2 classifiers.
      One-versus-the-rest: For n classes, it needs N classifiers.

   b. Key trade offs involved (such as complexity and robustness).
      1v1 requires more classifiers, so it can be more robust in the presence of noisy data. However, it takes more time to train and predict due to the larger number.

   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-the-rest will be the better one. It's computationally more efficient in both training and reference phases compared to one-versus-one. The reduced number of classifiers can lead to faster predictions, making it suitable for scenarios with limited resources.