NYCU Introduction to Machine Learning, Homework 4

110550168, 賴御安

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 = 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 = 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 = 1, gamma = 2):  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. $k(x, x') = k_1(x, x') + exp(x^T x')$

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

$\Rightarrow exp(x^T x)$

if $\phi(x) = x$ is basic function

$k_2(x, x') = x^T x$ is valid kernel

$\Rightarrow$ by ppt : 6.kernel_method 6.16 $\rightarrow exp(x^T x)$ is valid kernel

$\Rightarrow$ by 6.17, two valid kernel can sum up and become valid kernel

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

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

1. (b) $k(x, x^T) = k_1(x, x') - 1$

$\Rightarrow$ assume $X = \{x_1, x_2\}$ $\quad x_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ $\quad x_2 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$

$$K = \begin{bmatrix} 10 & 14 \\ 14 & 18 \end{bmatrix}$$

solve the eigen value $\det(K - \lambda I) = 0$

$\lambda = 2(7 \pm \sqrt{53})$ $\Rightarrow$ K is not positive semidefinite

$\Rightarrow$ not valid kernel

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

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

$\Rightarrow$ assume $X = \{x_1, x_2\}$, $x_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$, $x_2 = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$

$$K = \begin{bmatrix} exp(0) & exp(5) \\ exp(5) & exp(0) \end{bmatrix}$$

solve the eigen value $\det(K - \lambda I) = 0$

$\Rightarrow \lambda = 1 \pm exp(5)$ $\Rightarrow$ K is not positive semidefinite.

$\Rightarrow$ not valid kernel

(next page)

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

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

$\Rightarrow$ the taylor series of $exp(k_1(x, x'))$ is

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

$\Rightarrow$ sum up with $-k_1(x, x')$

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

by ppt 6. kernel_method (6.18)

we can know that $(k_1(x, x'))^d$, $d = 2, 3, \cdots$

are all valid kernel

and by 6.13 $\frac{(k_1(x, x'))^d}{d!}$, $d = 2, 3, \cdots$

are all valid kernel

and by 6.16, we can sum up all and have valid kernel

for constant 1, $k_2(x, x')$, $X = 1$, $K = 1$

$\det(K - \lambda I) = 0$

$\lambda = 1$, K is positive semidefinite

by 6.16, we can know that

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

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 $K_1(x,\ x')\ = x^Tx'$. Identify which rules you are employing at each step.

DATE . . NO.

2. ① by scaling $f(x) = \frac{1}{||x||}$ the linear kernel $K_1(x,x')=x^Tx'$

$\Rightarrow x^Tx' \Rightarrow \frac{1}{||x||}x^Tx'\frac{1}{||x'||} = \left(\frac{x}{||x||}\right)^T\left(\frac{x'}{||x||}\right)$

② by sum part ① and the constant kernel $K_0(x,x')=1$

$\Rightarrow 1+\left(\frac{x}{||x||}\right)^T\left(\frac{x'}{||x||}\right)$ is a valid kernel

③ by product part ② with part ②

$\Rightarrow \left(1+\left(\frac{x}{||x||}\right)^T\left(\frac{x'}{||x||}\right)\right)^2$ is a valid kernel

④ by product part ③ with part ②

$\Rightarrow \left(1+\left(\frac{x}{||x||}\right)^T\left(\frac{x'}{||x||}\right)\right)^3$ is finally built. #

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.

3. (a) For one-versus-one , it classified a pair of classes so if we need to classify n classes , we need $\frac{n(n-1)}{2}$ classifiers

For one-versus-rest . it classified one class from others , so if we need to classify n classes , we only need n classifiers

(next page)

DATE . . NO.

3. (b) Key trade off:

Complexity: For one-versus-one, the complexity is high if the number of the classes is large, since we need to calculate more classifiers, for n classes to n+1 classes, we need n+1 classifiers.
But for one-versus-rest, the complexity is more efficient. for n classes to n+1 classes, we need 1 classifiers only.

Robust: One-versus-one is stronger since the classifiers compare between a pair of classes, but for one-versus-rest the classifiers need to seperate one class from other classes.

3. (c) By the discussion above, I think that one-versus-vest is better. The first reason is that the complexity is smaller, so that we can use limit resources and get a good model. Also, we require a fast method but not a stronger method, so one-versus-rest perfectly fit the request.