NYCU Introduction to Machine Learning, Homework 4

110550167. 侯博軒

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 = 4):  0.83
```

```
54      C_ = 4
55      svc = SVC(kernel='precomputed', C=C_)
56      svc.fit(gram_matrix(X_train, X_train, linear_kernel), y_train)
```

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 = 0.4, degree = 3):  0.99
```

```
63      C_ = 0.4
64      svc = SVC(kernel='precomputed', C=C_)
65      svc.fit(gram_matrix(X_train, X_train, polynomial_kernel), y_train)
```

```
10  degree_ = 3 # degree_ should be an integer.
```

The constant to construct the polynomial k is 1.

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 = 0.1, gamma = 0.3):  0.99
```

```
71      C_ = 0.1
72      svc = SVC(kernel='precomputed', C=C_)
73      svc.fit(gram_matrix(X_train, X_train, rbf_kernel), y_train)
```

```
11  gamma_ = 0.3
```

Part. 2, Questions (50%):

1. (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.
   a. k(x,x')=k1 (x,x')+exp(xTx')
   b. kx, x'= k1(x, x')-1
   c. kx, x'= exp‖x-x'‖²
   d. kx, x'= exp(k1(x,x'))-k1(x, x')

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

$= k_1(x, x') + \exp(k'(x, x'))$ ... (6.16)

$k'(x, x') = (x^T \cdot x')$

$= k_1(x, x') + k_2(x, x')$ ... (6.17)

$= \phi(x)^T \phi(x')$

$= k_3(x, x')$ and k_3 is a valid kernel ∎

ϕ is the one dimension linear basis function, so $k(x, x')$ is a valid kernel

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

suppose $X = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, $k_1(x, x') = x^T x'$, $K_{22} = \begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix}$

$= \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} -2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$

$\Rightarrow \lambda_k = -2, 0$, K is not positive semidefinite

$k(x, x')$ is not a valid kernel

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

Suppose $X = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$, $K = \begin{bmatrix} 1 & e^2 \\ e^2 & 1 \end{bmatrix}$

by solving $\det(K - \lambda I) = 0$, we acquire $\lambda = 1 \pm e^2$

where the solution $\lambda = 1 - e^2 < 0$

$\Rightarrow$ K is not positive semidefinite, $k(x, x')$ is not a valid kernal

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

$= \left[1 + k_1(x, x') + \frac{k_1(x, x')^2}{2!} + \cdots\right] - k_1(x, x')$

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

by combining (6.13) (6.17) (6.18) we can prove that $K_2 = \sum_{n=2}^{\infty} \frac{k_1(x, x')^n}{n!}$ and K_2 is a valid kernal.

$= 1 + K_2(x, x')$

1 is a constant ≥ 0, so it is a valid kernel,

by applying (6.17) we can prove that $k(x, x')$ is a valid kernel.

(next page)

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) is 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:

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

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.

2. first start from the linear kernel

$k_1(x, x') = x^T x'$

apply the scaling rule

$k_1'(x, x') = f(x)(x^T x')f(x')$, $f(c) = \frac{1}{\|c\|}$, $f(c) \in \mathbb{R}$

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

apply the sum rule

$k_2(x, x') = k_0(x, x') + k_1'(x, x')$

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

apply the product rule

$k(x, x') = k_2(x, x') \times k_2(x, x') \times k_2(x, x')$

$= \left(1 + \left(\frac{x}{\|x\|}\right)^T \left(\frac{x'}{\|x'\|}\right)\right)^3$ cubic polynomial kernel

(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]

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.

a. One-versus-One: For a One-versus-one classifier if there are N class, $N * (N-1)/2$ classifiers will be needed. Each trained to distinguish between a pair of classes and use majority voting to decide the final predict class.

One-versus-the-Rest: For a One-versus-one classifier if there are N class, N classifiers will be needed. Each trained to distinguish a class against the rest.

b. One-versus-One: Requires more classifiers, especially as the number of classes increases. Training time scales with the square of the number of classes. Complexity for training is high. On the other hand, more robust because each classifier only needs to distinguish between two classes, which might result in better performance for each binary classification task.

One-versus-the-Rest: Requires fewer classifiers compared to One-versus-One. Training time scales linearly with the number of classes. Complexity and computation requirement is less than One-versus-One method. However, may be less robust than One-versus-One as each classifier needs to handle multiple classes, potentially leading to imbalanced datasets for individual classifiers.

c. If the platform has limited computing resources and requires a faster method for the inference phase, the better choice would often be One-versus-the-Rest. One-versus-the-Rest method involves training fewer classifiers compared to One-versus-One method, making it computationally more efficient during both training and inference.

In the inference phase, a new data point is passed through all classifiers, and the class with the highest confidence score is chosen. Therefore, One-versus-the-Rest usually requires less computation as it involves fewer classifiers.