NYCU Introduction to Machine Learning, Homework 4

110705013, 沈昱宏

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 = 3.0):  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 = 0.1, degree = 4):  0.99
```

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 = 3.0, gamma = 1.0):  0.98
```

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

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

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

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

Ans: with the following rules

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}') \tag{6.13}$$
$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \tag{6.14}$$
$$k(\mathbf{x}, \mathbf{x}') = q(k_1(\mathbf{x}, \mathbf{x}')) \tag{6.15}$$
$$k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}')) \tag{6.16}$$
$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \tag{6.17}$$
$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') \tag{6.18}$$
$$k(\mathbf{x}, \mathbf{x}') = k_3(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')) \tag{6.19}$$
$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}' \tag{6.20}$$
$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}_a') + k_b(\mathbf{x}_b, \mathbf{x}_b') \tag{6.21}$$
$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}_a')k_b(\mathbf{x}_b, \mathbf{x}_b') \tag{6.22}$$

a.

let $k_2(X, X') = X^T X'$

$k_2(X,X') = \phi(X)\phi(X')$ is a valid kernel with feature map $x \to [x]$

let $k_3(X,X') = \exp(k_2(X,X'))$

$k_3(X,X')$ is a valid kernel (6.16)

$k(X,X') = k_1(X,X') + k_3(X,X')$ is a valid kernel (6.17)

b.

consider case $X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ ↪ 2 data points (1D) and $k_1(X,X') = X^T X'$ (valid)

Gram matrix for $k(X,X')$: $K = \begin{bmatrix} x_1^2 - 1 & x_1x_2 - 1 \\ x_1x_2 - 1 & x_2^2 - 1 \end{bmatrix}$

assume $k(X,X')$ is valid

eigen values of $K \geq 0 \quad \forall (x_1, x_2) \in R^2$

consider case $X_1 = X_2 = 0$, $K = \begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix}$

$\det\left(\begin{bmatrix} -1-\lambda & -1 \\ -1 & -1-\lambda \end{bmatrix}\right) = 0$, $(1+\lambda)^2 - 1 = 0$, $\lambda = 0$ or $\underline{\underline{-2}}$

$\Rightarrow$ contradict, $k(X,X')$ invalid

c.

consider case $X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ and $k_1(X,X') = X'/$

Gram matrix for $k(X,X') = \begin{bmatrix} e^{\|x_1 - x_1\|^2} & e^{\|} \\ e^{\|x_2 - x_1\|^2} & e^{\|x_2 - x} \end{bmatrix}$

let $X = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $k(X,X') = \begin{bmatrix} 1 & e \\ e & 1 \end{bmatrix}$

$(1-\lambda)^2 - e^2 = 0$, $1 - \lambda = \pm e$, $\lambda = \underline{\underline{1-e}}$ or $1+e$

$\Rightarrow$ $k(X,X')$ is not a valid kernel <0

d.

$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} \ldots$

$\exp(k(X,X')) - k(X,X')$

$= 1 + \cancel{k(X,X')} + \frac{k(X,X')^2}{2!} + \frac{k(X,X')^3}{3!} \ldots - \cancel{k(X,X')}$

$= 1 + \frac{k(X,X')^2}{2!} + \frac{k(X,X')^3}{3!} \ldots$

according to 6.18 and 6.13 , $\frac{k(X,X')^2}{2!} + \frac{k(X,X')^3}{3!} \ldots$ is valid kernel

and 1 is also valid, and according to 6.17 , $k(X,X')$ is a 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 $K2(x, x')$ 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 $K_0 = 1$ and a linear kernel $K_1(x, x')$. Identify which rules you are employing at each step.

Ans:

let $f(x) = \frac{1}{||x||}$

$K_2(x,x') = \underline{f(x)\, K_1(x,x')\, f(x')}$ ... a. (scaling)

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

$K_3(x,x') = \underline{K_0(x,x') + K_2(x,x')} = 1 + \left(\frac{x}{||x||}\right)^T \left(\frac{x'}{||x'||}\right)$ ... b. (sum)

$K(x,x') = \underline{\left(K_3(x,x')\, K_3(x,x')\right) K_3(x,x')}$ ... c. (product) ×2

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

Then $K(x, x')$ is constructed.

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.

Ans:

a. Let the number of categories equal to n, n*(n-1)/2 classifiers are required when using one-versus-one method, and n-1 classifiers are required when using one-versus-the-rest method.
b. The space & training time complexity of one-versus-one method is higher than the one-versus-the-rest method because the model is number of model is higher (when n is big enough), so it requires more time and space to train and store them. As for the performance, one-versus-one is generally higher. Considering an imbalanced dataset with class A, B, C having 98% of A, 1% of B and 1% of C. Since SVM will accept some data point misclassified during training (depend on C), when you are training B v.s. the rest, the C might also be included, causing B and C unable to correctly classified. However, when using one-versus-one, we will train a classifier of B and C, so there won't be any problem classifying B and C when using one-versus-one in this case, and that is why I think one-versus-one generally perform better.
c. One-versus-the-rest is better in this case. Since you have limited computation resource, you would better choose the one with the lower memory requirement, which is one-versus-the-rest. As for running time, if you use one-versus-one, you may run the n*(n-1)/2 classifiers and do voting, which more time-consuming than running a maximum of n-1 classifiers when using one-versus-the-rest.