NYCU Introduction to Machine Learning, Homework 4

110550126 曾家祐

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):  0.82
```
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, 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 = 3, gamma = 0.4):  0.99
```

Part. 2, Questions (50%):

1. (20%) Given a valid kernel $k_1(x, x^{\prime})$, prove that the following proposed functions are or are not valid kernels. If one is not a valid kernel, give an example of $k(x, x^{\prime})$ that the co rresponding K is not positive semidefinite and shows its eigenvalues.

   a. $k(x, x^{\prime}) = k_1(x, x^{\prime}) + exp\,(x^T x^{\prime})$

   Suppose that ϕ(x) = x → it' s kernal function is $x^T x^{\prime}$ → it is valid kernal function and the exponential tern of a valid kernal will bbe valid kernal → $exp\left(x^T x^{\prime}\right)$ is valid kernal.

   since $k_1(x, x^{\prime})$ and $exp\,(x^T x^{\prime})$ are both valid kernal, by construction rules of sum → $k(x, x^{\prime}) = k_1(x, x^{\prime}) + exp\,(x^T x^{\prime})$ is valid kernal.

   b. $k\left(x, x^{\prime}\right) = k_1\left(x, x^{\prime}\right) - 1$

   $k\left(x, x^{\prime}\right)$ may not be a valid kernal.

   consider x = $[1\ 0]^T$, x' $=[0\ 1]^T$

   → K = $\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$ and the eigenvalue = 1 and -1

   → K is not positive semidefinite

   → This kernel is not an valid kernel

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

   $k\left(x, x^{\prime}\right)$ may not be a valid kernal.

   consider x = $[1\ 0]^T$, x' $=[3\ 0]^T$

➔ K = $\begin{pmatrix} 1 & \exp(4) \\ \exp(4) & 1 \end{pmatrix}$ and the eigenvalue = 1+exp(4) and 1-exp(4)

➔ K is not positive semidefinite

➔ This kernel is not an valid kernel

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

By Taylor' s expansions $exp\left(k_1\left(x, x^{'}\right)\right) = 1 + k_1\left(x, x^{'}\right) + \frac{k_1\left(x,x^{'}\right)^2}{2} \ldots + \frac{k_1\left(x,x^{'}\right)^n}{n!} \rightarrow exp\left(k_1\left(x, x^{'}\right)\right) - k_1\left(x, x^{'}\right) = 1 + \frac{k_1\left(x,x^{'}\right)^2}{2} \ldots + \frac{k_1\left(x,x^{'}\right)^n}{n!}$

1 is valid kernal and $k_1\left(x, x^{'}\right) \rightarrow \frac{k_1\left(x,x^{'}\right)^2}{2}, \cdots \frac{k_1\left(x,x^{'}\right)^n}{n!}$ are all valid kernal.

since the every term on the right side is valide kernal $\rightarrow k\left(x, x^{'}\right)$ are also valid kernal.

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 th en 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:

$$K(x,x^{\prime}) = \left(1 + \left(\frac{x}{||x||}\right)^T \left(\frac{x^{\prime}}{||x^{\prime}||}\right)\right)^3$$

You can assume that you already have a constant kernel $K_0(x,x^{\prime}) = 1$ and a linear kerne l $K_1(x,x^{\prime}) = x^T x^{\prime}$. Identify which rules you are employing at each step.

et $k_1\left(x,x^{\prime}\right) = 1$ and $k_2\left(x,x^{\prime}\right) = x^T x^{\prime}$

1. scaling : f(x) = $1/(x^T x)$ → from k_2 and f(x)

-> we get $\frac{x^T x^{\prime}}{\sqrt{x^T x}\sqrt{x^{\prime T} x^{\prime}}}$ → $\left(\frac{x}{||x||}\right)^T \left(\frac{x^{\prime}}{||x^{\prime}||}\right)$

2.sum: we add k_1 and the kernel we construct from scaling

->$1 + \left(\frac{x}{||x||}\right)^T \left(\frac{x^{\prime}}{||x^{\prime}||}\right)$

3. product: power is a kind of power and from the kernal from sum rule we get we can product it self another two times

→$(1 + \left(\frac{x}{||x||}\right)^T \left(\frac{x^{\prime}}{||x^{\prime}||}\right))*(1 + \left(\frac{x}{||x||}\right)^T \left(\frac{x^{\prime}}{||x^{\prime}||}\right))*1 + \left(\frac{x}{||x||}\right)^T \left(\frac{x^{\prime}}{||x^{\prime}||}\right))$

→$\left(1 + \left(\frac{x}{||x||}\right)^T \left(\frac{x^{\prime}}{||x^{\prime}||}\right)\right)^3$

We get the normalized cubic polynomial kernel by the construction rules.

3. (15%) A social media platform has posts with text and images spanning multiple topics lik e news, entertainment, tech, etc. They want to categorize posts into these topics using SV Ms. 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]

1. One-versus-one(OvO):

For N classes, we need to train N*(N-1)/2 binaryclassifiers. Each classifier is for each pair of classes, (ex: for three class A,B,C we need to train Avs B, B vs

C, A vs C. During prediction, each classifier votes for a class, and the class with the most votes is assigned.

2. One-versus-the-rest(OvR):

For N classes, we need to train N classifiers. one for each class against the rest of the classes, , (ex: for three class A,B,C we need to train Avs non-A, B vs non-B, C vs C). During prediction, the class with the highest confidence score from any of the N classifiers is assigned.

b. Key trade offs involved (such as complexity and robustness).

1. One-versus-one(OvO):

Num Number of Classifiers:

K*(K-1)/2, which can be computationally expensive if there are many classes.

Training Time:

Training K*(K-1)/2, but each classifier will be faster , since for each classifier the dataset is smaller.

Prediction Time:

Prediction involves comparing the outputs K*(K-1)/2 of classifiers, which can be slower than OvR for large K.

2. One-versus-the-rest(OvR):

Num Number of Classifiers:

K, which can be more efficient when the number of classes is large.

Training Time for each classifier:

Training K classifiers, but on larger datasets compared to OvO.

Prediction Time:

Prediction involves comparing the outputs K*(K-1)/2 of classifiers, which can be slower than OvR for large K.

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.

I think OVR is better than OVO, since OvR involves evaluating only K classifiers during prediction, making it computationally more efficient compared to the K*(K-1)/2 classifiers in OvO.