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 <u>Support Vector Machine Classifier</u> using only NumPy. After that, train your model and tune the hyperpara meter 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 <u>Gram matrix</u> of the given data with a n argument <u>kernel_function</u> 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 <u>polynomial kernel</u> between two vectors with an argument degree.
- Implement the *rbf_kernel* function to compute the value of the <u>rbf_kernel</u> between two vect ors with an argument gamma.

Tips:

- Your functions will be used in the SVM classifier from <u>scikit-learn</u> 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 automatical ly find the best hyperparameter, such as <u>GridSearch</u>. 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 $k_1(x, x^-)$, 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. $k(x,x^{-}) = k_{1}(x,x^{-}) + exp(x^{T}x^{-})$

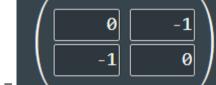
Suppose that $\phi(x) = x \rightarrow it'$ s kernal function is $x^T x \rightarrow it$ is valid kernal function and the exponential tern of a valid kernal will bbe valid kernal $\rightarrow exp(x^Tx)$ is valid kernal.

since $k_I(x, x^-)$ and $exp(x^Tx^-)$ are both valid kernal, by construction rules of sum $\Rightarrow k(x, x^-) = k_I(x, x^-) + exp(x^Tx^-)$ is valid kernal.

b.
$$k(x, x^{-}) = k_1(x, x^{-}) - 1$$

 $k(x, x^{-})$ may not be a valid kernal.

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

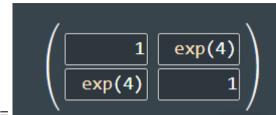


- $K = \frac{1}{2}$ and the eigenvalue = 1 and -1
- → K is not positive semidefinite
- → This kernel is not an valid kernel

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

 $k(x,x^{-})$ may not be a valid kernal.

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



- \star K = and the eigenvalue = 1+exp(4) and 1-exp(4)
- → K is not positive semidefinite
- → This kernel is not an valid kernel

d.
$$k(x,x) = exp(k_I(x,x)) - k_I(x,x)$$

By Taylor's expansions $exp(k_I(x,x^-)) = I + k_I(x,x^-) + \frac{k_I(x,x^-)^2}{2} \dots + \frac{k_I(x,$

$$\frac{k_{I}(x,x^{-})^{n}}{n!} \rightarrow exp\left(k_{I}(x,x^{-})\right) - k_{I}(x,x^{-}) = 1 + \frac{k_{I}(x,x^{-})^{2}}{2} \dots + \frac{k_{I}(x,x^{-})^{n}}{n!}$$

1 is valid kernal and $k_1(x, x^-) \rightarrow \frac{k_1(x, x^-)^2}{2}$, $\cdots \frac{k_1(x, x^-)^n}{n!}$ are all valid kernal. since the every term on the right side is valide kernal $\rightarrow k(x, x^-)$ 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 then so are
 - a. (scaling) $f(x)K_I(x,x^-)f(x^-), f(x) \in R$
 - 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 kerne $K_{0}(x, x^{-}) = x^{T}x^{-}$. Identify which rules you are employing at each step.

et
$$k_1(x,x^-) = 1$$
 and $k_2(x,x^-) = x^Tx^-$

1. scaling: $f(x) = 1/(x^T x) \rightarrow \text{from } k_2 \text{ and } f(x)$

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

2.sum: we add k₁ and the kernel we construct from scaling

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

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

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

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

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 th e 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 no n-B, C vs C). During prediction, the class with the highest confidence score from a ny 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 m any 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 larg e.

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 cla ssifiers during prediction, making it computationally more efficient compar ed to the K*(K-1)/2 classifiers in OvO.