NYCU Pattern Recognition, Homework 4

Deadline: May 17, 23:59

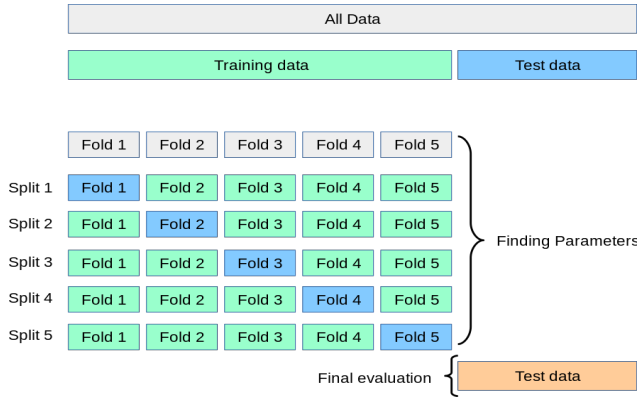
Part. 1, Coding (50%):

For this coding assignment, you are required to implement Cross-Validation and Grid Search using only NumPy. After that, you should train the SVM model from scikit-learn on the provided dataset and test the performance with the testing data. **You will get no points by simply calling** sklearn.model_selection.GridSearchCV.

(50%) K-Fold Cross-Validation & Grid Search

Requirements:

- Implement K-Fold Cross-Validation by creating a function that takes K as an argument and returns a list of K sublists.
  - Each sublist should contain two parts:
    - The first part contains the index of all training folds (index_x_train, index_y_train), for example, Fold 2 to Fold 5 in split 1.
    - The second part contains the index of the validation fold (index_x_val, index_y_val), for example, Fold 1 in split 1 .
  - You need to handle if the sample size is not divisible by K.
  - The first **n_samples % n_splits** folds should have a size of **n_samples // n_splits + 1**, and the other folds should have a size of **n_samples // n_splits**. Here, n_samples is the number of samples and n_splits is K.
  - Each of the samples should be used **exactly once** as the validation data.
  - Please **shuffle** your data before partition.



- Implement Grid Search & Cross-Validation:
  - Using sklearn.svm.SVC to train a classifier on the provided train set and perform **Grid Search** to find the best hyperparameters via cross-validation.

Criteria:

1. (10%) Implement K-fold data partitioning.

```python
def cross_validation(x_train, y_train, k=5):

    # Do not modify the function name and always take 'x_train, y_train, k' as the inputs.
    indices = np.arange(len(x_train))

    # use random seed to mak each samele at anytime is different.
    np.random.seed()
    np.random.shuffle(indices)
    folds = np.array_split(indices, k)
    folds = np.array(folds)

    k_fold = []
    for i in range(k):
        train_fold = np.delete(np.arange(k), i)
        k_fold.append([np.concatenate((folds[train_fold]), axis=None), folds[i]])
    return k_fold
```

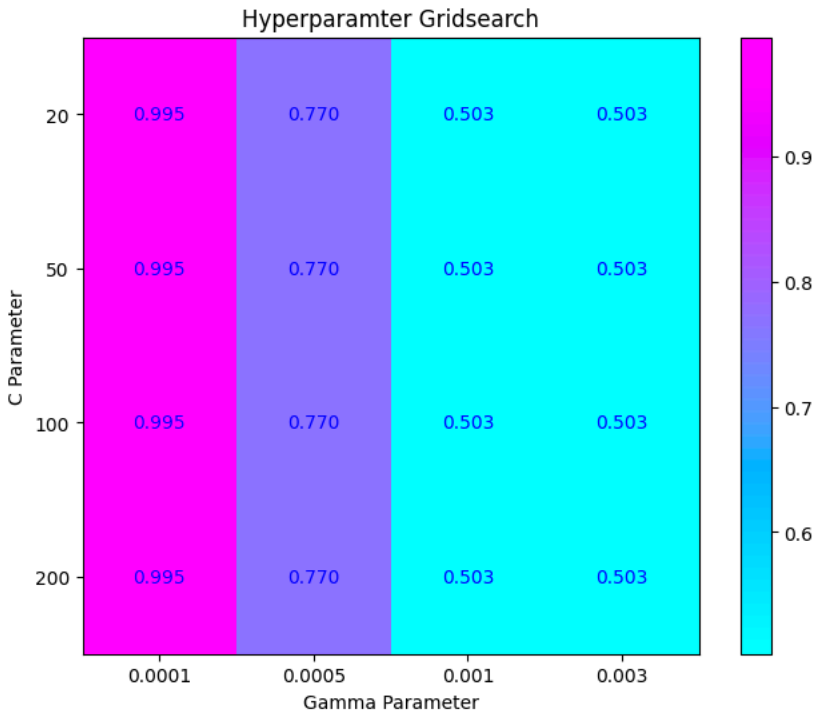
✓ 0.0s

2. (10%) Set the kernel parameter to 'rbf' and do grid search on the hyperparameters C and gamma to find the best values through cross-validation. Print the best hyperparameters you found. Note that we suggest using K=5 for the cross-validation.

```python
best_c, best_gamma = None, None

# TODO HERE
kfold_data = cross_validation(x_train, y_train, k=5)
C = [20, 50, 100 ,200 ]
gamma = [0.0001, 0.0005, 0.001 , 0.003]
# k-Flod Cross Validation and Grid Search
average_scores = np.zeros((len(C), len(gamma)))
best_score = 0
best_model = None
best_c = 0
best_gamma = 0
for i in range(len(C)):
    for j in range(len(gamma)):
        score = 0

        # use cross validation data to train and evaluate the model performance
        for traing_idx, validation_idx in kfold_data:
            clf = SVC(C=C[i], kernel='rbf', gamma=gamma[j])
            clf.fit(x_train[traing_idx], y_train[traing_idx])
            score += clf.score(x_train[validation_idx], y_train[validation_idx])

        average_scores[i][j] = score / len(kfold_data)

        # record the best model
        if average_scores[i][j] > best_score:
            best_score = average_scores[i][j]
            best_model = clf
            best_c = C[i]
            best_gamma = gamma[j]

best_parameters=(best_c, best_gamma)
```

✓ 7m 3.6s

3. (10%) Plot the results of your SVM's grid search. Use "gamma" and "C" as the x and y axes, respectively, and represent the average validation score with color. Below image is just for reference.



4. (20%) Train your SVM model using the best hyperparameters found in Q2 on the entire tra ining dataset, then evaluate its performance on the test set. Print your testing accuracy.

Points	Testing Accuracy
20 points	acc > 0.9
10 points	0.85 <= acc <= 0.9
0 points	acc < 0.85

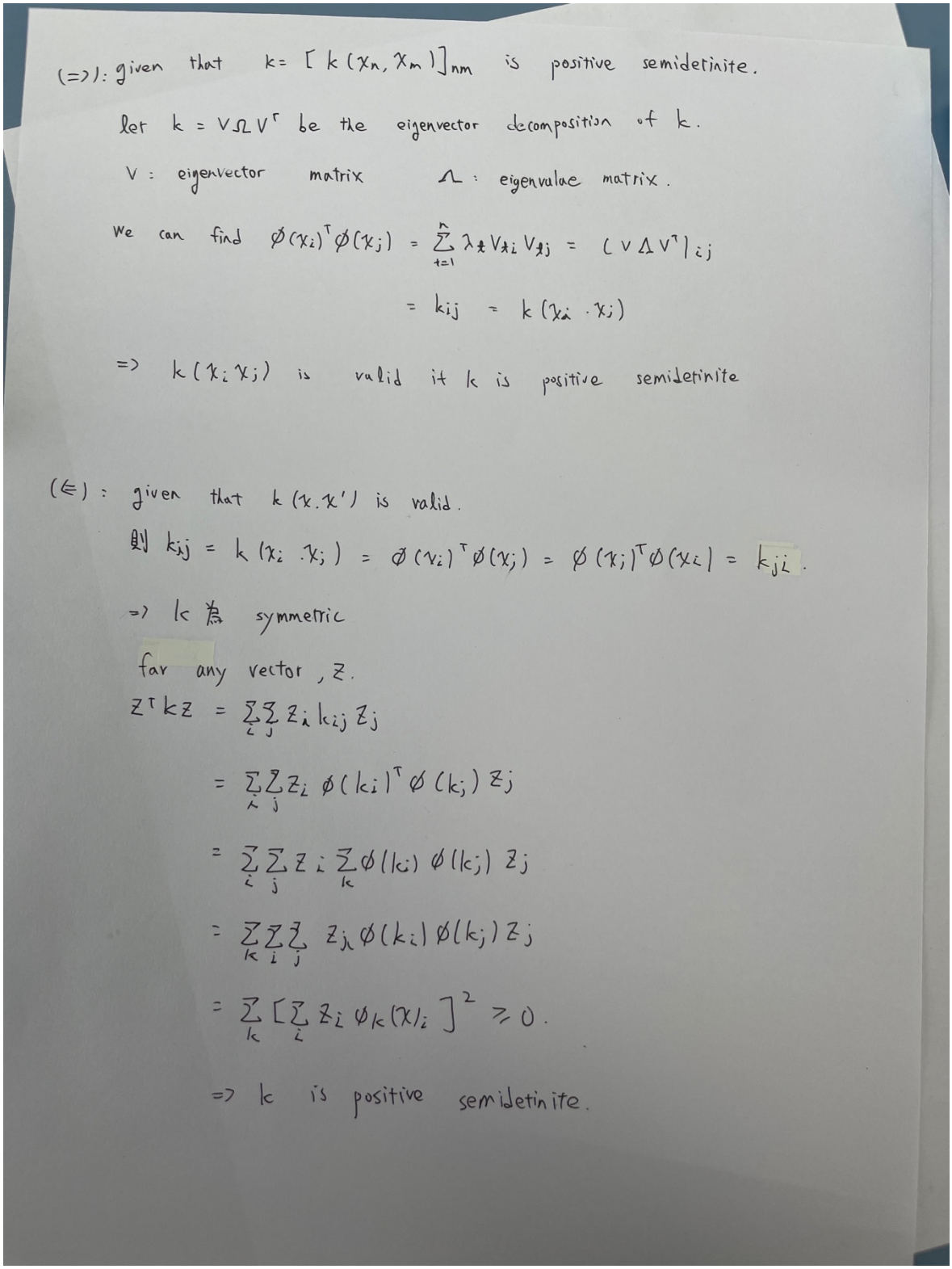
```
# Do Not Modify Below

best_model = SVC(C=best_parameters[0], gamma=best_parameters[1], kernel='rbf')
best_model.fit(x_train, y_train)

y_pred = best_model.predict(x_test)

print("Accuracy score: ", accuracy_score(y_pred, y_test))

# If your accuracy here > 0.9 then you will get full credit (20 points).
```

✓ 7.0s

```
Accuracy score:  0.995
```

Part. 2, Questions (50%):

1. (10%) Show that the kernel matrix $K = [k(x_n, x_m)]_{nm}$ should be positive semidefinite is the nece ssary and sufficient condition for $k(x, x\acute{})$ to be a valid kernel.



2. (10%) Given a valid kernel $k_1(x, x')$, explain that $k(x, x') = exp(k_1(x, x'))$ is also a v alid kernel. (Hint: Your answer may mention some terms like ____ series or ____ expansion.)

2.

from Taylor's expansion.

$$\exp(k_2(x,x')) = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

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

For positive constant 1, it's apparent that it's symmetric as well as PSD. which means that 1 is a valid kernel.

Given $k_1(x,x')$ is valid, so for the remaining part, we can find that they are all valid according to (6.13) and (6.14), Thus, from (6.14) the whole expansion is valid.

$\Rightarrow \exp(k_1(x,x'))$ is valid if $k_1(x,x')$ is valid.

3. (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 co rresponding K is not positive semidefinite and show its eigenvalues.

a. $k(x, x') = k_1(x, x') + x$

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

c. $k(x, x') = k_1(x, x')^2 + exp\left(\|x\|^2\right) * exp(\|x'\|^2)$

d. $k(x, x') = k_1(x, x')^2 + exp\left(k_1(x, x')\right) - 1$

3. (b) 取 $k_1(x,x')$ 為 x^Tx' $\Rightarrow$ $k(x,x') = x^Tx' - 1$

$x_1 = 1 \quad x_2 = -1$

gram matrix $= \begin{bmatrix} 0 & -2 \\ -2 & 0 \end{bmatrix}$ eigenvalue 為 $2, -2$.

非 PSD $\Rightarrow$ $k(x,x') = k_1(x,x')$ Not valid kernel.

(c) $k_1(x,x')$ valid $\rightarrow$ from 6.18. $k_1(x,x')\,k(x,x')$ valid.

So all we have to prove is $\exp(\|x\|^2) * \exp(\|x\|^2)$ is also a valid kernel, so that from (6.17). we can prove that $k_1(x,x')^2 + \exp(\|x^2\|) * \exp(\|x'\|^2)$ is also valid.

according to 6.14 : $f(x)\,k_1(x,x')\,f(x')$ is valid.

now we set $f(x') = \exp(\|x\|^2)$

$\Rightarrow f(x) * 1 * f(x')$ is valid.

$\Rightarrow \exp(\|x\|^2) * \exp(\|x'\|^2)$ is valid.

$\Rightarrow$ Q.E.D

(d) 已知 $k_1(x,x')$ 為 valid kernel $\quad k(x,x') = \underbrace{k_1(x,x')^2}_{①} + \underbrace{\exp(k_1(x,x')) - 1}_{②}$.

① is valid kernel according to 6.18.

② $\exp(k_1(x,x')) - 1$

$= (1 + k_1(x,x') + \frac{k_1(x,x')^2}{2!} + \frac{k(x,x')^3}{3!} \cdots) - 1$

$= k_1(x,x') + \frac{k(x,x')^2}{2!} + \frac{k(x,x')^3}{3!} + \cdots$

$\because k_1(x,x')$ valid $\Rightarrow$ from 6.13, 6.17, 6.18.

② is also valid.

$\because$ ①, ② valid $\Rightarrow$ from 6.17. $k(x,x') = ① + ②$ valid #

4. Consider the optimization problem

$$\begin{aligned} &\textit{minimize } (x-2)^2 \\ &\textit{subject to } (x+4)(x-1) \leq 3 \end{aligned}$$

State the dual problem. (Full points by completing the following equations)

$L(x,\lambda) =$ _____(x - 2)^2 + λ [(x + 4)(x - 1) − 3]_____

$\nabla_x L(x,\lambda) =$ _____2(x - 2) + λ (2x + 3) _____

when $\nabla_x L(x,\lambda) = 0$,

$x =$ _____(4 - 3 λ)/(2 + 2 λ) _____

$L(x,\lambda) = L(\lambda) =$ [(4 - 3 λ) / (2 + 2 λ) - 2]^2 + λ {[(4 - 3 λ) / (2 + 2 λ)) + 4][((4 - 3 λ) / (2 + 2 λ)) − 1] − 3}