NYCU Pattern Recognition, Homework 4

311553010, 陳姿羽

Part. 1, Coding (70%):

1. (5%) Implement K-fold data partitioning.

Q1

```python
def cross_validation(x_train, y_train, k=5):
    n_samples = len(y_train)
    fold_sizes = np.full(k, n_samples // k, dtype=np.int)
    fold_sizes[:n_samples % k] += 1
    current = 0
    kfold_data = []

    for fold_size in fold_sizes:
        start, stop = current, current + fold_size
        validation_index = np.arange(start, stop)
        train_index = np.concatenate([np.arange(0, start), np.arange(stop, n_samples)])
        kfold_data.append([train_index, validation_index])
        current = stop

    return kfold_data
```

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. (best_c, best_gamma) is (1, 0.0001)

Q2

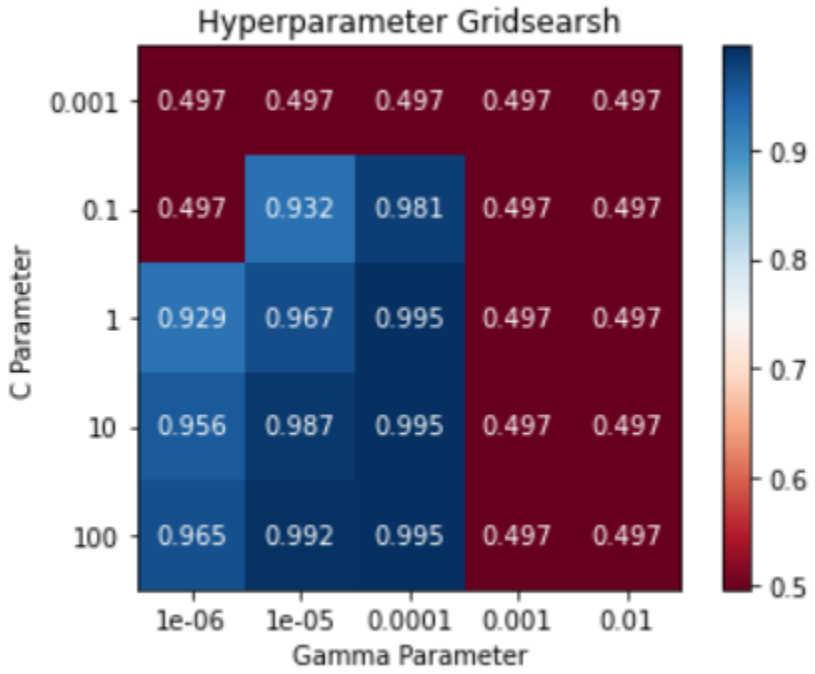
```
# (Example) Using SVC from sklearn

clf = SVC(C=1.0, gamma=0.01, kernel='rbf')
```

```
best_c, best_gamma = None, None
c_range = [0.001, 0.1, 1, 10, 100]
gamma_range = [0.000001, 0.00001, 0.0001, 0.001, 0.01]

best_score = 0

k = 5
kfold_data = cross_validation(x_train, y_train, k)

Gridsearsh = np.zeros((len(c_range), len(gamma_range)))

for c in c_range:
    for gamma in gamma_range:
        score_sum = 0
        for train_indices, val_indices in kfold_data:
            x_train_fold = x_train[train_indices]
            y_train_fold = y_train[train_indices]
            x_val_fold = x_train[val_indices]
            y_val_fold = y_train[val_indices]

            clf.set_params(C=c, gamma=gamma)
            clf.fit(x_train_fold, y_train_fold)
            y_val_pred = clf.predict(x_val_fold)
            score_sum += accuracy_score(y_val_fold, y_val_pred)

        average_score = score_sum / k
        Gridsearsh[c_range.index(c), gamma_range.index(gamma)] = average_score
        if average_score > best_score:
            best_score = average_score
            best_c = c
            best_gamma= gamma

best_parameters=(best_c, best_gamma)
```

```
print("(best_c, best_gamma) is ", best_parameters)
```

```
(best_c, best_gamma) is  (1, 0.0001)
```

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.



4. (5%) Train your SVM model using the best hyperparameters found in Q2 on the entire training dataset, then evaluate its performance on the test set. Print your testing accuracy.
   Accuracy score: 0.995

Q4

```
# 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).
```

```
Accuracy score:  0.995
```

Part. 2, Questions (30%):

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

   According to Mercer's theorem, for a valid kernel matrix K, the dot product of any vector with K must be greater than or equal to zero. Additionally, the kernel matrix K must be positive semidefinite, which means that all of its eigenvalues are greater than or equal to zero. If the kernel matrix K is not positive semidefinite, there may exist a vector n such that $n \cdot K < 0$, which means that K is not valid as a kernel function.

   Therefore, the kernel matrix $\mathrm{K} = [\mathrm{k}(x_n, x_m)]_{nm}$ being positive semidefinite is the necessary and sufficient condition for $\mathrm{k}(\mathrm{x}, \mathrm{x}')$ to be a valid kernel.

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

   K_1 is a valid kernel, so it is also positive semidefinite. According to the eigenvalue decomposition theorem, any positive semidefinite can be represented by eigenvectors and eigenvalues. Thus, K_1 can be expressed as $K_1 = \lambda\, v\, v^T$, where λ is the eigenvalue and v is the corresponding eigenvector.

   After expanding $exp\big(k_1(x,\ x')\big)$ using Taylor expansion, we substitute $k_1(x,\ x')$ into the definition of the kernel matrix to obtain a new matrix K.

$$exp\big(k_1(x, x')\big) = \sum_{m=0}^{\infty} \frac{k_1(x, x')}{m!}$$

   Since $k_1(x,\ x')$ is a real number, this Taylor expansion converges. The matrix K can be expressed as an infinite series sum.

$$\mathrm{K} = \sum_{m=0}^{\infty} \frac{1}{m!} K_1^m$$

   Since K_1 is positive semidefinite, raising it to the power of m, K_1^m, also results in positive semidefinite. Consequently, K is positive semidefinite as well. Therefore, $exp\big(k_1(x,\ x')\big)$ is a valid kernel.

3. 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 show its eigenvalues.

a、 $k(x,\ x') = k_1(x,\ x') + x$

b、 $k(x,\ x') = k_1(x,\ x') - 1$

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

d、 $k(x,\ x') = k_1(x,\ x')^2 + exp\big(k_1(x,\ x')\big) - 1$

4. Consider the optimization problem

$$minimize\ (x-2)^2$$
$$subject\ to\ (x+4)(x-1) \le 3$$

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

$$L(x,\ \lambda) = \underline{(x-2)^2 + \lambda((x+4)(x-1)-3)}$$

$$\nabla_x L(x,\ \lambda) = \underline{2(x-2) + \lambda(2x+3)}$$

when $\nabla_x L(x,\ \lambda) = 0$,
$(2(x-2) + \lambda(2x+3) = 0)$

$$x = \frac{4-3\lambda}{2+2\lambda}$$

$$L(x,\ \lambda) = L(\lambda) = \left(\left(\frac{4-3\lambda}{2+2\lambda}\right) - 2\right)^2 + \lambda\left(\left(\left(\frac{4-3\lambda}{2+2\lambda}\right) + 4\right)\left(\left(\frac{4-3\lambda}{2+2\lambda}\right) - 1\right) - 3\right)$$