Support Vector Machine (HW4)

Student: 張皓雲

Student Id: 310551031

- # Part. 1, Coding (50%):

1. (10%) K-fold data partition: Implement the K-fold cross-validation function. Your function should take K as an argument and return a list of lists (len(list) should equal to K), which contains K elements. Each element is a list containing two parts, the first part contains the index of all training folds (index_x_train, index_y_train), e.g., Fold 2 to Fold 5 in split 1. The second part contains the index of the validation fold, e.g., Fold 1 in split 1 (index_x_val, index_y_val)

```
(550, 300)
[0 1]
Split: 1, Training index: [ 2  3  5  6  7  8  9 10 11 12 13 14 16 17 18 19], Validation index: [ 0  1  4 15]
Split: 2, Training index: [ 0  1  2  4  5  6  8  9 10 11 13 14 15 17 18 19], Validation index: [ 3  7 12 16]
Split: 3, Training index: [ 0  1  2  3  4  5  6  7  8 10 12 14 15 16 18 19], Validation index: [ 9 11 13 17]
Split: 4, Training index: [ 0  1  2  3  4  5  7  9 11 12 13 14 15 16 17 19], Validation index: [ 6  8 10 18]
Split: 5, Training index: [ 0  1  3  4  6  7  8  9 10 11 12 13 15 16 17 18], Validation index: [ 2  5 14 19]
```

```python
def cross_validation(x_train, y_train, k=5):
    kfold_data = []
    single_fold_num = len(x_train)//k
    shuffle_index = np.arange(len(x_train))
    np.random.seed(14)
    np.random.shuffle(shuffle_index)
    for cur_k in range(0, len(x_train), single_fold_num):
        validation_fold = shuffle_index[cur_k:cur_k+single_fold_num]
        training_fold = np.array(
            [x for x in shuffle_index if x not in validation_fold])
        kfold_data.append([training_fold, validation_fold])

    if (len(x_train) % k) > 0:
        validation_fold = shuffle_index[cur_k:len(x_train)]
        training_fold = np.array(
            [x for x in shuffle_index if x not in validation_fold])
        kfold_data.append([training_fold, validation_fold])

    return kfold_data
```

2. (20%) Grid Search & Cross-validation: using sklearn.svm.SVC to train a classifier on the provided train set and conduct the grid search of “C” and “gamma,” “kernel’=’rbf’ to find the best hyperparameters by cross-validation. Print the best hyperparameters you found. Note: We suggest using K=5

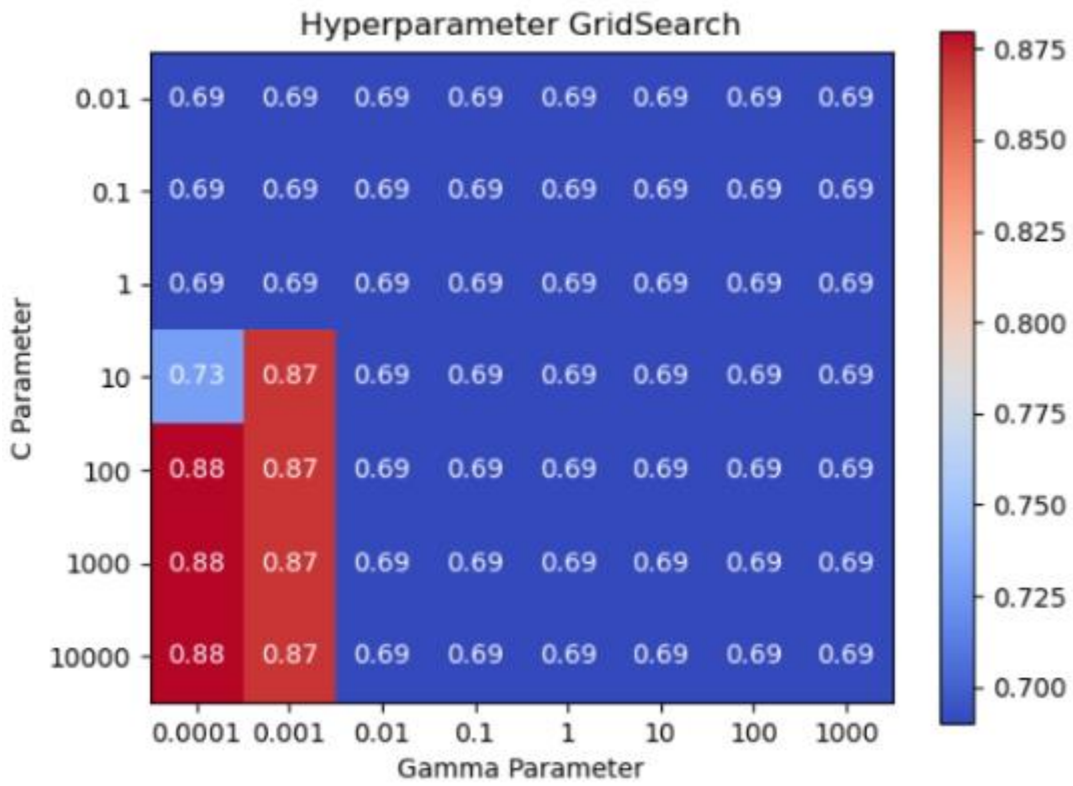
```
{'kernel': 'rbf', 'gamma': 0.0001, 'C': 100}
```

```python
kfold_data = cross_validation(x_train, y_train, k=5)
# clf = SVC(C=1.0, kernel='rbf', gamma=0.01)

maximum_score = -1
best_parameters = []
picture_data = []

for kernel in ['rbf']:
    for C in [0.01, 0.1, 1, 10, 100, 1000, 10000]:
        validate_acc = []
        for gmma in [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000]:
            average_score = []
            for idx, fold_data in enumerate(kfold_data):
                clf = SVC(C=C, kernel=kernel, gamma=gmma)
                clf.fit(x_train[fold_data[0]], y_train[fold_data[0]])
                y_pred = clf.predict(x_train[fold_data[1]])
                score = accuracy_score(y_pred, y_train[fold_data[1]])
                average_score.append(score)
            score = sum(average_score)/len(average_score)
            validate_acc.append(round(score, 2))
            if score > maximum_score:
                maximum_score = score
                best_parameters = {"kernel": kernel, "gamma": gmma, "C": C}
        picture_data.append(validate_acc)
```

3. (10%) Plot the grid search results of your SVM. The x and y represent “gamma” and “C” hyperparameters, respectively. And the color represents the average score of validation folds.



```python
def draw_heatmap(gamma_list, C_list, data):

    xlabel = gamma_list
    ylabel = C_list
    fig = plt.figure()
    ax = fig.add_subplot(111)
    ax.set_yticks(range(len(ylabel)))
    ax.set_yticklabels(ylabel)
    ax.set_xticks(range(len(xlabel)))
    ax.set_xticklabels(xlabel)
    ax.grid(which="minor", color="w", linestyle='-', linewidth=3)

    data = np.array(data)

    for x in range(data.shape[0]):
        for y in range(data.shape[1]):
            text = ax.text(y, x, data[x, y],
                           ha="center", va="center", color="w")

    im = ax.imshow(data, cmap=plt.cm.coolwarm)
    plt.colorbar(im)

    plt.title("Hyperparameter GridSearch")
    plt.xlabel("Gamma Parameter")
    plt.ylabel("C Parameter")

    plt.savefig("Hyperparameter_GridSearch.png")
    plt.close()

    return None
```

4. (10%) Train your SVM model by the best hyperparameters you found from question 2 on the whole training data and evaluate the performance on the test set.

```
{'kernel': 'rbf', 'gamma': 0.0001, 'C': 100}
Accuracy score:  0.9010416666666666
```

```python
# ## Question 4
# Train your SVM model by the best parameters you found from question 2 on

best_model = SVC(C=best_parameters["C"], kernel=best_parameters["kernel"],
                 gamma=best_parameters["gamma"])

best_model.fit(x_train, y_train)
y_pred = best_model.predict(x_test)
print("Accuracy score: ", accuracy_score(y_pred, y_test))
```

● Part. 2, Questions (50%):

1. (10%) Given a valid kernel $k_1(x, x')$, prove that the following proposed functions are or are not valid kernels.
   a. $k(x, x') = (k_1(x, x'))^2 + (k_1(x, x') + 1)^2$
   b. $k(x, x') = (k_1(x, x'))^2 + \exp(||x||^2) * \exp(||x'||^2)$

(a). $k(x,x') = (k_1(x,x'))^2 + (k_1(x,x')+1)^2$

證明 $k_c(x,x')=1$ is a valid kernel

positive semidefinite $\Rightarrow A^T k_c A \geq 0,\ \forall A \in R^d$

$\Rightarrow \sum_{i,j} a_i k_c(i,j) a_j = k_c(i,j) \sum_{i,j} a_i a_j$

$\Rightarrow C\|A\| \geq 0 \Rightarrow$ valid kernel ※

According to

$k(x,x') = k_1(x,x') + k_2(x,x')$ — (6.17)

$k(x,x') = k_1(x,x') k_2(x,x')$ — (6.18)

is a valid kernel

$\therefore k(x,x') = (k_1(x,x'))^2 + (k_1(x,x')+1)^2$ is a valid kernel ※

(b) $k(x,x') = (k_1(x,x'))^2 + \exp(\|x\|^2)\exp(\|x'\|^2)$

According to

$k(x,x') = \exp(k_1(x,x'))$ — (6.16)

$k(x,x') = k_1(x,x') + k_2(x,x')$ — (6.17) is valid kernel

$k(x,x') = k_1(x,x') k_2(x,x')$ — (6.18)

$\therefore k(x,x') = (k_1(x,x'))^2 + \exp(\|x\|^2) + \exp(\|x'\|^2)$

is a valid kernel

2. (10%) Show that the kernel matrix $\mathbf{K} = [k(\mathbf{x}_n, \mathbf{x}_m)]_{nm}$ should be positive semidefinite is the necessary and sufficient condition for $k(\mathbf{x}, \mathbf{x}')$ to be a valid kernel.

假設 gram matrix K 的維度是 $N \times N$

gram matrix 是 symmetric 且 positive semidefinite 的矩陣

if and only if $K_{nm} = \phi(x_n)^T \phi(x_m)$, $K_{nm} = K_{mn}$,且 $z \in R^d$

positive semidefinite $\Leftrightarrow$ $k(x, x')$ is valid

"$\Leftarrow$" support k is valid

$$z^T K z = \sum_{n=1}^{N} \sum_{m=1}^{N} z_n k_{nm} z_m = \sum_{n=1}^{N} \sum_{m=1}^{N} z_n \phi(x_n)^T \phi(x_m) z_m$$

$$= \sum_{n=1}^{N} \sum_{m=1}^{N} z_n \left(\sum_{k=1}^{N} \phi_k(x_n) \phi_k(x_m) \right) z_m$$

$$= \sum_{k=1}^{N} \sum_{n=1}^{N} \sum_{m=1}^{N} z_n \phi_k(x_n) \phi_k(x_m) z_m$$

$$= \sum_{k=1}^{N} \left(\sum_{n=1}^{N} z_n \phi_k(x_n) \right)^2 \geq 0 \; \#$$

"$\Rightarrow$" positive semidefinite is given

Let $K = UDU^T$ be the eigenvector decomposition of K

$$k_{ij} = (D^{\frac{1}{2}} u_i)^T (D^{\frac{1}{2}} u_j)$$

define $\phi(x_i) = D^{\frac{1}{2}} u_i$

$$\Rightarrow k(x_i, x_j) = \phi(x_i)^T \phi(x_j) = k_{ij}$$

$\therefore$ if K is position semidefinite

then $k(x, x')$ is a valid kernel.

3. (10%) Consider the dual formulation of the least-squares linear regression problem given on page 6 in the ppt of Kernel Methods. Show that the solution for the components $\mathbf{a_n}$ of the vector $\mathbf{a}$ can be expressed as a linear combination of the elements of the vector $\boldsymbol{\varphi}(\mathbf{x_n})$. Denoting these coefficients by the vector $\mathbf{w}$, show that the dual of the dual formulation is given by the original representation in terms of the parameter vector $\mathbf{w}$.

$$a_n = -\frac{1}{\lambda}\{w^T\phi(x_n) - t_n\}$$

$$= -\frac{1}{\lambda}\{W_1\phi_1(x_n) + W_2\phi_2(x_n) + \cdots + W_M\phi_M(x_n) - t_n\}$$

$$= -\frac{W_1}{\lambda}\phi_1(x_n) - \frac{W_2}{\lambda}\phi_2(x_n) - \cdots - \frac{W_M}{\lambda}\phi_M(x_n) + \frac{t_n}{\lambda}$$

$$= \left(C_n - \frac{W_1}{\lambda}\right)\phi_1(x_n) + \left(C_n - \frac{W_2}{\lambda}\right)\phi_2(x_n) + \cdots + \left(C_n - \frac{W_M}{\lambda}\right)\phi_M(x_n)$$

where $C_n = \dfrac{t_n/\lambda}{\phi_1(x_n) + \phi_2(x_n) + \cdots + \phi_M(x_n)}$

$\therefore$ a_n은 $\phi(x_n)$의 linear combination

4. (10%) Prove that the Gaussian kernel defined by (eq 1) is valid and show the function $\boldsymbol{\varphi}(\mathbf{x})$, where $x \in R^1$.

(eq1) $$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2\right) = \phi(x)^{\mathrm{T}} \phi(x')$$

$$k(x, x') = \exp\left(\frac{-|x-x'|^2}{2\sigma^2}\right) = \phi(x)^T\phi(x')$$

$$= \exp\left(\frac{-(x)^2 + 2xx' - (x')^2}{2\sigma^2}\right)$$

$$= \exp\left(\frac{-(x)^2}{2\sigma^2}\right)\exp\left(\frac{-(x')^2}{2\sigma^2}\right)\exp\left(\sum_{n=1}^{\infty}\frac{\left(\frac{2xx'}{2\sigma^2}\right)^n}{n!}\right)$$

$$= \sum_{n=0}^{\infty}\left(\exp\left(\frac{-(x)^2}{2\sigma^2}\right)\exp\left(\frac{-(x')^2}{2\sigma^2}\right)\sqrt{\frac{\left(\frac{1}{\sigma^2}\right)^n}{n!}}\sqrt{\frac{\left(\frac{1}{\sigma^2}\right)^n}{n!}}(x)^n(x')^n\right)$$

$$= \phi^T(x)\phi(x')$$

$$\phi(x) = \exp(-(x)^2)\cdot\left(1, \sqrt{\frac{\frac{1}{\sigma^2}}{1!}}\,x, \sqrt{\frac{\left(\frac{1}{\sigma^2}\right)^2}{2!}}\,x^2, \cdots\right)$$

According to

$$k(x, x') = \exp(k_1(x, x')) \quad - (6.16)$$

$$k(x, x') = f(x)k_1(x, x')f(x') \quad - (6.14)$$

$\therefore$ Gaussian kernel is valid

5. (10%) Consider the optimization problem

$$\text{minimize } (x - 2)^2$$
$$\text{subject to } (x+3)(x-1) \leqq 2$$

State the dual problem.

$$\begin{aligned}\text{Lagrange function} &= (x-2)^2 + \lambda((x+3)(x-1)-2) \\ &= x^2-4x+4+\lambda(x^2+2x-5) \\ &= (\lambda+1)x^2+(2\lambda-4)+(4-5\lambda)\end{aligned}$$

對x偏微

$$2x(\lambda+1)+(2\lambda-4)=0$$

$$\Rightarrow x=\frac{2-\lambda}{\lambda+1}, \quad \lambda \geq 0$$

帶回 Lagrange function

$$\frac{(2-\lambda)^2}{\lambda+1}+\frac{-2(2-\lambda)^2}{\lambda+1}+(4-5\lambda)$$

$$=\frac{-(2-\lambda)^2}{\lambda+1}+\frac{(4-5\lambda)(\lambda+1)}{\lambda+1}$$

$$=\frac{3\lambda(1-2\lambda)}{\lambda+1}$$

Maximum $\dfrac{3\lambda(1-2\lambda)}{\lambda+1}$

subject to $\lambda \geq 0$