Pattern Recognition HW4

M093781 趙宇涵

Part 1.

1. K-fold data partition:

Question 1

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 contains two parts, the first part contains the index of all training folds, e.g. Fold 2 to Fold 5 in split 1. The second part contains the index of validation fold, e.g. Fold 1 in split 1

```
In [5]: # import random
        # seed = random.randrange(100000)
        # np.random.seed(seed)
        # print("Seed was:", seed)

        def cross_validation(x_train, y_train, k=5):
            y_train = np.resize(y_train, (1,y_train.shape[0]))   # resize y train for append x and y.
            train = np.append(x_train, y_train.T, axis=1)          # append x and y together.
            np.random.seed(59527)
            train_rd = np.copy(train)                            # clone an array for shuttle.
            np.random.shuffle(train_rd)                          # shuffle the train array.
            kfold_data = []
            total_num = x_train.shape[0]                         # total number of training set.
            fold_num = int(total_num / k)                        # the number of dataset in a folder.
            for i in range(k):                                   # cut k part and store in to kfold_data
                valid_index = list(range(i*fold_num,(i+1)*fold_num))
                valid_data = train_rd[i*fold_num:(i+1)*fold_num]
                rest = np.delete(train_rd, valid_index, 0)
                kfold_data.append([rest, valid_data])
            return kfold_data
```

```
In [6]: kfold_data = cross_validation(x_train, y_train, k=10)
        assert len(kfold_data) == 10 # should contain 10 fold of data
        assert len(kfold_data[0]) == 2 # each element should contain train fold and validation fold
        assert kfold_data[0][1].shape[0] == 55 # The number of data in each validation fold should equal to training data di
```

In question 1, I implement the K-fold cross-validation function. Below is the steps:

(1) Combine the a and y together.

(2) Clone the array and random shuffle

(3) Calculate the number of data in 1 fold by $n = \frac{total\ number}{k}$

(4) Cut train and validation dataset.

2. Grid Search & Cross-validation

```
In [10]: ## your code
         def gridsearch(kfold_data, C_list, gamma_list):
             best_score = 0
             grid_acc = []
             for C in C_list:
                 gamma_acc = []
                 for gamma in gamma_list:
                     score_k = 0
                     for k in range(len(kfold_data)):
                         x_train_k = kfold_data[k][0][:,:-1]
                         y_train_k = kfold_data[k][0][:,-1]
                         x_valid_k = kfold_data[k][1][:,:-1]
                         y_valid_k = kfold_data[k][1][:,-1]
                         clf = SVC(C=C, kernel='rbf', gamma=gamma)
                         clf.fit(x_train_k, y_train_k)
                         score = clf.score(x_valid_k, y_valid_k)
                         score_k += score
                     score_final = score_k / len(kfold_data)
                     gamma_acc.append(score_final)
                     if score_final >= best_score:
                         best_score = score_final
                         best_parameters = [C, gamma]
                 grid_acc.append(gamma_acc)
             grid_acc = np.asarray(grid_acc)
             return best_parameters, best_score, grid_acc
```

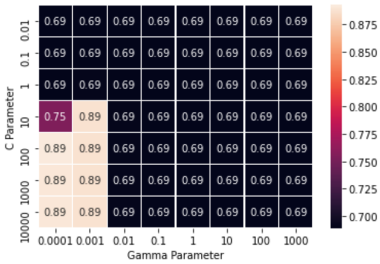
```
In [11]: C_list = [0.01, 0.1, 1, 10, 100, 1000, 10000]
         gamma_list = [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000]
         best_parameters, best_score, grid_acc = gridsearch(kfold_data, C_list, gamma_list)
```

```
In [12]: print(best_score)
         print(best_parameters)
         print(grid_acc.shape)

         0.8927272727272726
         [100, 0.0001]
         (7, 8)
```

In this part I use the k-fold dataset to be training and valid dataset, and use grid search to train with SVC package. The C parameter are [0.01, 0.1, 1, 10, 100, 1000, 10000], and the gamma parameter are [0.0001, 0.001, 0.1, 1, 10, 100, 1000]. Finally I get the best accuracy is 0.8927, and the best parameters are (C=100, gamma=0.0001).

3. Plot the grid search results of your SVM.
   Above is my grid search implement and result:

In [13]:
```
import seaborn as sns
import matplotlib.pylab as plt

ax = sns.heatmap(grid_acc, annot=True, xticklabels=gamma_list, yticklabels=C_list, linewidth=0.3)
ax.set(xlabel='Gamma Parameter', ylabel='C Parameter')
plt.show()
```



4. Train the SVM model and test with testing data.

Question 4

Train your SVM model by the best parameters you found from question 2 on the whole training set and evaluate the performance on the test set.

In [14]:
```
best_model = SVC(C=best_parameters[0], kernel='rbf', gamma=best_parameters[1])
best_model.fit(x_train, y_train)
```

Out[14]:
```
SVC(C=100, gamma=0.0001)
```

In [15]:
```
y_pred = best_model.predict(x_test)
print("Accuracy score: ", accuracy_score(y_pred, y_test))
```

```
Accuracy score:  0.9010416666666666
```

I observed the testing result is depending in the shuffle random seed in question 1, so I try a few times, and set the best random seed in question 1.
Finally, I could get net best result is 0.9010.

Part 2. Questions (50%)

1. (10%) Given valid kernel $k_1(x,x')$, prove a. b are or are not valid.

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

From formula (6.13) ~ (6.22) in textbook and slides.

$(k_1(x,x'))^2$ is the same type as (6.18), so it's valid.

$(k_1(x,x')+1)$ is the same type as (6.15), so it's valid. What's more,

$(k_1(x,x')+1)^2$ is also valid cause it's the same type as (6.18)

Finally, due to (6.21), we can validate a. is a valid kernel #.

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

Exponent property: If k is a positive definite kernel, $\exp(k(x,y))$ is valid.

Proof: we have $\exp(k(x,y)) = \lim_{N \to \infty} \sum_{n=0}^{N} \frac{1}{n!} k(x,y)^n$. By power, scaling, sum properties, we can proof exponent property.

Thus, $k_1(x,x')$, $\exp(\|x\|^2)$, $\exp(\|x'\|^2)$ are valid,

$\Rightarrow k(x,x')$ is valid. #

2. (10%) Show $\mathbb{K} = [k(x_n, x_m)]_{nm}$ should be positive semidefinite is necessary and sufficient condition for $k(x,x')$ to be a valid kernel.

<prof.> Let $\mathbb{K} = [k(x_n, x_m)]_{nm}$ as $\mathbb{A}$

Suppose $\mathbb{A} = \mathbb{B}'\mathbb{B}$ (Cholesky decompose), for any vector

$v'\mathbb{A}v = v'\mathbb{B}'\mathbb{B}v = \|\mathbb{B}v^2\|^2 \geq 0$ implying A is semi-definite.

set $\mathbb{A}V = V\Lambda$, $\mathbb{B} = \sqrt{\Lambda}V$

then $\mathbb{B}'B = V\sqrt{\Lambda}\sqrt{\Lambda}V' = V\Lambda V' = AVV' = \mathbb{A}$. #

if k is not semi-definite, the matrix $\mathbb{K}$ may also not be positive definite. Cholesky does not work and there's no corresponding inner product space.

3. (10%) Show $a_n = -\frac{1}{\lambda}\{w^T\phi(x_n) - t_n\}$ as a linear combination of the vector $\phi(x_n)$.

$$J(w) = \frac{1}{2}\sum_{n=1}^{N}(w^T\phi(x_n) - t_n)^2 + \frac{\lambda}{2}w^Tw \quad \text{with } \lambda > 0.$$

$$\frac{\partial J(w)}{\partial w} = \sum_{n=1}^{1}(w^T\phi(x_n) - t_n)\,\phi(x_n) + \lambda w = 0$$

$$w = -\frac{1}{\lambda}\sum_{n=1}^{N}(w^T\phi(x_n) - t_n)\,\phi(x_n)$$

$$= \sum_{n=1}^{N} a_n \phi(x_n) = \Phi^T a$$

where $\Phi = [\phi(x_1)^T \ldots \phi(x_n)^T]$

thus, w is a linear combination of $\phi(x_n)$ #

4. (10%) Prove the Gaussian kernel is valid and show the function $\varphi(x)$

$$k(x,x') = \exp(-\|x-x'\|^2/(2\sigma^2)) = \phi(x)^T\phi(x')$$

$$= \exp\left(-\frac{1}{2\sigma^2}\|x^2\|\right)\exp\left(\frac{1}{\sigma^2}x^Tx'\right)\exp\left(-\frac{1}{2\sigma^2}\|x'\|^2\right)$$

$\because x \in R$,

$\therefore$ by scaling property, $\exp\left(\frac{1}{\sigma^2}x^Tx'\right)$ is valid.

by exponents and functions properties, $\exp\left(-\frac{1}{2\sigma^2}\|x^2\|\right)$ and $\exp\left(-\frac{1}{2\sigma^2}\|x'\|^2\right)$ is valid as well

$\Rightarrow$ by multiple properties. Gaussian kernel is valid.

5. (10%) Consider the optimization problem :

minimize $(x-2)^2$

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

<sol>

by Lagrange function

$$L(x,a) = (x-2)^2 + a\left[(x+3)(x-1)-2\right]$$

$$= (x^2-4x+4) + (ax^2+2ax-5a)$$

$$= (a+1)x^2 + (2a-4)x + (-5a+4)$$

$$\frac{\partial L(x,a)}{\partial x} = 0$$

$$2(a+1)x + (2a-4) = 0$$

$$x = \frac{4-2a}{2a+2} \quad \#$$