ML HW05-2 – SVM

309553048_多工所_陳柏丞

- **Code Explanation：**

1. ReadCSV(filename)

```
def ReadCSV(filename):
    with open(filename, newline='') as csvfile:
        row = csv.reader(csvfile)
        data_list = [[float(ele) for ele in array] for array in row]

    data = np.array(data_list)
    row, col = data.shape

    return [data, row, col]
```

2. Data Prepare

```
X_train, X_train_row, X_train_col = ReadCSV("X_train.csv")
Y_train, Y_train_row, Y_train_col = ReadCSV("Y_train.csv")
Y_train = Y_train.reshape(5000,)

X_test, X_test_row, X_test_col = ReadCSV("X_test.csv")
Y_test, Y_test_row, Y_test_col = ReadCSV("Y_test.csv")
Y_test = Y_test.reshape(2500, )
```

3. Q1.

```
# Linear kernel
model = svm_train(Y_train, X_train, '-t 0 -r 1')
label, acc, val = svm_predict(Y_test, X_test, model)
```

```
Accuracy = 95.08% (2377/2500) (classification)
```

```
# Polynomial kernel
model = svm_train(Y_train, X_train, '-t 1 -r 1')
label, acc, val = svm_predict(Y_test, X_test, model)
```

```
Accuracy = 95.76% (2394/2500) (classification)
```

```
# RBF kernel
model = svm_train(Y_train, X_train, '-t 2 -r 1')
label, acc, val = svm_predict(Y_test, X_test, model)
```

```
Accuracy = 95.32% (2383/2500) (classification)
```

-t kernel_type : set type of kernel function (default 2)

0 -- linear: u'*v

1 -- polynomial: (gamma*u'*v + coef0)^degree

2 -- radial basis function: exp(-gamma*|u-v|^2)

3 -- sigmoid: tanh(gamma*u'*v + coef0)

-r coef0 : set coef0 in kernel function (default 0)

4. Q2

```python
# Linear kernel
C = [2**-3, 2**-1, 2**1]
G = [2**-1, 2**1, 2**3]

max_acc = 0.0
max_cost = 0.0
max_gamma = 0.0

for cost in C:
    for gamma in G:
        model = svm_train(Y_train, X_train, f'-t 0 -s 0 -c {cost} -g {gamma} -v 5')
        print(f'cost: {cost}, gamma: {gamma}, acc: {model}')
        if model > max_acc:
            max_acc = model
            max_cost = cost
            max_gamma = gamma

print(f'max_cost: {max_cost}, max_gamma: {max_gamma}, max_acc: {max_acc}')
model = svm_train(Y_train, X_train, f'-t 0 -s 0 -c {max_cost} -g {max_gamma}')
label, acc, val = svm_predict(Y_test, X_test, model)
```

```
Cross Validation Accuracy = 97%
cost: 0.125, gamma: 0.5, acc: 97.0
Cross Validation Accuracy = 96.94%
cost: 0.125, gamma: 2, acc: 96.94
Cross Validation Accuracy = 96.82%
cost: 0.125, gamma: 8, acc: 96.82
Cross Validation Accuracy = 96.24%
cost: 0.5, gamma: 0.5, acc: 96.24000000000001
Cross Validation Accuracy = 96.38%
cost: 0.5, gamma: 2, acc: 96.38
Cross Validation Accuracy = 96.12%
cost: 0.5, gamma: 8, acc: 96.12
Cross Validation Accuracy = 96.28%
cost: 2, gamma: 0.5, acc: 96.28
Cross Validation Accuracy = 96.06%
cost: 2, gamma: 2, acc: 96.06
Cross Validation Accuracy = 96.32%
cost: 2, gamma: 8, acc: 96.32
max_cost: 0.125, max_gamma: 0.5, max_acc: 97.0
Accuracy = 95.92% (2398/2500) (classification)
```

-g gamma : set gamma in kernel function (default 1/num_features)

-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)

-s svm_type : set type of SVM (default 0)

0 -- C-SVC

1 -- nu-SVC

2 -- one-class SVM

3 -- epsilon-SVR

4 -- nu-SVR

```
# RBF kernel
C = [2**-3, 2**-1, 2**1]
G = [2**-1, 2**1, 2**3]

max_acc = 0.0
max_cost = 0.0
max_gamma = 0.0

for cost in C:
    for gamma in G:
        model = svm_train(Y_train, X_train, f'-t 2 -s 0 -c {cost} -g {gamma} -v 5')
        print(f'cost: {cost}, gamma: {gamma}, acc: {model}')
        if model > max_acc:
            max_acc = model
            max_cost = cost
            max_gamma = gamma

print(f'max_cost: {max_cost}, max_gamma: {max_gamma}, max_acc: {max_acc}')
model = svm_train(Y_train, X_train, f'-t 2 -s 0 -c {max_cost} -g {max_gamma}')
label, acc, val = svm_predict(Y_test, X_test, model)
```

```
Cross Validation Accuracy = 22.4%
cost: 0.125, gamma: 0.5, acc: 22.400000000000002
Cross Validation Accuracy = 20.34%
cost: 0.125, gamma: 2, acc: 20.34
Cross Validation Accuracy = 78.96%
cost: 0.125, gamma: 8, acc: 78.96
Cross Validation Accuracy = 26.34%
cost: 0.5, gamma: 0.5, acc: 26.340000000000003
Cross Validation Accuracy = 20.38%
cost: 0.5, gamma: 2, acc: 20.380000000000003
Cross Validation Accuracy = 78.96%
cost: 0.5, gamma: 8, acc: 78.96
Cross Validation Accuracy = 46.12%
cost: 2, gamma: 0.5, acc: 46.12
Cross Validation Accuracy = 25.28%
cost: 2, gamma: 2, acc: 25.28
Cross Validation Accuracy = 20.92%
cost: 2, gamma: 8, acc: 20.919999999999998
max_cost: 0.125, max_gamma: 8, max_acc: 78.96
Accuracy = 78.64% (1966/2500) (classification)
```

5. LinearKernel

$$k(x, y) = x^T y + c$$

```
@jit
def LinearKernel(Xa, Xb):
    Xa = np.array(Xa)
    Xb = np.array(Xb)
    return Xa.T @ Xb
```

6. RBFkernel

$$k(x, y) = e^{\frac{-||x-y||^2}{2\sigma^2}} = e^{-\gamma||x-y||^2}$$

```
@jit
def RBFkernel(Xa, Xb):
    gamma = 1/784
    difference = np.array([Xa[i] - Xb[i] for i in range(len(Xa))])
    return np.exp(-1 * gamma * (difference.T @ difference))
```

7. GetDataByKernel

```
@jit
def GetDataByKernel(X_train):
    length = X_train.shape[0]
    output = []
    for i in range(length):
        tmp = []
        tmp.append(i + 1)
        for j in range(length):
            val = LinearKernel(X_train[i], X_train[j]) + RBFkernel(X_train[i], X_train[j])
            tmp.append(val)
        output.append(tmp)
    return np.array(output)
```

8. After kernel

```
new_train = GetDataByKernel(X_train)
```

```
new_test = GetDataByKernel(X_test)
```

```
model = svm_train(Y_train, new_train, '-t 4')
label, acc, val = svm_predict(Y_test, new_test, model)
```

- **Experiment settings and results：**

1. Q1：Among three kernels, the performance speed：Linear > Polynomial > RBF. I found if I use SVR model, the RBF would be slower, but more accurate.

   Result (Accuracy)：

   Linear：95.08%

   Polynomial：95.76%

   RBF：95.32%

   if I don't set "-r"：

```
# Linear kernel
model = svm_train(Y_train, X_train, '-t 0')
label, acc, val = svm_predict(Y_test, X_test, model)
```

```
Accuracy = 95.08% (2377/2500) (classification)
```

```
# Polynomial kernel
model = svm_train(Y_train, X_train, '-t 1')
label, acc, val = svm_predict(Y_test, X_test, model)
```

```
Accuracy = 34.68% (867/2500) (classification)
```

```
# RBF kernel
model = svm_train(Y_train, X_train, '-t 2')
label, acc, val = svm_predict(Y_test, X_test, model)
```

```
Accuracy = 95.32% (2383/2500) (classification)
```

   the accuracy in polynomial would be worse

2. Q2：

   In cost, I set three number：0.125, 0.5, 2

   In gamma, I set three number：0.5, 2, 8

   Result (Best parameter)：

   Linear：max_cost = 0.125, max_gamma = 0.5, max_acc = 97; after fitting,

   Accuracy = 95%

RBF：max_cost = 0.125, max_gamma = 8, max_acc = 78.96; after fitting, Accuracy = 78.64%

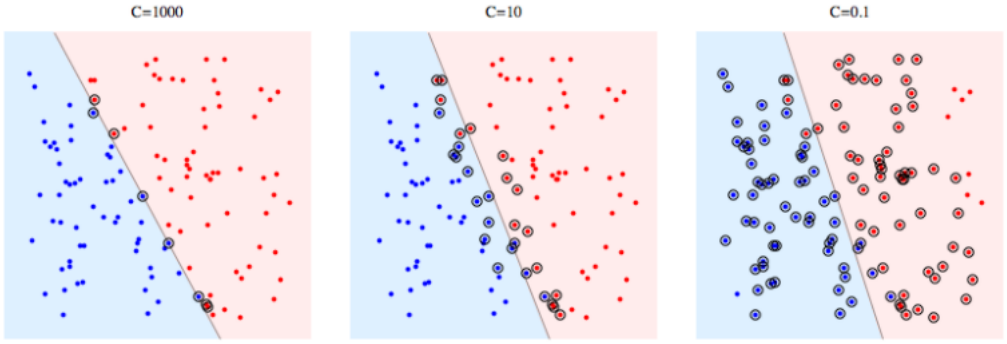
3. Q3.
   I didn't have result because the error "segmentation fault". After searching the error, it said the memory was not enough.

- **Observation and discussion：**

In Q1, if I didn't set "-b", meaning probability_estimates: whether to train a SVC or SVR model for probability estimates, 0 or 1 (default 0), then, the accuracy in polynomial would be worse.

In Q2, I didn't set many numbers to run because the runtime is long. However, if cost is bigger, support vectors is fewer and it's easy to be overfitting. If cost is smaller, support vectors is more and the margin is bigger.

$$L(w,b,\xi,\alpha,\mu)=\frac{1}{2}\|w\|^2+C\sum_i\xi_i-\sum_{i=1}^{N}\alpha_i\left[y_i\left(w^Tx_i-b\right)-1+\xi_i\right]-\sum_{i=1}^{N}\mu_i\xi_i$$



Gamma = How far the influence of a single training example reaches
If gamma is bigger, the influence of range by data points is closer and it's easy to be overfitting. If gamma is smaller, the influence of range by data points is more far.

In Q3, I have two questions, one is the performance of GetDataByKernel is bad, the other, I could train the model, but I couldn't predict the testing data. The error is "segmentation default". Thus, I thought the data produced by GetDataByKernel, occupying too much memory to work successfully when predicting the test data. Then, I use library "lamba" to solve the slow performance and its performance is obvious improve.