SVM

November 9, 2020

[1]:
```python
import numpy as np
from libsvm.python.svmutil import *
import matplotlib.pyplot as plt

from IPython.display import clear_output
```

[2]:
```python
# Read in the training and testing data
y_train, x_train = svm_read_problem('ncRNA_s.train.txt')
y_test, x_test = svm_read_problem('ncRNA_s.test.txt')
```

[3]:
```python
# Default constants
CV_SIZE = 1000
N_FOLD = 5
FOLD_SIZE = 200
```

1 Problem 1

[4]:
```python
tunning_param_list = [2**p for p in range(-4, 9)]

tunning_param_list
```

[4]:
```
[0.0625, 0.125, 0.25, 0.5, 1, 2, 4, 8, 16, 32, 64, 128, 256]
```

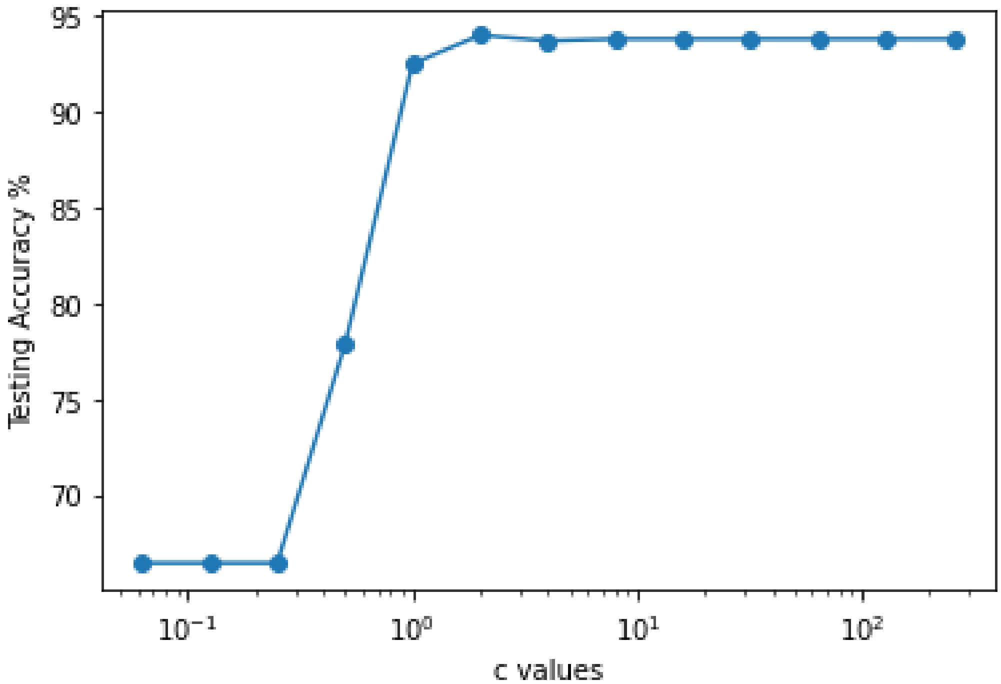
[5]:
```python
def svm_accuracy(y_train, x_train, y_test, x_test, c=1, t=0, g=None):

    gamma_arg = ''
    if t == 2 and g is not None:
        gamma_arg = f'-g {g}'

    # Train SVM on the training set with different c values
    model = svm_train(y_train, x_train, f'-c {c} -t {t} {gamma_arg}')

    # Test the model with testing set
    _, accuracy, _ = svm_predict(y_test, x_test, model, options='-q')

    return accuracy[0]
```

[6]:
```
history = []

for c in tunning_param_list:

    accuracy = svm_accuracy(y_train, x_train, y_test, x_test, c=c)

    history.append(accuracy)

history
```

[6]:
```
[66.43356643356644,
 66.43356643356644,
 66.43356643356644,
 77.82217782217782,
 92.5074925074925,
 94.00599400599401,
 93.7062937062937,
 93.8061938061938,
 93.8061938061938,
 93.8061938061938,
 93.8061938061938,
 93.8061938061938,
 93.8061938061938]
```

The test accuracy increased significantly from around 70% to 94%, as we increase the parameter c from values smaller than 1 to values that are greater than 1.

[7]:
```
plt.plot(tunning_param_list, history, 'o-')
plt.xscale('log')
plt.xlabel('c values')
plt.ylabel('Testing Accuracy %')
plt.show()
```



2 Problem 2

2.1 Part 1

[8]:
```
# Randomly select 1000 samples for cross-validation training
np.random.seed(5526)
y_size = len(y_train)
order = np.random.choice(range(y_size), size=CV_SIZE, replace=False)

y_cv = [[] for f in range(N_FOLD)]
x_cv = [[] for f in range(N_FOLD)]

i = 0

# Construct 5 subset for 5-fold cross-validation
for index in order:

    fold = i // FOLD_SIZE

    y_cv[fold].append(y_train[index])
    x_cv[fold].append(x_train[index])
```

```
    i += 1
```

[9]:
```
# Initialize an accuracy matrix
matrix_dim = len(tunning_param_list)
accuracy_matrix = np.zeros((matrix_dim, matrix_dim))

accuracy_matrix.shape
```

[9]: (13, 13)

[10]:
```
best_c = 0
best_g = 0
best_cv_accuracy = 0

for i in range(matrix_dim):

    for j in range(matrix_dim):

        fold_index = list(range(N_FOLD))

        accuracy = 0

        for index in range(N_FOLD):

            fold_index.remove(index)

            # Construct training set
            y_train_folds = [y_cv[i] for i in fold_index]
            y_cv_train = [sample for fold in y_train_folds for sample in fold]

            x_train_folds = [x_cv[i] for i in fold_index]
            x_cv_train = [sample for fold in x_train_folds for sample in fold]

            # Construct validation set
            y_cv_valid = y_cv[index]
            x_cv_valid = x_cv[index]

            # Restore fold_index
            fold_index = list(range(N_FOLD))

            # Evaluate the model for a given pair of c and gamma
            c = tunning_param_list[i]
            g = tunning_param_list[j]
            accuracy += svm_accuracy(y_cv_train, x_cv_train, y_cv_valid,␣
↪x_cv_valid, c=c, t=2, g=g)

        accuracy = accuracy / N_FOLD
```

```
        # Log the best parameters so far
        if accuracy > best_cv_accuracy:
            best_cv_accuracy = accuracy
            best_c = c
            best_g = g

        # Update the accuracy_matrix
        clear_output(wait=True)
        accuracy_matrix[i][j] = accuracy
        print(accuracy_matrix)
        print()
        print(f'Best c = {best_c}; Best gamma = {best_g}')
```

```
[[67.4 67.4 67.4 67.4 67.4 67.4 67.4 67.4 67.4 67.4 67.4 67.4 67.4]
 [67.4 67.4 67.4 67.4 67.4 67.4 67.4 67.4 67.5 67.6 67.5 67.4 67.4]
 [67.4 67.4 67.4 67.4 67.4 67.4 67.5 70.4 73.9 73.  71.7 68.4 67.4]
 [67.4 67.4 67.4 67.4 67.4 71.7 81.  83.8 83.6 80.3 75.8 72.6 68.5]
 [67.4 67.4 67.4 67.8 82.5 91.5 91.7 90.6 89.2 86.7 82.  77.5 72.8]
 [67.4 67.4 69.2 89.5 94.2 93.6 93.8 92.5 89.5 88.  84.2 79.  74.4]
 [67.4 69.7 90.9 95.3 94.8 94.8 94.2 93.1 91.1 87.9 83.9 78.6 74.1]
 [70.6 91.5 95.7 95.2 95.3 95.1 94.5 93.3 90.9 88.5 84.3 78.5 74.1]
 [92.  95.5 95.4 95.3 95.3 95.1 94.6 93.1 90.3 88.3 84.3 78.4 74.1]
 [95.6 95.5 95.  95.1 95.6 95.4 94.8 92.1 89.7 87.3 83.9 78.4 74.1]
 [95.5 95.7 95.4 95.3 95.4 95.4 93.2 91.4 89.7 86.7 83.9 78.4 74.1]
 [95.8 95.7 95.2 95.3 95.6 94.5 92.5 90.3 88.5 86.1 83.9 78.4 74.1]
 [95.6 95.6 95.4 95.1 95.2 93.7 91.8 89.6 87.1 86.1 83.9 78.4 74.1]]

Best c = 128; Best gamma = 0.0625
```

2.2 Part 2

The accuracy we can achieve by tunning parameters from cross-validation is:

[11]:
```
# Find the accuracy using the best c and gamma
accuracy = svm_accuracy(y_train, x_train, y_test, x_test, c=best_c, t=2,␣
 ↪g=best_g)

accuracy
```

[11]: 94.2057942057942