ex2

March 14, 2017

In [40]:
```
import numpy as np
import sys
from sklearn.svm import SVC
from itertools import product
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
import json

#train data = argv[1]
#train labels = argv[2]
#test data = argv[3]
#test labels = argv[4]

bases = ['A', 'C', 'G', 'T']
kmer_list_2 = sorted([''.join(p) for p in product(bases, repeat=2)])
kmer_list_3 = sorted([''.join(p) for p in product(bases, repeat=3)])
kmer_list_4 = sorted([''.join(p) for p in product(bases, repeat=4)])
train_2mers = []
train_3mers = []
train_4mers = []
test_2mers = []
test_3mers = []
test_4mers = []
train_data = []
test_data = []
train_Y = []
test_Y = []
train_Y_data = []
test_Y_data = []

kmers = kmer_list_2 + kmer_list_3 + kmer_list_4

fold_list = np.zeros((2175, 1))

train_X = np.zeros((2175, 336))
test_X = np.zeros((1000, 336))

i = 0
```

```
with open('Kfolds.txt') as f:
    f.readline()
    for line in f:
        stripped_line = line.strip().split(' ')
        fold_list[i][0] = stripped_line[1]
        i += 1

with open('train_sequence.txt') as f:
    f.readline()
    for line in f:
        train_data.append(line.strip())

for sequence in train_data:
    train_2mer_list = {kmer: 0 for kmer in kmer_list_2}
    train_3mer_list = {kmer: 0 for kmer in kmer_list_3}
    train_4mer_list = {kmer: 0 for kmer in kmer_list_4}
    for i in range(0, 59):
        train_2mer_list[sequence[i:i+2]] += 1
        if i <= 57:
            train_3mer_list[sequence[i:i+3]] += 1
        if i <= 56:
            train_4mer_list[sequence[i:i+4]] += 1

    train_2mers.append(train_2mer_list)
    train_3mers.append(train_3mer_list)
    train_4mers.append(train_4mer_list)


with open('train_kmer_counts.txt', 'w') as f:
    for d in train_2mers + train_3mers + train_4mers:
        f.write(str(d))
        f.write("\n")

for i in range(0, 2175):
    values = []
    for k in kmers:
        if len(k) == 2:
            values.append(train_2mers[i][k])
        elif len(k) == 3:
            values.append(train_3mers[i][k])
        else:
            values.append(train_4mers[i][k])
    train_X[i] = values


mean = np.mean(train_X, axis=0)
std = np.std(train_X, axis=0)
```

```
for i in range(0, 2175):
    for j in range(0, 336):
        train_X[i][j] = (train_X[i][j] - mean[j]) / std[j]

with open('train_label.txt') as f:
    f.readline()
    for line in f:
        train_Y.append(line.strip().split(' '))

for i in range(0, 2175):
    train_Y_data.append(train_Y[i][2])


with open('test_sequence.txt') as f:
    f.readline()
    for line in f:
        test_data.append(line.strip())

for sequence in test_data:
    test_2mer_list = {kmer: 0 for kmer in kmer_list_2}
    test_3mer_list = {kmer: 0 for kmer in kmer_list_3}
    test_4mer_list = {kmer: 0 for kmer in kmer_list_4}
    for i in range(0, 59):
        test_2mer_list[sequence[i:i+2]] += 1
        if i <= 57:
            test_3mer_list[sequence[i:i+3]] += 1
        if i <= 56:
            test_4mer_list[sequence[i:i+4]] += 1

    test_2mers.append(test_2mer_list)
    test_3mers.append(test_3mer_list)
    test_4mers.append(test_4mer_list)

with open('test_kmer_counts.txt', 'w') as f:
    for d in test_2mers + test_3mers + test_4mers:
        f.write(str(d))
        f.write("\n")


for i in range(0, 1000):
    values = []
    for k in kmers:
        if len(k) == 2:
            values.append(test_2mers[i][k])
        elif len(k) == 3:
            values.append(test_3mers[i][k])
        else:
            values.append(test_4mers[i][k])
```

```
        test_X[i] = values

    test_mean = np.mean(test_X, axis=0)
    test_std = np.std(test_X, axis=0)
    for i in range(0, 1000):
        for j in range(0, 336):
            test_X[i][j] = (test_X[i][j] - test_mean[j]) / test_std[j]

    with open('test_label.txt') as f:
        f.readline()
        for line in f:
            test_Y.append(line.strip().split(' '))

    for i in range(0, 1000):
        test_Y_data.append(test_Y[i][2])


    linear_svc = SVC(kernel='linear')
    deg_3_svc = SVC(kernel='poly', degree=3)
    deg_4_svc = SVC(kernel='poly', degree=4)
    deg_6_svc = SVC(kernel='poly', degree=6)
    gaussian_svc = SVC()

    linear_svc.fit(train_X, train_Y_data)
    deg_3_svc.fit(train_X, train_Y_data)
    deg_4_svc.fit(train_X, train_Y_data)
    deg_6_svc.fit(train_X, train_Y_data)
    gaussian_svc.fit(train_X, train_Y_data)


    linear_scores = cross_val_score(linear_svc, train_X, train_Y_data, groups=
    deg_3_scores = cross_val_score(deg_3_svc, train_X, train_Y_data, groups=fo
    deg_4_scores = cross_val_score(deg_4_svc, train_X, train_Y_data, groups=fo
    deg_6_scores = cross_val_score(deg_6_svc, train_X, train_Y_data, groups=fo
    gaussian_scores = cross_val_score(gaussian_svc, train_X, train_Y_data, gro


    predict_Y = gaussian_svc.predict(test_X)
    accuracy = gaussian_svc.score(test_X, test_Y_data)
    con_matrix = confusion_matrix(test_Y_data, predict_Y)

    precision = float(con_matrix[0,0]) / (con_matrix[0,0] + con_matrix[1, 0])

    recall = float(con_matrix[0,0]) / (con_matrix[0,0] + con_matrix[0,1])
```

```
In [3]: con_matrix
```

```
Out[3]: array([[386, 142],
               [122, 350]])
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: