In [1]:
```
import numpy as np
import random
```

In [2]:
```
f = open("pa3train.txt","r")
train = [line.strip() for line in f]
train = [[int(i) for i in line.split()] for line in train]

f1 = open("pa3test.txt","r")
test = [line.strip() for line in f1]
test = [[int(i) for i in line.split()] for line in test]

f2 = open("pa3dictionary.txt","r")
dictionary = [line.strip() for line in f2]
```

In [3]:
```
# 1: label1  -1: label2
train_12 = [line for line in train if line[-1] == 1 or line[-1] == 2]
train_12 = [line[:-1]+[1] if line[-1] == 1 else line[:-1]+[-1] for line
in train_12]

test_12 = [line for line in test if line[-1] == 1 or line[-1] == 2]
test_12 = [line[:-1]+[1] if line[-1] == 1 else line[:-1]+[-1] for line i
n test_12]
```

Q1

In [4]:
```
# Perceptron:
def perceptron(data, num_pass):
    data_s = data*num_pass
    w = [0]*(len(data_s[0])-1)

    for i in range(len(data_s)):
        x = data_s[i][:-1]
        y = data_s[i][-1]
        temp = y*np.dot(w,x)
        if temp <= 0:
            w = np.add(w,np.multiply(y,x))

    return w
```

In [5]:
```
for i in range(1,5):
    print(i, "pass")
    w = perceptron(train_12,i)
    train_result = [1 if np.dot(line[:-1],w) > 0 else -1
                    if np.dot(line[:-1],w) < 0 else random.choice([-1,1
]) for line in train_12]
    train_error = sum([train_result[i] != train_12[i][-1] for i in range
(len(train_12))])/len(train_12)
    print("train error :",train_error)

    test_result = [1 if np.dot(line[:-1],w) > 0 else -1
                   if np.dot(line[:-1],w) < 0 else random.choice([-1,1])
for line in test_12]
    test_error = sum([test_result[i] != test_12[i][-1] for i in range(le
n(test_12))])/len(test_12)
    print("test error :",test_error)
```

```
1 pass
train error : 0.04036697247706422
test error : 0.05305039787798409
2 pass
train error : 0.03761467889908257
test error : 0.058355437665782495
3 pass
train error : 0.02110091743119266
test error : 0.04509283819628647
4 pass
train error : 0.015596330275229359
test error : 0.04774535809018567
```

In [6]:
```
# voted perceptron
def votedperceptron(data, num_pass):

    data_s = data*num_pass
    wcpair = []

    w = [0]*(len(data_s[0])-1)
    c = 1
    wcpair += [(w,c)]
    for i in range(len(data_s)):
        x = data_s[i][:-1]
        y = data_s[i][-1]
        temp = y*np.dot(w,x)
        if temp <= 0:
            wcpair += [(w,c)]
            w = np.add(w,np.dot(y,x))
            c = 1
        else:
            c += 1
    wcpair += [(w,c)]

    return wcpair
```

In [7]:
```
for i in range(1,5):
    wcpair = votedperceptron(train_12,i)
    print(i,"pass")
    train_result = [sum([ c if np.dot(w,line[:-1]) > 0 else -c for w,c i
n wcpair]) for line in train_12]
    train_result = [1 if i > 0 else -1 if i < 0 else random.choice([1,-1
]) for i in train_result]
    train_error = sum([train_result[i] != train_12[i][-1] for i in range
(len(train_12))])/len(train_12)
    print("train error :",train_error)

    test_result = [sum([ c if np.dot(w,line[:-1]) > 0 else -c for w,c in
wcpair]) for line in test_12]
    test_result = [1 if i > 0 else -1 if i < 0 else random.choice([1,-1
]) for i in test_result]
    test_error = sum([test_result[i] != test_12[i][-1] for i in range(le
n(test_12))])/len(test_12)
    print("test error :",test_error)
```

```
1 pass
train error : 0.06697247706422019
test error : 0.08753315649867374
2 pass
train error : 0.04036697247706422
test error : 0.0610079575596817
3 pass
train error : 0.030275229357798167
test error : 0.04509283819628647
4 pass
train error : 0.024770642201834864
test error : 0.04509283819628647
```

In [8]:
```
def averagePerceptron(wcpair):
    return sum([np.dot(c,w) for w,c in wcpair])
```

In [9]:
```
for i in range(1,5):
    wcpair = votedperceptron(train_12,i)
    w = averagePerceptron(wcpair)
    print(i,"pass")
    train_result = [1 if np.dot(w,line[:-1]) > 0 else -1
                    if np.dot(line[:-1],w) < 0 else random.choice([-1,1
])for line in train_12]
    train_error = sum([train_result[i] != train_12[i][-1] for i in range
(len(train_12))])/len(train_12)
    print("train error :",train_error)

    test_result = [1 if np.dot(w,line[:-1]) > 0 else -1
                   if np.dot(line[:-1],w) < 0 else random.choice([-1,1])
for line in test_12]
    test_error = sum([test_result[i] != test_12[i][-1] for i in range(le
n(test_12))])/len(test_12)
    print("test error :",test_error)
```

```
1 pass
train error : 0.07706422018348624
test error : 0.11671087533156499
2 pass
train error : 0.05321100917431193
test error : 0.08222811671087533
3 pass
train error : 0.03669724770642202
test error : 0.0610079575596817
4 pass
train error : 0.03394495412844037
test error : 0.050397877984084884
```

Q2

In [10]:
```
w_avg = averagePerceptron(votedperceptron(train_12,3))

sort = sorted([(w_avg[i],i) for i in range(len(w_avg))])
lowest = [(i[1],dictionary[i[1]]) for i in sort[:3]]
print("lowest:",lowest)
highest = [(i[1],dictionary[i[1]]) for i in sort[-1:-4:-1]]
print("highest:",highest)
```

```
lowest: [(78, 'he'), (469, 'team'), (393, 'game')]
highest: [(438, 'file'), (466, 'program'), (203, 'line')]
```

Q3

In [11]:
```
train_1to6 = [[line[:-1]+[1] if line[-1] == i else line[:-1]+[-1] for li
ne in train] for i in range(1,7)]
C = [perceptron(l,1) for l in train_1to6]
```

In [12]:
```
test_report = []
for line in test:
    t = [np.dot(line[:-1],C[i]) > 0 for i in range(6)]
    if (sum(t) == 0 or sum(t) > 1):
        test_report += [("Don't know",line[-1])]
    else:
        test_report += [(t.index(1)+1,line[-1])]
```

In [13]:
```
confusion_matrix = np.zeros([7,6])
for t in test_report:
    i,j  = t
    if i == "Don't know":
        j -= 1
        confusion_matrix[6][j] += 1
    else:
        i -= 1
        j -= 1
        confusion_matrix[i][j] +=1
```

In [14]:
```
N = [sum([t[-1] == i for t in test_report]) for i in range(1,7)]
```

In [15]:
```
for i in range(7):
    for j in range(6):
        confusion_matrix[i][j] = confusion_matrix[i][j] / N[j]
```

In [16]:
```
print(confusion_matrix)
```

```
[[0.71891892 0.00520833 0.03428571 0.02173913 0.         0.        ]
 [0.01081081 0.65625    0.03428571 0.02717391 0.01282051 0.01851852]
 [0.         0.015625   0.37142857 0.         0.         0.02777778]
 [0.01621622 0.00520833 0.         0.69021739 0.         0.        ]
 [0.01621622 0.03125    0.07428571 0.00543478 0.80128205 0.12037037]
 [0.00540541 0.01041667 0.03428571 0.         0.07051282 0.49074074]
 [0.23243243 0.27604167 0.45142857 0.25543478 0.11538462 0.34259259]]
```

(a) The perceptron classifier has the highest accuracy for examples that belong to class 5.

(b) The perceptron classifier has the least accuracy for examples that belong to class 3.

(c) The perceptron classifier most often mistakenly classifies an example in class 6 as belonging to class 5. (i = 5, j = 6)