Result

Question 1

1 Pass Perception Train Error: 0.04036697247706422 Test Error: 0.05305039787798409

2 Pass Perception Train Error: 0.03853211009174312 Test Error: 0.058355437665782495

3 Pass Perception Train Error: 0.01926605504587156 Test Error: 0.04509283819628647

4 Pass Perception Train Error: 0.01834862385321101 Test Error: 0.04774535809018567

1 Pass Voted Perception Train Error: 0.06697247706422019 Test Error: 0.08753315649867374

2 Pass Voted Perception Train Error: 0.04036697247706422 Test Error: 0.0610079575596817

3 Pass Voted Perception Train Error: 0.030275229357798167 Test Error: 0.04509283819628647

4 Pass Voted Perception Train Error: 0.024770642201834864 Test Error: 0.04509283819628647

1 Pass Averaged Perception Train Error: 0.07889908256880734 Test Error: 0.11671087533156499

2 Pass Averaged Perception Train Error: 0.05321100917431193 Test Error: 0.08222811671087533

3 Pass Averaged Perception Train Error: 0.03669724770642202 Test Error: 0.0610079575596817

4 Pass Averaged Perception Train Error: 0.03211009174311927 Test Error: 0.050397877984084884

Question 2

Highest coordinates and words: [(438, 'file'), (466, 'program'), (203, 'line')]

Lowest coordinates and words: [(78, 'he'), (469, 'team'), (393, 'game')]

Question 3

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.

Code

Import Packages and Read Files

In [1]:

```
import numpy as np
import random
```

In [2]:

```
ft = open("pa3train.txt","r")
fs = open("pa3test.txt","r")
fd = open("pa3dictionary.txt","r")

train = [[int(i) for i in l.strip().split()] for l in ft]
test = [[int(i) for i in l.strip().split()] for l in fs]
dic = [l.strip() for l in fd]
```

In [3]:

```
sub_train = [l for l in train if l[-1] == 1 or l[-1] == 2]
sub_train = [l[:-1] + [-1] if l[-1] != 1 else l for l in sub_train]
```

In [4]:

```
sub_test = [l for l in test if l[-1] == 1 or l[-1] == 2]
sub_test = [l[:-1] + [-1] if l[-1] != 1 else l for l in sub_test]
```

In [5]:

```
ova_train = [[l[:-1]+[1] if l[-1] == x else l[:-1]+[-1] for l in train]
for x in range(1,7,1)]
```

Functions

In [6]:

```
def perception(d, n):
    length = (len(d[0]) - 1)
    wi = [0] * length
    d = d * n

    for i in range(len(d)):
        xi = d[i][:-1]
        yi = d[i][-1]
        t = yi * np.dot(wi, xi)
        if(t <= 0):
            wi = np.add(wi, np.multiply(xi,yi))
    return wi
```

In [7]:

```
def voted_perception(d, n):
    length = (len(d[0]) - 1)
    wi = [0] * length
    d = d * n
    c = 1
    pair = [(wi,c)]

    for i in range(len(d)):
        xi = d[i][:-1]
        yi = d[i][-1]
        t = yi * np.dot(wi, xi)
        if(t <= 0):
            pair.append((wi,c))
            c = 1
            wi = np.add(wi, np.multiply(xi,yi))
        else:
            c += 1
    pair.append((wi,c))
    return pair
```

In [8]:

```
def averaged_perception(pair):
    return sum([np.dot(wi, ci) for wi, ci in pair])
```

Question 1

In [9]:

```
for i in range(1,5):
    w = perception(sub_train,i)
    result_train = [1 if np.dot(l[:-1],w) > 0 else -1
                    if np.dot(l[:-1],w) < 0 else random.choice([-1,1]) for l in sub_train]
    train_error = sum([result_train[i] != sub_train[i][-1] for i in range(len(sub_train))])/len(sub_train)

    result_test = [1 if np.dot(l[:-1],w) > 0 else -1
                   if np.dot(l[:-1],w) < 0 else random.choice([-1,1]) for l in sub_test]
    test_error = sum([result_test[i] != sub_test[i][-1] for i in range(len(sub_test))])/len(sub_test)

    print(i, "Pass Perception")
    print("Train Error: ",train_error)
    print("Test Error: ",test_error)
    print("")
```

```
1 Pass Perception
Train Error:  0.040366972477064222
Test Error:  0.05305039787798409

2 Pass Perception
Train Error:  0.03853211009174312
Test Error:  0.058355437665782495

3 Pass Perception
Train Error:  0.01926605504587156
Test Error:  0.04509283819628647

4 Pass Perception
Train Error:  0.01834862385321101
Test Error:  0.04774535809018567
```

In [15]:

```
for i in range(1,5):
    pair = voted_perception(sub_train,i)
    result_train = [sum([c if np.dot(w,l[:-1]) > 0 else -c for w,c in pair]) for l in sub_train]
    result_train = [1 if x > 0 else -1 if x < 0 else random.choice([1,-1]) for x in result_train]
    train_error = sum([result_train[i] != sub_train[i][-1] for i in range(len(sub_train))])/len(sub_train)

    result_test = [sum([c if np.dot(w,l[:-1]) > 0 else -c for w,c in pair]) for l in sub_test]
    result_test = [1 if x > 0 else -1 if x < 0 else random.choice([1,-1]) for x in result_test]
    test_error = sum([result_test[i] != sub_test[i][-1] for i in range(len(sub_test))])/len(sub_test)

    print(i, "Pass Voted Perception")
    print("Train Error: ",train_error)
    print("Test Error: ",test_error)
    print("")
```

```
1 Pass Voted Perception
Train Error:  0.06697247706422019
Test Error:  0.08753315649867374

2 Pass Voted Perception
Train Error:  0.04036697247706422
Test Error:  0.0610079575596817

3 Pass Voted Perception
Train Error:  0.030275229357798167
Test Error:  0.04509283819628647

4 Pass Voted Perception
Train Error:  0.024770642201834864
Test Error:  0.04509283819628647
```

In [16]:

```
for i in range(1,5):
    pair = voted_perception(sub_train,i)
    w = averaged_perception(pair)
    result_train = [1 if np.dot(l[:-1],w) > 0 else -1
                    if np.dot(l[:-1],w) < 0 else random.choice([-1,1]) for l in sub_train]
    train_error = sum([result_train[i] != sub_train[i][-1] for i in range(len(sub_train))])/len(sub_train)

    result_test = [1 if np.dot(l[:-1],w) > 0 else -1
                   if np.dot(l[:-1],w) < 0 else random.choice([-1,1]) for l in sub_test]
    test_error = sum([result_test[i] != sub_test[i][-1] for i in range(len(sub_test))])/len(sub_test)

    print(i, "Pass Averaged Perception")
    print("Train Error: ",train_error)
    print("Test Error: ",test_error)
    print("")
```

```
1 Pass Averaged Perception
Train Error:  0.07889908256880734
Test Error:  0.11671087533156499

2 Pass Averaged Perception
Train Error:  0.05321100917431193
Test Error:  0.08222811671087533

3 Pass Averaged Perception
Train Error:  0.03669724770642202
Test Error:  0.0610079575596817

4 Pass Averaged Perception
Train Error:  0.03211009174311927
Test Error:  0.050397877984084884
```

Question 2

In [13]:

```
pair = voted_perception(sub_train, 3)
w = averaged_perception(pair)
values = [(w[i], i) for i in range(len(w))]
values = sorted(values)
most_negative = [(l[1], dic[l[1]]) for l in values[:3]]
values.reverse()
most_positive = [(l[1], dic[l[1]]) for l in values[:3]]
print("Most strongly positive coordinates and words:")
print(most_positive)
print("")
print("Most strongly negative coordinates and words:")
print(most_negative)
```

```
Most strongly positive coordinates and words:
[(438, 'file'), (466, 'program'), (203, 'line')]

Most strongly negative coordinates and words:
[(78, 'he'), (469, 'team'), (393, 'game')]
```

Question 3

In [14]:

```
classes = [perception(l,1) for l in ova_train]
confusion_component = []

for l in test:
    t = [(np.dot(l[:-1], classes[i]) > 0) for i in range(6)]
    if(sum(t) != True):
        confusion_component.append(("X", l[-1]))
    else:
        confusion_component.append((t.index(1)+1, l[-1]))

confusion_mat = np.zeros([7,6])
for c in confusion_component:
    x,y = c
    if(x == "X"):
        y -= 1
        confusion_mat[6][y] += 1
    else:
        x -= 1
        y -= 1
        confusion_mat[x][y] += 1

N = [sum([c[-1] == i for c in confusion_component]) for i in range(1,7,1
)]

for i in range(len(confusion_mat)):
    for j in range(len(confusion_mat[0])):
        confusion_mat[i][j] /= N[j]

print("Confusion Matrix:")
print("")
print(confusion_mat)
```

```
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]]
```