PA1

In [1]:
```
f1 = open("pa1train.txt", "r")
pa1train = [line.strip() for line in f1]

pa1train = [[int(i) for i in line.split()] for line in pa1train]
train = [i[:784] for i in pa1train]
train_label = [i[-1] for i in pa1train]
```

In [2]:
```
f2 = open("pa1validate.txt","r")
pa1validate = [line.strip() for line in f2]
pa1validate = [[int(i) for i in line.split()] for line in pa1validate]

validate = [i[:784] for i in pa1validate]
validate_label = [i[-1] for i in pa1validate]
```

In [3]:
```
f3 = open("pa1test.txt","r")
pa1test = [line.strip() for line in f3]
pa1test = [[int(i) for i in line.split()] for line in pa1test]
test = [i[:784] for i in pa1test]
test_label = [i[-1] for i in pa1test]
```

Q1:

In [4]:
```
def dist(a,b):
    return sum([(a[i]-b[i])**2 for i in range(len(a))])
```

In [5]:
```
def total_dist(data1, training):
    total = []
    for i in range(len(data1)):
        q = []
        for j in range(len(training)):
            q+=[(dist(data1[i],training[j]),j)]
        total += [[l[1] for l in sorted(q)]]
    return total
```

In [6]:
```
dist_total = total_dist(train,train)
```

In [7]:
```
validate_total = total_dist(validate,train)
```

In [8]:
```
test_total = total_dist(test,train)
```

In [9]:
```
import random
def majority(list1):
    dict1 = dict()
    for i in list1:
        if i not in dict1:
            dict1[i] = 0
        dict1[i] += 1
    major =  [j for j in dict1 if dict1[j] == max([dict1[i] for i in dic
t1])]
    return random.choice(major)
```

In [10]:
```
def KNN(k):
    inx = [[train_label[j] for j in l[:k]] for l in dist_total]
    train_label_1 = [majority(l) for l in inx]
    train_error = sum(train_label[i] != train_label_1[i] for i in range(
len(train_label)))/len(train_label)
    print("Training error:",train_error)

    inx_valid = [[train_label[j] for j in l[:k]] for l in validate_total
]
    validate_label_1 = [majority(l) for l in inx_valid]
    validate_error = sum(validate_label[i] != validate_label_1[i] for i
in range(len(validate_label)))/len(validate_label)
    print("Validation error:",validate_error)

    inx_test = [[train_label[j] for j in l[:k]] for l in test_total]
    test_label_1 = [majority(l) for l in inx_test]
    test_error = sum(test_label[i] != test_label_1[i] for i in range(len
(test_label)))/len(test_label)
    print("Test error:",test_error)
```

In [11]:
```
KNN(1)
```

```
Training error: 0.0
Validation error: 0.082
Test error: 0.094
```

In [12]:
```
KNN(5)
```

```
Training error: 0.055
Validation error: 0.096
Test error: 0.092
```

In [13]:
```
KNN(9)
```

```
Training error: 0.07
Validation error: 0.109
Test error: 0.102
```

In [14]:
```
KNN(15)
```

```
Training error: 0.093
Validation error: 0.105
Test error: 0.116
```

In [15]:
```
KNN(3)
```

```
Training error: 0.0425
Validation error: 0.098
Test error: 0.09
```

Answer: The classifier of k=1 performs the best on validation data.

Its test error is 0.094.

Q2:

In [16]:
```
f4 = open("projection.txt", "r")
projection = [line.strip() for line in f4]
project = []
for i in range(20):
    p = []
    for line in projection:
        p += [float(line.split()[i])]
    project += [p]
```

In [17]:
```
def dot(list1,list2):
    return sum([list1[i] * list2[i] for i in range(len(list1))])
def add(list1,list2):
    return [list1[i] + list2[i] for i in range(len(list1))]
def mul(c, list1):
    return [c*list1[i] for i in range(len(list1))]
```

In [18]:
```
def Projection(matrix):
    proj = []
    for vec in matrix:
        q = [0]*len(vec)
        for i in project:
            q = add(q, mul(dot(vec,i),i))
        proj += [q]
    return proj
```

In [19]:
```
train_proj = Projection(train)
dist_total_proj = total_dist(train_proj,train_proj)
```

In [20]:
```
validation_proj = Projection(validate)
validation_total_proj = total_dist(validation_proj,train_proj)
```

In [21]:
```
test_proj = Projection(test)
test_total_proj = total_dist(test_proj,train_proj)
```

In [22]:
```
def KNN_proj(k):
    inx = [[train_label[j] for j in l[:k]] for l in dist_total_proj]
    train_label_1 = [majority(l) for l in inx]
    train_error = sum(train_label[i] != train_label_1[i] for i in range(
len(train_label)))/len(train_label)
    print("Training error:",train_error)

    inx_valid = [[train_label[j] for j in l[:k]] for l in validation_tot
al_proj]
    validate_label_1 = [majority(l) for l in inx_valid]
    validate_error = sum(validate_label[i] != validate_label_1[i] for i
in range(len(validate_label)))/len(validate_label)
    print("Validation error:",validate_error)

    inx_test = [[train_label[j] for j in l[:k]] for l in test_total_proj
]
    test_label_1 = [majority(l) for l in inx_test]
    test_error = sum(test_label[i] != test_label_1[i] for i in range(len
(test_label)))/len(test_label)
    print("Test error:",test_error)
```

In [70]:
```
KNN_proj(1)
```

```
Training error: 0.0
Validation error: 0.32
Test error: 0.314
```

In [71]:
```
KNN_proj(5)
```

```
Training error: 0.1905
Validation error: 0.299
Test error: 0.305
```

In [72]:
```
KNN_proj(9)
```

```
Training error: 0.2295
Validation error: 0.297
Test error: 0.285
```

In [73]:
```
KNN_proj(15)
```

```
Training error: 0.261
Validation error: 0.295
Test error: 0.297
```

In [69]:
```
KNN_proj(3)
```

```
Training error: 0.162
Validation error: 0.314
Test error: 0.3
```

Answer: The classifier of k=15 performs the best on validation data.

Its test error is 0.297.

After projection, the classification accuracy becomes less accurate.

The running becomes faster when run on projected data.

In [ ]: