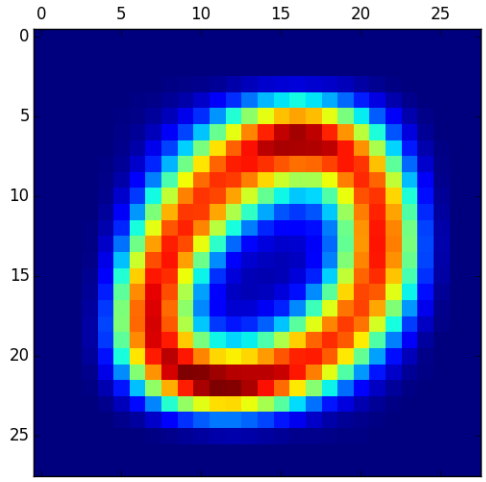
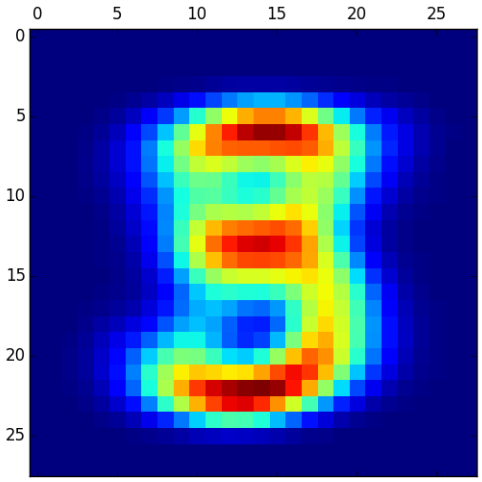
CS189–FALL 2015 — Homework 7 Write up

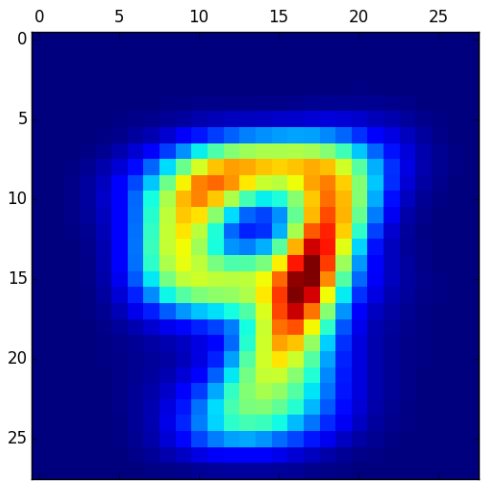
ZUBO GU, SID 25500921, gu.zubo@berkeley.edu

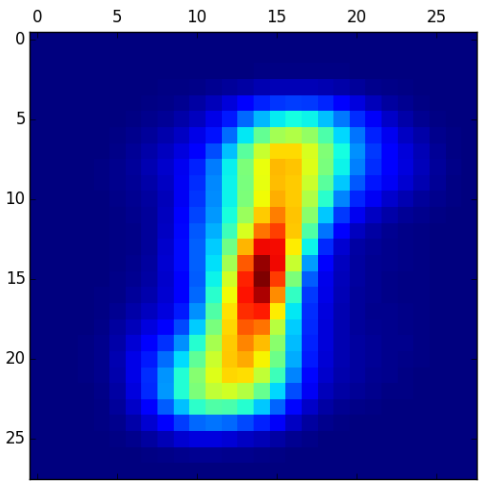
Problem 1

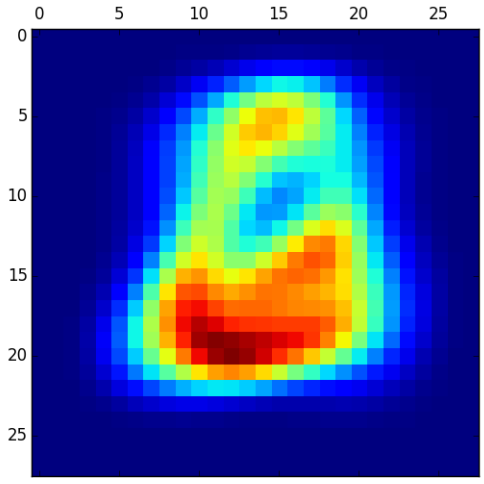
1.2 . With K = 5, visualize the cluster centers as below.



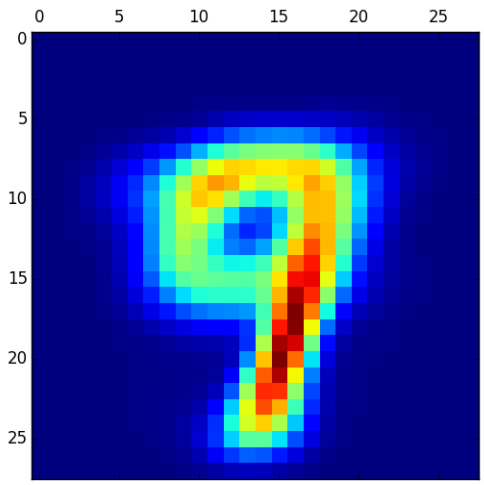


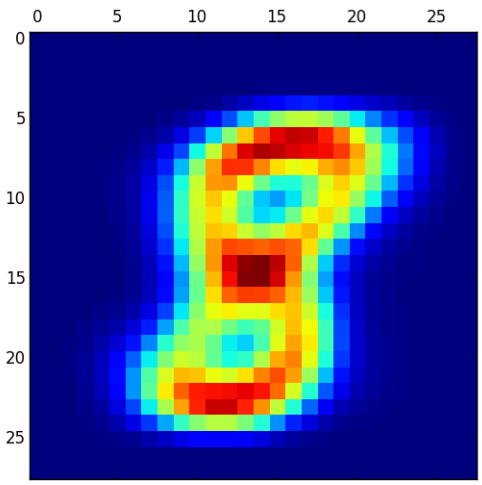


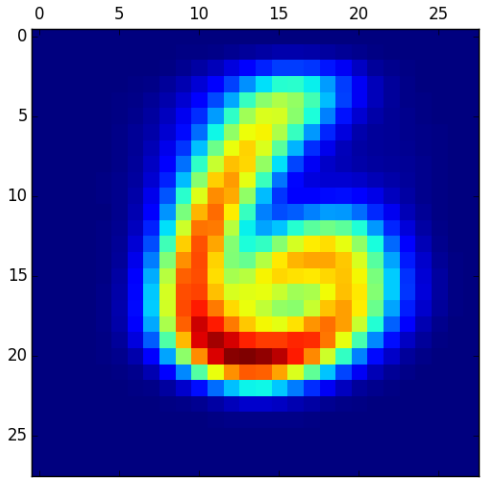


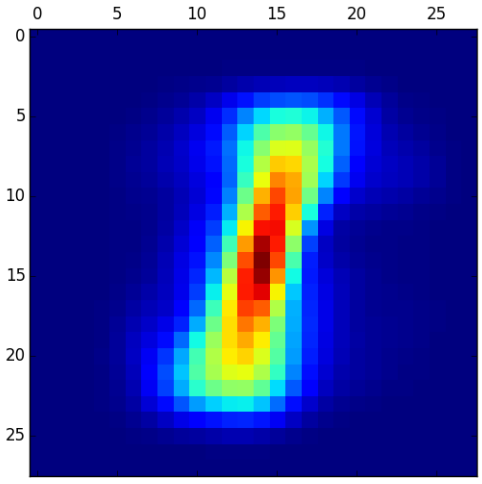


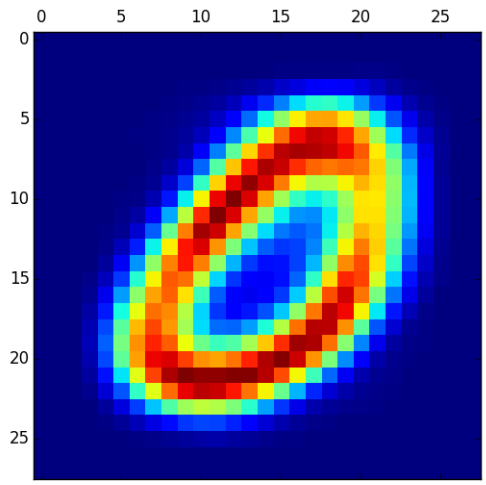
With K = 10, visualize the cluster centers as below.

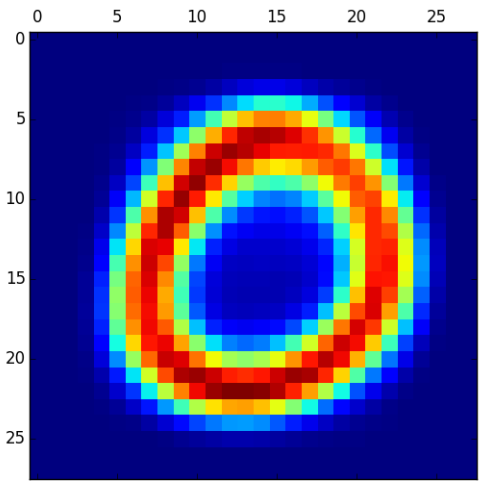

0 5 10 15 20 25
0 5 10 15 20 25

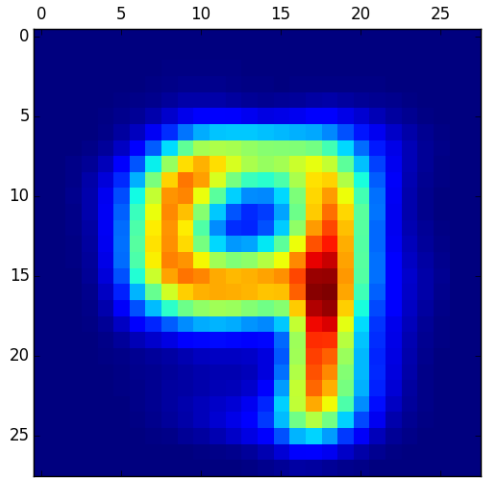

0 5 10 15 20 25
0 5 10 15 20 25

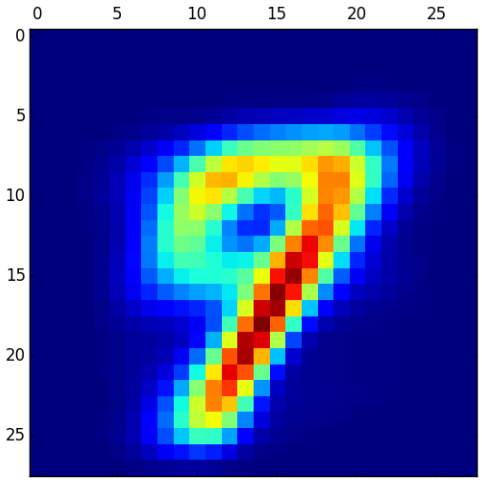

0 5 10 15 20 25
0 5 10 15 20 25

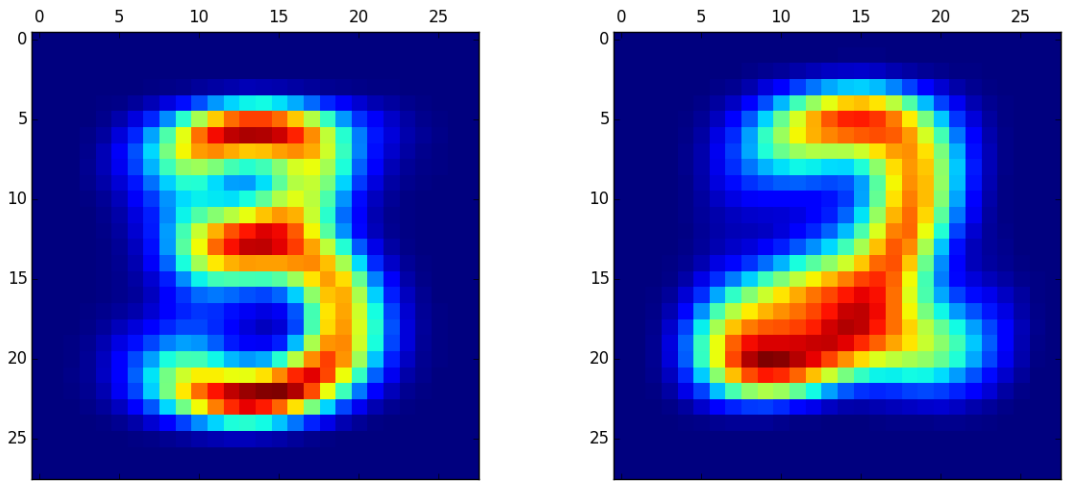

0 5 10 15 20 25
0 5 10 15 20 25


0 5 10 15 20 25
0 5 10 15 20 25

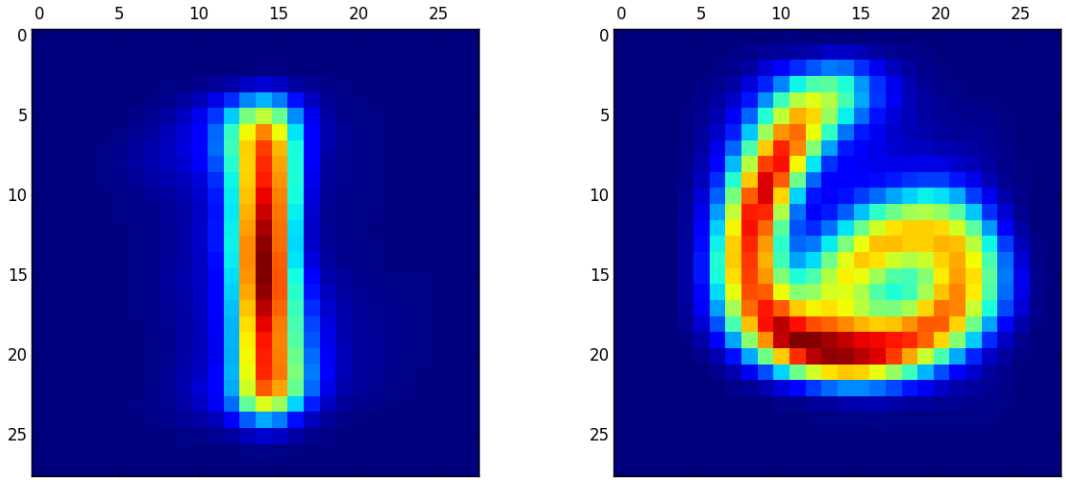

0 5 10 15 20 25
0 5 10 15 20 25

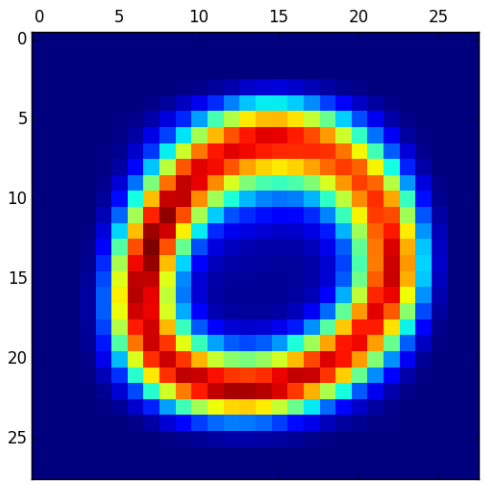

0 5 10 15 20 25
0 5 10 15 20 25

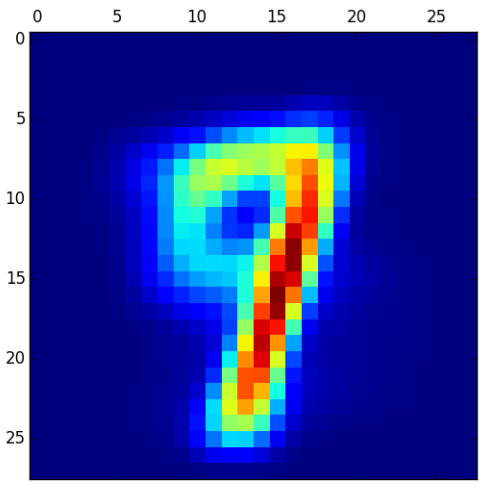

0 5 10 15 20 25
0 5 10 15 20 25

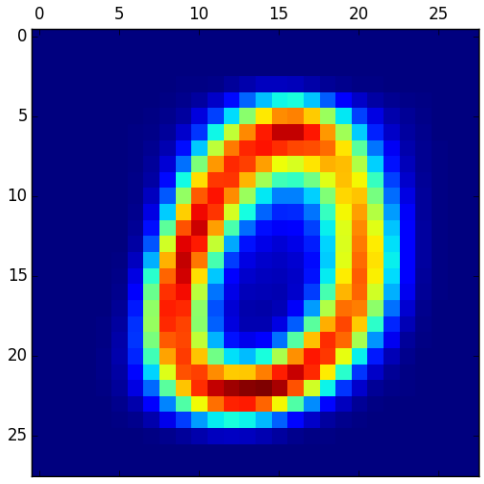


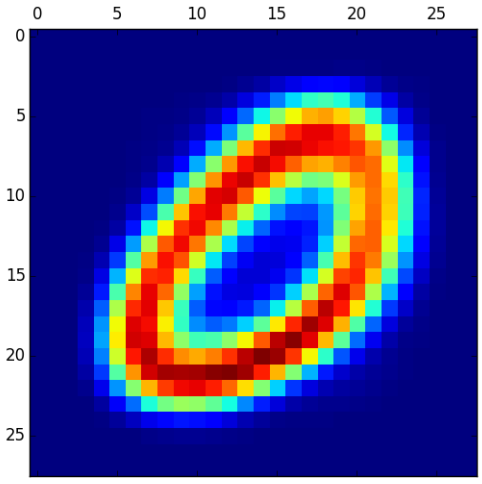
With K = 20, visualize the cluster centers as below.

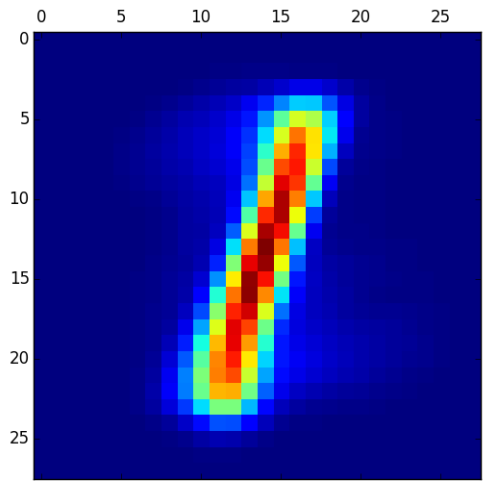


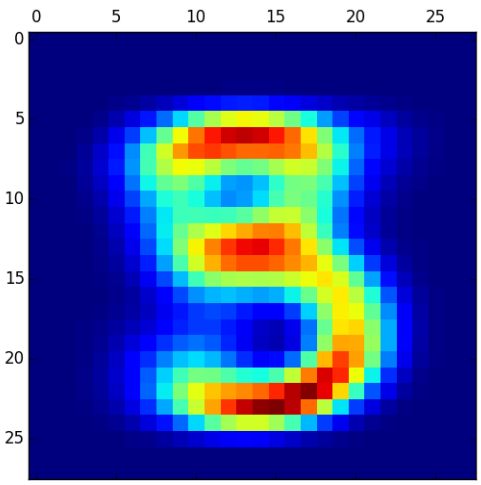

0 5 10 15 20 25
0 5 10 15 20 25

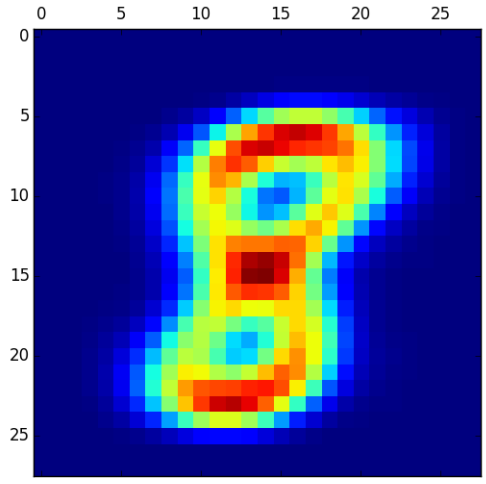

0 5 10 15 20 25
0 5 10 15 20 25

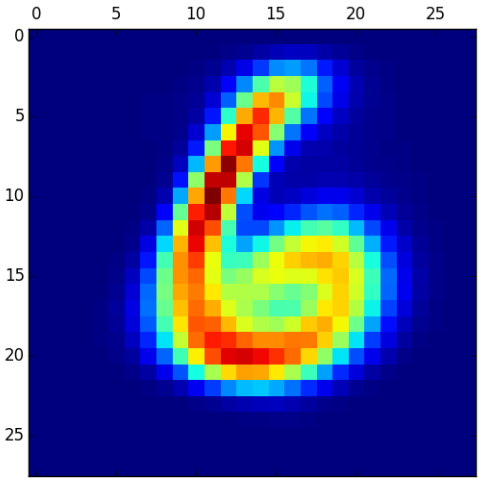

0 5 10 15 20 25
0 5 10 15 20 25

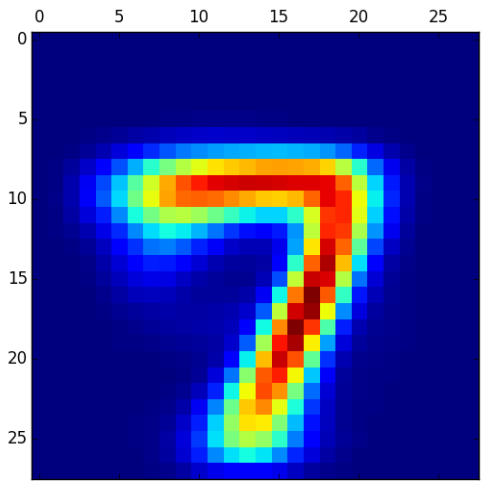

0 5 10 15 20 25
0 5 10 15 20 25

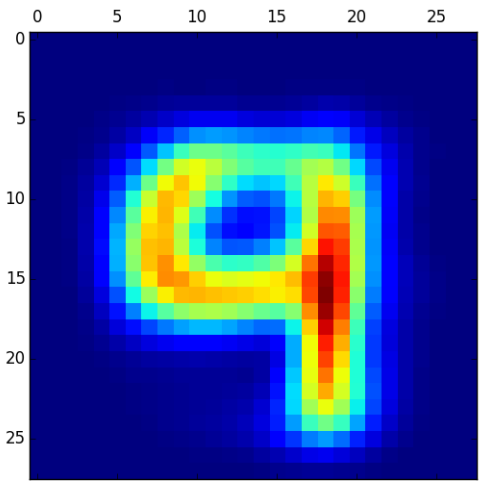

0
5
10
15
20
25

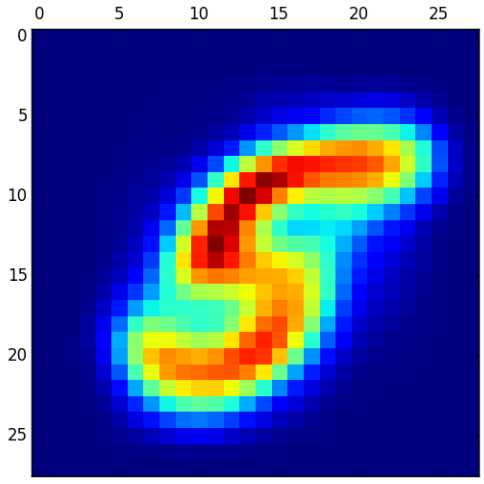

0
5
10
15
20
25

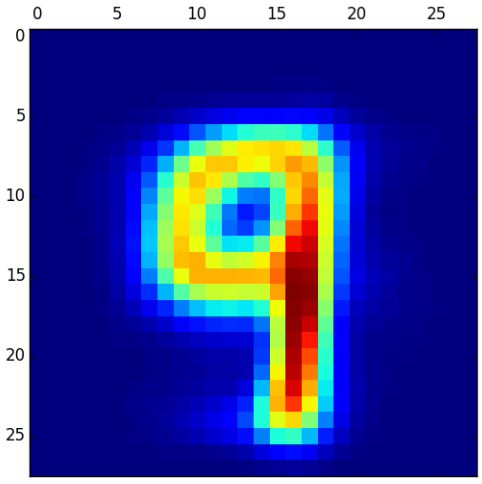

0
5
10
15
20
25

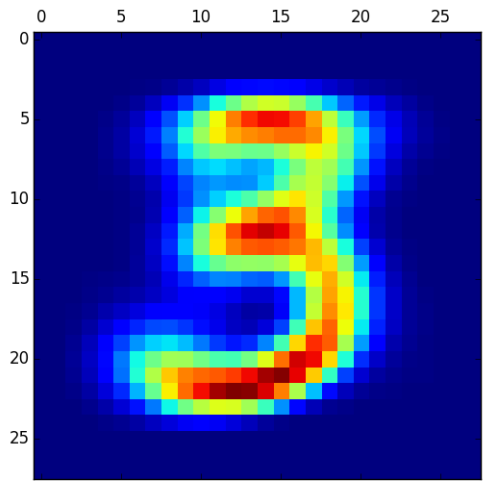

0
5
10
15
20
25

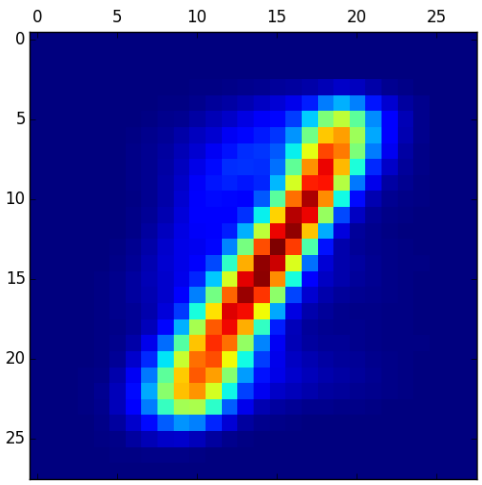

0
5
10
15
20
25

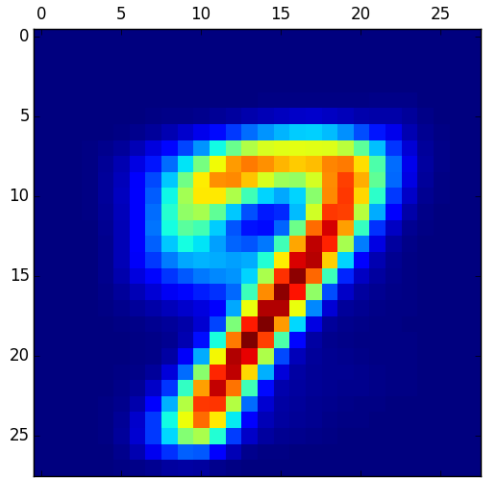

0
5
10
15
20
25

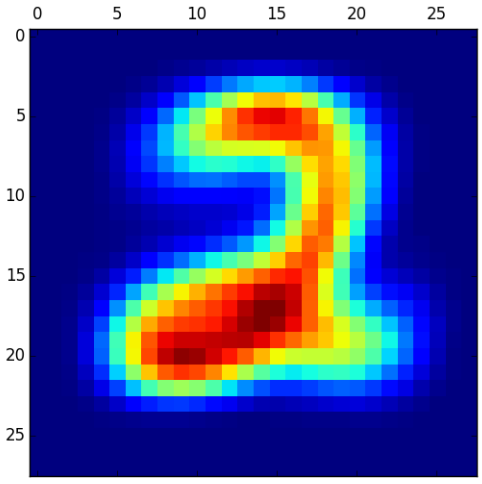

0
5
10
15
20
25

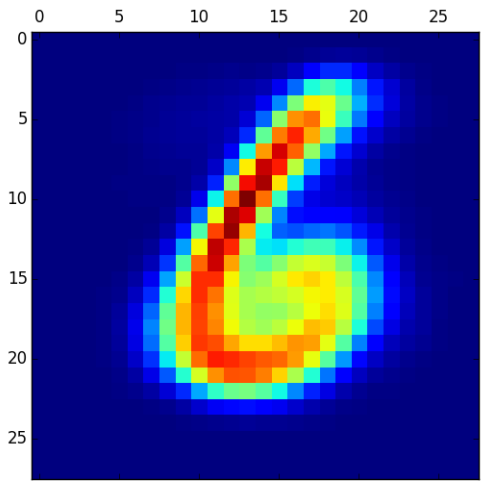

0
5
10
15
20
25

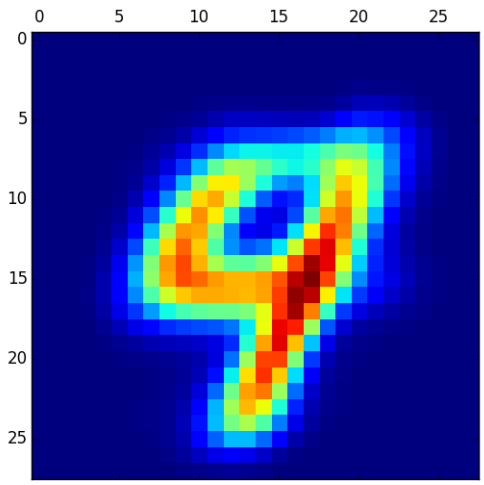

0 5 10 15 20 25
0 5 10 15 20 25


0 5 10 15 20 25
0 5 10 15 20 25


0 5 10 15 20 25
0 5 10 15 20 25


0 5 10 15 20 25
0 5 10 15 20 25





1.3 .

the k-mean loss does vary in different runs.

Problem 2

2.2 .

Recommend the joke by its average rating in the training set, the validation set accuracy is 0.6203252032520326

Find the k nearest neighbors of him/her, then make the prediction by averaging the ratings of these neighbors.

With K = 10, the validation set accuracy is 0.6490514905149052

With K = 100, the validation set accuracy is 0.6894308943089431

With K = 1000, the validation set accuracy is 0.6940379403794038

The accuracies are all higher than the accuracy that we got from the simple system.

2.3.1 . See code in q2.3.py

2.3.2 .

MSE is 20257100.7948 for d = 2

Validation Set accuracy is 0.7094850948509485 for d = 2

MSE is 18839053.5157 for d = 5

Validation Set accuracy is 0.713550135501355 for d = 5

MSE is 17092099.6887 for d = 10

Validation Set accuracy is 0.7132791327913279 for d = 10

MSE is 14192908.8435 for d = 20

Validation Set accuracy is 0.6859078590785908 for d = 20

Thus, MSE is decrease as variable d increase.

2.3.3 . See code in q2.3.3.py

2.3.4 .

MSE is 15111947.0226 for d = 2

Validation Set accuracy is 0.7073170731707317 for d = 2

MSE is 12380230.026 for d = 5

Validation Set accuracy is 0.7092140921409215 for d = 5

MSE is 9632977.97809 for d = 10

Validation Set accuracy is 0.7184281842818429 for d = 10

MSE is 5656292.52527 for d = 20

Validation Set accuracy is 0.6658536585365854 for d = 20

Compare to step 2, for the same d value, the MSE in is much lower in step 3. The validation set accuracies are almost same in both step 2 and 3, not having significant difference.

2.4 .

The best kaggle result is 0.72573

Code Q1

```
import scipy.io as sio
import numpy as np
from numpy.linalg import inv
import matplotlib.pyplot as plt
from sklearn.utils import shuffle


MnistData = sio.loadmat('./mnist_data/images.mat')
Images = MnistData['images']


x_dim = len(Images)
y_dim = len(Images[0])
image_index = len(Images[0][0])
Images = Images.transpose((2, 0, 1))
Images = Images.reshape(image_index, x_dim * y_dim)

def Kmean(images, k):
        #initial clusters and centers
        centers = np.random.randn(k, images[0].size)
        clusters = [[] for i in range(k)]
        for point in images:
                squres = [np.square(np.linalg.norm(center - point)) for center in centers]
                index = np.argmin(squres)
                clusters[index] += [point]
        centers = [np.mean(data, axis=0) for data in clusters]

        #repeat assign point to new centers until no change
        changed = True
        iteration = 0
        while changed:
                print(iteration)
                changed = False
                newclusters = [[] for i in range(k)]
                for i in range(k):
                        cluster = clusters[i]
                        for point in cluster:
                                squres = [np.square(np.linalg.norm(center - point)) for center in centers]
                                index = np.argmin(squres)
                                newclusters[index] += [point]
                                if i != index:
                                        changed = True
                clusters = newclusters
                centers = [np.mean(data, axis=0)for data in clusters]
                iteration += 1
        return centers

k = 5
```

```
centers = Kmean(Images, k)
print("done for k = 5")
for i in range(k):
    plt.matshow(np.reshape(centers[i], (28, 28)))
    plt.show()

k = 10
centers = Kmean(Images, k)
print("done for k = 10")
for i in range(k):
    plt.matshow(np.reshape(centers[i], (28, 28)))
    plt.show()

k = 20
centers = Kmean(Images, k)
print("done for k = 20")
for i in range(k):
    plt.matshow(np.reshape(centers[i], (28, 28)))
    plt.show()
```

Code Q2.2

```
import scipy.io as sio
import numpy as np
from numpy.linalg import inv
import matplotlib.pyplot as plt
from sklearn.utils import shuffle
from sklearn.preprocessing import Imputer
import operator

JokeData = sio.loadmat('./joke_data/joke_train.mat')
Images = JokeData['train']

data = [[int(i) for i in (line.strip().split(','))] for line in open("./joke_data/validation.txt", 'r')]
data = np.array(data)
ValidationSet = data[:, :2]
ValidationLabels = data[:, 2]

def calculateAccuracy(expect, actual):
        same = [i for i in range(len(actual)) if expect[i] == actual[i]]
        return len(same) / len(actual)

#get nearst Neighbors with increase in distance
def findNearestNeighbors(Images, k, point):
        distances = []
        for i in range(len(Images)):
                dist = np.linalg.norm(point - Images[i])
                distances.append((Images[i], dist))
        distances = sorted(distances, key=operator.itemgetter(1))
        neighbors = []
        for i in range(1, k + 1):
                neighbors.append(distances[i][0])
        return np.array(neighbors)

#2.2 Warm-up predict with average rating
averageRateValue = np.nanmean(Images, axis=0)
averageRatePredict = [ 1 if value > 0 else 0 for value in averageRateValue]
predict = []
for data in ValidationSet:
        index =  data[1]
        predict.append(averageRatePredict[index - 1])

accuracy = calculateAccuracy(predict, ValidationLabels)
print("Rating by its average rating, the validation set accuracy is", accuracy)

#2.2 Warm-up with predict the k nearest neighbors

#replace Nan with 0
Ks = [10, 100, 1000]
Images[np.isnan(Images)] = 0
```

```
accuracy = 0
for k in Ks:
        print('k =', k)
        predict = []
        userId = 0
        averageRatePredict = []
        for data in ValidationSet:
                newuserId = data[0]
                if not (newuserId == userId):
                        NearestNeighbors = findNearestNeighbors(Images, k, Images[newuserId - 1])
                        averageRateValue = np.mean(NearestNeighbors, axis=0)
                        averageRatePredict = [1 if value > 0 else 0 for value in averageRateValue]
                        predict.append(averageRatePredict[data[1] - 1])
                        userId = newuserId
                else:
                        predict.append(averageRatePredict[data[1] - 1])
        accuracy = calculateAccuracy(predict, ValidationLabels)
        print('the validation set accuracy is ', accuracy)


```

Code Q2.3.2

```
import scipy.io as sio
import numpy as np
from sklearn import preprocessing
from sklearn.preprocessing import Imputer
import operator
import csv

JokeData = sio.loadmat('./joke_data/joke_train.mat')
Images = JokeData['train']

data = [[int(i) for i in (line.strip().split(','))] for line in open("./joke_data/validation.txt", 'r')]
data = np.array(data)
ValidationSet = data[:, :2]
ValidationLabels = data[:, 2]

query = [[int(i) for i in (line.strip().split(','))] for line in open("./joke_data/query.txt", 'r')]
query = np.array(query)
query = query[:, 1:3]

def calculateAccuracy(expect, actual):
        same = [i for i in range(len(actual)) if expect[i] == actual[i]]
        return len(same) / len(actual)

def predictWithUV(Dataset, vectorU, vectorV):
        predict = []
        for data in Dataset:
                estimate = vectorU[data[0] - 1].dot(vectorV[:, data[1] - 1])
                predict += [1] if estimate > 0 else [0]
        return predict

# newImages replace Nan with 0
newImages = Images
newImages[np.isnan(newImages)] = 0
# newImages = preprocessing.normalize(newImages.astype("float"), norm='l2', axis=0)

# 2.3.1 compute U and V
U, s, V = np.linalg.svd(newImages, full_matrices=False)
U = U.dot(np.diag(s))

#2.3.2 with d vary compute MSE and validation set accuracies
Ds = [2, 5, 10, 20]
for d in Ds:
        newU = U[:, :d+1]
        newV = V[:d+1, :]
        newR = newU.dot(newV)
        MSE = np.square(np.linalg.norm(newR - newImages))
        print('MSE is', MSE, 'for d = ', d)
        predict = predictWithUV(ValidationSet, newU, newV)
```

```
49          accuracy = calculateAccuracy(predict, ValidationLabels)
50          print('Validation Set accuracy is ', accuracy, 'for d = ', d)
51
52          # creat Kaggle submission file
53          predict_labels = predictWithUV(query, newU, newV)
54          indexs = [i for i in range(1, len(query) + 1)]
55          data = []
56          data += [indexs]
57          data += [predict_labels]
58          data = np.transpose(data, (1, 0)).tolist()
59          first_row = [['Id', 'Category']]
60          with open('2spampredict d = ' + str(d) + '.csv', 'w') as f:
61              a = csv.writer(f)
62              a.writerows(first_row)
63              a.writerows(data)
64
```

Code Q2.3.3

```
import scipy.io as sio
import numpy as np
from sklearn import preprocessing
from sklearn.preprocessing import Imputer
import operator
import csv

JokeData = sio.loadmat('./joke_data/joke_train.mat')
Images = JokeData['train']

data = [[int(i) for i in (line.strip().split(','))] for line in open("./joke_data/validation.txt", 'r')]
data = np.array(data)
ValidationSet = data[:, :2]
ValidationLabels = data[:, 2]

query = [[int(i) for i in (line.strip().split(','))] for line in open("./joke_data/query.txt", 'r')]
query = np.array(query)
query = query[:, 1:3]

#2.3.3
def calculateAccuracy(expect, actual):
        same = [i for i in range(len(actual)) if expect[i] == actual[i]]
        return len(same) / len(actual)

def predictWithUV(Dataset, vectorU, vectorV):
        predict = []
        for data in Dataset:
                estimate = vectorU[data[0] - 1].dot(vectorV[:, data[1] - 1])
                predict += [1] if estimate > 0 else [0]
        return predict

def calculateLoss(Images, U, V, lamb):
        loss = 0
        for i in range(len(Images)):
                for j in range(len(Images[0])):
                        if not np.isnan(Images[i][j]):
                                loss += np.square(np.linalg.norm(U[i].T.dot(V[ : , j]) - Images[i][j]))\
                                                + lamb * np.square(np.linalg.norm(U[i]))\
                                                + lamb * np.square(np.linalg.norm(V[ : , j]))
        return loss

def findUV(Images, d, lamb):
        U = np.random.randn(len(Images), d)
        V = np.random.randn(d, len(Images[0]))
        # Changed = True
        iterantions = 0
        while iterantions < 100:
                # Changed = False
```

```
                iterantions += 1
                newU = []
                orignLoss = calculateLoss(Images, U, V, lamb)
                for i in range(len(Images)):
                        left = lamb * np.identity(d)
                        right = np.zeros(d)
                        for j in range(len(Images[0])):
                                if not np.isnan(Images[i][j]):
                                        left += np.outer(V[ : , j], V[ : , j])
                                        right += int(Images[i][j]) * V[ : , j].T
                        newU.append(right.dot(np.linalg.inv(left)))
                newU = np.array(newU)
                U = newU

                newV = []
                for j in range(len(Images[0])):
                        left = lamb * np.identity(d)
                        right = np.zeros(d)
                        for i in range(len(Images)):
                                if not np.isnan(Images[i][j]):
                                        left += np.outer(U[i].T, U[i].T)
                                        right += int(Images[i][j]) * U[i].T
                        newV.append(right.dot(np.linalg.inv(left)))
                newV = np.array(newV).T
                V = newV

                newLoss = calculateLoss(Images, U, V, lamb)
                # diff = orignLoss - newLoss
                # print('diff is', diff)
                print('MSE is ', newLoss)
                predict = predictWithUV(ValidationSet, U, V)
                accuracy = calculateAccuracy(predict, ValidationLabels)
                print('Validation Set accuracy is ', accuracy, 'for d = ', d)
                # if abs(diff) > 3:
                #         Changed = True
        # creat Kaggle submission file
        predict_labels = predictWithUV(query, U, V)
        indexs = [i for i in range(1, len(query) + 1)]
        data = []
        data += [indexs]
        data += [predict_labels]
        data = np.transpose(data, (1, 0)).tolist()
        first_row = [['Id', 'Category']]
        with open('3spampredict d = ' + str(d) + '.csv', 'w') as f:
            a = csv.writer(f)
            a.writerows(first_row)
            a.writerows(data)
        return U, V


```

```
lamb = 0.1
Ds = [2, 5, 10, 20]

for d in Ds:
        print('begin train with d = ', d)
        U, V = findUV(Images, d, lamb)
        MSE = calculateLoss(Images, U, V, lamb)
        print('MSE is', MSE, 'for d = ', d)
        predict = predictWithUV(ValidationSet, U, V)
        accuracy = calculateAccuracy(predict, ValidationLabels)
        print('Validation Set accuracy is ', accuracy, 'for d = ', d)
        print('done train with d = ', d)


```