METHOD DECRIPTION

1. First step was endpointing, which is basically clipping the audio such that only the frames with energy above a threshold were retained. Thus, only the single words were included. Silence was clipped out.This was done to train and test files.
2. Then preemphasis was added to remove noise.This was done to train and test files.
3. Then 39 mfcc features were extracted from the train and test files.
4. Then, the vq codebook matching method was carried out using kmeans clustering.
5. K-means clustering is an iterative process of assigning each data point to the groups and slowly data points get clustered based on similar features. The objective is to minimize the sum of distances between the data points and the cluster centroid, to identify the correct group each data point should belong to.
6. Using this, training and testing was done
7. For noisy test files, training data was augmented with noise and the model was retrained on it and then tested on noisy data

TESTING OBSERVATIONS AND RESULTS

1. On increasing the number of clusters from 4 to 8 in K-means clustering, the accuracy improved from 31.2 to 39.5 percent on clean test data
2. The k=4 models and k=8 models trained on clean data gave 25.54 and 29.9 accuracies on the noisy test data respectively so noise reduced prediction accuracy
3. On augmentation of training data using noise, the k=4 models and k=8 models gave 20.2 and 23.9 accuracies on the noisy test data respectively so surprisingly noise augmentation reduced prediction accuracy on noisy data
4. We can also observe from the confusion matrix that the best model performs the best on 'off' and worst on 'right', so short, more vowel-like words perform better than longer words which have consonants at their start and end.

DISCUSSION

Thus we can conclude that increasing k in kmeans clustering improved accuracy in this task. mfcc feature extraction is a good way for audio file characterization and classification, especially classification of short words. The accuracy of this method is low and i expect gmmhmm method to give better accuracy. Noise augmentation did not improve results on noisy test data, but it ideally should, this can be attributed to the type of adding noise or the way it was added.

CODE

```
import math
from scipy import signal
import numpy as np
import matplotlib.pyplot as plt
from scipy.io.wavfile import write
from scipy.io import wavfile
import scipy.signal as signal
from IPython.display import Audio, display
from scipy.fftpack import fft, dct
import os
from sklearn.metrics import confusion_matrix
from scipy.cluster.vq import vq, kmeans, whiten
from hmmlearn import hmm
```

ENDPOINTING - Taking out only the parts of speech with audio in it for training and testing sets

```
words=['yes','go','down','left','right','on','no','off','up','stop']
for word in words:
    trainyes=os.listdir("Commands Dataset-20200305T135856Z-001/Commands Dataset/train/"+word)
    for yesfile in trainyes:
      read = wavfile.read('Commands Dataset-20200305T135856Z-001/Commands Dataset/train/'+word+'/'+ yesfile)
      inp= read[1]
      norminp = inp/max(abs(inp))
      noofframes = len(norminp)//480
      Senergy = []
      for i in range(noofframes):
            ham = np.zeros_like(norminp)
            ham[480*i:(i+1)*480] = np.hamming(480)
            Senergyind = sum((norminp**2)*(ham**2))
            Senergy.append(Senergyind)
      frame=5
      while(frame<len(Senergy)):
            if(Senergy[frame]>=0.5):
                cut=inp[(frame-2)*480:(frame+18)*480]
                write('./cut_trained/'+word+'/' + yesfile, 16000, cut)
                frame+=27
            else: frame+=1
```

```
words=['yes','go','down','left','right','on','no','off','up','stop']
for word in words:
    trainyes=os.listdir("Commands Dataset-20200305T135856Z-001/Commands Dataset/test_clean/"+word)
    for yesfile in trainyes:
      read = wavfile.read('Commands Dataset-20200305T135856Z-001/Commands Dataset/test_clean/'+word+'/'+ yesfile)
      inp= read[1]
      norminp = inp/max(abs(inp))
      noofframes = len(norminp)//480
      Senergy = []
      for i in range(noofframes):
            ham = np.zeros_like(norminp)
            ham[480*i:(i+1)*480] = np.hamming(480)
            Senergyind = sum((norminp**2)*(ham**2))
            Senergy.append(Senergyind)
      frame=5
      while(frame<len(Senergy)):
            if(Senergy[frame]>=0.5):
                cut=inp[(frame-2)*480:(frame+18)*480]
                write('./cut_test_clean/'+word+'/' + yesfile, 16000, cut)
                frame+=27
            else: frame+=1


words=['yes','go','down','left','right','on','no','off','up','stop']
for word in words:
    trainyes=os.listdir("Commands Dataset-20200305T135856Z-001/Commands Dataset/test_noisy/"+word)
    for yesfile in trainyes:
      read = wavfile.read('Commands Dataset-20200305T135856Z-001/Commands Dataset/test_noisy/'+word+'/'+ yesfile)
      inp= read[1]
      norminp = inp/max(abs(inp))
      noofframes = len(norminp)//480
      Senergy = []
      for i in range(noofframes):
            ham = np.zeros_like(norminp)
            ham[480*i:(i+1)*480] = np.hamming(480)
            Senergyind = sum((norminp**2)*(ham**2))
            Senergy.append(Senergyind)
      frame=5
      while(frame<len(Senergy)):
            if(Senergy[frame]>=0.5):
                cut=inp[(frame-2)*480:(frame+18)*480]
                write('./cut_test_noisy/'+word+'/' + yesfile, 16000, cut)
                frame+=27
            else: frame+=1
```

PREEMPHASIS addition in training and testing sets

```
words=['yes','go','down','left','right','on','no','off','up','stop']
for word in words:
    trainyes=os.listdir("./cut_trained/"+word)
    for yesfile in trainyes:
        read = wavfile.read('./cut_trained/'+word+'/'+ yesfile)
        inp= read[1]
        pe=np.zeros_like(inp)
        for i in range(len(inp)):
          if (i==0 or i==1):
            pe[i]=inp[i]
          else:
            pe[i]=(inp[i]-0.95*inp[i-1])
        write('./preemp/'+word+'/' + yesfile, 16000, pe)


words=['yes','go','down','left','right','on','no','off','up','stop']
for word in words:
    trainyes=os.listdir("./cut_test_clean/"+word)
    for yesfile in trainyes:
        read = wavfile.read('./cut_test_clean/'+word+'/'+ yesfile)
        inp= read[1]
        pe=np.zeros_like(inp)
        for i in range(len(inp)):
          if (i==0 or i==1):
            pe[i]=inp[i]
          else:
            pe[i]=(inp[i]-0.95*inp[i-1])
        write('./preemp_tc/'+word+'/' + yesfile, 16000, pe)


words=['yes','go','down','left','right','on','no','off','up','stop']
for word in words:
    trainyes=os.listdir("./cut_test_noisy/"+word)
    for yesfile in trainyes:
```

```
        read = wavfile.read('./cut_test_noisy/'+word+'/'+ yesfile)
        inp= read[1]
        pe=np.zeros_like(inp)
        for i in range(len(inp)):
          if (i==0 or i==1):
            pe[i]=inp[i]
          else:
            pe[i]=(inp[i]-0.95*inp[i-1])
        write('./preemp_tn/'+word+'/' + yesfile, 16000, pe)
```

EXTRACTING 39 MFCC features for training set

```
def melbin(s, f, filters, fs):
    bin=np.linspace(2595*np.log10(1 + (s/700)), 2595*np.log10(1 + (f/700)), filters+2)
    for i in range(bin.shape[0]):
        bin[i]=10**(bin[i]/2595)-1
        bin[i]=np.floor((257)*bin[i]*700/16000)
    return bin

def melfil(f, filters):
    fbank = np.zeros([filters,256+1])
    for m in range(filters):
        for k in range(int(f[m]), int(f[m+1])):
            fbank[m,k] = (k - f[m]) / (f[m+1]-f[m])
        for k in range(int(f[m+1]), int(f[m+2])):
            fbank[m,k] = (f[m+2]-k) / (f[m+2]-f[m+1])
    return fbank


filters=26
traincodebook= np.zeros((10,2350*60,39))
words=['yes','go','down','left','right','on','no','off','up','stop']
i=0
for word in words:
    j=0
    for file in os.listdir('./preemp/'+word):
        read = wavfile.read('./preemp/'+word+'/'+file)
        inp=read[1]
        if (len(inp)<9600):
            inp=np.append(inp,np.zeros(9600-len(inp)))
        norminp = inp/max(abs(inp))
        nFrames = len(norminp)//160
        feat=[]
        for k in range(nFrames):
            frame = norminp[k*160:(k+1)*160]*np.hamming(160)
            p=(np.abs(np.fft.fft(frame, n=512))**2)/frame.shape[0]
            bins = melbin(300, 4000, filters, 16000)
            fbank = melfil(bins, filters)
            c=np.dot(p[:257],fbank.T)
            if (0.0 in c):
                c[c == 0.0] = 0.000001
            c=dct(20*np.log10(c))
            feat.append(c[:13])
        mfcc13 = np.array(feat)
        mfcc39 = np.zeros((mfcc13.shape[0], 39))
        mfcc39[:,:13] = mfcc13
        mfcc39[:,13:26] = mfcc13
        mfcc39[:,26:39] = mfcc13
        for l in range(1,12):
            mfcc39[:,13+l] = (mfcc13[:,l+1] - mfcc13[:,l-1])/2.0
        for l in range(2,11):
            mfcc39[:,26+l] = (mfcc13[:,l+2] - mfcc13[:,l-2])/2.0
        traincodebook[i,j*60:(j+1)*60,:] = mfcc39
        j=j+1
    i=i+1
    print('extractedtrainingfeatureforlabel', i)
np.save('codebook',traincodebook)
```

```
 extractedtrainingfeatureforlabel 1
 extractedtrainingfeatureforlabel 2
 extractedtrainingfeatureforlabel 3
 extractedtrainingfeatureforlabel 4
 extractedtrainingfeatureforlabel 5
 extractedtrainingfeatureforlabel 6
 extractedtrainingfeatureforlabel 7
 extractedtrainingfeatureforlabel 8
 extractedtrainingfeatureforlabel 9
 extractedtrainingfeatureforlabel 10
```

```
print(traincodebook.shape)
```

```
(10, 141000, 39)
```

TRAINING VQ codebook using K-means clustering for k=4 and 8.

```
vqcodebook4=np.zeros((10,4,39))
for i in range(10):
    vqcodebook4[i] = kmeans(traincodebook[i],4)[0]
    print('traininglabel',i)
```

```
traininglabel 0
traininglabel 1
traininglabel 2
traininglabel 3
traininglabel 4
traininglabel 5
traininglabel 6
traininglabel 7
traininglabel 8
traininglabel 9
```

```
vqcodebook8=np.zeros((10,8,39))
for i in range(10):
    vqcodebook8[i] = kmeans(traincodebook[i],8)[0]
    print('traininglabel',i)
```

```
traininglabel 0
traininglabel 1
traininglabel 2
traininglabel 3
traininglabel 4
traininglabel 5
traininglabel 6
traininglabel 7
traininglabel 8
traininglabel 9
```

```
np.save('model4',vqcodebook4)
np.save('model8',vqcodebook8)
print(vqcodebook4.shape)
print(vqcodebook8.shape)
```

```
(10, 4, 39)
(10, 8, 39)
```

EXTRACTING 39 MFCC features for clean testing set

```
filters=26
testccodebook= np.zeros((10,240*60,39))
words=['yes','go','down','left','right','on','no','off','up','stop']
i=0
for word in words:
    j=0
    for file in os.listdir('./preemp_tc/'+word):
        read = wavfile.read('./preemp_tc/'+word+'/'+file)
        inp=read[1]
        if (len(inp)<9600):
            inp=np.append(inp,np.zeros(9600-len(inp)))
        norminp = inp/max(abs(inp))
        nFrames = len(norminp)//160
        feat=[]
        for k in range(nFrames):
            frame = norminp[k*160:(k+1)*160]*np.hamming(160)
            p=(np.abs(np.fft.fft(frame, n=512))**2)/frame.shape[0]
            bins = melbin(300, 4000, filters, 16000)
            fbank = melfil(bins, filters)
            c=np.dot(p[:257],fbank.T)
            if (0.0 in c):
                c[c == 0.0] = 0.000001
            c=dct(20*np.log10(c))
            feat.append(c[:13])
        mfcc13 = np.array(feat)
        mfcc39 = np.zeros((mfcc13.shape[0], 39))
        mfcc39[:,:13] = mfcc13
        mfcc39[:,13:26] = mfcc13
        mfcc39[:,26:39] = mfcc13
        for l in range(1,12):
            mfcc39[:,13+l] = (mfcc13[:,l+1] - mfcc13[:,l-1])/2.0
        for l in range(2,11):
            mfcc39[:,26+l] = (mfcc13[:,l+2] - mfcc13[:,l-2])/2.0
```

```
        testccodebook[i,j*60:(j+1)*60,:] = mfcc39
        j=j+1
    i=i+1
    print('extractedcleantestfeatureforlabel', i)
np.save('testccodebook',testccodebook)
```

```
 extractedcleantestfeatureforlabel 1
 extractedcleantestfeatureforlabel 2
 extractedcleantestfeatureforlabel 3
 extractedcleantestfeatureforlabel 4
 extractedcleantestfeatureforlabel 5
 extractedcleantestfeatureforlabel 6
 extractedcleantestfeatureforlabel 7
 extractedcleantestfeatureforlabel 8
 extractedcleantestfeatureforlabel 9
 extractedcleantestfeatureforlabel 10
```

```
print(testccodebook.shape)
```

```
 (10, 16800, 39)
```

TESTING on clean test data to get accuracy

```
pred= []
gt= []
for i in range(10):
    for j in range(240):
        feat = testccodebook[i, 60*j:60*(j+1),:]
        labeldist=[]
        for label in range(10):
            sumdist=0
            for l in range (60):
                dists=[]
                for k in range(4):
                    dist=0.0
                    for m in range(39):
                        dist=dist+ (feat[l][m]-vqcodebook4[label,k][m])**2
                    dists.append(dist)
                sumdist= sumdist+  min(dists)
            labeldist.append(sumdist)
        prediction=np.argmin(labeldist)
        pred.append(prediction)
        gt.append(i)
    print('testing on label',i)
true=0
for q in range (len(gt)):
    if (gt[q]==pred[q]):
        true=true+1
print("Accuracy=",true*100/len(gt))
print(confusion_matrix(gt, pred))
```

```
 testing on label 0
 testing on label 1
 testing on label 2
 testing on label 3
 testing on label 4
 testing on label 5
 testing on label 6
 testing on label 7
 testing on label 8
 testing on label 9
 Accuracy= 31.208333333333332
 [[ 85  11   6  36   7   4  27   3  31  30]
  [ 13  74  13  12  10   5  59   6  25  23]
  [ 29   7  66  11  14  12  48   3  26  24]
  [ 50   6   2  48  17  15  23  11  30  38]
  [ 36  16  12  22  33   7  52   3  32  27]
  [  8  23  11  11   5  85  46  11  28  12]
  [ 30  33  16  13   8   4  99   1  21  15]
  [  7   2   4   4   1   9   7 147  33  26]
  [  7  10   5   8   5  13  13  69  51  59]
  [ 14  12   4  21   4  16   8  54  46  61]]
```

```
pred= []
gt= []
for i in range(10):
    for j in range(240):
        feat = testccodebook[i, 60*j:60*(j+1),:]
        labeldist=[]
        for label in range(10):
            sumdist=0
            for l in range (60):
```

```
                dists=[]
                for k in range(8):
                    dist=0.0
                    for m in range(39):
                        dist=dist+ (feat[l][m]-vqcodebook8[label,k][m])**2
                    dists.append(dist)
                sumdist= sumdist+  min(dists)
            labeldist.append(sumdist)
        prediction=np.argmin(labeldist)
        pred.append(prediction)
        gt.append(i)
    print('testing on label',i)
true=0
for q in range (len(gt)):
    if (gt[q]==pred[q]):
        true=true+1
print("Accuracy=",true*100/len(gt))
print(confusion_matrix(gt, pred))
```

```
 testing on label 0
 testing on label 1
 testing on label 2
 testing on label 3
 testing on label 4
 testing on label 5
 testing on label 6
 testing on label 7
 testing on label 8
 testing on label 9
 Accuracy= 39.541666666666664
 [[108  10   7  45  13   4  22   5  13  13]
  [  4  91  12  14  16  10  55   4  18  16]
  [  1   4  89  18  21  18  50   4  13  22]
  [ 35  13   8  72  29  10  15   7  13  38]
  [ 21  19  21  37  59   6  44   2   9  22]
  [  2  21  26   9   2  89  41  11  12  27]
  [  9  24  21  14   8   9 130   2   8  15]
  [  1   5   3   2   0  16   3 150  22  38]
  [  2  15   6  15   6  12   5  44  60  75]
  [  0  25   4  15   6  11   5  32  41 101]]
```

TESTING on noisy test data

```
filters=26
testncodebook= np.zeros((10,240*60,39))
words=['yes','go','down','left','right','on','no','off','up','stop']
i=0
for word in words:
    j=0
    for file in os.listdir('./preemp_tn/'+word):
        read = wavfile.read('./preemp_tn/'+word+'/'+file)
        inp=read[1]
        if (len(inp)<9600):
            inp=np.append(inp,np.zeros(9600-len(inp)))
        norminp = inp/max(abs(inp))
        nFrames = len(norminp)//160
        feat=[]
        for k in range(nFrames):
            frame = norminp[k*160:(k+1)*160]*np.hamming(160)
            p=(np.abs(np.fft.fft(frame, n=512))**2)/frame.shape[0]
            bins = melbin(300, 4000, filters, 16000)
            fbank = melfil(bins, filters)
            c=np.dot(p[:257],fbank.T)
            if (0.0 in c):
                c[c == 0.0] = 0.000001
            c=dct(20*np.log10(c))
            feat.append(c[:13])
        mfcc13 = np.array(feat)
        mfcc39 = np.zeros((mfcc13.shape[0], 39))
        mfcc39[:,:13] = mfcc13
        mfcc39[:,13:26] = mfcc13
        mfcc39[:,26:39] = mfcc13
        for l in range(1,12):
            mfcc39[:,13+l] = (mfcc13[:,l+1] - mfcc13[:,l-1])/2.0
        for l in range(2,11):
            mfcc39[:,26+l] = (mfcc13[:,l+2] - mfcc13[:,l-2])/2.0
        testncodebook[i,j*60:(j+1)*60,:] = mfcc39
        j=j+1
    i=i+1
    print('extractedcleantestfeatureforlabel', i)
np.save('testncodebook',testncodebook)
```

```
extractedcleantestfeatureforlabel 1
extractedcleantestfeatureforlabel 2
extractedcleantestfeatureforlabel 3
extractedcleantestfeatureforlabel 4
extractedcleantestfeatureforlabel 5
extractedcleantestfeatureforlabel 6
extractedcleantestfeatureforlabel 7
extractedcleantestfeatureforlabel 8
extractedcleantestfeatureforlabel 9
extractedcleantestfeatureforlabel 10
```

```
pred= []
gt= []
for i in range(10):
    for j in range(240):
        feat = testncodebook[i, 60*j:60*(j+1),:]
        labeldist=[]
        for label in range(10):
            sumdist=0
            for l in range (60):
                dists=[]
                for k in range(4):
                    dist=0.0
                    for m in range(39):
                        dist=dist+ (feat[l][m]-vqcodebook4[label,k][m])**2
                    dists.append(dist)
                sumdist= sumdist+  min(dists)
            labeldist.append(sumdist)
        prediction=np.argmin(labeldist)
        pred.append(prediction)
        gt.append(i)
    print('testing on label',i)
true=0
for q in range (len(gt)):
    if (gt[q]==pred[q]):
        true=true+1
print("Accuracy=",true*100/len(gt))
print(confusion_matrix(gt, pred))
```

```
testing on label 0
testing on label 1
testing on label 2
testing on label 3
testing on label 4
testing on label 5
testing on label 6
testing on label 7
testing on label 8
testing on label 9
Accuracy= 25.541666666666668
[[ 60   5  18  37   7   8  14  30  25  36]
 [  8  50  22  19   5  19  40  27  24  26]
 [ 29   4  59  34   7  18  25  17  27  20]
 [ 40   4  10  42  14  16  12  39  28  35]
 [ 29   9  22  43  17  15  34  15  27  29]
 [  6  11  19  10   4  74  25  39  34  18]
 [ 32  19  30  32   4  15  54  14  20  20]
 [  4   2   7   2   2  16   6 146  26  29]
 [  5   2   7   8   1   9   5  91  46  66]
 [ 11   4   7   9   2  14   3  83  42  65]]
```

```
pred= []
gt= []
for i in range(10):
    for j in range(240):
        feat = testncodebook[i, 60*j:60*(j+1),:]
        labeldist=[]
        for label in range(10):
            sumdist=0
            for l in range (60):
                dists=[]
                for k in range(8):
                    dist=0.0
                    for m in range(39):
                        dist=dist+ (feat[l][m]-vqcodebook8[label,k][m])**2
                    dists.append(dist)
                sumdist= sumdist+  min(dists)
            labeldist.append(sumdist)
        prediction=np.argmin(labeldist)
        pred.append(prediction)
        gt.append(i)
    print('testing on label',i)
true=0
```

```
for q in range (len(gt)):
    if (gt[q]==pred[q]):
        true=true+1
print("Accuracy=",true*100/len(gt))
print(confusion_matrix(gt, pred))
```

```
 testing on label 0
 testing on label 1
 testing on label 2
 testing on label 3
 testing on label 4
 testing on label 5
 testing on label 6
 testing on label 7
 testing on label 8
 testing on label 9
 Accuracy= 29.916666666666668
 [[ 73   4  13  66   5   6  13  34  16  10]
  [  7  56  32  25   9  11  34  36  20  10]
  [  3   2  75  49  10  21  22  19  10  29]
  [ 26   9  12  71  19  10   7  46  18  22]
  [ 14  13  19  77  35   4  26  12  18  22]
  [  1  10  36  12   2  64  21  51  18  25]
  [  8  14  38  44   8  13  68  14  18  15]
  [  1   2  11   0   0   8   2 161  20  35]
  [  1   3   7  16   6   5   2  88  51  61]
  [  1  11   8  14   3   7   4 100  28  64]]
```

ADDING NOISE augmentations to clean training data

```
words=['yes','go','down','left','right','on','no','off','up','stop']
noise=['doing_the_dishes.wav','dude_miaowing.wav','exercise_bike.wav','pink_noise.wav','running_tap.wav','white_noise.wav']
from add_noise import *
for word in words:
    trainyes=os.listdir("./preemp/"+word)
    for file in trainyes:
        add_noise("./preemp/"+word+"/"+file,
                  "Commands Dataset-20200305T135856Z-001/Commands Dataset/_background_noise_/"+ np.random.choice(noise),
                  "./preemp_noise/"+word+"/"+file)
```

```
 /home/amruta/code/add_noise.py:17: WavFileWarning: Chunk (non-data) not understood, skipping it.
   sampFreq, noise = wavfile.read(noisefile)
```

TRAINING on augmented noisy data

```
filters=26
trainncodebook= np.zeros((10,2350*60,39))
words=['yes','go','down','left','right','on','no','off','up','stop']
i=0
for word in words:
    j=0
    for file in os.listdir('./preemp_noise/'+word):
        read = wavfile.read('./preemp_noise/'+word+'/'+file)
        inp=read[1]
        if (len(inp)<9600):
            inp=np.append(inp,np.zeros(9600-len(inp)))
        norminp = inp/max(abs(inp))
        nFrames = len(norminp)//160
        feat=[]
        for k in range(nFrames):
            frame = norminp[k*160:(k+1)*160]*np.hamming(160)
            p=(np.abs(np.fft.fft(frame, n=512))**2)/frame.shape[0]
            bins = melbin(300, 4000, filters, 16000)
            fbank = melfil(bins, filters)
            c=np.dot(p[:257],fbank.T)
            if (0.0 in c):
                c[c == 0.0] = 0.000001
            c=dct(20*np.log10(c))
            feat.append(c[:13])
        mfcc13 = np.array(feat)
        mfcc39 = np.zeros((mfcc13.shape[0], 39))
        mfcc39[:,:13] = mfcc13
        mfcc39[:,13:26] = mfcc13
        mfcc39[:,26:39] = mfcc13
        for l in range(1,12):
            mfcc39[:,13+l] = (mfcc13[:,l+1] - mfcc13[:,l-1])/2.0
        for l in range(2,11):
            mfcc39[:,26+l] = (mfcc13[:,l+2] - mfcc13[:,l-2])/2.0
        trainncodebook[i,j*60:(j+1)*60,:] = mfcc39
        j=j+1
    i=i+1
```

```
    print('extractedtrainingfeatureforlabel', i)
np.save('ncodebook',trainncodebook)
```

```
 extractedtrainingfeatureforlabel 1
 extractedtrainingfeatureforlabel 2
 extractedtrainingfeatureforlabel 3
 extractedtrainingfeatureforlabel 4
 extractedtrainingfeatureforlabel 5
 extractedtrainingfeatureforlabel 6
 extractedtrainingfeatureforlabel 7
 extractedtrainingfeatureforlabel 8
 extractedtrainingfeatureforlabel 9
 extractedtrainingfeatureforlabel 10
```

```
nvqcodebook4=np.zeros((10,4,39))
for i in range(10):
    nvqcodebook4[i] = kmeans(trainncodebook[i],4)[0]
    print('traininglabel',i)
```

```
 traininglabel 0
 traininglabel 1
 traininglabel 2
 traininglabel 3
 traininglabel 4
 traininglabel 5
 traininglabel 6
 traininglabel 7
 traininglabel 8
 traininglabel 9
```

```
nvqcodebook8=np.zeros((10,8,39))
for i in range(10):
    nvqcodebook8[i] = kmeans(trainncodebook[i],8)[0]
    print('traininglabel',i)
```

```
 traininglabel 0
 traininglabel 1
 traininglabel 2
 traininglabel 3
 traininglabel 4
 traininglabel 5
 traininglabel 6
 traininglabel 7
 traininglabel 8
 traininglabel 9
```

TESTING on noisy test data

```
pred= []
gt= []
for i in range(10):
    for j in range(240):
        feat = testncodebook[i, 60*j:60*(j+1),:]
        labeldist=[]
        for label in range(10):
            sumdist=0
            for l in range (60):
                dists=[]
                for k in range(4):
                    dist=0.0
                    for m in range(39):
                        dist=dist+ (feat[l][m]-nvqcodebook4[label,k][m])**2
                    dists.append(dist)
                sumdist= sumdist+  min(dists)
            labeldist.append(sumdist)
        prediction=np.argmin(labeldist)
        pred.append(prediction)
        gt.append(i)
    print('testing on label',i)
true=0
for q in range (len(gt)):
    if (gt[q]==pred[q]):
        true=true+1
print("Accuracy=",true*100/len(gt))
print(confusion_matrix(gt, pred))
```

```
 testing on label 0
 testing on label 1
 testing on label 2
 testing on label 3
 testing on label 4
 testing on label 5
 testing on label 6
```

```
testing on label 7
testing on label 8
testing on label 9
Accuracy= 20.166666666666668
[[ 26   5   4  55   4   1   3  10 127   5]
 [ 19  71   6  26   4   7   7   8  83   9]
 [ 16   6  30  38   4  14  13   6  96  17]
 [ 14   6   3  42   3   4  10  10 145   3]
 [  5  11   4  68  15   6  17   7  96  11]
 [ 11  13  16   9   1  25  11  26 106  22]
 [ 30  29  12  37   4  12  30  13  70   3]
 [  4   7   1   1   0   4   1  42 165  15]
 [  2   2   2   5   0   7   2  17 188  15]
 [  3   4   6   3   1   4   0  20 184  15]]
```

```
pred= []
gt= []
for i in range(10):
    for j in range(240):
        feat = testncodebook[i, 60*j:60*(j+1),:]
        labeldist=[]
        for label in range(10):
            sumdist=0
            for l in range (60):
                dists=[]
                for k in range(8):
                    dist=0.0
                    for m in range(39):
                        dist=dist+ (feat[l][m]-nvqcodebook8[label,k][m])**2
                    dists.append(dist)
                sumdist= sumdist+  min(dists)
            labeldist.append(sumdist)
        prediction=np.argmin(labeldist)
        pred.append(prediction)
        gt.append(i)
    print('testing on label',i)
true=0
for q in range (len(gt)):
    if (gt[q]==pred[q]):
        true=true+1
print("Accuracy=",true*100/len(gt))
print(confusion_matrix(gt, pred))
```

```
testing on label 0
testing on label 1
testing on label 2
testing on label 3
testing on label 4
testing on label 5
testing on label 6
testing on label 7
testing on label 8
testing on label 9
Accuracy= 23.875
[[ 67   2   5  41  13   0   1  10 100   1]
 [ 21  77   8  20  11   0   7  13  71  12]
 [ 21   7  24  26   9  10   6  11 108  18]
 [ 31   3   4  50   2   1   4  10 132   3]
 [ 40   5   4  53  26   2   5   7  90   8]
 [  6  14  12   5   4  19  11  43 100  26]
 [ 42  23   8  25  13   3  18  13  86   9]
 [  1   3   2   4   0   3   1  81 131  14]
 [  2   1   4   8   1   1   1  18 189  15]
 [  1   0   3  13   2   1   0  25 173  22]]
```