In [305]:
```
from sklearn.decomposition import PCA
import matplotlib.patches as mpatches
```

In [334]:

```
%matplotlib inline
#%matplotlib nbagg

import wave, random, struct, math, numpy as np, matplotlib.mlab as mlab,
pylab as pl
import matplotlib.pyplot as pyplot, collections
from scipy import linalg
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.lda import LDA


fwine = open('wine.data', 'r')
winedata = np.loadtxt("wine.data", comments="#", delimiter=",", unpack=F
alse) #178x14
class12= winedata[[winedata[:,0] > 1]]
class12 = class12[:,1:len(winedata[2,:])]

fmnisttest = open('test.csv', 'r')
mtestdata = np.loadtxt("test.csv", delimiter = ",", unpack=False) #sampl
es x feats #col = samples
fmnisttrain = open('train.csv', 'r')
mtraindata = np.loadtxt("train.csv", delimiter = ",", unpack=False) #sam
ples x feats #col = samples


#------------MNIST Data Computations-----------#
#Mnist data transpose
mtesttr = mtestdata.transpose(); #col = features
[sampm,featm] = np.shape(mtesttr)
mtraintr = mtraindata.transpose(); # col = features
[sampm,featm] = np.shape(mtraintr)


# train data #
# Separating into classes- class0 train
temp1 = mtraintr #columns = features
temp2 = temp1[temp1[:,featm-1]==0]
class0mtrain = temp2[:,0:featm-1]
class0mtrain = class0mtrain.transpose() #Col = samples
#Separating Class 1
temp2 = temp1[temp1[:,featm-1]==1]
class1mtrain = temp2[:,0:featm-1]
class1mtrain = class1mtrain.transpose()  #Col = samples
#Separating Class 3
temp2 = temp1[temp1[:,featm-1]==3]
class3mtrain = temp2[:,0:featm-1]
class3mtrain = class3mtrain.transpose() #Col = samples
#Separating Class 5
temp2 = temp1[temp1[:,featm-1]==5]
class5mtrain = temp2[:,0:featm-1]
class5mtrain = class5mtrain.transpose() #Col = samples
```

```
# test data #
# Separating into classes- class0 test
temp1 = mtesttr #columns = features
temp2 = temp1[temp1[:,featm-1]==0]
class0mtest = temp2[:,0:featm-1]
class0mtest = class0mtest.transpose() #Col = samples
#Separating Class 1
temp2 = temp1[temp1[:,featm-1]==1]
class1mtest = temp2[:,0:featm-1]
class1mtest = class1mtest.transpose() #Col = samples
#Separating Class 3
temp2 = temp1[temp1[:,featm-1]==3]
class3mtest = temp2[:,0:featm-1]
class3mtest = class3mtest.transpose()#Col = samples
#Separating Class 5
temp2 = temp1[temp1[:,featm-1]==5]
class5mtest = temp2[:,0:featm-1]
class5mtest = class5mtest.transpose() #Col = samples

# other matrices of train data
#Transpose of Class0,1,3,and 5
class0mtraintr = class0mtrain.transpose();
class1mtraintr = class1mtrain.transpose(); #Col = features
class3mtraintr = class3mtrain.transpose();
class5mtraintr = class5mtrain.transpose();
# Mean of differnt samples of one single feature at a time.
mu0mtrain = np.mean(class0mtraintr,0) #1xfeatm
mu1mtrain = np.mean(class1mtraintr,0) #1xfeatm
mu3mtrain = np.mean(class3mtraintr,0)
mu5mtrain = np.mean(class5mtraintr,0)
#Mean of entire data
temp = (mtraindata[0:featm-1,:]).transpose()
mumtrain = np.mean(temp,0) # 1xfeatm
#Length of each class
N0mtrain = len(class0mtrain[0,:])
N1mtrain = len(class1mtrain[0,:])
N3mtrain = len(class3mtrain[0,:])
N5mtrain = len(class5mtrain[0,:])


# other matrices of test data
#Transpose of Class0,1,3,and 5
class0mtesttr = class0mtest.transpose();
class1mtesttr = class1mtest.transpose(); #Col = features
class3mtesttr = class3mtest.transpose();
class5mtesttr = class5mtest.transpose();
# Mean of differnt samples of one single feature at a time.
mu0mtest = np.mean(class0mtesttr,0) #1xfeatm
mu1mtest = np.mean(class1mtesttr,0) #1xfeatm
mu3mtest = np.mean(class3mtesttr,0)
mu5mtest = np.mean(class5mtesttr,0)
#Mean of entire data
temp = (mtestdata[0:featm-1,:]).transpose()
```

```
mumtest = np.mean(temp,0) # 1xfeatm
#Length of each class
N0mtest = len(class0mtest[0,:])
N1mtest = len(class1mtest[0,:])
N3mtest = len(class3mtest[0,:])
N5mtest = len(class5mtest[0,:])

print(N0mtest,N1mtest,N3mtest,N5mtest)

mtesttr = np.roll(mtesttr,1,axis = 1)
mtraintr = np.roll(mtraintr,1,axis = 1)

#------------Wine data computations-----------#
#WIne data transpose
winetr = winedata.transpose(); #14x178
[feat,samp] = np.shape(winetr)
# Separating into classes- class1
temp1 = winetr.transpose(); #columns = feature
temp2 = temp1[temp1[:,0] == 1]
class1 = temp2.transpose() #rows - features
class1 = class1[1:feat,:] #13x59 #Col = samples
#Separating Class 2
temp2 = temp1[temp1[:,0] == 2]
class2 = temp2.transpose()
class2 = class2[1:feat,:]
#Separating Class 3
temp2 = temp1[temp1[:,0] == 3]
class3= temp2.transpose()
class3 = class3[1:feat,:]
#Transpose of Class1,2,and 3
class1tr = class1.transpose(); #59x13 #Col = features
class2tr = class2.transpose();
class3tr = class3.transpose();
# Mean of differnt samples of one single feature at a time.
mu1 = np.mean(class1tr,0) #1x13
mu2 = np.mean(class2tr,0)
mu3 = np.mean(class3tr,0)
#Mean of entire data
temp = (winetr[1:14,:]).transpose()
mu = np.mean(temp,0) #13x1
#Length of each class
N1 = len(class1[0,:]) #59
N2 = len(class2[0,:]) #71
N3 = len(class3[0,:])

#pyplot.imshow(testdata[0,:])
#pyplot.show()

#print(np.shape(testdata[0,:]))
```

```
980 1135 1010 892
```

In [571]:
```
print(np.shape(class5mtrain))
```

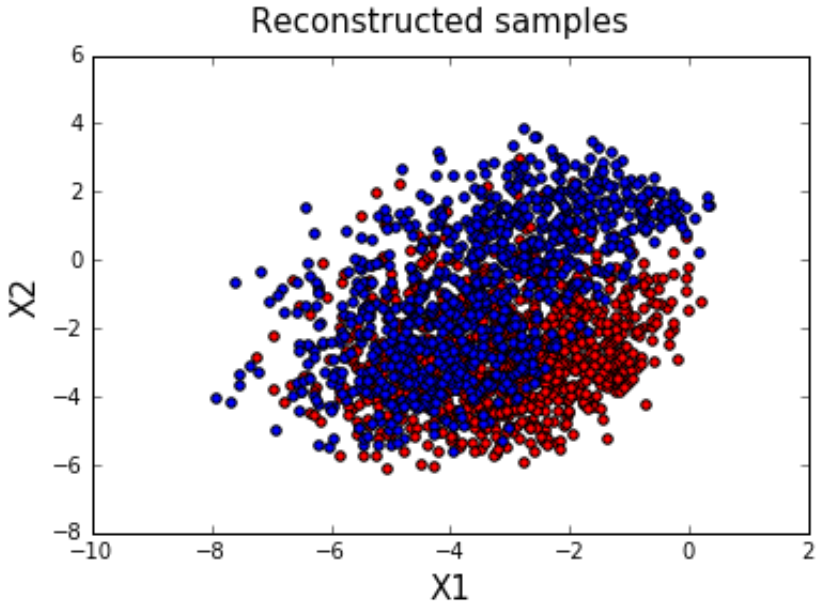
```
(784, 5421)
```

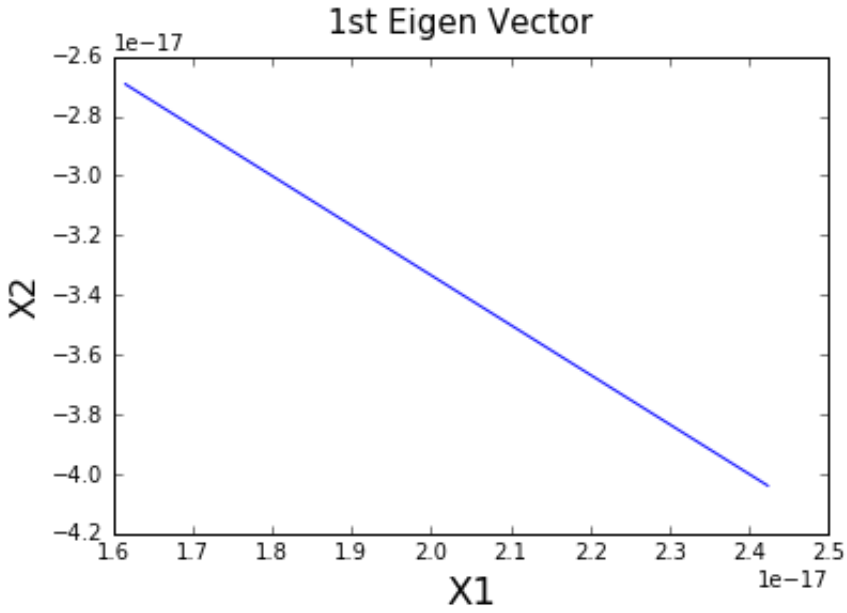
In [705]:

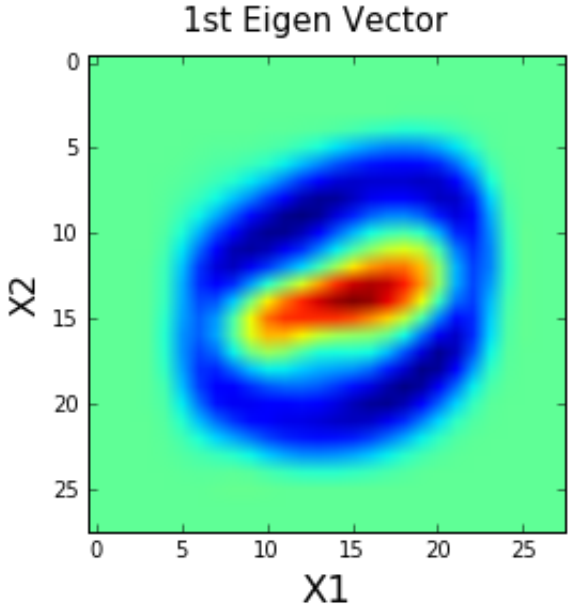
```
#-------PCA--------#
[M,N] = np.shape(winedata)
wmean = winedata.mean(0)
wmean = wmean[1:N]
trainsize = 50
mnist = 1;


ind = random.sample(range(1,M),trainsize)
winetrain = []; #5x13
for i in range(len(ind)):
    if i == 0:
        winetrain = winedata[ind[i],:] #
    else:
        winetrain = np.vstack((winetrain,winedata[ind[i],:]))
#
winetrain = winetrain[:,1:N]
winetest = np.delete(winedata, (ind), axis=0)

temp2 = winetest[winetest[:,0]==1] #columns = features
class1wtest = temp2[:,1:N]

temp2 = winetest[winetest[:,0]==2] #columns = features
class2wtest = temp2[:,1:N]

#---For Mnist
if (mnist == 1):
    [M,N] = np.shape(mtraintr)
    winetrain = mtraintr[:,1:N] # col - features
    winetest = mtesttr[:,1:N]
    class1wtest = class3mtesttr # col = features
    class2wtest = class5mtesttr
#------

trainsize = np.shape(winetrain[:,0])
trainsize = trainsize[0]
wmean = np.mean(winetrain,axis = 0)
print(np.shape(np.tile(wmean,(24217,1))))
X = winetrain - np.tile(wmean,(trainsize,1))

#xcov = np.dot(X.transpose(),X);
U,S,V = np.linalg.svd(X,full_matrices= False)
print(np.shape(V))

#SVD in python gives us transpose of the eigen vector(or Principal compo
nent) matrix.
PC1 = np.dot(V.transpose(),-1)
PC = PC1[:,0:2]

#---Projections---#
#testlabels = winetest[:,0]

pyplot.suptitle('Reconstructed samples', fontsize=15)
```

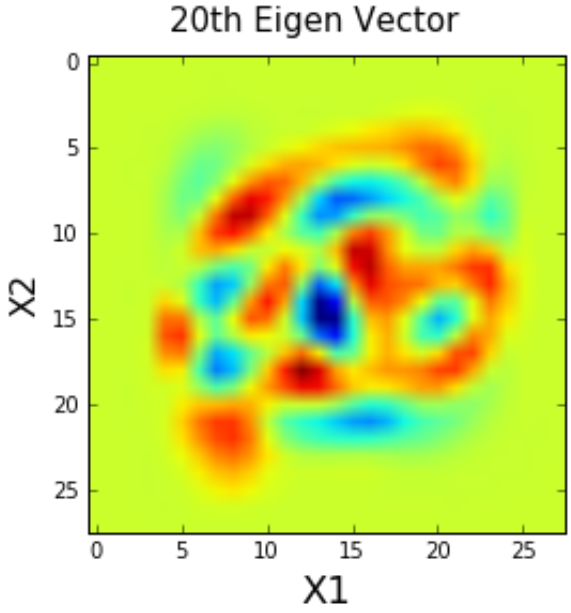
```
pyplot.xlabel('X1', fontsize=15)
pyplot.ylabel('X2', fontsize=15)
projpca1 =  np.dot(PC.transpose(),class1wtest.transpose())
pyplot.figure(1)
pyplot.scatter(projpca1[0,:],projpca1[1,:],c='r')
projpca2 =  np.dot(PC.transpose(),class2wtest.transpose())
pyplot.scatter(projpca2[0,:],projpca2[1,:],c='b')

if (mnist == 1):
    projpca1train = np.dot(PC.transpose(),class0mtraintr.transpose())
    projpca2train = np.dot(PC.transpose(),class1mtraintr.transpose())
    fullproj1test = np.dot(PC1.transpose(),class0mtesttr.transpose())
    #

pyplot.figure(2)
pyplot.suptitle('1st Eigen Vector', fontsize=15)
pyplot.xlabel('X1', fontsize=18)
pyplot.ylabel('X2', fontsize=16)
t = range(28,43)
line_x1 = np.dot(t,V.transpose()[0,0])
line_y1 = np.dot(t,V.transpose()[0,1])
pyplot.plot(line_x1, line_y1)

#---------Plot image of eigen vector for Mnist
if (mnist == 1):
    ev1 = PC1[:,0]
    ev20 = PC1[:,19]

    imev1 = ev1.reshape((28,28))
    imev20 = ev20.reshape((28,28))

    pyplot.figure(3)
    pyplot.suptitle('1st Eigen Vector', fontsize=15)
    pyplot.xlabel('X1', fontsize=18)
    pyplot.ylabel('X2', fontsize=16)
    pyplot.imshow(imev1)

    pyplot.figure(4)
    pyplot.suptitle('20th Eigen Vector', fontsize=15)
    pyplot.xlabel('X1', fontsize=18)
    pyplot.ylabel('X2', fontsize=16)
    pyplot.imshow(imev20)

    pyplot.figure(5)
    pyplot.suptitle('Reconstructed Example', fontsize=15)
    ev5 = fullproj1test[:,20]
    imev5 = ev5.reshape((28,28))
    pyplot.imshow(imev5)
```

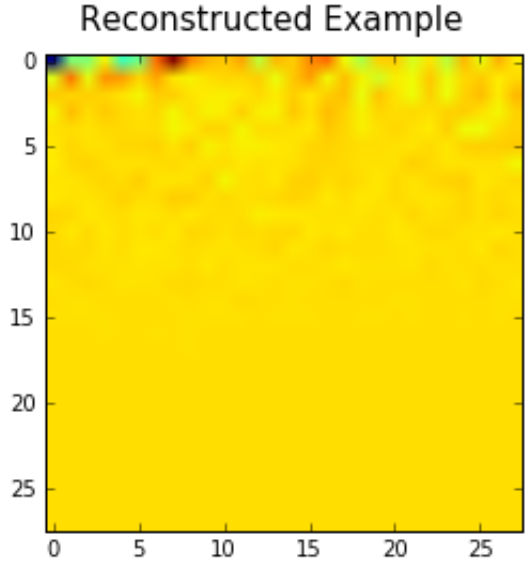
```
(24217, 784)
(784, 784)
```






1st Eigen Vector
X2
X1
0
5
10
15
20
25


20th Eigen Vector
X2
X1
0
5
10
15
20
25

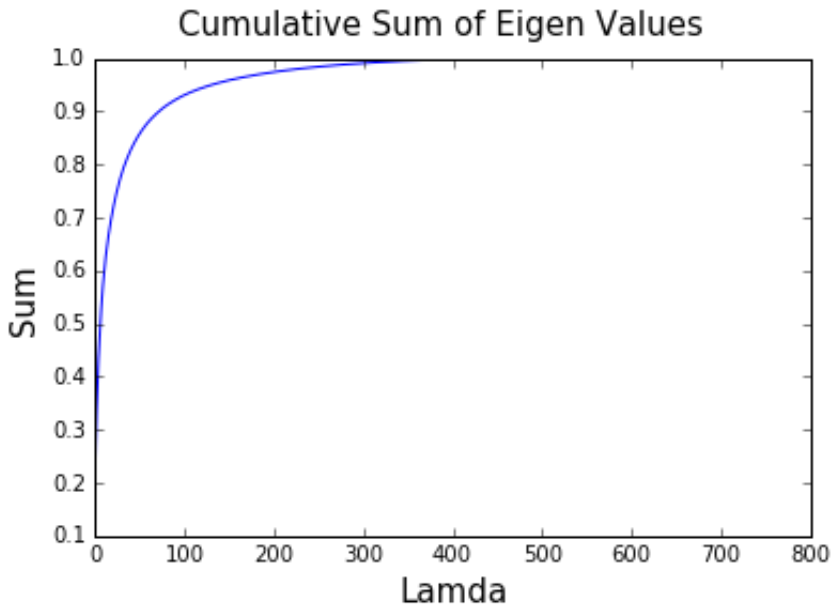

Reconstructed Example
0
5
10
15
20
25

In [596]:
```
#print(np.sum(class0mtraintr))
#(np.sum(class1mtraintr))
np.shape(PC1)
```

Out[596]: (784, 784)

In [706]:
```
#--------PCA Cumulative Sum--------#
eigval = np.multiply(S,S)/(M-1)
cumcov = [];
for i in range(len(eigval)):
    temp = sum(eigval[0:i+1])
    cumcov.append(temp)

cumcov = cumcov /sum(eigval)
pyplot.suptitle('Cumulative Sum of Eigen Values', fontsize=15)
pyplot.xlabel('Lamda', fontsize=15)
pyplot.ylabel('Sum', fontsize=15)
pyplot.plot(cumcov)
#print(eigval)

lamda = np.sum(eigval)
lamda = lamda - eigval[0] - eigval[1]
ReconError = lamda/np.sum(eigval)
print(ReconError)
```

```
0.730744247346
```



In [ ]:

In [709]:

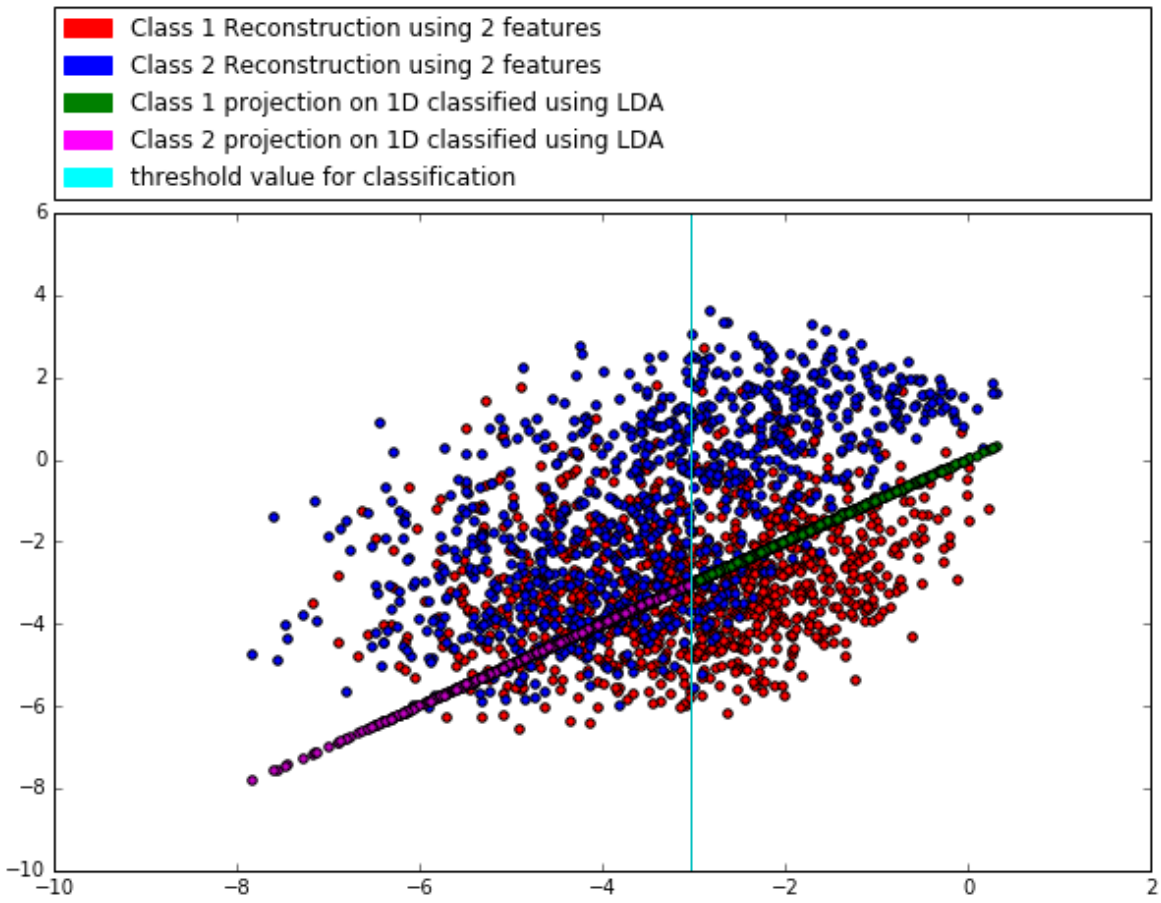
```
#-----------LDA------------#

#a = np.array([[1,2,1,1],[5,6,7,8]])#,[9,10,11,12]])

#[feat,samp] = np.shape(winetr)
#class1 #13x59 #Col = samples
#Class 2 #Rows = features, #Col = samples
#Class 3 #Rows = features, #Col = samples
#class1tr = class1.transpose(); #59x13 #Col = features
#class2tr = class2 transpose
#class3tr = class3 transpose

#mu1, mu2, mu3 #Mean of differnt samples of one single feature at a tim
e. #1x13
#mu - mean of entire dataset
#N1, N2, N3 - lenght of each class - number of samples in each class

#-------Mnist
if (mnist == 1):
    class1wtrain = projpca1train.transpose(); # 6742 x 2 #6742 = number
of samples # col = features
    class2wtrain = projpca2train.transpose();
    class1wtest = projpca1.transpose(); # 980x2 980 = number of samples
# col = features
    class2wtest = projpca2.transpose()
    trainlth = len(projpca1train[:,0])
    mu1 = np.mean(class1wtrain, axis = 0)
    mu2 = np.mean(class2wtrain, axis = 0)
#----------

#--------Wine
if (mnist == 0):
    trainlth = 5;
    ind = [];
    for i in range(5):
        ind = np.hstack((ind,0))

    ind = random.sample(range(1,N1),trainlth)
    #ind = np.array([0,1,2,3,4])
    class1train = []; #5x13
    for i in range(len(ind)):
        if i == 0:
            class1wtrain = class1tr[ind[i],:]
        else:
            class1wtrain = np.vstack((class1wtrain,class1tr[ind[i],:]))

    class1wtest = np.delete(class1tr, (ind), axis=0) #59x13

    ind = random.sample(range(1,N2),trainlth)
    #ind = np.array([0,1,2,3,4])
    class2train = []; #5x13
    for i in range(len(ind)):
        if i == 0:
```

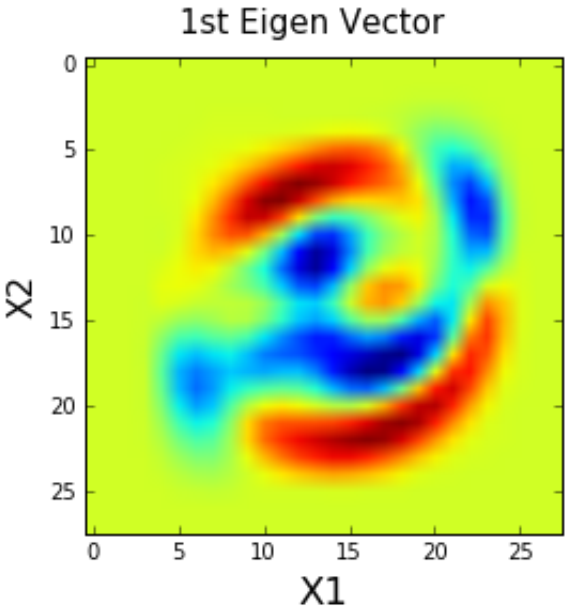
```
            class2wtrain = class2tr[ind[i],:] # 5x13
        else:
            class2wtrain = np.vstack((class2wtrain,class2tr[ind[i],:]))

    class2wtest = np.delete(class2tr, (ind), axis=0)
#------------------

print("Data Loaded")
SW = 0;
SB = 0;
for i in range(1 , 3):
    if i == 1:
        Nx = trainlth
        classx = class1wtrain.transpose() # 13x5 #2x980
        mux = np.mean(class1wtrain,axis=0);
        mu1 = mux;
    elif i == 2:
        Nx = trainlth;
        classx = class2wtrain.transpose();
        #mux = mu2;
        mux = np.mean(class2wtrain,axis = 0)
        mu2 = mux;
    #
    S = 0
    S = np.cov(classx);
    SW = SW + S;

#
print("SW generated")
SB = np.multiply.outer((mu1 - mu2),(mu1 - mu2))
print("SB generated")

SWinv = np.linalg.inv(SW);
div = np.dot(SWinv,SB);
e, v = np.linalg.eig(div)
print("eigen values generated")

v_sorted = v[np.argsort(np.abs(e))]
v_sorted = v_sorted[::-1]

#--Plotting eigel vectors
#pyplot.figure(1)
#pyplot.suptitle('13th Eigen Vector', fontsize=15)
#pyplot.xlabel('X1', fontsize=15)
#pyplot.ylabel('X2', fontsize=15)
#t = range(28,43)
#line_x1 = np.dot(t,v_sorted.transpose()[12,0])
#line_y1 = np.dot(t,v_sorted.transpose()[12,1])
#pyplot.plot(line_x1, line_y1)

#----Finding threshold for classification---#
Wmat = v[:,0:2]; #Columns = new vector features
Wmattemp = Wmat[:,0]
```

```
linproj1 = np.dot(Wmattemp,class1wtrain.transpose())
mulin1 = np.mean(linproj1)
Wmattemp = Wmat[:,0]
linproj2 = np.dot(Wmattemp,class2wtrain.transpose())
mulin2 = np.mean(linproj2)
thresh = (mulin1+mulin2)/2

if (mulin2 >= mulin1):
    rightclass = 2;
    leftclass = 1;
else:
    rightclass = 1;
    leftclass = 2;
#
testdata = np.vstack((class1wtest,class2wtest))
projecteddata = np.dot(Wmat.transpose(),testdata.transpose())
ind = np.where(projecteddata[0,:]>= thresh)
leftind = np.where(projecteddata[0,:] < thresh)
projecteddata[1,leftind]= leftclass;
projecteddata[1,ind] = rightclass;

trueprob1 = np.array([1])
trueprob1 = np.tile(trueprob1,len(class1wtest[:,0]))

trueprob2 = ([2])
trueprob2 = np.tile(trueprob2,len(class2wtest[:,0]))

trueprob = np.hstack((trueprob1,trueprob2))
projecteddata = projecteddata.real
confmatlda = confusion_matrix(trueprob,projecteddata[1,:])
accscorelda = accuracy_score(trueprob,projecteddata[1,:])
print("Accuracy of LDA is ",accscorelda*100)
print("conusion matrix is ")
print(confmatlda)

#---Projected classes---#
Wmat = v[:,0:2]; #Columns = new vector features
projected2D1 = np.dot(Wmat.transpose(),class1wtest.transpose())
projected2D2 = np.dot(Wmat.transpose(),class2wtest.transpose())

#pyplot.figure(2)
fig = pyplot.figure(2,figsize = (10,6))
pyplot.scatter(projected2D1[0,:], projected2D1[1,:],c='r')
pyplot.scatter(projected2D2[0,:], projected2D2[1,:],c='b')
pyplot.scatter(projecteddata[0,ind], projecteddata[0,ind],c='g')
pyplot.scatter(projecteddata[0,leftind], projecteddata[0,leftind], c
='m')
pyplot.axvline(x = thresh, c = 'c')
redp = mpatches.Patch(color='red', label='Class 1 Reconstruction using 2
features')
bluep = mpatches.Patch(color='blue', label='Class 2 Reconstruction using
2 features')
greenp = mpatches.Patch(color='green', label='Class 1 projection on 1D c
```

```
lassified using LDA')
magp = mpatches.Patch(color='magenta', label='Class 2 projection on 1D c
lassified using LDA')
cyanp = mpatches.Patch(color='cyan', label='threshold value for classifi
cation')
pyplot.legend(bbox_to_anchor=(0., 1.02, 1., .102), loc=3,
           ncol=1, mode="expand", borderaxespad=0., handles=[redp,bluep,
greenp,magp,cyanp])

reconex2D = fullproj1test[:,22];
#reconex2D = projected2D1[:,0];
#zer = np.tile(np.array([0]),782)
#reconex2D = np.hstack((reconex2D,zer))
#print(np.shape(reconex2D))

pyplot.figure(3)
#pyplot.imshow(reconex2D.reshape((28,28)))
if (mnist == 1):
    ev1 = PC1[:,2]
    ev20 = PC1[:,9]

    imev1 = ev1.reshape((28,28))
    imev20 = ev20.reshape((28,28))

    pyplot.figure(3)
    pyplot.suptitle('1st Eigen Vector', fontsize=15)
    pyplot.xlabel('X1', fontsize=18)
    pyplot.ylabel('X2', fontsize=16)
    pyplot.imshow(imev1)

#ldatemp = LDA()
#ldatemp.fit(wine[:,1:feat])
#ldatemp.
```

```
Data Loaded
SW generated
SB generated
eigen values generated
Accuracy of LDA is  44.6372239748
conusion matrix is
[[488 522]
 [531 361]]
```



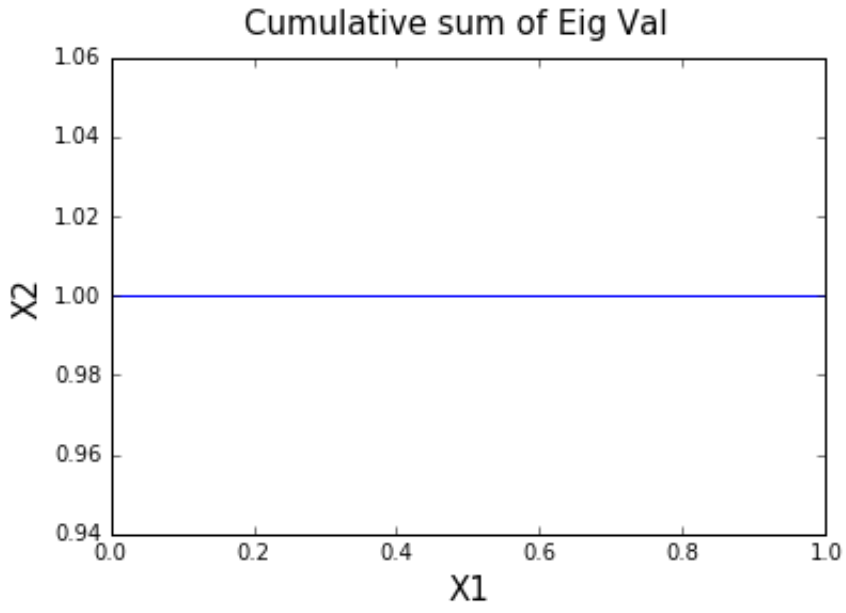


In [697]:
```
#----Display matrices for debugging-----#

#print(np.shape(fullproj1test))
np.shape(ev1)
#projecteddata[1,:]
```

Out[697]: (2,)

In [710]:
```
#--------LDA Cumulative Sum--------#
eigval = sorted(e, reverse = True)
cumcov = [];
for i in range(len(eigval)):
    temp = sum(eigval[0:i+1])
    cumcov.append(temp)

cumcov = cumcov /sum(eigval)
pyplot.suptitle('Cumulative sum of Eig Val', fontsize=15)
pyplot.xlabel('X1', fontsize=15)
pyplot.ylabel('X2', fontsize=15)
pyplot.plot(cumcov)
```

Out[710]: [<matplotlib.lines.Line2D at 0x10c57390>]



In [714]:

```
#------Naive Baye's-------#

#--------Wine data----
[M,N] = np.shape(winedata)
numsamp = 50
ind = [];
for i in range(numsamp):
    ind = np.hstack((ind,0))

ind = random.sample(range(1,N1),numsamp)
#ind = np.array([0,1,2,3,4])
class1train = []; #5x13
for i in range(len(ind)):
    if i == 0:
        class1train = class1tr[ind[i],:]
    else:
        class1train = np.vstack((class1train,class1tr[ind[i],:]))

class1test = np.delete(class1tr, (ind), axis=0)

ind = random.sample(range(1,N2),numsamp)
#ind = np.array([0,1,2,3,4])
class2train = []; #5x13
for i in range(len(ind)):
    if i == 0:
        class2train = class2tr[ind[i],:]
    else:
        class2train = np.vstack((class2train,class2tr[ind[i],:]))

class2test = np.delete(class2tr, (ind), axis=0)
#---------------
#-------Mnist data
if (mnist == 1):
    [M,N] = np.shape(mtraintr);
    class1train = class3mtraintr; #col = features
    class2train = class5mtraintr;
    class1test = class3mtesttr;
    class2test = class5mtesttr
#------------
mu1 = np.mean(class1train,0)
mu2 = np.mean(class2train,0)
mu1sq = np.square(mu1)
mu2sq = np.square(mu2)

wmat = []
trainset = np.vstack((class1train,class2train))
covarc = np.cov(trainset,rowvar = 0)
varc = np.diag(covarc)
for i in range(len(varc)):
    if (varc[i] != 0):
        wmat.append(np.divide((mu1[i]-mu2[i]),varc[i]))
    elif (varc[i] == 0):
        wmat.append(0);
```

```
#wmat = np.divide((mu1-mu2),varc)


pi1 = 5/10
pi2 = 5/10
#------Mnist Prior probabilities
if (mnist == 1):
    ncl0 = len(class0mtraintr[:,0])
    ncl1 = len(class1mtraintr[:,0])
    nboth = ncl0 + ncl1
    pi1 = ncl0/nboth
    pi2 = ncl1/nboth
#-------------

feat = len(class1train[0,:]) #13

w0 = math.log((1-pi1)/pi1)
#for i in range(feat):
    #w0 = w0 + ((mu2sq[i]-mu1sq[i])/(2*varc[i]))
for i in range(len(varc)):
    if (varc[i] != 0):
        w0 = w0 + ((mu2sq[i]-mu1sq[i])/(2*varc[i]))
    elif (varc[i] == 0):
        w0 = 0;
#
probmat = [];
debug = [0];
testmat = np.vstack((class1test,class2test))
for i in range(len(testmat[:,0])):
    wixi = 0;
    #for j in range(feat):
    #    wixi = wixi + np.multiply(wmat[j], testmat[i,j])
    #
    wixi = np.dot(wmat,testmat[i,:])
    debug.append(wixi)
    pC0x = np.divide((math.exp(w0+wixi)),(1+math.exp(w0+wixi)))
    if pC0x >= pi1:
        probmat.append(1)
    else:
        probmat.append(2)#2
    #
#

probmat.count(1)
trueprob1 = np.array([1])
trueprob1 = np.tile(trueprob1,len(class1test[:,0]))

trueprob2 = ([2])
trueprob2 = np.tile(trueprob2,len(class2test[:,0]))

trueprob = np.hstack((trueprob1,trueprob2))

confmatrix = confusion_matrix(trueprob, probmat)
```

```
accscore = accuracy_score(trueprob, probmat)

submat = []
submat = np.subtract(probmat,trueprob)
diff = probmat - trueprob
#for s in range()
#conf21 = diff.count(-1)
#conf12 = diff.count(1)
#conf11 = len(class1test - conf12)
#conf22 = len(class2test - conf21)

#confmat = np.array([[conf11,conf12],[conf21,conf22]])

#print("Accuracy_Score is ",accscore)
print("Accuracy percentage is ",accscore*100)
print("Confusion Matrix is ")
print(confmatrix)
```

```
Accuracy percentage is  79.2849631966
Confusion Matrix is
[[1006    4]
 [ 390  502]]
```

In [682]:
```
#fmnisttest = open('test.csv', 'r')
#mtestdata = np.loadtxt("test.csv", delimiter = ",", skiprows = 1, unpac
k=False) #samples x feats
#x = (wmat[0])
#y = (testmat[0,0])
#print(sum(mu1-mu2))
#print(wmat)
#wmat
#print(range(0,100))
#a = np.cov(trainset)
#b = np.diag(a)
#print(np.shape(trainset))
#a = np.array([[1,2,3,4],[3,4,5,6]])
#np.var(trainset[:,0])
#trainset[:,0]
np.shape(probmat)
```

Out[682]: (30,)

In [ ]: