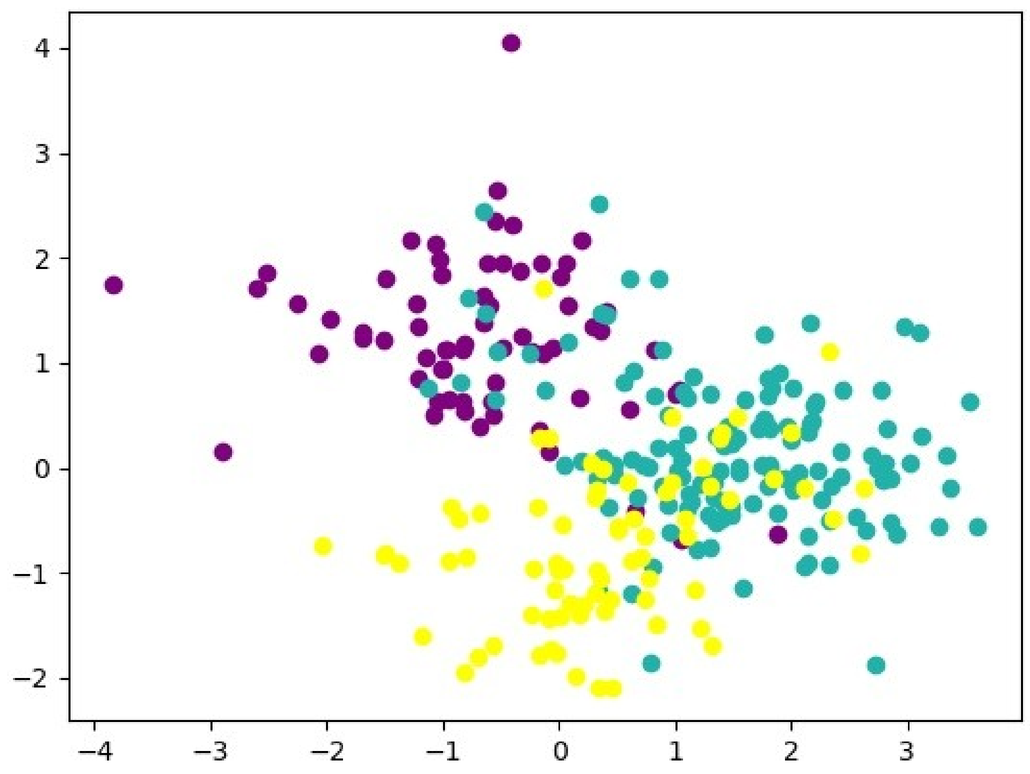
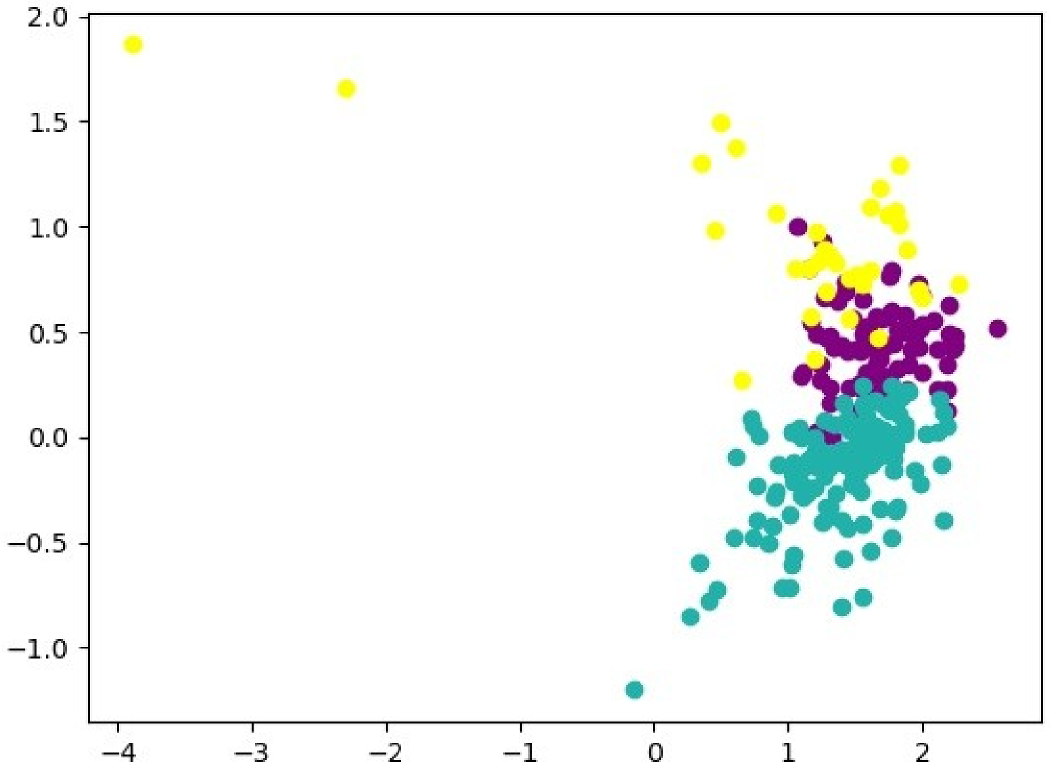
Nate Dauterman
ECE 49595
Feb 15, 2022

Cho PCA Graph



Iyer PCA Graph



pca Function from pca_template.py

```
def pca(dataMat, PC_num=2):
    '''
    Input:
        dataMat: obtained from the loadDataSet function, each row represents an observation
                 and each column represents an attribute
        PC_num:  The number of desired dimensions after applyting PCA. In this project keep it to 2.
    Output:
        lowDDataMat: the 2-d data aPCfter PCA transformation
    '''

    dataMat = numpy.matrix(dataMat)

    #print(len(dataMat))
    #print(dataMat)

    means = []

    for row in dataMat:
        if len(means) == 0:
            for col in row:
                means.append(col)
        else:
            means = [a + b for a, b in zip (means, row)]

    for i, item in enumerate(means):
        means[i] = item / len(dataMat)

    #print(means)
    #print(dataMat.transpose())

    for i in range(len(dataMat)):
        dataMat[i] = [a - b for a, b in zip(dataMat[i], means)]

    n = dataMat.size

    #print(numpy.matmul(dataMat.transpose(), dataMat))
    covariance = numpy.matmul(dataMat.transpose(), dataMat) / (n - 1)

    #print(covariance)

    eigvals, eigvecs = numpy.linalg.eig(covariance)
    #print(eigvals)
    #print(eigvecs)

    eigvecs = eigvecs.transpose()

    zippedEigs = zip(eigvals, eigvecs)
    sortedZipped = sorted(zippedEigs, reverse=True)
    sortedVecs = [element for i, element in (sortedZipped)]

    #print(sortedVecs)

    useEigvecs = sortedVecs[0:PC_num]

    #print(useEigvecs)

    lowDDataMat = []
    for i in range(len(dataMat)):
        newRow = numpy.matmul(useEigvecs, numpy.array(dataMat[i].transpose()))
        lowDDataMat.append(newRow)

    #print(lowDDataMat)

    return array(lowDDataMat)
```

plot Function from pca_template.py

```
def plot(lowDDataMat, labelMat, figname):
    '''
    Input:
        lowDDataMat: the 2-d data after PCA transformation obtained from pca function
        labelMat: the corresponding label of each observation obtained from loadData
    '''
    sets = []

    for row in lowDDataMat:
        temp = []
        for j in range(len(row)):
            temp.append(list(row[j])[0][0]) #to strip the extra list encapsulations
        sets.append(temp)


    for point, label in zip(sets, labelMat):
        if label == 1:
            plt.plot(point[0], point[1], marker='o', markeredgecolor='purple', markerfacecolor='purple')
        if label == 2:
            plt.plot(point[0], point[1], marker='o', markeredgecolor='lightseagreen',
markerfacecolor='lightseagreen')
        if label == 3:
            plt.plot(point[0], point[1], marker='o', markeredgecolor='yellow', markerfacecolor='yellow')


    #plt.plot(sets[0], sets[1], 'ro')
    #plt.show()
    plt.savefig(figname)
```