CSE 469: Assignment 1

Principle Component Analysis

Peter M. VanNostrand

09/21/2019

# Assignment 1: Principle Componenet Analysis

## PCA Scatter Plots

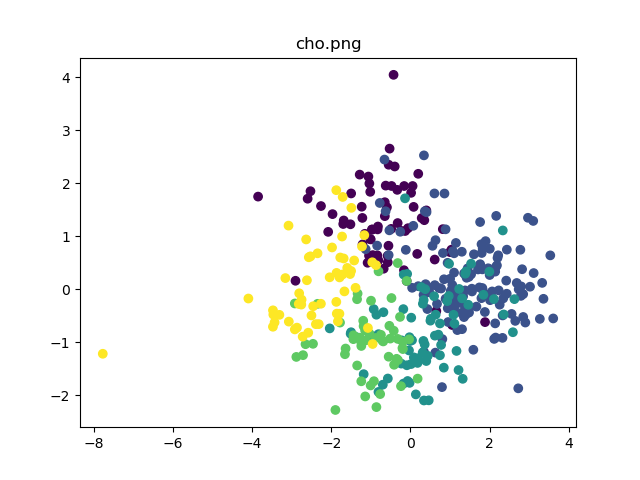


Figure 1: PCA on Cho Dataset

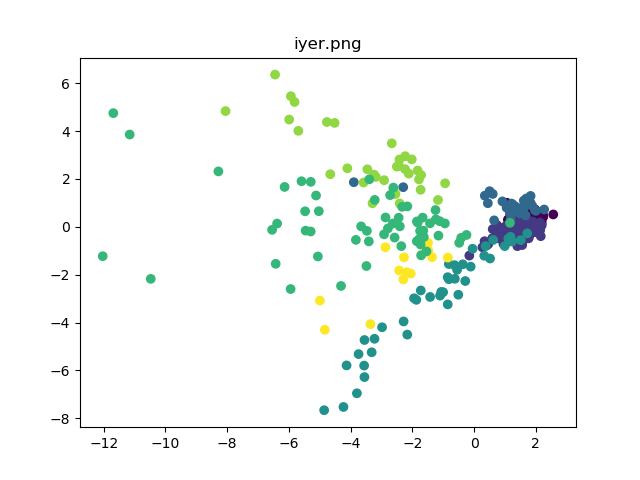


Figure 2: PCA on Iyer Dataset

## PCA Code

# coding: utf-8

# In[1]:

from matplotlib import pyplot as plt

import sys

import numpy as np

def loadDataSet(fileName = 'iris.csv'):

    dataMat=[]

    labelMat=[]

    fr = open(fileName)

    for line in fr.readlines():

        lineArray=line.strip().split(',')

        records = []

        for np.attr in lineArray[:-1]:

            records.append(float(np.attr))

        dataMat.append(records)

        labelMat.append(int(lineArray[-1]))

    dataMat = np.array(dataMat)

    labelMat = np.array(labelMat)

    return dataMat,labelMat

def normalize(matrix):

    '''

    center all columns around zero using x-xbar

    '''

    # create a copy of the array

    x = np.copy(matrix)

    # take the avarage of each column

    avg = np.mean(matrix, axis=0)

    # adjust the values by the average

    for i in range(x.shape[0]): x[i] -= avg

    return x

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

    '''

    plt.figure()

    plot\_data = np.transpose(lowDDataMat)

    plt.scatter(plot\_data[0], plot\_data[1], c=labelMat)

    plt.title(figname)

    plt.savefig(figname)

    plt.show()

def top\_n\_eigen(matrix, n):

    '''

    Returns the top n eigen values and vectors of a matrix sorted by eigen value in descending order

    '''

    # compute the eigen values and vectors with numpy

    # eigenvectors are given by the columns of of vecs

    vals, vecs = np.linalg.eig(matrix)

    # combine these into one matrix

    vals\_vecs = np.vstack((vals, vecs))

    # sort the matrix by largest eigenvalue

    vals\_vecs\_sort = np.transpose(sorted(np.transpose(vals\_vecs),key=lambda x: x[0], reverse=True))

    # extract and return the sorted eigen values and vectors

    vals\_sorted = vals\_vecs\_sort[:1,:n]

    vecs\_sorted = vals\_vecs\_sort[1:,:n]

    return vals\_sorted, vecs\_sorted

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 after PCA transformation

    '''

    # normalize the data

    x = normalize(dataMat)

    # compute the covariance matrix

    n = x.shape[0]

    xT = np.transpose(x)

    covariance = np.dot(xT, x) / (n-1)

    # get top PC\_num eigen vectors

    vals, vecs = top\_n\_eigen(covariance, PC\_num)

    # calculate points in new dimension

    lowDDataMat = np.dot(x, vecs)

    return lowDDataMat

if \_\_name\_\_ == '\_\_main\_\_':

    if len(sys.argv) == 2:

        filename = sys.argv[1]

    else:

        filename = 'iris.csv'

    figname = filename

    figname = figname.replace('csv','png')

    dataMat, labelMat = loadDataSet(filename)

    lowDDataMat = pca(dataMat)

    plot(lowDDataMat, labelMat.T, figname)