Programming Assignment 3: PCA and Fastmap Algorithm

Member: Luyao Wang, Di Jin, Yingqi Lin

Individual contributions:

(1) Luyao Wang:
   Implementation of PCA
   Implementation of Fastmap Algorithm
   The first part of report

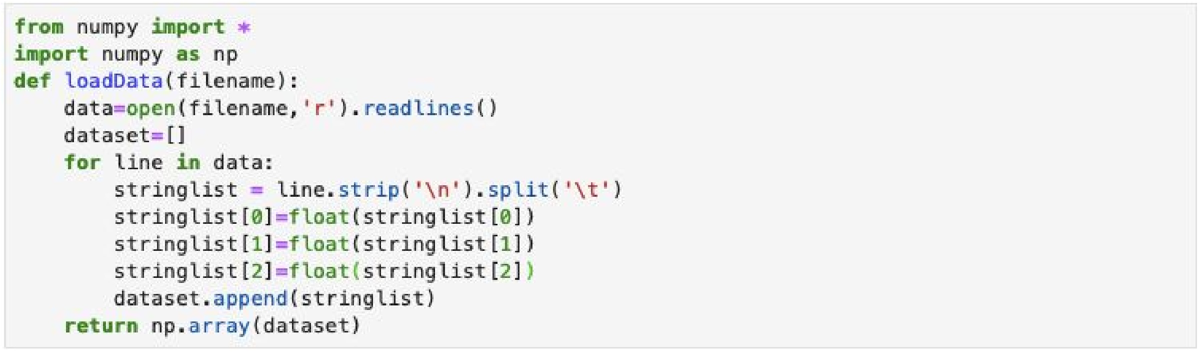
(2) Di Jin:
   Implementation of PCA
   Implementation of Fastmap Algorithm
   The second part and third part of report

(3) Yingqi Lin:
   Discussion of PCA
   Implementation of GMM Algorithm
   The second part and third part of report

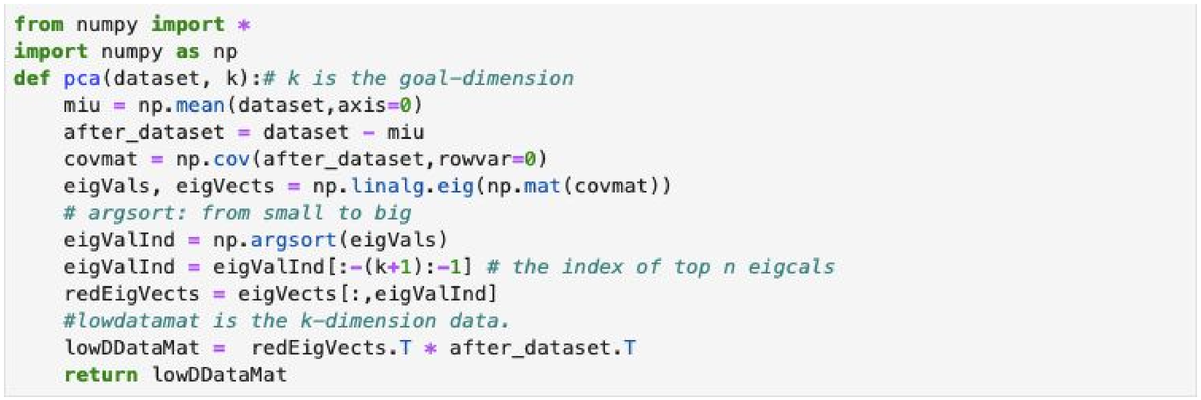
Part 1 :

(1) Implement of PCA

(a) Stricture of data storage



```
from numpy import *
import numpy as np
def loadData(filename):
    data=open(filename,'r').readlines()
    dataset=[]
    for line in data:
        stringlist = line.strip('\n').split('\t')
        stringlist[0]=float(stringlist[0])
        stringlist[1]=float(stringlist[1])
        stringlist[2]=float(stringlist[2])
        dataset.append(stringlist)
    return np.array(dataset)
```



```
from numpy import *
import numpy as np
def pca(dataset, k):# k is the goal-dimension
    miu = np.mean(dataset,axis=0)
    after_dataset = dataset - miu
    covmat = np.cov(after_dataset,rowvar=0)
    eigVals, eigVects = np.linalg.eig(np.mat(covmat))
    # argsort: from small to big
    eigValInd = np.argsort(eigVals)
    eigValInd = eigValInd[:-(k+1):-1] # the index of top n eigcals
    redEigVects = eigVects[:,eigValInd]
    #lowdatamat is the k-dimension data.
    lowDDataMat =  redEigVects.T * after_dataset.T
    return lowDDataMat
```

We choose an array of float as the structure of data storage. Because we can call some methods or functions easily. We use list to store eigvals and eigvects, so that we can easily get the top k elements by their index. And the results which pca() returns is a type of list of lists.

(b) Optimization on code level

We seperate the loaddata function from the pca function. So the steps of pca algorithm are more clear and the code is more readable.

We try to learn some methods from numpy, and we use some of them, such as mean(), cov(),eig(), in order to reduce the complexity of codes and make them look more elegant.

(c) Challenge

When we get the sorted eigvals, it is from the smallest to the biggest, so we need to withdraw k elements from the last one to the (last-k). However, we are not familiar with the syntax [m:n:k]. It takes a lot of time to figure out how to get top k elements from eigvals.

(d) Results

```python
dataset=loadData('pca-data.txt')
print(pca(dataset,2))
```

```
[[ 10.87667009 -12.68609992   0.43255106 ...  -2.92254009  11.18317124
   14.2299014 ]
 [  7.37396173  -4.24879151   0.26700852 ...   2.41914881   4.20349275
    5.64409544]]
```

(2) implement of FastMap

(a) Stricture of data storage

We use array as the structure of data which is downloaded from txt files.

We use tuple (oa, ob,ditance_ab) as the result of find_function() so that we can easily get the value of oa, ob, and their distance.

In the function cos(), we choose a dictionary to store the xi of each object, the key is the index of each object and the value is the value of its xi.

We also use a list of lists to store the values of each dictionary, thus each row represents new k-dimension data based on each original data. In this case, we use the 2D list to draw a picture of points.

```python
from numpy import *
import numpy as np
def loadData(filename):
    data=open(filename,'r').readlines()
    dataset=[]
    for line in data:
        stringlist = line.strip('\n').split('\t')
        if len(stringlist)>1:
            stringlist[0]=float(stringlist[0])
            stringlist[1]=float(stringlist[1])
            stringlist[2]=float(stringlist[2])
        dataset.append(stringlist)
    return np.array(dataset)
Dataset=loadData('fastmap-data.txt')
word=loadData('fastmap-wordlist.txt')
```

```python
def find_function(Dataset):
    Farthest= -1
    for items in Dataset:
        if items[2]>Farthest:
            Farthest=items[2]
    for items in Dataset:
        if items[2]==Farthest:
            (oa,ob,d_ab)=(items[0],items[1],Farthest)
            break
    return (oa,ob,d_ab)
```

```python
def cos(oa,ob,d_ab,Dataset,wordlist):
    dt=list(Dataset)
    wl=list(wordlist)
    dictionary=dict()
    for i in range(len(wl)):
        oi=i+1 #first xi is correct,oi should start from 1 not 0.
        d_ai=0
        d_bi=0
        for d in dt:
            if (d[0]==oa and d[1]==oi) or (d[0]==oi and d[1]==oa):
                d_ai=d[2]
            if (d[0]==ob and d[1]==oi) or (d[0]==oi and d[1]==ob):
                d_bi=d[2]
            if d_ai and d_bi:
                break
        xi=(d_ai**2+d_ab**2-d_bi**2)/(2*d_ab)
        dictionary[i]=xi
    return dictionary
```

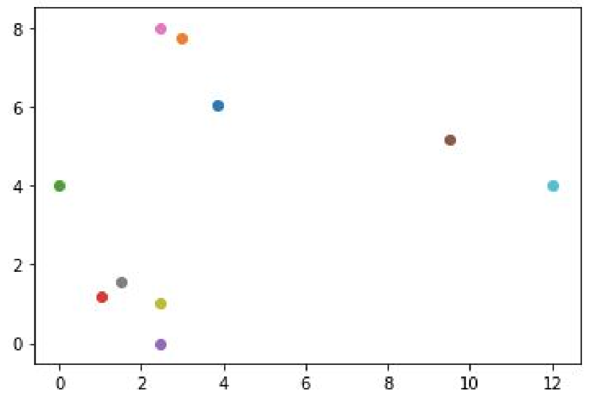
```python
#update
import math
def update(dictionary,dataset):
    dt=list(dataset)
    for oi in range(1,11):
        for element in dictionary:
            if element+1==oi:#element start from 1
                xi=dictionary[element]
                break
        for oj in range(oi+1,11):
            for element in dictionary:
                if element+1==oj:#element start from 1
                    xj=dictionary[element]
                    break
            #print(dt)
            for d in dt:
                if (d[0]==oi and d[1]==oj) or (d[1]==oi and d[0]==oj):
                    d_old_ij=d[2]
                    d_new_ij=math.pow(abs(d_old_ij**2-(xi-xj)**2),0.5)
                    d[2]=d_new_ij
    return np.array(dt)
```

```python
import matplotlib.pyplot as plt
def fastmap(Dataset,word):
    dt=list(Dataset)
    wl=list(word)
    oa,ob,d_ab=find_function(Dataset)
    dictionary=cos(oa,ob,d_ab,Dataset,word)
    return dictionary

def main(Dataset,word):
    dictionary1=fastmap(Dataset,word)
    sub_dataset=update(dictionary1,Dataset)
    print(sub_dataset)
    dictionary2=fastmap(sub_dataset,word)
    list_2D=list()
    for i in range(len(word)):
        x=dictionary1[i]
        y=dictionary2[i]
        list_2D.append([x,y])

    fig, ax = plt.subplots()
    for i in range(10):
        ax.scatter(list_2D[i][0], list_2D[i][1])
    plt.show()

    return list_2D

main(Dataset,word)
```

(b) Optimization on code level
We divided the whole complex program into a few functions, and then we call them by the steps of fastmap algorithm in function main(). In this way, we can easily find mistakes and check our thoughts on coding. Also, this way makes our structure of the program more clear.

(c) Challenge
When we finished our fastmap program and got a result, we found the result is a little different from the result which was obtained by running another public fastmap program. This is because we made a mistake on indexs of the dictionary of xi and the array of data. We debugged our code and finally found the mistake and corrected it.

(d) Result

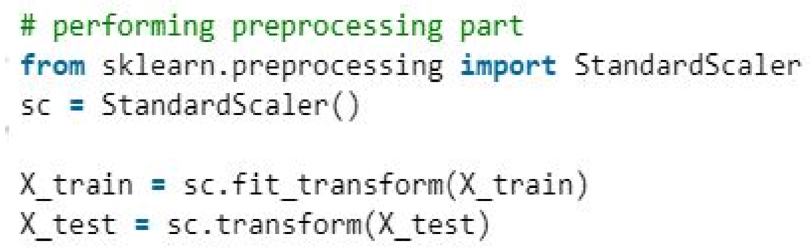


```
[[3.875, 6.0625],
 [3.0, 7.749999999999999],
 [0.0, 4.0],
 [1.0416666666666667, 1.1875],
 [2.4583333333333335, 0.0],
 [9.5, 5.1875],
 [2.4583333333333335, 8.0],
 [1.5, 1.5624999999999996],
 [2.4583333333333335, 1.0],
 [12.0, 4.0]]
```

Part 2:

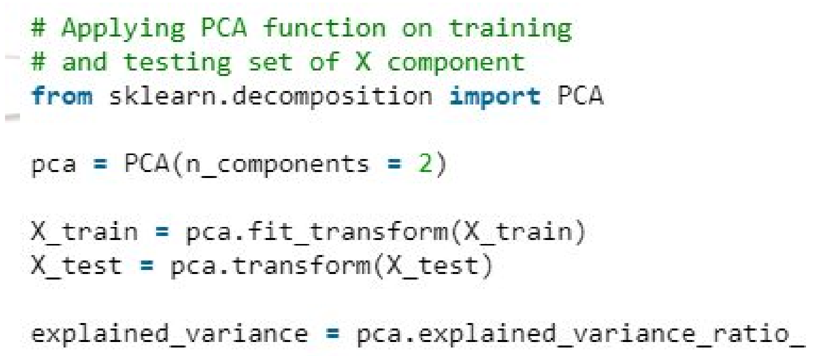
(1) PCA

(a) How to use PCA
The library function to use the algorithm PCA is sklearn.decomposition.PCA. In order to use the library function PCA, there are the following steps we should use. Firstly, importing required libraries including from sklearn.decomposition import PCA. Secondly, importing or loading the dataset. Then standardized the data through the function StandardScaler.



```
# performing preprocessing part
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

So we should from sklearn.preprocessing import StandardScaler so that standardize the dataset's features onto unit scale (mean = 0 and variance = 1).



```
# Applying PCA function on training
# and testing set of X component
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)

X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)

explained_variance = pca.explained_variance_ratio_
```

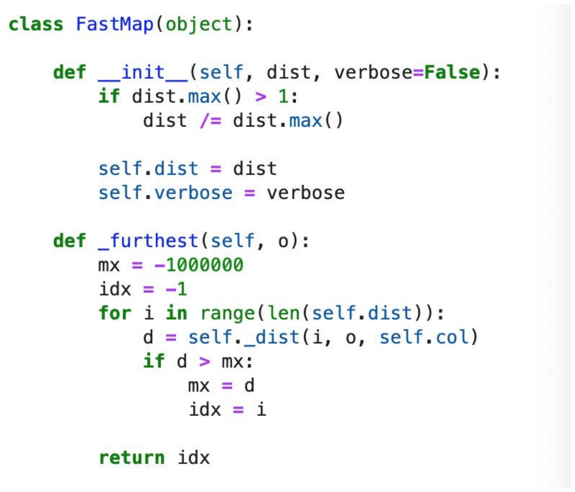
Then apply PCA function for analysis. Finally, visualize the projection.

(b) Compare and improve

When we use the library function to reduce the dimension, it can concise our code in calculate. We do not need to calculate the data by ourselves step by step. We only need to preprocess data before applying the library function. After preprocessing the data including standardized the data and divided into the training set and test set, we can import the library function and apply the library function in our code.

(2) FastMap

(a) How to use FastMap



```
class FastMap(object):

    def __init__(self, dist, verbose=False):
        if dist.max() > 1:
            dist /= dist.max()

        self.dist = dist
        self.verbose = verbose

    def _furthest(self, o):
        mx = -1000000
        idx = -1
        for i in range(len(self.dist)):
            d = self._dist(i, o, self.col)
            if d > mx:
                mx = d
                idx = i

        return idx
```

__init__ method is used to initialize data. The algorithm uses _furthest method to find the index of the furthest point.

```python
def _pickPivot(self):
    o1 = random.randint(0, len(self.dist) - 1)
    o2 = -1

    i = DISTANCE_ITERATIONS

    while i > 0:
        o = self._furthest(o1)
        if o == o2:
            break
        o2 = o
        o = self._furthest(o2)
        if o == o1:
            break
        o1 = o
        i -= 1

    self.pivots[self.col] = (o1, o2)
    return (o1, o2)
```

The algorithm uses _pickPivot to find the two most distant points o1 and o2.

```python
def _map(self, K):
    if K == 0:
        return

    px, py = self._pickPivot()

    if self.verbose:
        print ("Picked %d, %d at K = %d" % (px, py, K))

    if self._dist(px, py, self.col) == 0:
        return

    for i in range(len(self.dist)):
        self.res[i][self.col] = self._x(i, px, py)

    self.col += 1
    self._map(K - 1)

def _x(self, i, x, y):
    dix = self._dist(i, x, self.col)
    diy = self._dist(i, y, self.col)
    dxy = self._dist(x, y, self.col)
    return (dix + dxy - diy) / 2 * math.sqrt(dxy)
```

The algorithm uses _x method to project the i'th point onto the line defined by x and y.
The algorithm uses _map method to update the matrix.

```python
def _dist(self, x, y, k):
    if k == 0:
        return self.dist[x, y] ** 2

    rec = self._dist(x, y, k - 1)
    resd = (self.res[x][k] - self.res[y][k]) ** 2
    return rec - resd

def map(self, K):
    self.col = 0
    self.res = scipy.zeros((len(self.dist), K))
    self.pivots = scipy.zeros((K, 2), "i")
    self._map(K)
    return self.res

def fastmap(dist, K):
    """dist is a NxN distance matrix
    returns coordinates for each N in K dimensions
    """

    return FastMap(dist, verbose=True).map(K)
```

The algorithm uses _dist method to recursively compute the distance based on previous projections.

(b) Compare and improve

The library function's algorithm is simpler than our algorithm. It has a more concise structure. Our algorithm only has two recursions and can not change the times of recursion. If the dataset has too many dimensions, our algorithm will create a lot of dictionaries, which will take up a lot of storage space. Therefore, we can change our storage structure like the library function's algorithm.

Part 3:

(1) Application of PCA

PCA can be applied in fields like facial recognition, computer vision, and image compression as a dimensionality reduction technique. For the field of image processing, PCA can also be used to filter noisy datasets to help process the image. It is also used for finding patterns in data of high dimension in the field of finance, data mining, bioinformatics, psychology, etc so that we can reduce the dimension to simplify the problem and solve the problem.

(2) Application of FastMap

The Fastmap algorithm can break a song into a series of sounds, convert the audio into a sequence of symbols, and store it in a database in the form of a matrix. Fastmap makes each data object can be regarded as a point in Euclidean space. The distance between points represents the distance between objects, which is helpful for the research of song matching search. The Fastmap algorithm is faster than the MDS algorithm and also allows indexing.