07.1ex_Algorithms

March 14, 2020

1. **PCA on 3D dataset**

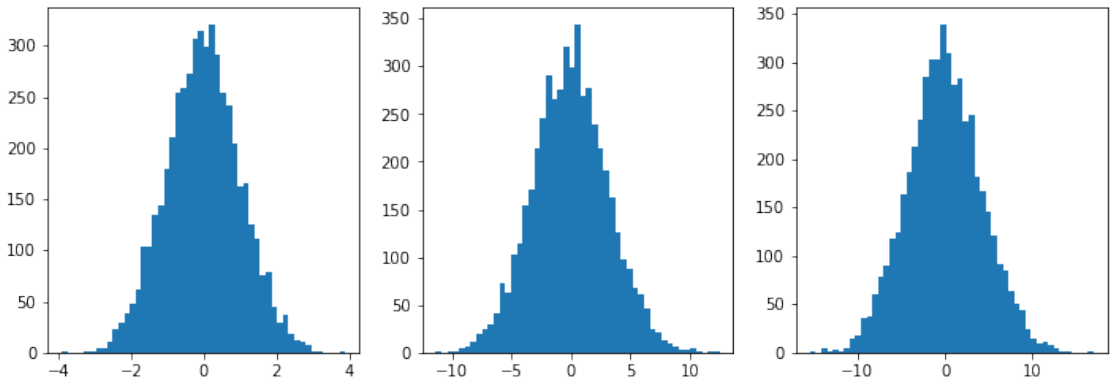
- Generate a dataset with 3 features each with N entries (N being $\mathcal{O}(1000)$). With $N(\mu, \sigma)$ the normali distribution with mean μ and σ standard deviation, generate the 3 variables $x_{1,2,3}$ such that:
  - x_1 is distributed as $N(0, 1)$
  - x_2 is distributed as $x_1 + N(0, 3)$
  - x_3 is given by $2x_1 + x_2$
- Find the eigenvectors and eigenvalues of the covariance matrix of the dataset
- Find the eigenvectors and eigenvalues using SVD. Check that they are two procedure yields to same result
- What percent of the total variability is explained by the principal components? Given how the dataset was constructed, do these make sense? Reduce the dimensionality of the system so that at least 99% of the total variability is retained.
- Redefine the data in the basis yielded by the PCA procedure
- Plot the data points in the original and the new coordiantes as a set of scatter plots. Your final figure should have 2 rows of 3 plots each, where the columns show the (0,1), (0,2) and (1,2) proejctions.

In [1]:
```
import pandas as pd
import numpy as np
import scipy as sp
from scipy import linalg
import matplotlib.pyplot as plt
```

In [2]:
```
N=5000

a1=np.random.normal(0,1, N)
a2=np.random.normal(0,3, N)

x1=a1
x2=a1+a2
x3=2*x1+x2

data=pd.DataFrame({'x1':x1,'x2':x2,'x3':x3})
#print(data)
```

```
%matplotlib inline
fig, (fig1, fig2, fig3)= plt.subplots(nrows=1, ncols=3, figsize=(12, 4))
fig1.hist(data["x1"], bins="auto")
fig2.hist(data["x2"], bins="auto")
fig3.hist(data["x3"], bins="auto")
```

Out[2]:
```
(array([  1.,   0.,   4.,   1.,   3.,   2.,   5.,  14.,  18.,  35.,  37.,
         61.,  78.,  90., 117., 124., 164., 186., 213., 241., 284., 303.,
        302., 339., 309., 277., 283., 239., 246., 181., 166., 145., 121.,
         92.,  84.,  64.,  50.,  44.,  24.,  15.,  10.,  11.,   8.,   5.,
          1.,   2.,   0.,   0.,   0.,   1.]),
 array([-15.42970289, -14.78168707, -14.13367125, -13.48565543,
        -12.83763961, -12.1896238 , -11.54160798, -10.89359216,
        -10.24557634,  -9.59756052,  -8.9495447 ,  -8.30152888,
         -7.65351307,  -7.00549725,  -6.35748143,  -5.70946561,
         -5.06144979,  -4.41343397,  -3.76541815,  -3.11740234,
         -2.46938652,  -1.8213707 ,  -1.17335488,  -0.52533906,
          0.12267676,   0.77069258,   1.4187084 ,   2.06672421,
          2.71474003,   3.36275585,   4.01077167,   4.65878749,
          5.30680331,   5.95481913,   6.60283494,   7.25085076,
          7.89886658,   8.5468824 ,   9.19489822,   9.84291404,
         10.49092986,  11.13894567,  11.78696149,  12.43497731,
         13.08299313,  13.73100895,  14.37902477,  15.02704059,
         15.6750564 ,  16.32307222,  16.97108804]),
 <a list of 50 Patch objects>)
```



In [3]:
```
%precision 3

covariance_matrix=np.cov(data, rowvar=False)
eigVal, eigVec = sp.linalg.eig(covariance_matrix)
print("Eingenvalues with cov matrix:\n", eigVal)
print("Eingenvectors with cov matrix:")
```

```
        for i in range(len(eigVec)):
            print(eigVec[:,i])

        u,s,vh = sp.linalg.svd(data)
        eigValSvd = s**2/(N-1)
        print("Eingenvalues with SVD:\n", eigValSvd)
        print("Eingenvectors with SVD:")
        for i in range(len(vh)):
            print(vh[i,:])
```

```
Eingenvalues with cov matrix:
 [ 2.698e+01+0.j -9.853e-16+0.j  2.066e+00+0.j]
Eingenvectors with cov matrix:
[-0.116 -0.577 -0.808]
[-0.816 -0.408  0.408]
[ 0.566 -0.707  0.424]
Eingenvalues with SVD:
 [2.698e+01 2.066e+00 6.960e-31]
Eingenvectors with SVD:
[0.116 0.577 0.808]
[ 0.566 -0.707  0.424]
[-0.816 -0.408  0.408]
```

0.1 COMMENT

The result is the same except for the second and third eigenvalue/vector which is swapped between the 2 methods, all the values differ for a -1 factor but it's not a problem since they are eigenvalues/eigenvectors.

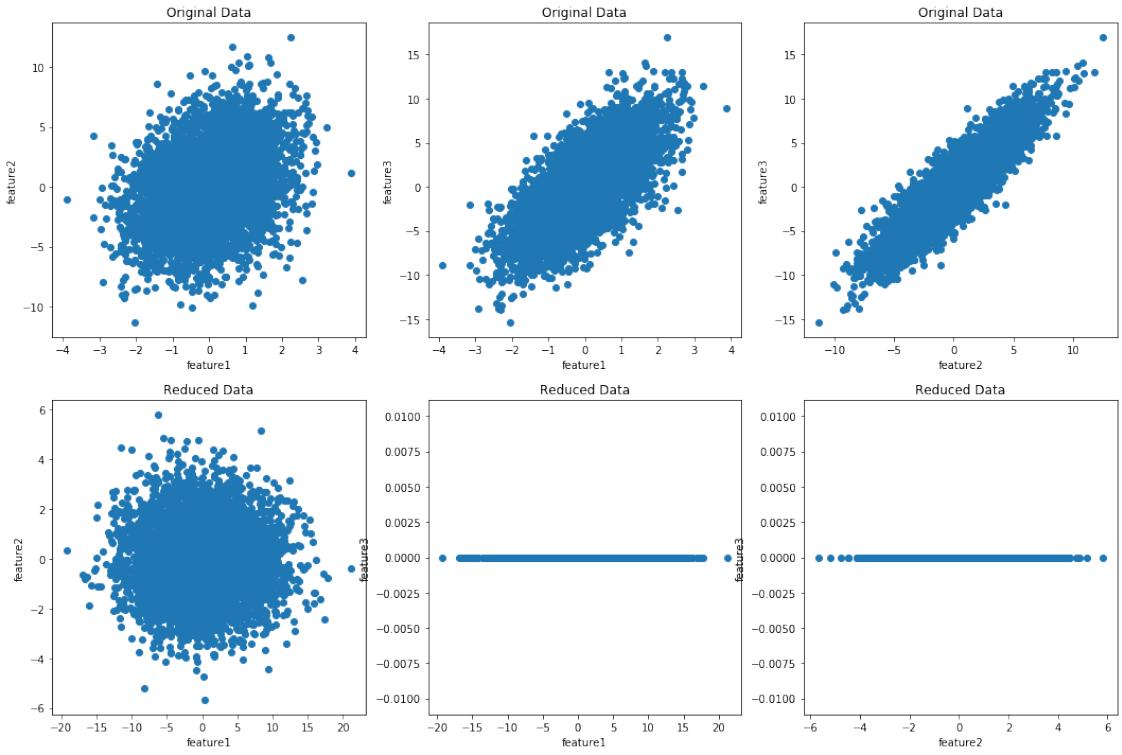
The vfalue of the second eigenvalue may seem very different but it can be because it is very small when compared to the others but they can both be considered zero.

Since the third component is a linear combination of the first 2 components the variability is explained by the first 2 compents so when reducing we need only 2 vectors as base (since the matrix is rank2)

```
In [4]: data_rotation=np.dot(vh, data.T)
        data_reduction=np.delete(data_rotation, 2, axis=0)

        covMatrix_rotated=np.cov(data_reduction)
        eigVal, eigVec=sp.linalg.eig(covMatrix_rotated)

        fig, subfig= plt.subplots(nrows=2, ncols=3, figsize=(18,12))
        subfig[0,0].scatter(data["x1"],data["x2"])
        subfig[0,0].set_xlabel("feature1")
        subfig[0,0].set_ylabel("feature2")
        subfig[0,0].set_title("Original Data")

        subfig[0,1].scatter(data["x1"],data["x3"])
```

```
subfig[0,1].set_xlabel("feature1")
subfig[0,1].set_ylabel("feature3")
subfig[0,1].set_title("Original Data")

subfig[0,2].scatter(data["x2"],data["x3"])
subfig[0,2].set_xlabel("feature2")
subfig[0,2].set_ylabel("feature3")
subfig[0,2].set_title("Original Data")

subfig[1,0].scatter(data_rotation[0,:],data_rotation[1,:])
subfig[1,0].set_xlabel("feature1")
subfig[1,0].set_ylabel("feature2")
subfig[1,0].set_title("Reduced Data")

subfig[1,1].scatter(data_rotation[0,:],data_rotation[2,:])
subfig[1,1].set_xlabel("feature1")
subfig[1,1].set_ylabel("feature3")
subfig[1,1].set_title("Reduced Data")

subfig[1,2].scatter(data_rotation[1,:],data_rotation[2,:])
subfig[1,2].set_xlabel("feature2")
subfig[1,2].set_ylabel("feature3")
subfig[1,2].set_title("Reduced Data")
```

Out[4]: Text(0.5, 1.0, 'Reduced Data')



2. **PCA on a nD dataset**

Start from the dataset you have genereted in the previous exercise and add uncorrelated random noise. Such noise should be represented by other 10 uncorrelated variables normal distributed, with standar deviation much smaller (say, a factor 50) than those used to generate the x_1 and x_2.

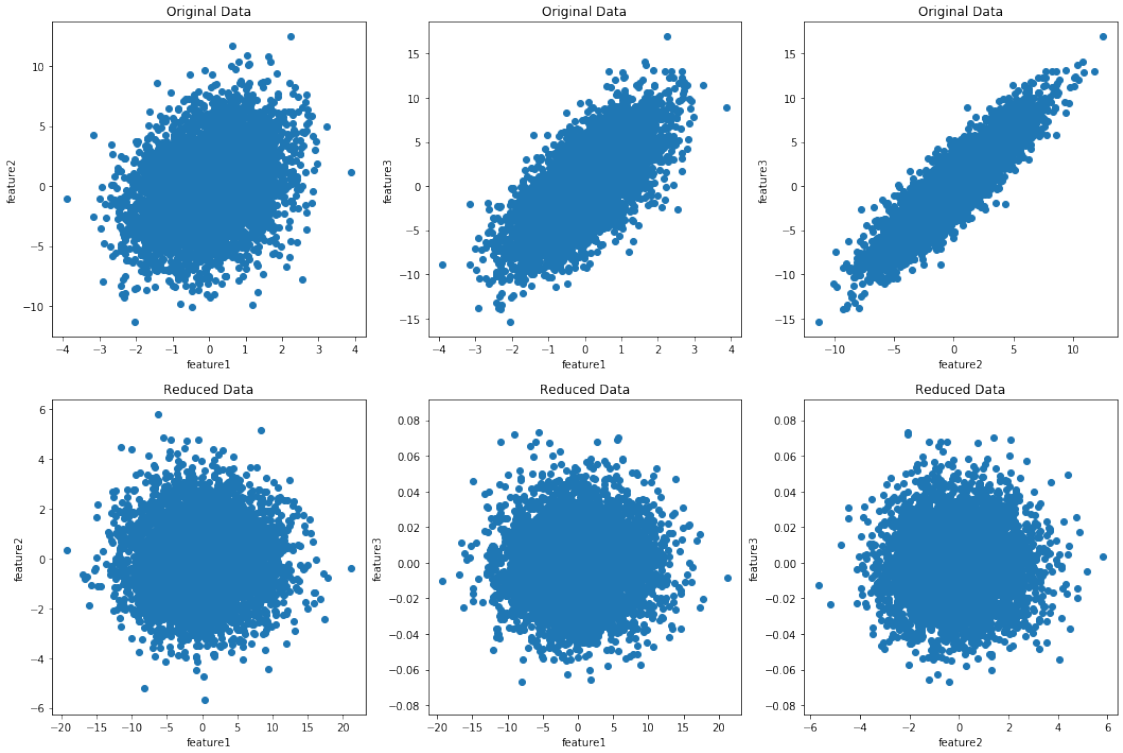
Repeat the PCA procedure and compare the results with what you obtained before

```
In [5]: new_data=data

        sigmas=np.random.uniform(0.01,0.02, 10)
        for s in sigmas:
            new_data=np.column_stack((new_data, np.random.normal(0, s, N)))

        print(new_data.shape)
        print(new_data)


        u,s,vh = sp.linalg.svd(new_data)
        eigValSvd = s**2/(N-1)
        data_rotation=np.dot(vh, new_data.T)
        data_reduction=np.delete(data_rotation, 2, axis=0)

        covMatrix_rotated=np.cov(data_reduction)
        eigVal, eigVec=sp.linalg.eig(covMatrix_rotated)

        fig, subfig= plt.subplots(nrows=2, ncols=3, figsize=(18,12))
        subfig[0,0].scatter(data["x1"],data["x2"])
        subfig[0,0].set_xlabel("feature1")
        subfig[0,0].set_ylabel("feature2")
        subfig[0,0].set_title("Original Data")

        subfig[0,1].scatter(data["x1"],data["x3"])
        subfig[0,1].set_xlabel("feature1")
        subfig[0,1].set_ylabel("feature3")
        subfig[0,1].set_title("Original Data")

        subfig[0,2].scatter(data["x2"],data["x3"])
        subfig[0,2].set_xlabel("feature2")
        subfig[0,2].set_ylabel("feature3")
        subfig[0,2].set_title("Original Data")

        subfig[1,0].scatter(data_rotation[0,:],data_rotation[1,:])
        subfig[1,0].set_xlabel("feature1")
        subfig[1,0].set_ylabel("feature2")
        subfig[1,0].set_title("Reduced Data")

        subfig[1,1].scatter(data_rotation[0,:],data_rotation[2,:])
```

```
        subfig[1,1].set_xlabel("feature1")
        subfig[1,1].set_ylabel("feature3")
        subfig[1,1].set_title("Reduced Data")

        subfig[1,2].scatter(data_rotation[1,:],data_rotation[2,:])
        subfig[1,2].set_xlabel("feature2")
        subfig[1,2].set_ylabel("feature3")
        subfig[1,2].set_title("Reduced Data")
```

```
(5000, 13)
[[-4.185e-01  2.430e+00  1.593e+00 ...  1.159e-02 -2.205e-03 -2.517e-02]
 [-4.194e-02 -2.930e+00 -3.014e+00 ... -3.095e-02 -2.098e-02  1.460e-02]
 [-3.458e-01  6.996e-01  7.978e-03 ... -4.168e-03  2.338e-02 -8.542e-03]
 ...
 [ 3.551e-01 -4.948e-01  2.153e-01 ... -1.259e-02  1.043e-03  1.777e-03]
 [-7.842e-01 -3.702e+00 -5.270e+00 ... -1.538e-02 -1.055e-02 -2.773e-03]
 [ 1.182e+00  1.222e+00  3.587e+00 ...  5.589e-03 -3.758e-03 -4.835e-03]]
```

```
Out[5]: Text(0.5, 1.0, 'Reduced Data')
```



The main difference from before is that the scatter plot of the 1v3 and 2v3 feature is not a line like before.

The previous plot showed that the reduction was almost perfect and that we were in a basis of the dataset in this case the result is similar.

We cannot say that the reduction is perfect but for sure the rotation obtained can be considered still a good base and the reduced dataset is still to be cosidered good since the scatter plots of the reduced data shows that the deviation from the zero value of the 2nd and 3rd plot are 2 orders of magnitude inforior to the 1st one so they are neglectable

3 . **Looking at an oscillating spring** (optional)

Imagine you have n cameras looking at a spring oscillating along the x axis. Each camera record the motion of the spring looking at it along a given direction defined by the pair (θ_i, ϕ_i), the angles in spherical coordinates.

Start from the simulation of the records (say $\mathcal{O}(1000)$) of the spring's motion along the x axis, assuming a little random noise affects the measurements along the y. Rotate such dataset to emulate the records of each camera.

Perform a Principal Component Analysis on the thus obtained dataset, aiming at finding the only one coordinate that really matters.

4. **PCA on the MAGIC dataset** (optional)

Perform a PCA on the magic04.data dataset

In [ ]: