Augusto Ruiz Vazquez

Carlos Alberto Garcia Moncada

Report of the "PCA Tutorial" from Lindsey I Smith

In this report we understand that the PCA is a technique which helps us find the directions of maximum variance in high-dimensional data and project it onto a smaller dimensional subspace while retaining most of the information.

PCA could divide in five steps that we explain in detail that we understand.

Step 1: Get some data

In this step we need to have a dataset that could be a n-dimension matrix.

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Wed May 04 00:19:52 2016
4
5  @author: Augusto Ruiz Vazquez
6  """
7
8  import numpy as np
9  import matplotlib.pyplot as plt
10 from mpl_toolkits.mplot3d import Axes3D
11
12
13 from sklearn import datasets
14
15 np.random.seed(5)
16
17 centers = [[1, 1], [-1, -1], [1, -1]]
18 iris = datasets.load_iris()
19 X = iris.data
20 y = iris.target
21 print X
```

This is how the IRIS dataset is loaded and finally part of the dataset is printed

```
[[ 5.1  3.5  1.4  0.2]
 [ 4.9  3.   1.4  0.2]
 [ 4.7  3.2  1.3  0.2]
 [ 4.6  3.1  1.5  0.2]
 [ 5.   3.6  1.4  0.2]
 [ 5.4  3.9  1.7  0.4]
 [ 4.6  3.4  1.4  0.3]
 [ 5.   3.4  1.5  0.2]
 [ 4.4  2.9  1.4  0.2]
 [ 4.9  3.1  1.5  0.1]
 [ 5.4  3.7  1.5  0.2]
 [ 4.8  3.4  1.6  0.2]
 [ 4.8  3.   1.4  0.1]
 [ 4.3  3.   1.1  0.1]
 [ 5.8  4.   1.2  0.2]
 [ 5.7  4.4  1.5  0.4]
 [ 5.4  3.9  1.3  0.4]]
```

Step 2: Subtract the mean

We need to get the values' mean of each dimension (plus all the n-values and divide by n) and substract the mean to the original values.

```
30 vectorpromedio = np.mean(X, axis=0)
31 print vectorpromedio
32
33 matrizdecov = (X - vectorpromedio).T.dot((X - vectorpromedio)) / (X.shape[0]-1)
34 print('Matriz de covarianza \n%s' %matrizdecov)
```

```
Promedio de x y y
[ 5.84333333  3.054       3.75866667  1.19866667]
```

Step 3: get the variance and covariance in a covariance matrix

If we have a 2-dimension matrix we could have a 2x2 covariance matrix.

To get the variance of each dimension of our dataset, we apply the following formula:

$$var(X) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})}{(n-1)}$$

To get the covariance we need to follow the next formula:

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

```
33 matrizdecov = (X - vectorpromedio).T.dot((X - vectorpromedio)) / (X.shape[0]-1)
34 print('Matriz de covarianza \n%s' %matrizdecov)
```

```
Matriz de covarianza
[[ 0.68569351 -0.03926846  1.27368233  0.5169038 ]
 [-0.03926846  0.18800403 -0.32171275 -0.11798121]
 [ 1.27368233 -0.32171275  3.11317942  1.29638747]
 [ 0.5169038  -0.11798121  1.29638747  0.58241432]]
```

Here we need the covariance matrix to find the eigenvectors and eigenvalues

Step 4: calculate the eigenvalues and eigenvectors of the covariance matrix

An eigenvector is a matrix that multiply the covariance matrix and from this matrix we can obtain the values.

An eigenvalue is the value that works like a escalar number and should be a value that make multiple the matrix product betwen the covariance matrix and eigenvector.

To get the eigenvalue we need to substract the identity matrix from covariance matrix find the determinant of the covariance matrix, and isolate from the resulting characteristic equation. The roots of the characteristic equation will be the eigenvalues.

To get the eigenvector we need to replace each eigenvalue of each lambda in the substract of the identity matrix to the covariance matrix. Then multiply by the column matrix and make it equal to zero. And finally we get the vectors equations.

```
36 valprop, vecprop = np.linalg.eig(matrizdecov)
37
38 print('vectorespropios \n%s' %vecprop)
39 print('\valorespropios de mayor a menor \n%s' %valprop)
```

```
vectorespropios
[[ 0.36158968 -0.65653988 -0.58099728  0.31725455]
 [-0.08226889 -0.72971237  0.59641809 -0.32409435]
 [ 0.85657211  0.1757674   0.07252408 -0.47971899]
 [ 0.35884393  0.07470647  0.54906091  0.75112056]]
♂alorespropios de mayor a menor
[ 4.22484077  0.24224357  0.07852391  0.02368303]
```

Step 5: Choosing components and forming a feature vector

The highest eigenvalue is the principle component of the data set and the eigenvector with the larges eigenvalue was the one that pointed down the middle of the data.

```
[[ 0.36158968 -0.08226889  0.85657211  0.35884393]
 [-0.65653988 -0.72971237  0.1757674   0.07470647]
 [-0.58099728  0.59641809  0.07252408  0.54906091]]
```
feature vector.

In general, once eigenvectors are found from the covariance matrix, the next step is to order them by eigenvalue, highest to lowest. This gives you the components in order of significance. Now, if you like, you can decide to ignore the components of lesser significance. You do lose some information, but if the eigenvalues are small, you don't lose much. If you leave out some components, the final data set will have less dimensions than the original.

```
41 nmc = vecprop[:,[0,1,2]]
42 print nmc
```

We delete the column 3 because is the less significant

Then you need to form a feature vector, which is just a fancy name for a matrix of vectors. This is constructed by taking the eigenvectors that you want to keep from the list of eigenvectors, and forming a matrix with these eigenvectors in the columns.

```
ueva matriz de covarianza sin la columna 3
[[ 0.36158968 -0.65653988 -0.58099728]
 [-0.08226889 -0.72971237  0.59641809]
 [ 0.85657211  0.1757674   0.07252408]
 [ 0.35884393  0.07470647  0.54906091]]
```

This the final step in PCA, and is also the easiest. Once we have chosen the components (eigenvectors) that we wish to keep in our data and formed a feature vector, we simply take the transpose of the vector and multiply it on the left of the original data set, transposed.

```
ueva matriz de covarianza transpuesta
[[ 0.36158968 -0.08226889  0.85657211  0.35884393]
 [-0.65653988 -0.72971237  0.1757674   0.07470647]
 [-0.58099728  0.59641809  0.07252408  0.54906091]]
```

$$FinalData = RowFeatureVector \times RowDataAdjust,$$

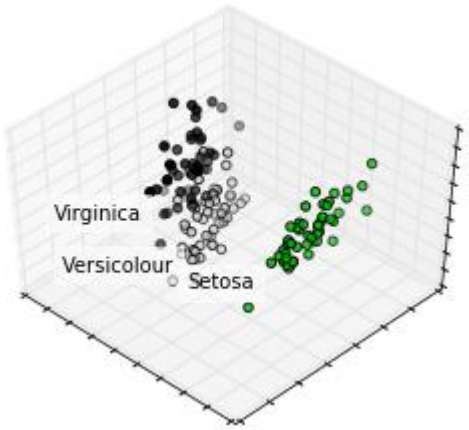
where *RowFetatureVector* is the matrix with the eigenvectors in the columns transposed so that the eigenvectors are now in the rows, with the most significant eigenvector at the top, and *RowDataAdjust* is the mean-adjusted data transposed, ie. the data items are in each column, with each row holding a separate dimension

```
48 nds = (ncov_matT).dot(dataT)
49 print nds
```

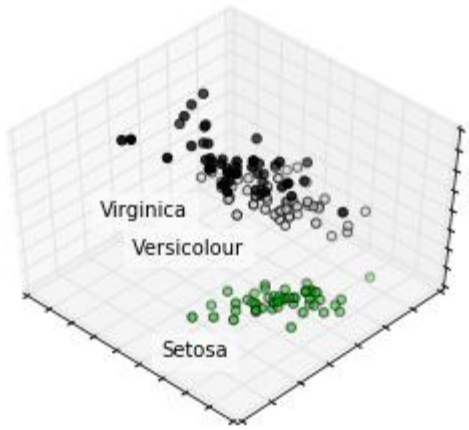
New Data Set

```
(150L, 3L)  array([[ 2.82713597, -5.64133105, -0.66427693],
                   [ 2.79595248, -5.14516688, -0.84628652],
```

With 150 values and 3 variables.



Plot after PCA



Plot before PCA

Referencias

Pedregosa, F. "Sklearn.decomposition.PCA." Scikit Learn. 2014. De Scikit. Consultation date: April 24th. 2016. Available online: http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

R. A. Fisher. *The use of multiple measurements in taxonomic problems*. Annals of Eugenics 7 (2): 179-188. doi:10.1111/j.1469-1809.1936.tb02137.x\, 1936.

Smith, Lindsay I. "A tutorial on principal components analysis." Cornell University, USA 51.52 (2002): 65.