PCA_example

June 9, 2017

```
In [18]: import numpy as np
         import matplotlib.pyplot as plt
         from sklearn import decomposition
         from sklearn import datasets
         from sklearn import preprocessing
```

1 Load Iris data

```
In [19]: iris = datasets.load_iris()
         print(iris.DESCR)
```

```
Iris Plants Database
====================

Notes
-----
Data Set Characteristics:
    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica
    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20  0.76     0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

This is a copy of UCI ML iris datasets.
http://archive.ics.uci.edu/ml/datasets/Iris

The famous Iris database, first used by Sir R.A Fisher

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

References
----------
   - Fisher,R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda,R.O., & Hart,P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...
```

```
In [20]: X = iris.data
         X.shape
```

```
Out[20]: (150, 4)
```

2 Optionally standardize the data (mean 0, std 1)

Note that PCA is sensitive to the scaling of the inputs!

In the Iris data the inputs have similar ranges so standardizing the scaling is not strictly necessary. We do it anyway here...

```
In [21]: Xs = preprocessing.StandardScaler(copy=True,
                                           with_mean=False,
                                           with_std=True).fit_transform(X)


         print('std before: %s' % X.std(axis=0))
         print('std after: %s' % Xs.std(axis=0))
```

```
std before: [ 0.82530129  0.43214658  1.75852918  0.76061262]
std after: [ 1.  1.  1.  1.]
```

3 Principle Component Analysis

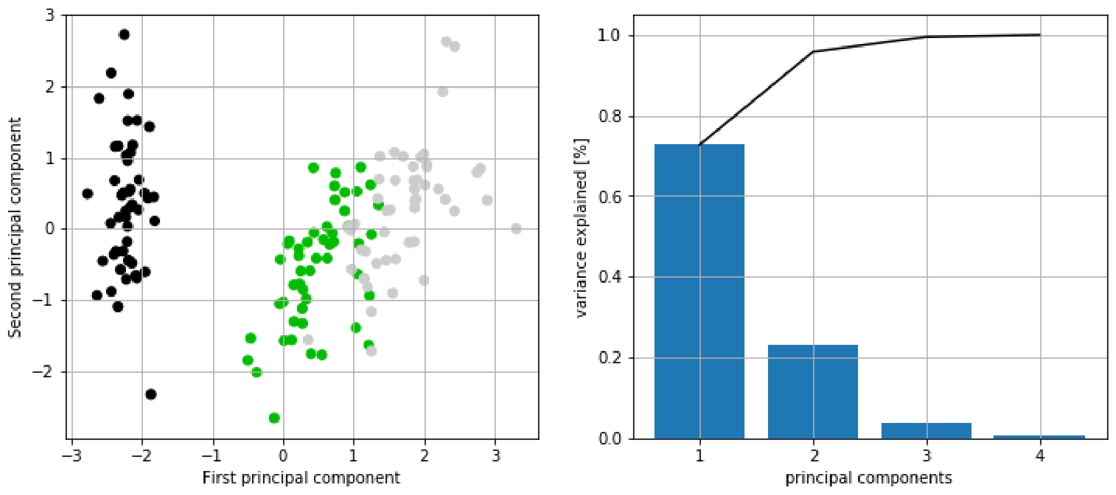
In the following we calculate a PCA with as many components as there are dimensions in the dataset (4). This allows investigating how much variance is explained by which principal components (see right plot). If we are only interested in dimensionality reduction, it is often more efficient to only calculate the desired number of components, e.g. 2.

```
In [22]: pca = decomposition.PCA(n_components=4)
         pca.fit(Xs)
         Z = pca.transform(Xs)
         Z.shape
```

```
Out[22]: (150, 4)
```

```
In [23]: fig,ax = plt.subplots(1,2, figsize=(12, 5))
         ax[0].scatter(Z[:, 0], Z[:, 1], c=iris.target, cmap=plt.cm.spectral)
         ax[0].set_xlabel('First principal component')
         ax[0].set_ylabel('Second principal component')
         ax[0].grid()

         var_explained = pca.explained_variance_ratio_
         ax[1].bar(np.arange(len(var_explained)),var_explained)
         ax[1].set_xticks(np.arange(len(var_explained)))
         ax[1].set_xticklabels(1+np.arange(len(var_explained)))
         ax[1].set_xlabel('principal components')
         ax[1].set_ylabel('variance explained [%]')
         ax[1].plot(np.cumsum(var_explained),'k-')
         ax[1].grid()
         plt.show()
```



Note that the first two principal components explain almost the complete variance of the four-dimensional dataset.