//

How to deal with Principal Component Analysis

---

maXbox Starter 58 - Data Science with PCA

"Errors should never pass silently.(Unless explicitly silenced.)"
- The Zen of Python

Principal component analysis (PCA) is often the first thing to try out if you want to cut down the number of features and do not know what feature extraction method to use.

PCA is limited as its a linear method, but chances are that it already goes far enough for your model to learn well enough.

Add to this the strong mathematical properties it offers and the speed at which it finds the transformed feature data space and is later able to transform between original and transformed features; we can almost guarantee that it also will become one of your frequently used machine learning tools.

This tutor will go straight to an overview to PCA.

The script 811_mXpcatest_dmath_datascience.pas (pcatest.pas) (located in the demo\console\curfit subdirectory) performs a principal component analysis on a set of 4 variables.

(Example taken from: P. DAGNELIE, Analyse statistique ↕a plusieurs variables, Presses Agronomiques de Gembloux, Belgique, 1982). The program prints:

☼ The mean vector and variance-covariance matrix of the original variables
☼ The correlation coefficients between the original variables
☼ The eigenvalues and eigenvectors of the correlation matrix
☼ The correlation coefficients between principal factors and the original variables
☼ The values of the principal factors for each point

Summarizing it, given the original feature space, PCA finds a linear projection of itself in a lower dimensional space that has the following two properties:

- The conserved variance is maximized.
- The final reconstruction error (when trying to go back from transformed features to the original ones) is minimized.

As PCA simply transforms the input data, it can be applied both to classification and regression problems. In this section, we will use a classification task to discuss the method.

The script can be found at:

http://www.softwareschule.ch/examples/811_mXpcatest_dmath_datascience.pas
..\examples\811_mXpcatest_dmath_datascience.pas

It may be seen that:

- High correlations exist between the original variables, which are therefore not independent

- According to the eigenvalues, the last two principal factors may be neglected since they represent less than 11 % of the total variance. So, the original variables depend mainly on the first two factors

- The first principal factor is negatively correlated with the second and fourth variables, and positively correlated with the third variable

• The second principal factor is positively correlated with the first variable

• The table of principal factors show that the highest scores are usually associated with the first two principal factors, in agreement with the previous results

```
Const
  N    = 11;  { Number of observations }
  Nvar = 4;   { Number of variables }

{ Data }
var X : array[1..N] of array[1..Nvar] of Float;
 =
(( 87.9, 19.6,    1  ,   1661),
 ( 89.9, 15.2,   90.1,    968),
 (153  , 19.7,   56.6,   1353),
 (132.1, 17  ,   91  ,   1293),
 ( 88.8, 18.3,   93.7,   1153),
 (220.9, 17.8,  106.9,   1286),
 (117.7, 17.8,   65.5,   1104),
 (109  , 18.3,   41.8,   1574),
 (156.1, 17.8,   57.4,   1222),
 (181.5, 16.8,  140.6,    902),
 (181.4, 17  ,   74.3,   1150));//}

var
  XX     : TMatrix;  { Data }
  M      : TVector;  { Mean vector }
  V      : TMatrix;  { Variance-covariance matrix }
  R      : TMatrix;  { Correlation matrix }
  S      : TVector;  { Standard deviations }
  Lambda : TVector;  { Eigenvalues of correlation matrix }
  C      : TMatrix;  { Eigenvectors of correlation matrix }
  Rc     : TMatrix;  { Correlation factors/variables }
  Z      : TMatrix;  { Scaled variables }
  F      : TMatrix;  { Principal factors }
  I, J   : Integer;  { Loop variables }
```

PCA involves a lot of linear algebra, which we do not want to go into. Nevertheless, the basic algorithm can be easily described as follows:

1. Center the data by subtracting the mean from it.
2. Calculate the covariance matrix.
3. Calculate the eigenvectors of the covariance matrix.

If we start with N features, then the algorithm will return a transformed feature space again with N dimensions (we gained nothing so far). The nice thing about this algorithm, however, is that the eigenvalues indicate how much of the variance is described by the corresponding eigenvector.

Lets assume we start with N = 2000 features and we know that our model does not work well with more than 15 features. Then, we simply pick the 15 eigenvectors with the highest eigenvalues.

>>>

Mean vector:

```

138.027272727273
17.7545454545455
74.4454545454545
1242.36363636364

Variance-covariance matrix

   1793.9074      -4.4733     726.8760   -2217.5917
     -4.4733       1.4879     -26.6170     197.4529
    726.8760     -26.6170    1224.4170   -6202.4802
  -2217.5917     197.4529   -6202.4802   48104.2314

Correlation matrix

      1.0000      -0.0866       0.4905      -0.2387
     -0.0866       1.0000      -0.6236       0.7380
      0.4905      -0.6236       1.0000      -0.8082
     -0.2387       0.7380      -0.8082       1.0000


Eigenvalues of correlation matrix:

2.59317860210044
0.978288683253008
0.285238674676994
0.143294039969554

Eigenvectors (columns) of correlation matrix:

      0.2908       0.8713       0.3322       0.2142
     -0.5062       0.4248      -0.7423       0.1107
      0.5773       0.1360      -0.4184      -0.6879
     -0.5709       0.2047       0.4043      -0.6846

Correlations between factors (columns) and variables (lines):

      0.4682       0.8618       0.1774       0.0811
     -0.8152       0.4201      -0.3965       0.0419
      0.9296       0.1345      -0.2235      -0.2604
     -0.9194       0.2024       0.2159      -0.2592

Principal factors:

     -3.4114      -0.2835       0.1338       0.0511
      1.7022      -2.0748       0.4841       0.0733
     -1.2870       1.0193      -0.6491       0.2578
      0.4137      -0.2731       0.3081      -0.5820
     -0.0141      -0.8313      -1.1131      -0.2990
      0.9719       1.8876       0.3148      -0.3509
      0.0542      -0.5662      -0.3352       0.5091
     -1.8275      -0.2247       0.4421      -0.4907
     -0.1230       0.3023       0.2804       0.4942
      2.6719       0.5015      -0.4967      -0.1048
      0.8489       0.5427       0.6309       0.4420


By the way the well known dmath.dll is not statically linked:
Put the file dmath.dll therefore in your PATH and you can change to newer versions.
By default, DMath is intended to be used as a shared library, such as a Windows DLL. Compilation scripts are supplied in the dll subdirectory to generate the shared library file, the interface file and (in the case of FPC/Lazarus) the object file.
```

It is sometimes more efficient to link statically with individual units rather than calling the shared library. This may be done by replacing the reference to unit dmath by references to the individual units.

```
  procedure VecMean(X           : TMatrix;
                    Lb, Ub, Nvar : Integer;
                    M            : TVector); external 'VecMean@dmath.dll';
{ Computes the mean vector M from matrix X }

  procedure MatVarCov(X           : TMatrix;
                      Lb, Ub, Nvar : Integer;
                      M            : TVector;
                      V            : TMatrix); external
'MatVarCov@dmath.dll';
 { Computes the variance-covariance matrix V from matrix X }

  procedure MatCorrel(V   : TMatrix;
                      Nvar : Integer;
                      R    : TMatrix); external 'MatCorrel@dmath.dll';
{ Computes the correlation matrix R from the var-cov matrix V }

  procedure VecSD(X           : TMatrix;
                  Lb, Ub, Nvar : Integer;
                  M, S         : TVector); external 'VecSD@dmath.dll';
{ Computes the vector of standard deviations S from matrix X }

  procedure ScaleVar(X           : TMatrix;
                     Lb, Ub, Nvar : Integer;
                     M, S         : TVector;
                     Z            : TMatrix); external 'ScaleVar@dmath.dll';
{ Scales a set of variables by subtracting means and dividing by SD's }

  procedure PCA(R      : TMatrix;
                Nvar   : Integer;
                Lambda : TVector;
                C, Rc  : TMatrix); external 'PCA@dmath.dll';
{ Performs a principal component analysis of the correlation matrix R }

  procedure PrinFac(Z           : TMatrix;
                    Lb, Ub, Nvar : Integer;
                    C, F         : TMatrix); external 'PrinFac@dmath.dll';
{ Computes principal factors }
```

Of course, its not always this and that simple. Often, we dont know what number of dimensions is advisable in upfront. In that case, we leave n_components or Nvar parameter unspecified when initializing PCA to let it calculate the full transformation. After fitting the data, explained_variance_ratio_ contains an array of ratios in decreasing order: The first value is the ratio of the basis vector describing the direction of the highest variance, the second value is the ratio of the direction of the second highest variance, and so on.

Being a linear method, PCA has, of course, its limitations when we are faced with strange data that has non-linear relationships. We wont go into much more details here, but its sufficient to say that there are extensions of PCA.

Ref:

Building Machine Learning Systems with Python
Second Edition March 2015

```
    DMath
    Math library for Delphi, FreePascal and Lazarus May 14, 2011

    http://fann.sourceforge.net

Doc:
    Neural Networks Made Simple: Steffen Nissen
    http://fann.sourceforge.net/fann_en.pdf
    http://www.softwareschule.ch/examples/datascience.txt
    https://maxbox4.wordpress.com
    https://www.tensorflow.org/


https://sourceforge.net/projects/maxbox/files/Examples/13_General/811_mXpcatest_dmath_datascienpas/download
https://sourceforge.net/projects/maxbox/files/Examples/13_General/809_FANN_XorSample_traindata.pas/download


Plots displaying the explained variance over the number of components is called a Scree plot. A nice example of combining a Screeplot with a grid search to find the best setting for the classification problem can be found at

http://scikit-learn.sourceforge.net/stable/auto_examples/plot_digits_pipe.html.

Although, PCA tries to use optimization for retained variance, multidimensional scaling (MDS) tries to retain the relative distances as much as possible when reducing the dimensions. This is useful when we have a high-dimensional dataset and want to get a visual impression.
>>> Building Machine Learning Systems with Python
```