Principal Component Analysis (PCA) & NIPALS algorithm

Henning Risvik

May 10, 2007

1 Contents

2 PCA

A short description of the technique is: simplifying a data set by reducing multidimensional data sets to lower dimensions for analysis. It can be used for exploratory data analysis and to make predictive models.

In this description of the PCA we will be looking at an example data set for some explanations, the Cheese data set. In the data set 12 assessors have

tested 14 samples 2 times, giving a score between 1-9 on 17 variables. This leaves us with a data matrix of (12*14*2)=336 objects, X with n=336 and p=17 (336 x 17). Each sample represents a cheese sample that was stored under certain conditions. The variables are various flavours and odours of a cheese.

The first principal component will stretch out in the direction where there is most variance, of variable space, and form the first PC axis. The next PC is orthogonal to this axis, and has the direction where there is second most spread of variance orthogonally to the first axis, for the next where there is third most spread, and so on. With our data set we have a 17-dimensional variable space, so to visualize how the first PCs form we must simulate a 2d or 3d variable space of the data set.

2.1 Preprocessing

The preprocessing part can make the difference between a useful model and no model at all. One term for preprocessing is called autoscaling, which is the combination of mean centering and standardization. Standardization is one type of scaling where each value is scaled by 1/STD. In our data set we know that all the scores lies between 1 and 9, so scaling will not be necessary. In other cases attributes/variables can have scores where there is natural to have other values than just 1-9 (i.e. any other type: age, length, volume, time, newton, etc.), in these cases we probably need to do a standardization.

We mean center each attribute vector X_i by this formula:

$$X_{ij} = X_{ij} - \overline{X_i} \tag{1}$$

2.2 PC Model

The method involves decomposing a data matrix X into a structure part and a noise part. The PC model is the matrix product TP^T (the Structure):

$$X = TP^T + E = Structure + Noise \tag{2}$$

In figure 1 (on next page) we can see how scores and loadings form the TP (Structure) part of the equation. This summation will get us back to X, but what we are interested in is each of the scores and loadings which are collected in TP^T. We assume that X can be split into the sum of the matrix

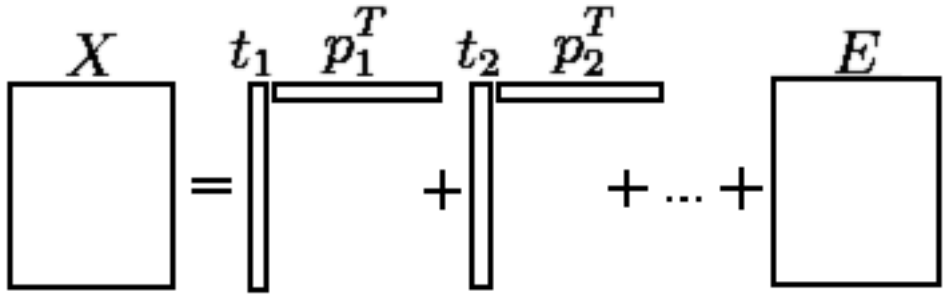

Figure 1: Collection of scores and loadings

product TP^T and the residual matrix E. And wish to use T and P instead of X.

About the number of PCs, there is a maximum, and that is the least of n or p (min(n,p)) of the data matrix. We have 336 objects and 17 variables in our cheese data set (n=336, p=17), so that means there can be a maximum of 17 PCs. If we use all 17 we will have a full model, but then we have only replaced the variable space with a new coordinate system, the PC-space. And there has been no separation, so E will be equal to 0. The separation of the structure part and the noise part is one the strengths of a PCA. That is why we are typically most interested in the first few PCs (usually PC_1 and PC_2), where there should be most explained variance, and if not min(n,p)=2 should yield separation of structure and noise.

We want the structure part to have a high percentage of explained variance and the noise to be collected in E. For this to happen, as best as possible, we need to find an optimum number of PCs. This can be a non-trivial task. Because we don't want to be removing information about our data set, but still there is great advantage in the separation of the structure and the noise. To search for the optimum number of PCs, we use the E-matrix.

2.3 Scores: T

The Scores, structure part of the PCA. Summary of the original variables in X that describe how the different rows in X (observations) relate to each other. In the T-matrix column 1 (t_1) is the scores of the first PC. The second column contains the scores of the second PC, and so on. The term scores can be ambiguous, what we mean are the elements in the T-matrix. Without

any further specification.

The scores give us one of the most powerful tools that Principal-Component-based methods can offer us. The Score plot. It is simply scores plotted against each other. The most usual score plot has scores of PC1 plotted against the scores of PC2. It could also be called a map of samples.

2.4 Loadings: P

The Loadings, structure part of the PCA. The weights (influence) of the variables in X on the scores T. Of the loadings we can see which variables that are responsible for patterns found in scores, T, using the Loadings plot. This plot is simply the loadings of a PC plotted against the loadings of another PC. We can see how the scores and loadings relate, and that is very important about this plot. The loadings plot could be called a map of variables.

2.5 Residuals: E

The Residuals (E-matrix), is the noise part of the PCA, a n x p large Matrix. E is not part of the model. It will be the part of X which is not explained by the model TP^T. That is why it should generally not be "large", that would mean we have removed much information. Which, furthermore, will give us a poor PC model. Hopefully, it should be no more than noise of some sort (i.e. differences, on a score on same sample and variable, between assessors).

The evaluation of E is always relative to the average object of X (in variable space), which is the origo in the PC-space. This average object can also be thought of as the first approximation to X, or the 0th PC (PC_0).

As we calculate more PCs, the residual variance will change. We start off with a E_0 with a residual variance of 100% and an explained variance of 0% (in the model). Actually, at this stage we don't have the TP^T term, yet. TP^T equals to 0. The other way around, when we have a full model, E equals to 0 and we have 100% explained variance (but also with all the noise). Throughout the calculation of PCs:

$$ExplainedVariance + ResidualVariance = 100\% \tag{3}$$

This is what we use to find the optimum number of PCs. We can plot the change in residual variance, as the number of PCs increase. And, by human interpretation, select the best number.

The size of E is expressed in terms of squared variance. This is by proper statistical fashion, because E is an error term. So, there will be summation of squares of the elements in the E-matrix. This can be done either along the rows of the E-matrix, which will give us object residuals, or down along columns, which will give us variable residuals. Let us take a look at object residuals. For each row we will get a sum e_i^2 for object i. We will be looking at the total residual variance, for all the objects simultaneously. Though we could have been looking at the residual variance for each object or variable, but this we would only do in special cases (i.e. with outliers. Note: The PCA module must be modified to return such results). We get a measure of the distance between the objects in variable space and the representation of the objects in PC-space by this formula:

$$e_{tot}^2 = \sum_{i=1}^{n} e_i^2 \tag{4}$$

The smaller these distances are, the closer the PC model representation of the objects is to the original objects. This sum will be decreasing as the number of PCs increase ($e_{tot0}^2 > e_{tot1}^2$).

3 Correlation Loadings

Correlation Loadings can be calculated after the PCA is run. It is the correlations between the values of a column in X (variable space) and the scores of a PC (correlations of X_i and t_i).

4 NIPALS algorithm

The NIPALS ("Nonlinear Iterative Partial Least Squares") algorithm is one of the many methods that exist for finding the eigenvectors (another example is SVD). It was originally made for PCA, but it has been used in other methods as well.
This is an overview of the algorithm:

X is a mean centered data matrix
$E_{(0)} = X$ The E-matrix for the zero-th PC (PC_0) is mean centered X
t vector is set to a column in X

t will be the scores for PC_i
p will be the loadings for PC_i
$threshold = 0.00001$ Just a low value, to do the convergence check

Iterations (i=1 to number-of-PCs):

1. Project X onto t to find the corresponding loading p
   $p = (E_{(i-1)}^T t)/(t^T t)$
2. Normalise loading vector p to length 1
   $p = p * (p^T p)^{-0.5}$
3. Project X onto p to find corresponding score vector t
   $t = (E_{(i-1)} p)/(p^T p)$
4. Check for convergence. If difference between eigenvalues $\tau_{new} = (t^T t)$ and τ_{old} (from last iteration) is larger than $threshold * \tau_{new}$ return to step 1.
5. Remove the estimated PC component from $E_{(i-1)}$
   $E_{(i)} = E_{(i-1)} - (tp^T)$

References

[1] K. Esbensen: *Multivariate Analysis - in practice*, CAMO, 3rd edt. (1998)

[2] H. Lohninger: *Teach/Me Data Analysis*, Springer-Verlag, (1999)
`http://www.vias.org/tmdatanaleng/dd_nipals_algo.html`