Cross Validated

How to perform PCA for data of very high dimensionality?

Asked 10 years, 6 months ago Active 1 year, 6 months ago Viewed 11k times

14

10

To perform principal component analysis (PCA), you have to subtract the means of each column from the data, compute the correlation coefficient matrix and then find the eigenvectors and eigenvalues. Well, rather, this is what I did to implement it in Python, except it only works with small matrices because the method to find the correlation coefficient matrix (corrcoef) doesn't let me use an array with high dimensionality. Since I have to use it for images, my current implementation doesn't really help me.

I've read that it's possible to just take your data matrix D and compute $DD^\top/n$ instead of $D^\top D/n$, but that doesn't work for me. Well, I'm not exactly sure I understand what it means, besides the fact that it's supposed to be a $n \times n$ matrix instead of $p \times p$ (in my case $p \gg n$). I read up about those in the eigenfaces tutorials but none of them seemed to explain it in such a way I could really get it.

In short, is there a simple algorithmic description of this method so that I can follow it?

pca python

Share Cite Edit Follow Flag

edited Feb 7 '15 at 22:09
amoeba
89.4k 28 262 311

asked Feb 11 '11 at 22:56
user3164

1 What you read is correct. The matrix $DD^\top$ is called the Gram matrix. Its eigenvectors are (scaled) principal components. Its eigenvalues are exactly identical, up to the factor $1/n$, to the eigenvalues of the covariance matrix $D^\top D/n$. – amoeba Feb 7 '15 at 22:18

4 Answers

Active Oldest Votes

11

The easiest way to do standard PCA is to center the columns of your data matrix (assuming the columns correspond to different variables) by subtracting the column means, and then perform an SVD. The left singular vectors, multiplied by the corresponding singular value, correspond to the (estimated) principal components. The right singular vectors correspond to the (estimated) principal component directions — these are the same as the eigenvectors given by PCA. The singular values correspond to the standard deviations of the principal components (multiplied by a factor of root n, where n is the number of rows in your data matrix) — the same as the square root of the eigenvalues given by PCA.

If you want to do PCA on the correlation matrix, you will need to standardize the columns of

your data matrix before applying the SVD. This amounts to subtracting the means (centering) and then dividing by the standard deviations (scaling).

This will be the most efficient approach if you want the full PCA. You can verify with some algebra that this gives you the same answer as doing the spectral decomposition of the sample covariance matrix.

There are also efficient methods for computing a partial SVD, when you only need a few of the PCs. Some of these are variants of the power iteration. The Lanczos algorithm is one example that is also related to partial least squares. If your matrix is huge, you may be better off with an approximate method. There are also statistical reasons for regularizing PCA when this is the case.

Share Cite Edit Follow Flag

edited Feb 7 '15 at 22:09
amoeba
89.4k 28 262 311

answered Feb 12 '11 at 1:19
vqv
2,394 1 22 16

1 Correct me if I am wrong, but I think Lanczos algorithm performs eigendecomposition and not SVD. – amoeba Feb 7 '15 at 22:11

2 An interested reader can look here for the further details on performing PCA via SVD: Relationship between SVD and PCA. How to use SVD to perform PCA? – amoeba Feb 7 '15 at 22:14

11

What you're doing right now is close, but you need to make sure you multiply the eigenvectors of `(data . data.T) / lines` on the left by `data.T`, in order to get the eigenvectors of `(data.T . data) / lines`. This is sometimes called the "transpose trick".

Here are some more details. Suppose you have a matrix A that you want to perform PCA on; for simplicity, suppose that the columns of A have already been normalized to have zero mean, so that we just need to compute the eigenvectors of the covariance matrix $A^T A$.

Now if A is an $m \times n$ matrix, with $n >> m$, then $A^T A$ is a very large $n \times n$ matrix. So instead of computing the eigenvectors of $A^T A$, we might like to compute the eigenvectors of the much smaller $m \times m$ matrix AA^T -- assuming we can figure out a relationship between the two. So how are the eigenvectors of $A^T A$ related to the eigenvectors of AA^T?

Let v be an eigenvector of AA^T with eigenvalue λ. Then

- $AA^T v = \lambda v$
- $A^T(AA^T v) = A^T(\lambda v)$
- $(A^T A)(A^T v) = \lambda(A^T v)$

In other words, if v is an eigenvector of AA^T, then $A^T v$ is an eigenvector of $A^T A$, with the same eigenvalue. So when performing a PCA on A, instead of directly finding the eigenvectors of $A^T A$ (which may be very expensive), it's easier to find the eigenvectors v of AA^T and then multiply these on the left by A^T to get the eigenvectors $A^T v$ of $A^T A$.

Share Cite Edit Follow Flag

edited Jun 7 '11 at 21:15

answered Feb 13 '11 at 7:21

raegtin
8,860 11 46 52

1 This sounds like the "kernel trick" applied to PCA. en.wikipedia.org/wiki/Kernel_PCA It's a very good way of handling certain large matrices. – Gilead Feb 14 '11 at 23:11

1 +1. Perhaps one should add that $AA^\top$ is called the Gram matrix. – amoeba Feb 7 '15 at 22:16

9

It sounds like what you want is the NIPALS algorithm for performing PCA. It's a very popular algorithm among statisticians. It has many advantages:

- Computationally less expensive than SVD or eigenvalue decomposition methods if only the first few components are required.
- Has more modest storage requirements in general because the covariance matrix is never formed. This is a very important property for very large datasets.
- Can handle missing data in the dataset (though that's not an issue in your problem, since you're dealing with images).

Description
http://en.wikipedia.org/wiki/Non-linear_iterative_partial_least_squares

Algorithm
Here's a simple and excellent description of the algorithm (in section 1.2)
http://stats4.eng.mcmaster.ca/w/mediafiles/mediawiki/f/f7/Section-Extra-Class-1.pdf

Remember to mean-center-scale first before doing PCA as it is scale-sensitive.

Share Cite Edit Follow Flag

answered Feb 12 '11 at 3:43

Gilead
361 1 2 7

5

To add on Gilead's answer, they are computationally less expensive algorithms for truncated PCAs. NIPALS is indeed very popular, but I have had a lot of success with approximate methods that perform a succession of fits on partial data (what is often called PCA by random projection). This was discussed in a metaoptimize thread.

As you mention Python, let me point out that the algorithm is implemented in the scikit-learn: the PCA class. In particular, it is used in an example demonstrating eigenfaces.

Share Cite Edit Follow Flag

edited Feb 5 '20 at 16:42

community wiki
4 revs, 3 users 77%
Gael Varoquaux