

Cross Validated is a question and answer site for people interested in statistics, machine learning, data analysis, data mining, and data visualization. It only takes a minute to sign up.

Anybody can ask a question



Anybody can answer

Sign up to join this community

The best answers are voted up and rise to the top



How to reverse PCA and reconstruct original variables from several principal components?

Asked 7 years, 6 months ago Modified 6 years, 7 months ago Viewed 200k times



172

Principal component analysis (PCA) can be used for dimensionality reduction. After such dimensionality reduction is performed, how can one approximately reconstruct the original variables/features from a small number of principal components?



Alternatively, how can one remove or discard several principal components from the data?

In other words, how to reverse PCA?



Given that PCA is closely related to singular value decomposition (SVD), the same question can be asked as follows: how to reverse SVD?

pca

dimensionality-reduction

svd

Share Cite

Improve this question Follow

edited Aug 12, 2016 at 11:57

asked Aug 9, 2016 at 23:52



amoeba

104k

35

311

338

11 I am posting this Q&A thread, because I am tired of seeing dozens of questions asking this very thing and not being able to close them as duplicates because we do not have a canonical thread on this topic. There are several similar threads with decent answers but all seem to have serious

limitations, like e.g. focusing exclusively on R. – amoeba Aug 9, 2016 at 23:54

- 4 I appreciate the effort -- I think there is a dire need to collect together information about PCA, what it does, what it doesn't do, into one or several high-quality threads. I'm glad that you've taken it upon yourself to do this! – Sycorax Aug 10, 2016 at 18:01
- 3 I am not convinced that this canonical answer "cleanup" serves its purpose. What we have here is an excellent, *generic* question and answer, but each of the questions had some subtleties to it about PCA in practise which are lost here. Essentially you have taken all the questions, done PCA on them, and discarded the lower principal components, where sometimes, rich and important detail is hidden. Moreover, you have reverted to textbook Linear Algebra notation which is precisely what makes PCA opaque to many people, instead of using the lingua franca of casual statisticians, which is R. – Thomas Browne Aug 12, 2016 at 16:32
- 2 @Thomas Thanks. I think we have a disagreement, happy to discuss it [in chat](#) or in Meta. Very briefly: (1) It might indeed be better to answer each question individually, but the harsh reality is that it does not happen. Many questions just stay unanswered, as yours probably would have. (2) The community here strongly prefers generic answers useful for many people; you can look at what sort of answers get upvoted the most. (3) Agree about maths, but that's why I did give R code here! (4) Disagree about lingua franca; personally, I don't know R. – amoeba Aug 12, 2016 at 21:42

@ThomasBrowne Well, you just click on the link I gave above:

chat.stackexchange.com/rooms/18/ten-fold -- and it takes you to chat. By now the conversation about your comment went a bit behind, so here is a link to the relevant chat history:

chat.stackexchange.com/transcript/message/31659626#31659626 – amoeba Aug 14, 2016 at 21:57

1 Answer

Sorted by: Highest score (default)



237



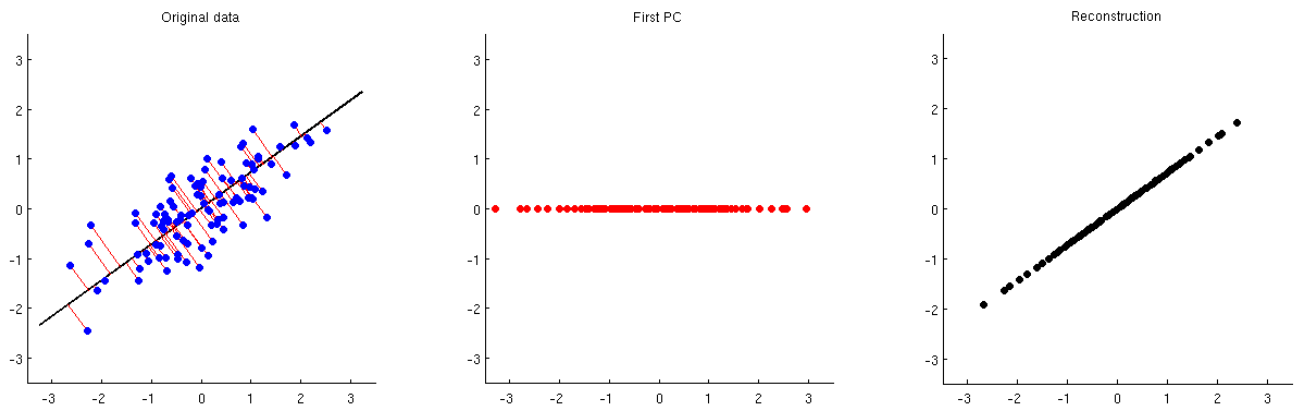
+100



PCA computes eigenvectors of the covariance matrix ("principal axes") and sorts them by their eigenvalues (amount of explained variance). The centered data can then be projected onto these principal axes to yield principal components ("scores"). For the purposes of dimensionality reduction, one can keep only a subset of principal components and discard the rest. (See [here for a layman's introduction to PCA.](#))

Let \mathbf{X}_{raw} be the $n \times p$ data matrix with n rows (data points) and p columns (variables, or features). After subtracting the mean vector $\boldsymbol{\mu}$ from each row, we get the *centered* data matrix \mathbf{X} . Let \mathbf{V} be the $p \times k$ matrix of some k eigenvectors that we want to use; these would most often be the k eigenvectors with the largest eigenvalues. Then the $n \times k$ matrix of PCA projections ("scores") will be simply given by $\mathbf{Z} = \mathbf{XV}$.

This is illustrated on the figure below: the first subplot shows some centered data (the same data that I use in [my animations](#) in the linked thread) and its projections on the first principal axis. The second subplot shows only the values of this projection; the dimensionality has been reduced from two to one:



In order to be able to reconstruct the original two variables from this one principal component, we can map it back to p dimensions with \mathbf{V}^T . Indeed, the values of each PC should be placed on the same vector as was used for projection; compare subplots 1 and 3. The result is then given by $\hat{\mathbf{X}} = \mathbf{Z}\mathbf{V}^T = \mathbf{X}\mathbf{V}\mathbf{V}^T$. I am displaying it on the third subplot above. To get the final reconstruction $\hat{\mathbf{X}}_{\text{raw}}$, we need to add the mean vector $\boldsymbol{\mu}$ to that:

$$\text{PCA reconstruction} = \text{PC scores} \cdot \text{Eigenvectors}^T + \text{Mean}$$

Note that one can go directly from the first subplot to the third one by multiplying \mathbf{X} with the $\mathbf{V}\mathbf{V}^T$ matrix; it is called a *projection* matrix. If all p eigenvectors are used, then $\mathbf{V}\mathbf{V}^T$ is the identity matrix (no dimensionality reduction is performed, hence "reconstruction" is perfect). If only a subset of eigenvectors is used, it is not identity.

This works for an arbitrary point \mathbf{z} in the PC space; it can be mapped to the original space via $\hat{\mathbf{x}} = \mathbf{z}\mathbf{V}^T$.

Discarding (removing) leading PCs

Sometimes one wants to discard (to remove) one or few of the leading PCs and to keep the rest, instead of keeping the leading PCs and discarding the rest (as above). In this case all the formulas stay *exactly the same*, but \mathbf{V} should consist of all principal axes *except* for the ones one wants to discard. In other words, \mathbf{V} should always include all PCs that one wants to keep.

Caveat about PCA on correlation

When PCA is done on correlation matrix (and not on covariance matrix), the raw data \mathbf{X}_{raw} is not only centered by subtracting $\boldsymbol{\mu}$ but also scaled by dividing each column by its standard deviation σ_i . In this case, to reconstruct the original data, one needs to back-scale the columns of $\hat{\mathbf{X}}$ with σ_i and only then to add back the mean vector $\boldsymbol{\mu}$.

Image processing example

This topic often comes up in the context of image processing. Consider [Lenna](#) -- one of the standard images in image processing literature (follow the links to find where it comes from). Below on the left, I display the grayscale variant of this 512×512 image (file [available here](#)).



We can treat this grayscale image as a 512×512 data matrix \mathbf{X}_{raw} . I perform PCA on it and compute $\hat{\mathbf{X}}_{\text{raw}}$ using the first 50 principal components. The result is displayed on the right.

Reverting SVD

PCA is very closely related to singular value decomposition (SVD), see [Relationship between SVD and PCA. How to use SVD to perform PCA?](#) for more details. If a $n \times p$ matrix \mathbf{X} is SVD-ed as $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ and one selects a k -dimensional vector \mathbf{z} that represents the point in the "reduced" U -space of k dimensions, then to map it back to p dimensions one needs to multiply it with $\mathbf{S}_{1:k,1:k} \mathbf{V}_{:,1:k}^T$.

Examples in R, Matlab, Python, and Stata

I will conduct PCA on the [Fisher Iris data](#) and then reconstruct it using the first two principal components. I am doing PCA on the covariance matrix, not on the correlation matrix, i.e. I am not scaling the variables here. But I still have to add the mean back. Some packages, like Stata, take care of that through the standard syntax. Thanks to @Stask and @Kodiologist for their help with the code.

We will check the reconstruction of the first datapoint, which is:

5.1 3.5 1.4 0.2

Matlab

```
load fisheriris
X = meas;
mu = mean(X);

[eigenvectors, scores] = pca(X);

nComp = 2;
Xhat = scores(:,1:nComp) * eigenvectors(:,1:nComp)';
Xhat = bsxfun(@plus, Xhat, mu);

Xhat(1,:)
```

Output:

5.083 3.5174 1.4032 0.21353

R

```
X = iris[,1:4]
mu = colMeans(X)

Xpca = prcomp(X)

nComp = 2
Xhat = Xpca$x[,1:nComp] %*% t(Xpca$rotation[,1:nComp])
Xhat = scale(Xhat, center = -mu, scale = FALSE)

Xhat[1,]
```

Output:

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
5.0830390	3.5174139	1.4032137	0.2135317

For worked out R example of PCA reconstruction of images see also [this answer](#).

Python

```
import numpy as np
import sklearn.datasets, sklearn.decomposition

X = sklearn.datasets.load_iris().data
mu = np.mean(X, axis=0)
```

```
pca = sklearn.decomposition.PCA()
pca.fit(X)

nComp = 2
Xhat = np.dot(pca.transform(X)[:,:nComp], pca.components_[:nComp,:])
Xhat += mu

print(Xhat[0,])
```

Output:

```
[ 5.08718247  3.51315614  1.4020428  0.21105556]
```

Note that this differs slightly from the results in other languages. That is because Python's version of the Iris dataset contains mistakes.

Stata

```
webuse iris, clear
pca sep* pet*, components(2) covariance
predict _seplen _sepwid _petlen _petwid, fit
list in 1
```

iris	seplen	sepwid	petlen	petwid	_seplen	_sepwid	_petlen
_petwid							
setosa	5.1	3.5	1.4	0.2	5.083039	3.517414	1.403214
.2135317							

Share Cite Improve this answer

Follow

edited Apr 13, 2017 at 12:44



Community Bot

1

answered Aug 9, 2016 at 23:52



amoeba

104k


35

311

338

-
- 2 In MATLAB you can retrieve mu from the standard PCA outputs, and also can supply the number of components in inputs. – [Aksakal](#) Aug 11, 2016 at 17:51
-
- 3 @Aksakal I tried to make all three code excerpts as similar (and as clear) as possible; in particular, I wanted to compute μ by hand before calling `pca()`, and also to run PCA with all components and to use only `nComp` components when performing dot product between scores and eigenvectors. I have now modified the Python code to follow the same pattern. – [amoeba](#) Aug 11, 2016 at 20:17
-
- 3 I would remove everything from the answer that is not related to the direct answer to the question, such as that cute girl's image and image processing. If someone's not interested in images it makes consumption difficult. Remember that whoever is asking the question is already deeply confused. – [Aksakal](#) Aug 12, 2016 at 14:15
-
- 12 Lenna is about as standard a data set as iris is. – [StasK](#) Aug 12, 2016 at 14:35
-
- 3 @amoeba I was talking about size, bit-depth, even black pixels in the border. I have not a

definitive version <http://www.ece.rice.edu/~wakin/images/>: "There seem to be many versions of the Lena (aka "Lenna") test image available. This problem was noted by Shapiro in his 1993 zerotree paper, and it remains surprisingly true today" – [Laurent Duval](#) [Aug 12, 2016 at 21:40](#)

 **Highly active question.** Earn 10 reputation (not counting the [association bonus](#)) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.