How to use scikit-learn PCA for features reduction and know which features are discarded

I am trying to run a PCA on a matrix of dimensions m x n where m is the number of features and n the number of samples.

Suppose I want to preserve the `nf` features with the maximum variance. With `scikit-learn` I am able to do it in this way:

```
from sklearn.decomposition import PCA

nf = 100
pca = PCA(n_components=nf)
# X is the matrix transposed (n samples on the rows, m features on the columns)
pca.fit(X)

X_new = pca.transform(X)
```

Now, I get a new matrix `X_new` that has a shape of n x nf. Is it possible to know which features have been discarded or the retained ones?

Thanks

python machine-learning scikit-learn pca feature-selection

edited Apr 25 '14 at 14:47

asked Apr 25 '14 at 13:30
gc5
2,948 13 53 91

Thanks Tom, I was thinking PCA could be used for feature selection, but (correct if I am wrong) it is only used to rescale the data on the principal components. As you read it I think I'll close the question. – gc5 Apr 25 '14 at 14:00

1 Your output matrix should be of shape `(n, nf)` , not `(nf, n)` . – eickenberg Apr 25 '14 at 14:31

3 Answers

The features that your `PCA` object has determined during fitting are in `pca.components_` . The vector space orthogonal to the one spanned by `pca.components_` is discarded.

Please note that PCA does not "discard" or "retain" any of your pre-defined features (encoded by the columns you specify). It mixes all of them (by weighted sums) to find orthogonal directions of maximum variance.

If this is not the behaviour you are looking for, then PCA dimensionality reduction is not the way to go. For some simple general feature selection methods, you can take a look at `sklearn.feature_selection`

edited Apr 25 '14 at 14:53 answered Apr 25 '14 at 14:34

eickenberg
9,129 1 18 33

1 I finally understood what PCA does (hopefully). Is there any preferred correlation function to compute if a feature is correlated with a principal component? In this way I think to be able to find the most representative dimensions in my dataset.. (correct me if I am wrong) .. may I use just Pearson or cosine similarity? – gc5 Apr 25 '14 at 18:59

5 Thumbs up for understanding PCA ;) -- In order to be able to answer your question, we need to be very clear about what is meant by *feature* and *dimension*. There is potential for confusion with both. The features you specified are the columns of your matrix. In order to see whether PCA component 0 makes use of feature `i` , you can compare `pca.components_[0, i]` to the rest of `pca.components_[0]` . So if I understand your question correctly, then the answer is to look at a given PC and see which of your features have the strongest weights. – eickenberg Apr 25 '14 at 19:27

5 Disclaimer: If you select features according to weights in your principal components you may or may not obtain something interesting. Once again, PCA is not made for throwing away features as defined by the canonical axes. In order to be sure what you are doing, try selecting `k` features using

categorical – eickenberg Apr 25 '14 at 19:54

1 Ok I'll take a look to those. To answer your previous question, I see components as pseudo-samples, is it wrong? I use feature and dimension interchangeably. However, in order to get k features (as a kind of feature selection), I think I have to swap samples and features, to obtain PCs that are pseudo-features (and not pseudo-samples). I do not know if it's clear. In this scenario I could correlate each feature with each PC, to see if it shows the same behavior across all samples. Thanks anyway for the effort :) – gc5 Apr 25 '14 at 19:57

1 Ok, maybe another step forward: PCs are not pseudo-samples but arrays of projections of the features on each principal component. So, if I did it correctly, if some of the features are over a certain threshold together in a PC (e.g. A = 0.75 and B = 0.9), and not relevant in the other PCs (say A = 0.1 and B = 0.05), maybe we can say that they can be summarized with B (if our objective is feature selection).. – gc5 Apr 25 '14 at 21:28

The projected features onto principal components will retain the important information (axes with maximum variances) and drop axes with small variances. This behavior is like to `compression` (Not discard).

And `X_proj` is the better name of `X_new` , because it is the projection of `X` onto `principal components`

You can reconstruct the `X_rec` as

```
X_rec = pca.inverse_transform(X_proj) # X_proj is originally X_new
```

Here, `X_rec` is close to `X` , but the `less important` information was dropped by PCA. So we can say `X_rec` is denoised.

In my opinion, I can say `the noise` is discard.

edited Apr 25 '14 at 14:40

answered Apr 25 '14 at 14:22
emeth
3,983 19 36

The answer marked above is incorrect. The sklearn site clearly states that the components_

> components_ : array, [n_components, n_features] Principal axes in feature space, representing the directions of maximum variance in the data. The components are sorted by explained_variance_.

http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

answered Feb 26 '17 at 1:37

Pramod Kalipatnapu
11