Using Demixed Principal Component Analysis

An Introduction for Python and Matlab

Wieland Brendel
Version 23 . 11 . 2011
Contact: wieland.brendel@neuro.fchampalimaud.org

Principal Component Analysis is a widespread data exploration tool. However, the results of this analysis are usually hard to interpret in terms of the underlying parameters that were present in the experiment. Demixed Principal Component Analysis (DPCA) is a new method targeted to simplify the interpretation of the principal components. This method basically uses the rotational symmetry of PCA to find a new coordinate system that demixes different task variables (given additional metadata).

In this article we describe the use of the Matlab and Python code. You can find a copy of the code at

http://sourceforge.net/projects/dpca/

The technical details have been published in NIPS 2011 *Demixed Principal Component Analysis*. This article is organized as follows: Necessary preprocessing steps can be found in section one. The return of the algorithm is described in section two. Known pitfalls are collected in the FAQ.

Finally, the authors appreciate all comments on the code and your story of using it. Please don't hesitate to contact them at

wieland.brendel@neuro.fchampalimaud.org

Preprocessing

There are versions of the algorithm in Python and Matlab. Both have different preprocessing requirements.

a | Python directions

Your Python installation has to include the **latest** Numpy 1.6 (for the function 'einsum'). The code is tested in Python 2.7 but should work from 2.6+ onward (Python 3+ not supported!). Please make sure you have the latest Numpy version since some bugs in 'einsum' have been fixed in small updates of Numpy 1.6!

Algorithm installation

The easiest way to use the algorithm is to put the script DPCA.py in your preferred directory (referred to as $PATH). In your analysis script please add

Table 1

DPCA(Y, n, tol=10e-3, maxloop=100)

Parameters:	Y : ndarray Data matrix n : int Number of components tol : float (default = 10e-3) Tolerance, relative change of evidence before break maxloop : int (default = 100) Maximum number of loops before break
Returns:	W : ndarray Mixing matrix

```
import sys
sys.path.append($PATH)
from DPCA import *
```

in the header. The function you will use in your algorithm is called DPCA. Please look at the input and output variables described in table 1.

Data format

The most important preprocessing step is to get your data into right format. As an example, consider a data set with N-dimensional data points (e.g. N recorded cells). You have identified two parameters *t* and *s* (like time bin and stimulus). Assume that *t* can take any value between 0 and 100 whereas *s* is just a binary variable. For every dimension (for every cell) you have a bunch of observations. Each observation carries a specific label *t* and *s*. You now average over all observations with the same label. After averaging you end up with data set Y that has the shape

Y.shape = (N,100,2)

It is precisely this format that you hand over to the function DPCA.

Keep in mind

If you process neural recordings, time has a rather distinct meaning: cues might

be presented early in time, rewards are scheduled for specific parts of the trial. Therefore, responses for different parameters will usually be convolved with time. Although dPCA treats all parameters - including time - seperately keep in mind that other parameters and time will usually be mixed.

b | Matlab directions

Compared to Python, Matlab is not fully designed to handle multidimensional matrices. Especially more complex operations like high-dimensional tensor products are only available through additional mex-packages. However, we focused our attention on the usability of the code and the method. Therefore, we opted to implement in Matlab a slightly less demanding version of DPCA based on a simple gradient ascent algorithm. The output is by and large identical to the Python version. If you experience any significant differences, please give us a note.

Algorithm installation

Just download the two *.m-files from Sourgeforge and store them in a path were Matlab can find them.

Data format

The most important preprocessing step is to get your data into right format. As an example, consider a data set with N-dimensional data points (e.g. N recorded cells). You have identified two parameters *t* and *s* (like time bin and stimulus). Assume that *t* can take any value between 0 and 100 whereas *s* is just a binary variable. For every dimension (for every cell) you have a bunch of observations. Each observation carries a specific label *t* and *s*. You now average over all observations with the same label. After averaging you end up with data set Y stored as a multidimensional array with the size

size(Y) = [N,100,2]

It is precisely this format that you hand over to the function DPCA. Note: the order of the metaparameters doesn't matter but make sure that the index of the observations is along the first axis!

Usage

The main function, namely dpca, is described in table 2. To get the loading matrix (the equivalent of the matrix of eigenvectors of the covariance matrix in PCA), just add

W = dpca(Y,10,[],[]);

where the arguments "[]" denote that you opt for the default values.

Return

In both Matlab and Python the function DPCA returns a two-dimensional array W with the shape (N,n) where n is the number of components you chose. The rows of W span the coordinate axis on which you will project your data by means of $Z(t,s) = W^T Y(t,s)$ for every combination of parameters *t,s*. Now, Z are the components you searched for.

Table2

dpca(Y, n, tol, maxloop)

Parameters:	Y : array Data matrix n : int Number of components tol : float (default = 10e-6) Tolerance, minimum relative change of objective function maxloop : int (default = 100) Maximum number of loops
Returns:	W : array Mixing matrix

FAQ

What you always wanted to know about DPCA

How do I choose the right number of components?

The right number of components solely depends on how much variance of your data you want to describe. Just calculate the variance that remains undescribed by the components you chose.

Is this method biasing the representation of the data set?

No. We are simply using the rotational symmetry of PCA to find a better representation of your data set. On the other hand, if there is no demixing of different parameters possible on the population level, dPCA won't be able to find a good representation.

Ain't I am loosing variance compared with PCA?

Usually you loose less than 1% of variance compared with PCA (depending on the number of components you choose). We are mainly relying on a rotational symmetry of PCA that leaves the total variance described untouched.

I have neural recordings and get strange looking components. They show signs of the stimulus before the onset of the stimulus presentation.

First, please check that the presentation of your stimulus really was random and unpredictable. If that is the case, check if the effect you see really is significant. Usually these effects are artefacts caused by an insufficient statistic: if there are only small numbers of trials for different stimuli a bunch of neurons will show different firing rates for different stimuli - even before their presentation - but the difference is not statistically significant. Successively, these effects might add up and be represented on the population level. Therefore, always check your error intervals.

Isn't this method similar to ICA or CCA?

No, both ICA and CCA do not incorporate any kind of metadata and can therefore, by design, not achieve the same results as DPCA.

Can't you achieve the same with one of the many regression methods?

No. Regression methods are hard to use in dimensionality reduction.

Can I use trials with different temporal length?

No. Just like in PCA trials have to have the same number of time bins. You have two ways to shape your data into that format: You can either stretch (and interpolate) your trials to match each other. Or you opt to align the trials on one event in the trial (like cue presentation) and you use only one fixed time interval before and after that event for your analysis.