Whitening

From Ufldl

Contents

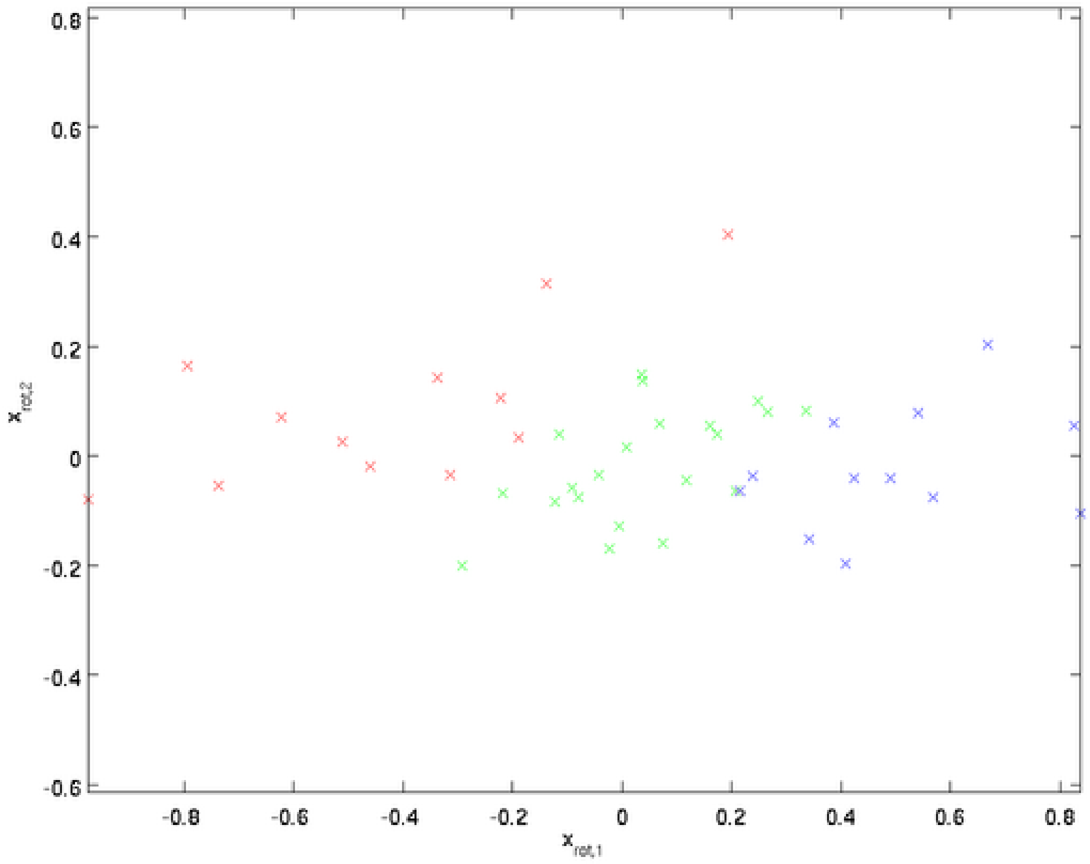
Introduction

We have used PCA to reduce the dimension of the data. There is a closely related preprocessing step called **whitening** (or, in some other literatures, **sphering**) which is needed for some algorithms. If we are training on images, the raw input is redundant, since adjacent pixel values are highly correlated. The goal of whitening is to make the input less redundant; more formally, our desiderata are that our learning algorithms sees a training input where (i) the features are less correlated with each other, and (ii) the features all have the same variance.

2D example

We will first describe whitening using our previous 2D example. We will then describe how this can be combined with smoothing, and finally how to combine this with PCA.

How can we make our input features uncorrelated with each other? We had already done this when computing $x_{\rm rot}^{(i)} = U^T x^{(i)}$. Repeating our previous figure, our plot for $x_{\rm rot}$ was:



The covariance matrix of this data is given by:

$$\begin{bmatrix} 7.29 & 0 \\ 0 & 0.69 \end{bmatrix}.$$

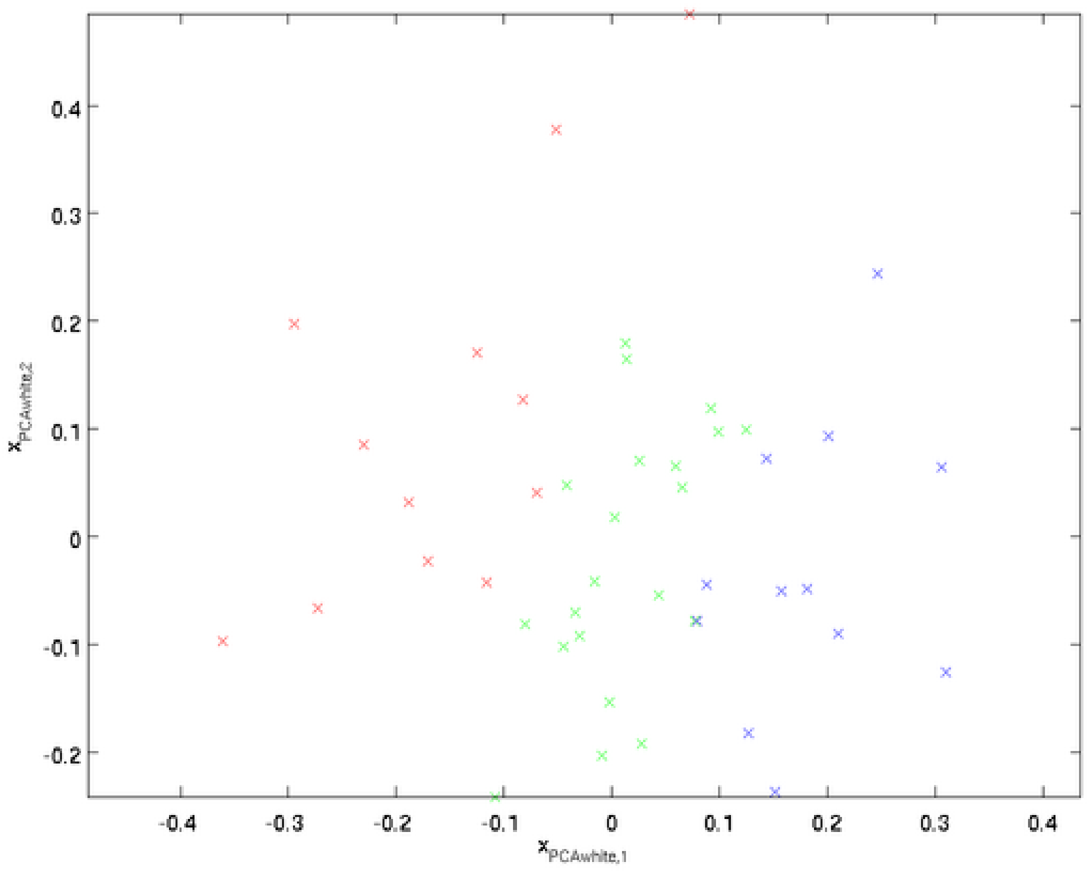
(Note: Technically, many of the statements in this section about the "covariance" will be true only if the data has zero mean. In the rest of this section, we will take this assumption as implicit in our statements. However, even if the data's mean isn't exactly zero, the intuitions we're presenting here still hold true, and so this isn't something that you should worry about.)

It is no accident that the diagonal values are λ_1 and λ_2. Further, the off-diagonal entries are zero; thus, $x_{\mathrm{rot},1}$ and $x_{\mathrm{rot},2}$ are uncorrelated, satisfying one of our desiderata for whitened data (that the features be less correlated).

To make each of our input features have unit variance, we can simply rescale each feature $x_{\mathrm{rot},i}$ by $1/\sqrt{\lambda_i}$. Concretely, we define our whitened data $x_{\mathrm{PCAwhite}} \in \Re^n$ as follows:

$$x_{\mathrm{PCAwhite},i} = \frac{x_{\mathrm{rot},i}}{\sqrt{\lambda_i}}.$$

Plotting x_{PCAwhite}, we get:



This data now has covariance equal to the identity matrix I. We say that $x_{\rm PCAwhite}$ is our **PCA whitened** version of the data: The different components of $x_{\rm PCAwhite}$ are uncorrelated and have unit variance.

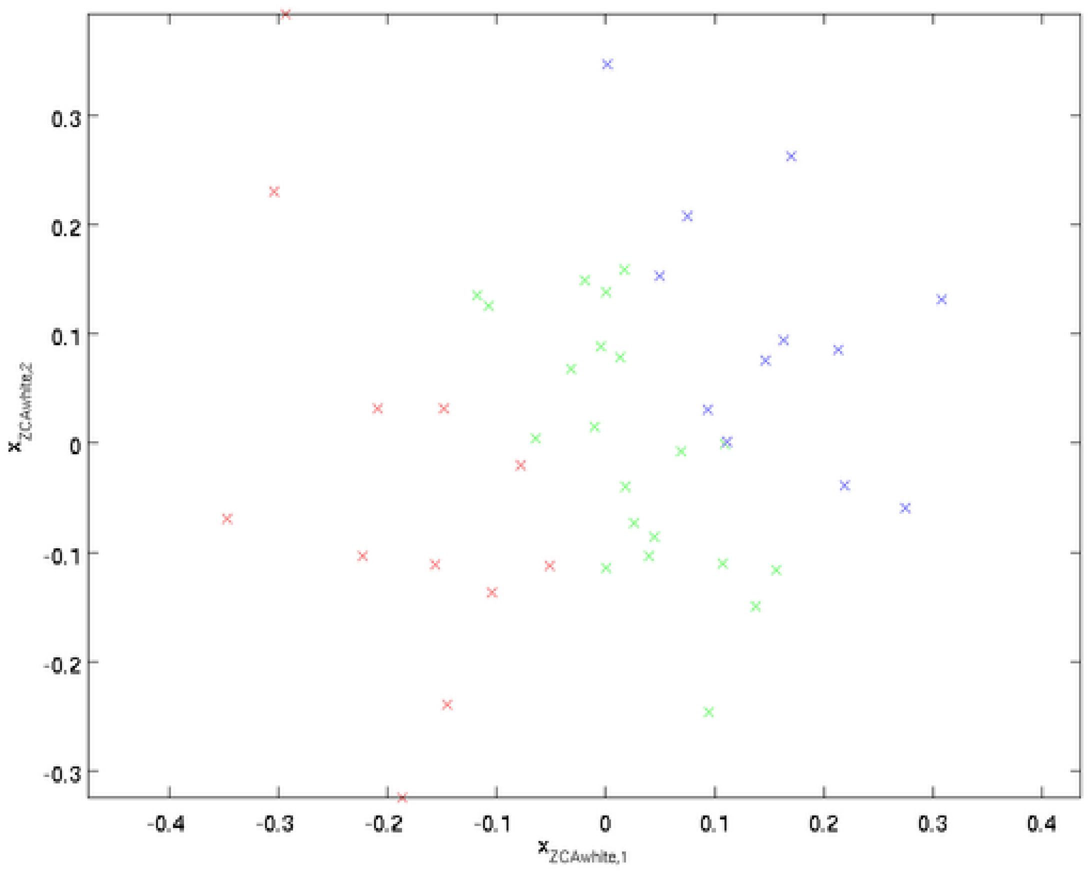
Whitening combined with dimensionality reduction. If you want to have data that is whitened and which is lower dimensional than the original input, you can also optionally keep only the top k components of $x_{\rm PCAwhite}$. When we combine PCA whitening with regularization (described later), the last few components of $x_{\rm PCAwhite}$ will be nearly zero anyway, and thus can safely be dropped.

ZCA Whitening

Finally, it turns out that this way of getting the data to have covariance identity I isn't unique. Concretely, if R is any orthogonal matrix, so that it satisfies $RR^T = R^TR = I$ (less formally, if R is a rotation/reflection matrix), then $R\,x_{\rm PCAwhite}$ will also have identity covariance. In **ZCA whitening**, we choose $R = U$. We define

$$x_{\rm ZCAwhite} = U x_{\rm PCAwhite}$$

Plotting $x_{\rm ZCAwhite}$, we get:



It can be shown that out of all possible choices for R, this choice of rotation causes $x_{\rm ZCAwhite}$ to be as close as possible to the original input data x.

When using ZCA whitening (unlike PCA whitening), we usually keep all n dimensions of the data, and do not try to reduce its dimension.

Regularizaton

When implementing PCA whitening or ZCA whitening in practice, sometimes some of the eigenvalues λ_i will be numerically close to 0, and thus the scaling step where we divide by $\sqrt{\lambda_i}$ would involve dividing by a value close to zero; this may cause the data to blow up (take on large values) or otherwise be numerically unstable. In practice, we therefore implement this scaling step using a small amount of regularization, and add a small constant ϵ to the eigenvalues before taking their square root and inverse:

$$x_{{\rm PCAwhite},i} = \frac{x_{{\rm rot},i}}{\sqrt{\lambda_i + \epsilon}}.$$

When x takes values around $[-1,1]$, a value of $\epsilon \approx 10^{-5}$ might be typical.

For the case of images, adding ϵ here also has the effect of slightly smoothing (or low-pass filtering) the input image. This also has a desirable effect of removing aliasing artifacts caused by the way pixels are laid out in an image, and can improve the features learned (details are beyond the scope of these notes).

ZCA whitening is a form of pre-processing of the data that maps it from x to $x_{\rm ZCAwhite}$. It turns out that this is also a rough model of how the biological eye (the retina) processes images. Specifically, as your eye perceives images, most adjacent "pixels" in your eye will perceive very similar values, since adjacent parts of an image tend to be highly correlated in intensity. It is thus wasteful for your eye to have to transmit every pixel separately (via your optic nerve) to your brain. Instead, your retina performs a decorrelation operation (this is done via retinal neurons that compute a function called "on center, off surround/off center, on surround") which is similar to that performed by ZCA. This results in a less redundant representation of the input image, which is then transmitted to your brain.

Language : 中文