PCA Whitening

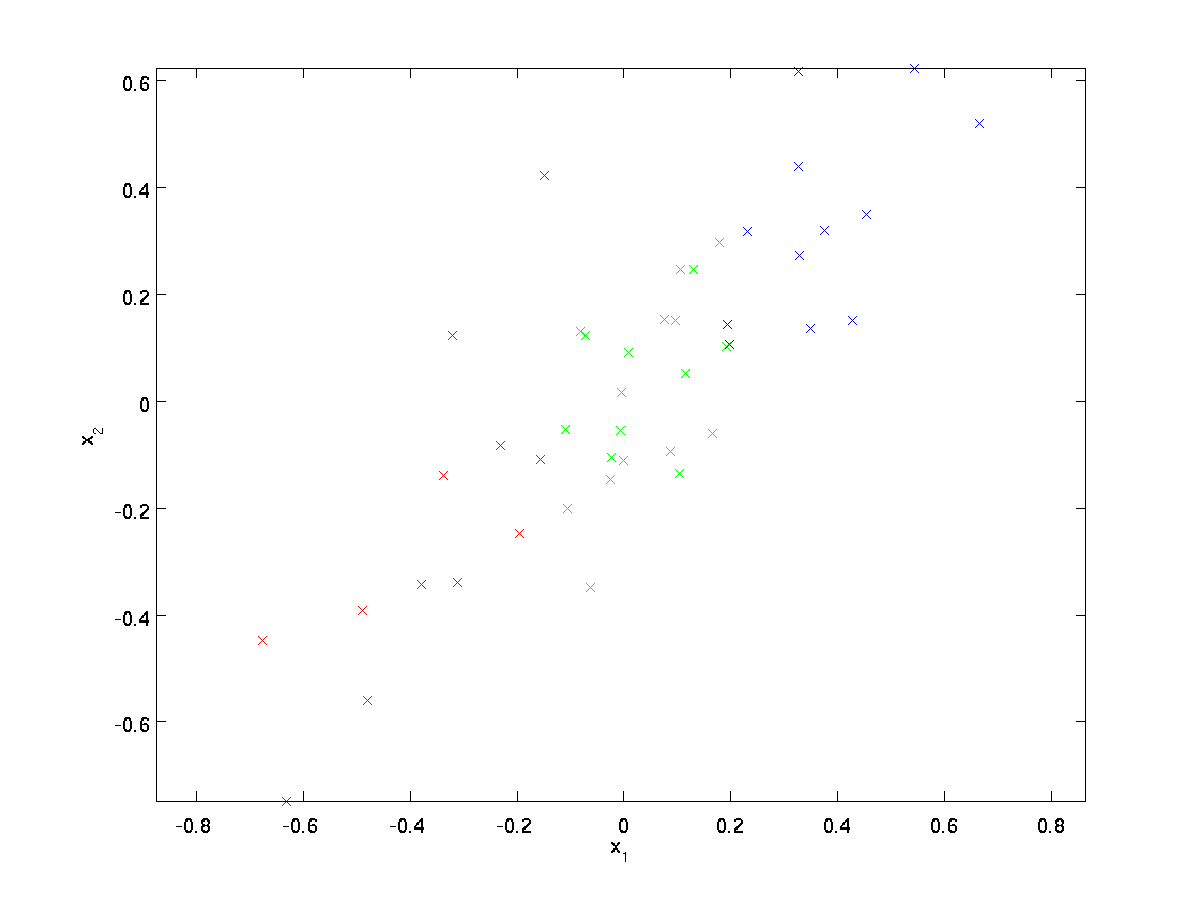
**1.Introduction**

Principal Components Analysis (PCA) is a dimensionality reduction algorithm that can be used to significantly speed up your unsupervised feature learning algorithm. More importantly, understanding PCA will enable us to later implement **whitening**, which is an important pre-processing step for many algorithms.

Suppose you are training your algorithm on images. Then the input will be somewhat redundant, because the values of adjacent pixels in an image are highly correlated. Concretely, suppose we are training on 16x16 grayscale image patches. Then are 256 dimensional vectors, with one feature corresponding to the intensity of each pixel. Because of the correlation between adjacent pixels, PCA will allow us to approximate the input with a much lower dimensional one, while incurring very little error.

**2. Example and Mathematical Background**

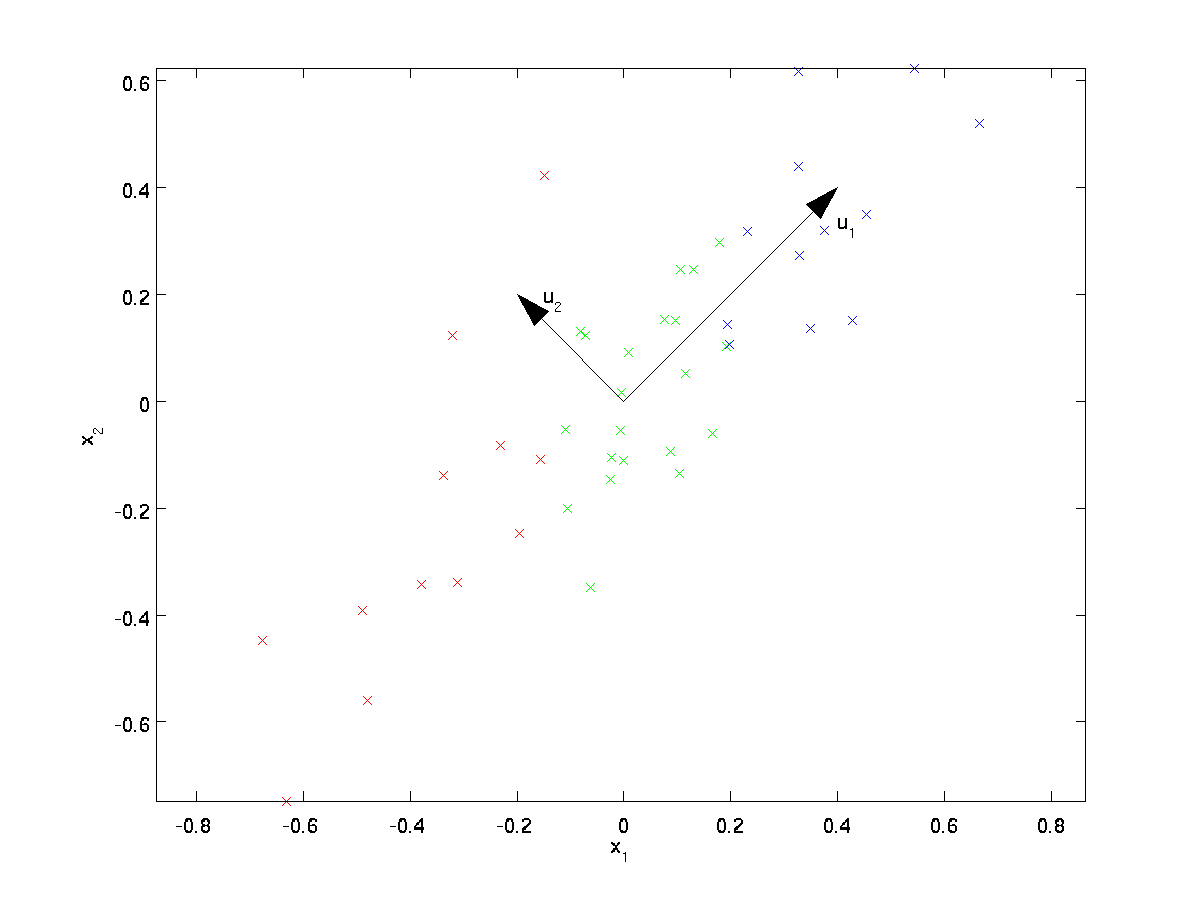
For our running example, we will use a dataset  with n=2 dimensional inputs, so that . Suppose we want to reduce the data from 2 dimensions to 1. (In practice, we might want to reduce data from 256 to 50 dimensions, say; but using lower dimensional data in our example allows us to visualize the algorithms better.) Here is our dataset:



This data has already been pre-processed so that each of the features x1 and x2 have about the same mean (zero) and variance.

For the purpose of illustration, we have also colored each of the points one of three colors, depending on their x1 value; these colors are not used by the algorithm, and are for illustration only.

PCA will find a lower-dimensional subspace onto which to project our data.  
From visually examining the data, it appears that u1 is the principal direction of variation of the data, and u2 the secondary direction of variation:



I.e., the data varies much more in the direction u1 than u2. To more formally find the directions u1 and u2, we first compute the matrix Σ as follows:



If x has zero mean, then Σ is exactly the covariance matrix of x. (The symbol ”Σ”, pronounced “Sigma”, is the standard notation for denoting the covariance matrix. Unfortunately it looks just like the summation symbol, as in ; but these are two different things.)

It can then be shown that u1—the principal direction of variation of the data—is the top (principal) eigenvector of Σ, and u2 is the second eigenvector.

### 3. Whitening

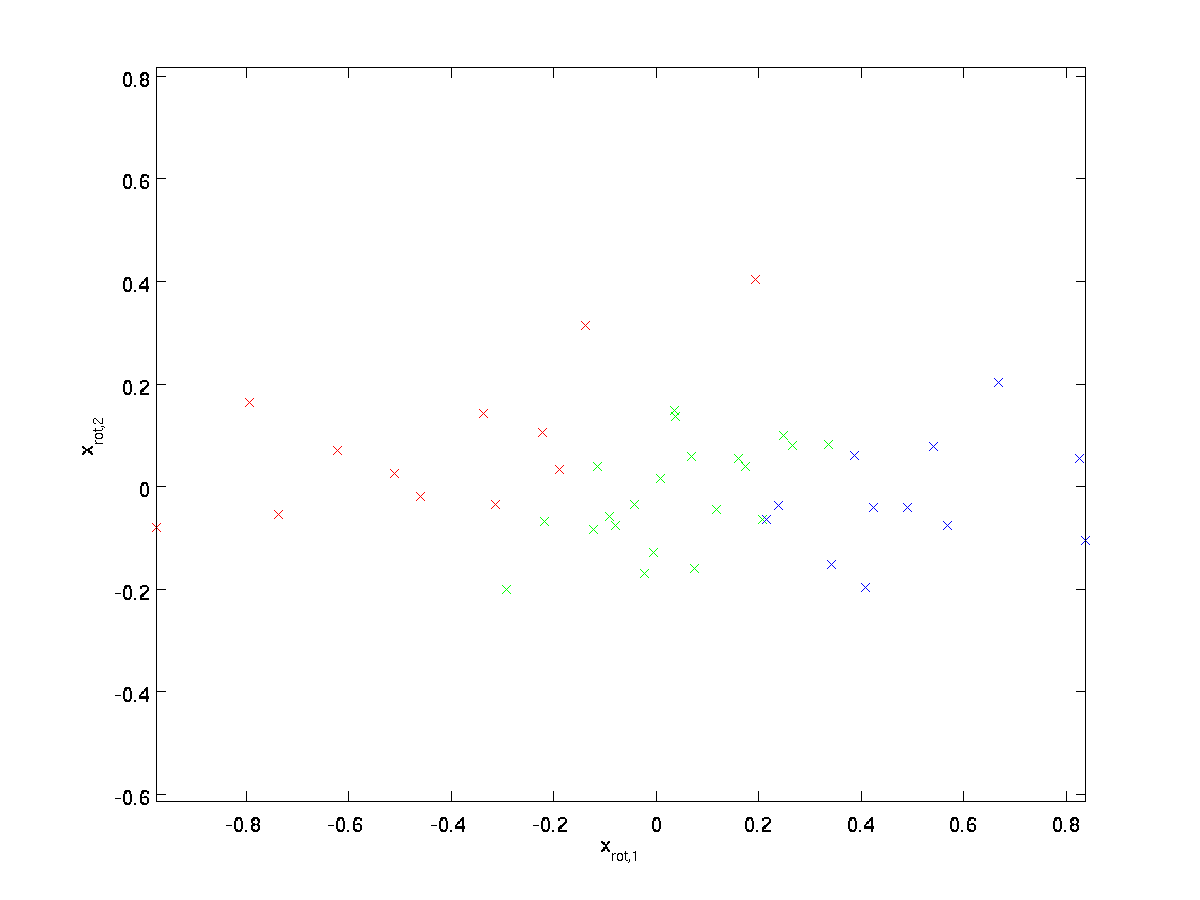
We have used PCA to reduce the dimension of the data. There is a closely related preprocessing step called **whitening** (or, in some other literatures, **sphering**) which is needed for some algorithms. If we are training on images, the raw input is redundant, since adjacent pixel values are highly 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.

### 3.1 2D example

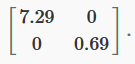
We will first describe whitening using our previous 2D example. We will then describe how this can be combined with smoothing, and finally how to combine this with PCA.

How can we make our input features uncorrelated with each other? We had already done this when computing .

Repeating our previous figure, our plot for xrot was:



The covariance matrix of this data is given by:



(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 λ1 and λ2. Further, the off-diagonal entries are zero; thus, xrot,1 and xrot,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 xrot,i by . Concretely, we define our whitened data  as follows:



Plotting, we get:

