From SVD to PCA

Singular Value Decomposition (SVD) and Principle Components Analysis (PCA) are often talked about synonomously, for good reasons, but they are different.

SVD is the decomposition of any matrix into three parts including a diagonal singular value matrix.

PCA is a data analysis technique (that uses SVD), but it make assumptions about what the data means and what you are aiming to do with it, so it is a little different at least conceptually.

You should, at this point, already know SVD, so we will start with PCA, and further, we'll start by considering what we are aiming to do.

Data

Data is commonly presented in tabular form (a table of values) where we interpret

1. The columns to correspond to measured variables;
2. The rows correspond to a sample, or an instance, or a single entity that has been measured.

When the data is numerical, we can treat our table like a matrix (or an order-2 tensor). For example in the following we have 4 samples, each with three variables.

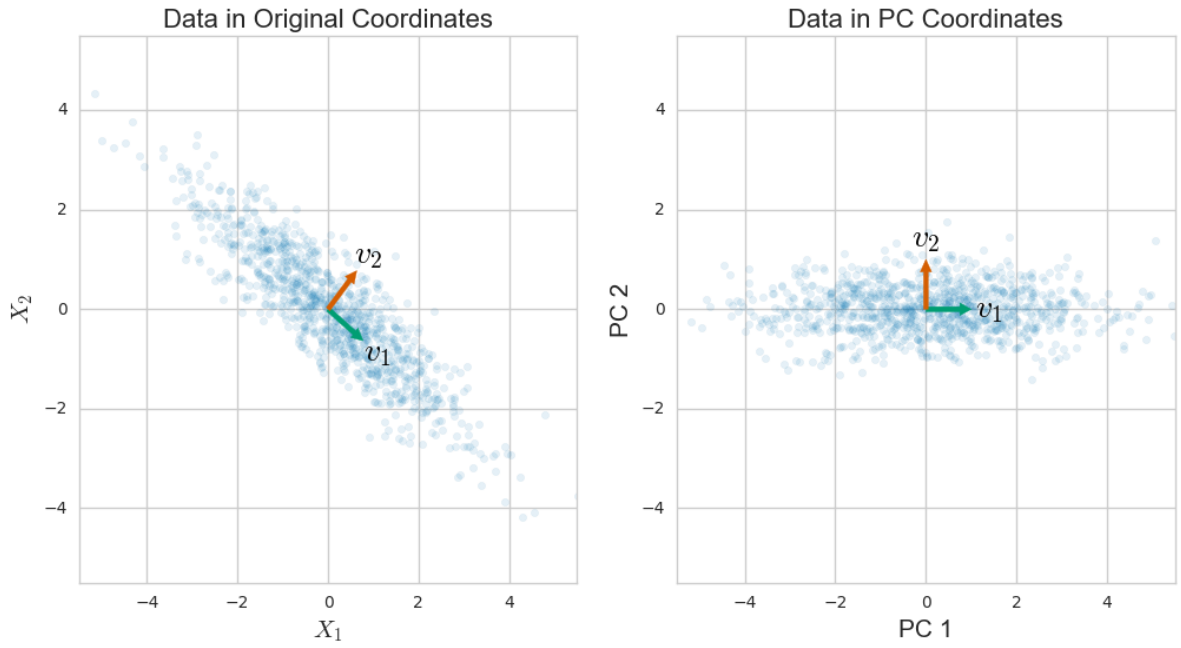
Age	Height	Weight (kg)
25	1.80	65.7
57	1.59	91.2
21	1.75	120.4
26	1.8	78.2

So imagine we have m samples, each of which contains n variables, then our data is a $m \times n$ matrix X.

Another way to think of this is that we have m vectors $\mathbf{x}_i \in \mathbb{R}^n$. Typically n is quite large (as we have discussed before).

Imagine that these data were distributed as a joint normal random variable with mean μ and covariance Σ. Then if $n = 1$, they are just a 1D normal, and in 2D they form an elliptical blob centred around the mean.

For instance look at the blob of data in the following image:



The first goal of PCA is to reorient the axes such that they are aligned with the directions of the axes in the ellipses. They are called the principle axes.

PCA

Let's start by calculating the sample mean and subtracting it from the matrix (that's standard stats so I'm not going to do it for you). From now on I will assume the columns of X have zero mean.

Then the sample covariance matrix is just the $n \times n$ matrix

$$S = \frac{1}{m-1} X^T X.$$

That means that $[S]_{ij}$ tells you about the correlation between variables i and j.

The matrix S is square and symmetric (check it) and normal (that means $SS^T = S^T S$), so it's an easy one to work with. In fact those properties mean that its eigenvalues will be real and non-negative, and that the matrix will always be diagonalisable.

But remember that the singular values of X are the square root of the eigenvalues of $X^T X$, so we can use SVD of X instead of calculating S and performing eigenanalysis (though we should be careful about the extra scale factor of $1/(n-1)$).

So now we can interpret:

- the columns of U as the *left singular vectors* of X.
- the columns of V as the *right singular vectors* of X, which are the eigenvectors of S, and which form the basis of the space we will transform data into.

There are nice proofs that these directions will do what we want, *i.e.,* line up with the axes of the ellipsoids.

The intuitive version of this is that the eigenvector directions are orthogonal basis vectors that correspond to a new set of variables (consisting of a linear combination of the original variables) and that these new variables are more meaningful than the originals. Because S is square and U and V are unitary, they correspond to rotations to reorient the axes in these directions. The variables corresponding to the largest singular values are most important because they account for the largest variance in the data.

That leads to the critical difference between SVD and PCA: in SVD we are concerned with decomposing the matrix, but in PCA we use the decomposition to perform a coordinate transformation. We can transform the original data into the new space with

$$T = XV = U\Sigma,$$

(remembering the U and V are unitary so, *e.g.*, $U^{-1} = U^T$). So a coordinate vector in the new space $\mathbf{t}$ (one of the rows of T) is given by the transform (remember our data vectors are being treated like row-vectors here):

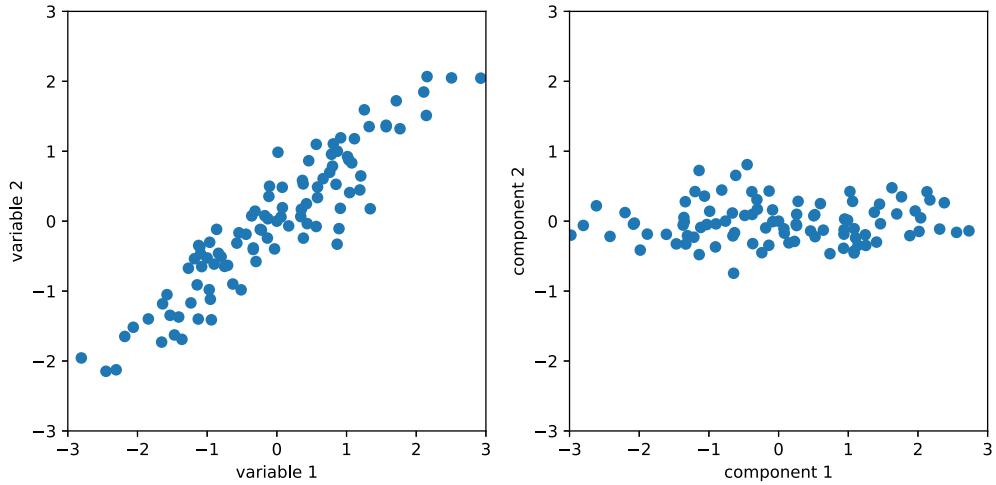
$$\mathbf{t} = \mathbf{x}V,$$

In the new space, the variables corresponding to the axes are now approximately independent, so this is also sometimes called *whitening*.

We can also apply the same transform to new data to map it into the same space.

An Example

The following takes a pair of jointly normal random variables with 100 samples, computes the SVD of the sample data X and then projects it into the new space, such that the principle components lie along the axes.



See `pca_example.ipynb` for the code to create it. Have a play around.

Dimension reduction

So now we get to the second goal of PCA, which is dimension reduction. The same approximation we used for A by only using the first r singular values, can be used here. We can reconstruct the matrix X using only the most important parts.

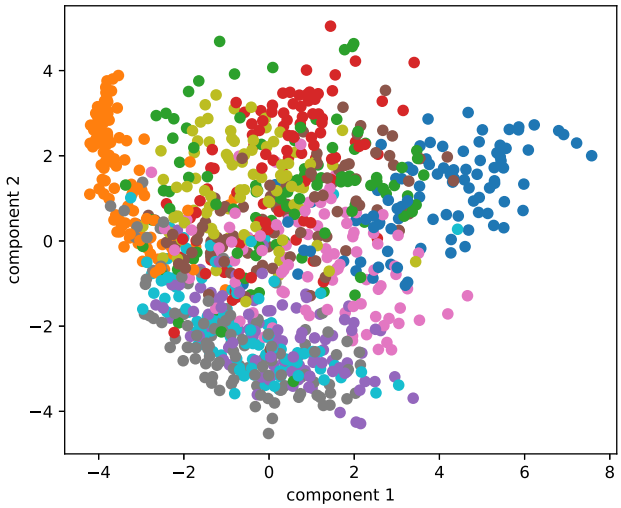
However, now we think about this somewhat differently. No longer is this about approximating the matrix as a whole (think about the racoon image example). Now we are projecting the data (the rows of X) from the original n dimensional space down into a r-dimensional space. The transformation is given by the matrices used for diagonalisation.

The procedure is almost obvious though. Instead of truncating the diagonal matrix, we truncate the transform V, so to include the top k singular values we take the matrix V_k that has the first k columns of V, and then compute the projection

$$\mathbf{t}^{(k)} = \mathbf{x}V_k.$$

An Example

The following is the PCA performed on the first 1000 MNIST training images. The colour of the point depends on the number depicted. We can see some quite strong groupings in the different images, but also large overlaps. This particular compression/projection of the images is revealing, but not a great idea if you want to use it for classification.



What next?

PCA is just one dimensionality reduction technique. Essentially it is doing a linear transform into a space that has certain properties with respect to the data, so unlike the Fourier transform it is customised around the data. It is customising its basis to the data. That is a good thing.

Hence PCA used a lot, but it isn't always the right approach.

1. PCA is "optimal" only in one sense, and that is closely linked with the idea that the data is jointly normal. The idea is based on the assumption that the mean and covariance of the data characterise the distribution, which intrinsically links to the data being normal. That can cause problems, for instance when there are outliers in the data that don't fit normality they can skew the directions unduly.
2. Some data (*e.g.,* heights) is intrinsically non-negative, and hence mean removal results in non-physical negative values.
3. PCA is a linear transform. Not all data lives on a linear space.
4. PCA dimension reduction is aimed a preserving or enhancing a particular property, but that doesn't necessarily solve all embedding problems – sometimes you might aim to preserve distances between points in the new space, for instance.
5. PCA's direction vectors tend to include bits over everything. That isn't ideal because they aren't *explanitory*. It would be more appealing if the basis vectors themselves had some meaning. For instance, Fourier basis vectors have a meaning in terms of frequency, and sparse basis vectors are groupings of variables.
6. PCA starts from a set of measurements in $\mathbb{R}^n$. That precludes non-numerical data (*e.g.,* categorical data), as well as data for which there is no simple mapping to a finite dimensional space (*e.g.,* strings).

There are many other approaches:

- Independent Components Analysis (ICA), which is aimed at separating components of the signal (the direction vectors shouldn't overlap). It's used to separate out multiple signals that have been mixed together (*e.g.,* speech in a noisy room).
- Multidimensional Scaling (MDS), which starts from a distance matrix between the data points, so it can handle almost any type of data.

- VAE autoencoders: a neural network approach to dimension reduction.
- word2vec: a neural network approach specialised for embedding words.

Links

- https://intoli.com/blog/pca-and-svd/
- https://www.math.tamu.edu/~dallen/m640_03c/lectures/chapter6.pdf