Making sense of principal component analysis, eigenvectors & eigenvalues

Imagine a big family dinner, where everybody starts asking you about PCA. First you explain it to your great-grandmother; then to you grandmother; then to your mother; then to your spouse; finally, to your daughter (who is a mathematician). Each time the next person is less of a layman. Here is how the conversation might go.

Great-grandmother: I heard you are studying "Pee-See-Ay". I wonder what that is...
You: Ah, it's just a method of summarizing some data. Look, we have some wine bottles standing here on the table. We can describe each wine by its colour, by how strong it is, by how old it is, and so on (see this very nice visualization of wine properties taken from here). We can compose a whole list of different characteristics of each wine in our cellar. But many of them will measure related properties and so will be redundant. If so, we should be able to summarize each wine with fewer characteristics! This is what PCA does.

Grandmother: This is interesting! So this PCA thing checks what characteristics are redundant and discards them?
You: Excellent question, Vivek! No, PCA is not selecting some characteristics and discarding the others. Instead, it constructs some *new* characteristics that turn out to summarize our list of wines well. Of course these new characteristics are constructed using the old ones; for example, a new characteristic might be computed as wine age minus wine acidity level or some other combination like that (we call them *inear combinations*).
In fact, PCA finds the best possible characteristics, the ones that summarize the list of wines as well as only possible (among all conceivable linear combinations). This is why it is so useful.

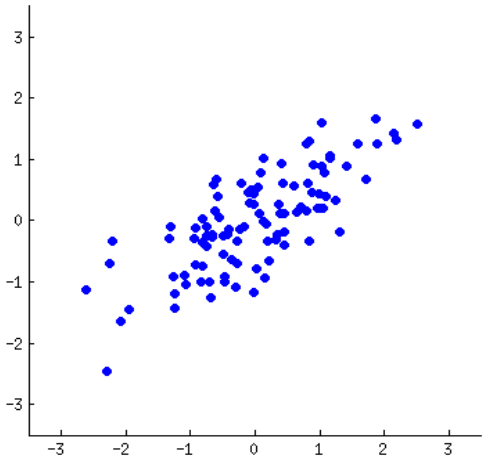
Mother: Hmmm, this certainly sounds good, but I am not sure I understand. What do you actually mean when you say that these new PCA characteristics "summarize" the list of wines?
You: I guess I can give two different answers to this question. First answer is that you are looking for some wine properties (characteristics) that strongly differ across wines. Indeed, imagine that you come up with a property that is the same for most of the wines. This would not be very useful, would it? Wines are very different, but your new property makes them all look the same! This would certainly be a bad summary. Instead, PCA looks for properties that show as much variation across wines as possible.
The second answer is that you look for the properties that would allow you to predict, or "reconstruct", the original wine characteristics. Again, imagine that you come up with a property that has no relation to the original characteristics; if you use only this new property, there is no way you could reconstruct the original ones! This, again, would be a bad

summary. So PCA looks for properties that allow to reconstruct the original characteristics as well as possible.

Surprisingly, it turns out that these two aims are equivalent and so PCA can kill two birds with one stone.

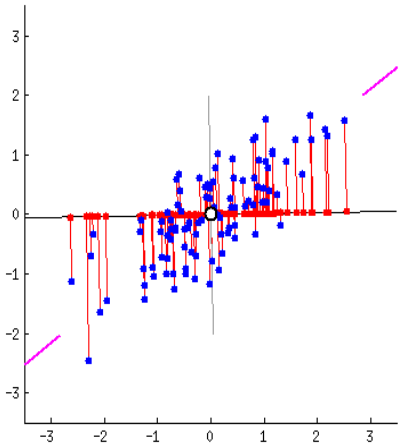
Spouse: But darling, these two "goals" of PCA sound so different! Why would they be equivalent?

You: Hmmm. Perhaps I should make a little drawing *(takes a napkin and starts scribbling)*. Let us pick two wine characteristics, perhaps wine darkness and alcohol content -- I don't know if they are correlated, but let's imagine that they are. Here is what a scatter plot of different wines could look like:



Each dot in this "wine cloud" shows one particular wine. You see that the two properties (x and y on this figure) are correlated. A new property can be constructed by drawing a line through the center of this wine cloud and projecting all points onto this line. This new property will be given by a linear combination w1x+w2y, where each line corresponds to some particular values of w1 and w2.

Now look here very carefully -- here is how these projections look like for different lines (red dots are projections of the blue dots):



As I said before, PCA will find the "best" line according to two different criteria of what is the "best". First, the variation of values along this line should be maximal. Pay attention to how the "spread" (we call it "variance") of the red dots changes while the line rotates; can you see when it reaches maximum? Second, if we reconstruct the original two characteristics (position of a blue dot) from the new one (position of a red dot), the reconstruction error will be given by the length of the connecting red line. Observe how the length of these red lines changes while the line rotates; can you see when the total length reaches minimum?

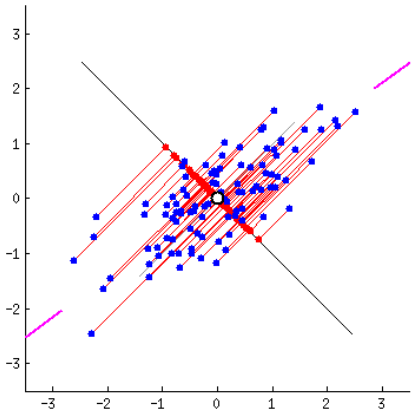
If you stare at this animation for some time, you will notice that "the maximum variance" and "the minimum error" are reached at the same time, namely when the line points to the magenta ticks I marked on both sides of the wine cloud. This line corresponds to the new wine property that will be constructed by PCA.

By the way, PCA stands for "principal component analysis" and this new property is called "first principal component". And instead of saying "property" or "characteristic" we usually say "feature" or "variable".

Daughter: Very nice, papa! I think I can see why the two goals yield the same result: it is essentially because of the Pythagoras theorem, isn't it? Anyway, I heard that PCA is somehow related to eigenvectors and eigenvalues; where are they on this picture?

You: Brilliant observation. Mathematically, the spread of the red dots is measured as the average squared distance from the center of the wine cloud to each red dot; as you know, it is called the *variance*. On the other hand, the total reconstruction error is measured as the average squared length of the corresponding red lines. But as the angle between red lines and the black line is always 90°, the sum of these two quantities is equal to the average squared distance between the center of the wine cloud and each blue dot; this is precisely Pythagoras theorem. Of course this average distance does not depend on the orientation of the black line, so the higher the variance the lower the error (because their sum is constant). This hand-wavy argument can be made precise (see here).

By the way, you can imagine that the black line is a solid rod and each red line is a spring. The energy of the spring is proportional to its squared length (this is known in physics as the Hooke's law), so the rod will orient itself such as to minimize the sum of these squared distances. I made a simulation of how it will look like, in the presence of some viscous friction:



Regarding eigenvectors and eigenvalues. You know what a *covariance matrix* is; in my example it is a 2×2 matrix that is given by

$$\begin{pmatrix}1.07 & 0.63\\ 0.63 & 0.64\end{pmatrix}.$$

What this means is that the variance of the x variable is 1.07, the variance of the y variable is 0.64, and the covariance between them is 0.63. As it is a square symmetric matrix, it can be diagonalized by choosing a new orthogonal coordinate system, given by its eigenvectors (incidentally, this is called *spectral theorem*); corresponding eigenvalues will then be located on the diagonal. In this new coordinate system, the covariance matrix is diagonal and looks like that:

$$\begin{pmatrix}1.52 & 0\\ 0 & 0.19\end{pmatrix},$$

meaning that the correlation between points is now zero. It becomes clear that the variance of any projection will be given by a weighted average of the eigenvalues (I am only sketching the intuition here). Consequently, the maximum possible variance (1.52) will be achieved if we simply take the projection on the first coordinate axis. It follows that the direction of the first principal component is given by the first eigenvector of the covariance matrix. (More details here.)

You can see this on the rotating figure as well: there is a gray line there orthogonal to the black one; together they form a rotating coordinate frame. Try to notice when the blue dots become uncorrelated in this rotating frame. The answer, again, is that it happens precisely when the black line points at the magenta ticks. Now I can tell you how I found them: they mark the direction of the first eigenvector of the covariance matrix, which in this case is equal to (0.81,0.58