Making sense of principal component analysis, eigenvectors & eigenvalues

In today's pattern recognition class my professor talked about PCA, eigenvectors & eigenvalues.

I got the mathematics of it. If I'm asked to find eigenvalues etc. I'll do it correctly like a machine. But I didn't **understand** it. I didn't get the purpose of it. I didn't get the feel of it. I strongly believe in

> you do not really understand something unless you can explain it to your grandmother -- Albert Einstein

Well, I can't explain these concepts to a layman or grandma.

1. Why PCA, eigenvectors & eigenvalues? What was the *need* for these concepts?
2. How would you explain these to a layman?

pca | intuition | eigenvalues

edited Aug 22 '16 at 15:57   asked Sep 15 '10 at 20:05

amoeba   **claws**
52.2k 13 177 235   **3,999**   3   11   9

---

83 Good question. I agree with the quote as well. I believe there are many people in statistics and mathematics who are highly intelligent, and can get very deep into their work, but don't deeply understand what they are working on. Or they do, but are incapable of explaining it to others.I go out of my way to provide answers here in plain English, and ask questions demanding plan English answers. – Neil McGuigan Sep 15 '10 at 21:43

7 This was asked on the Mathematics site in July, but not as well and it didn't get many answers (not surprising, given the different focus there). math.stackexchange.com/questions/1146/… – whuber ♦ Sep 16 '10 at 5:03

4 Similar to explanation by Zuur et al in Analyzing ecological data where they talk about projecting your hand on an overhead projector. You keep rotating your hand so that the projection on the wall looks pretty similar to what you think a hand should look like. – Roman Luštrik Sep 16 '10 at 9:00

7 This question lead me to a good paper, and even though I think that is a great quote it is not from Einstein. This is a common misattribution, and the more likely original quote is probably this one from Ernest Rutherford who said, "If you can't explain your physics to a barmaid it is probably not very good physics." All the same thanks for starting this thread. – gavaletz Apr 15 '13 at 15:03

17 Alice Calaprice, *The ultimate quotable Einstein*, Princeton U.P. 2011 flags the quotation here as one of many "Probably not by Einstein". See p.482. – Nick Cox Jun 20 '13 at 10:01

27 Answers

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, granny! 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 *linear combinations*).

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, wouldn't 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:
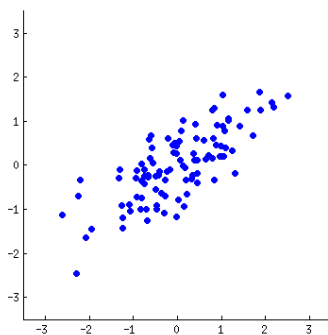
Each dot in this "wine cloud" shows one particular wine. You see that the two properties (x and y on this figure) are correlated. A new property can be constructed by drawing a line through the center of this wine cloud and projecting all points onto this line. This new property will be given by a linear combination $w_1 x + w_2 y$, where each line corresponds to some particular values of w_1 and w_2.

Now look here very carefully -- here is how these projections look like for different lines (red dots are projections of the blue dots):
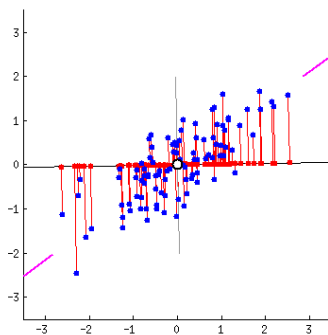
As I said before, PCA will find the "best" line according to two different criteria of what is the "best". First, the variation of values along this line should be maximal. Pay attention to how the "spread" (we call it "variance") of the red dots changes while the line rotates; can you see when it reaches 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

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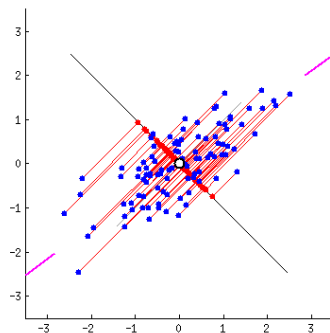
Regarding eigenvectors and eigenvalues. You know what a *covariance matrix* is; in my example it is a 2×2 matrix that is given by

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

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 $(0.81, 0.58)$.

Per popular request, I shared the Matlab code to produce the above animations.

This manuscript really helped me grok PCA. I think it's still too complex for explaining to your grandmother, but it's not bad. You should skip first few bits on calculating eigens, etc. Jump down to the example in chapter 3 and look at the graphs.

I have some examples where I worked through some toy examples so I could understand PCA vs. OLS linear regression. I'll try to dig those up and post them as well.

edit: You didn't really ask about the difference between Ordinary Least Squares (OLS) and PCA but since I dug up my notes I did a blog post about it. The very short version is OLS of y ~ x minimizes error perpendicular to the independent axis like this (yellow lines are examples of two errors):
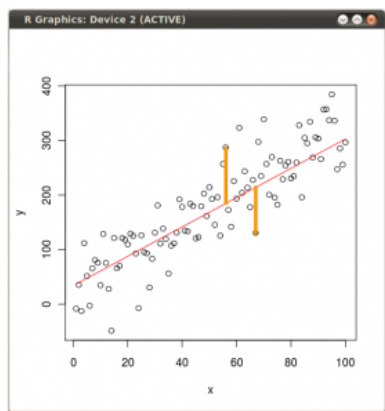
If you were to regress x ~ y (as opposed to y ~ x in the first example) it would minimize error like this:
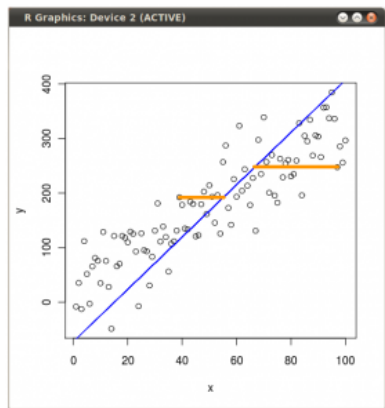
and PCA effectively minimizes error orthogonal to the model itself, like so:
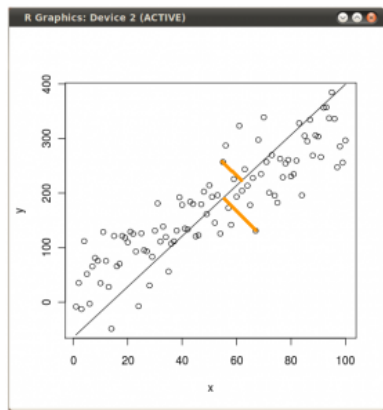
More importantly, as others have said, in a situation where you have a WHOLE BUNCH of independent variables, PCA helps you figure out which linear combinations of these variables matter the most. The examples above just help visualize what the first principal component looks like in a really simple case.

In my blog post I have the R code for creating the above graphs and for calculating the first principal component. It might be worth playing with to build your intuition around PCA. I tend to not really *own* something until I write code that reproduces it.

edited Aug 22 '16 at 14:19   answered Sep 15 '10 at 21:42

amoeba   JD Long
52.2k   13   177   235   **4,614**   1   12   15

---

9   Good call on the Lindsay I Smith manuscript - just read it today; very helpful. – Stedy Oct 23 '10 at 5:58

6   So is PCA equivalent to Total Least Squares if it optimizes orthogonal distances from points to the fit line? – Marcin Apr 16 '11 at 14:25

2   @Marcin - this is correct. You can re-phrase PCA as finding the best rank m estimate ($1 \leq m \leq p$) of the original p variables ($\hat{x}_{ij}$ $i = 1, \ldots, n$ $j = 1, \ldots, p$), with an objective function of $\sum_{i=1}^{n} \sum_{j=1}^{p} (x_{ij} - \hat{x}_{ij})^2$. Choosing the number of PCs is equivalent to choosing the rank of the predictions. – probabilityislogic Sep 5 '11 at 7:24

3   Small math error in Smith: "all the eigenvectors of a matrix are perpendicular... no matter how many dimensions you have" only applies to symmetric matrices, here's one with them 45 degrees apart. Smith notes the symmetry of covariance matrices earlier, but not the implication - symmetry ensures n perpendicular eigenvectors. In fact, not all real $n \times n$ matrices even have real eigenvalues (eg {{0,1},{-1,0}}) and of those that do, not all have n independent eigenvectors (eg {{1,1},{0,1}})! Symmetry matters! – Silverfish Aug 16 '13 at 20:08

7   As mathematician teaching eigenvectors, I have to cringe when reading this Lindsay Smith manuscript. "... resulting vector is an integer multiple of the original ..." - what is the point of mentioning **integer**? An eigenvector of matrix A is any vector X such that AX is a multiple of X. **Not an integer multiple, just a multiple!** A non-integer multiple is also ok! Jeez why creating unnecessary confusion where is none? – Dmitri Zaitsev Apr 17 '15 at 2:09

Let's do (2) first. PCA fits an ellipsoid to the data. An ellipsoid is a multidimensional generalization of distorted spherical shapes like cigars, pancakes, and eggs. These are all neatly described by the directions and lengths of their principal (semi-)axes, such as the axis of the cigar or egg or the plane of the pancake. No matter how the ellipsoid is turned, the eigenvectors point in those principal directions and the eigenvalues give you the lengths. The smallest eigenvalues correspond to the thinnest directions having the least variation, so ignoring them (which collapses them flat) loses relatively little information: that's PCA.

(1) Apart from simplification (above), we have needs for pithy description, visualization, and insight. Being able to reduce dimensions is a *good* thing: it makes it easier to describe the data and, if we're lucky to reduce them to three or less, lets us draw a picture. Sometimes we can even find useful ways to interpret the combinations of data represented by the coordinates in the picture, which can afford insight into the joint behavior of the variables.

The figure shows some clouds of 200 points each, along with ellipsoids containing 50% of each cloud and axes aligned with the principal directions. In the first row the clouds have essentially one principal component, comprising 95% of all the variance: these are the cigar shapes. In the second row the clouds have essentially two principal components, one about twice the size of the other, together comprising 95% of all the variance: these are the pancake shapes. In the third row all three principal components are sizable: these are the egg shapes.
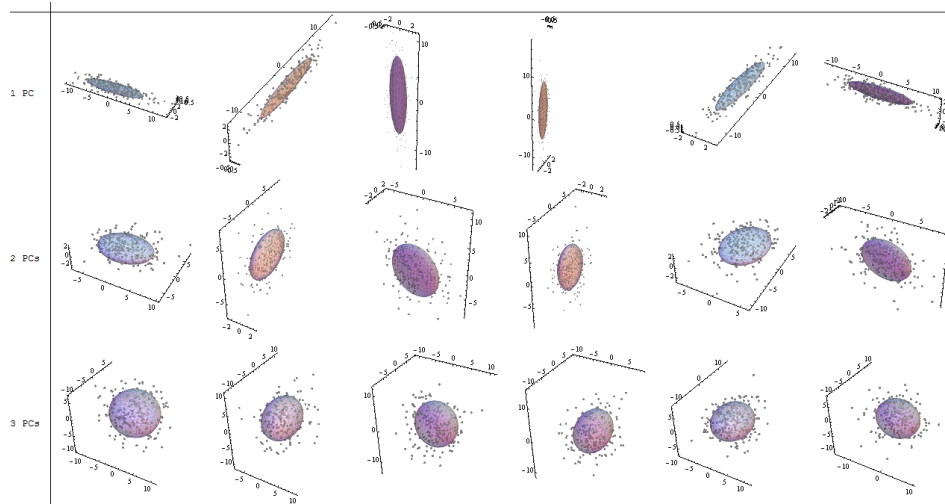
Any 3D point cloud that is "coherent" in the sense of not exhibiting clusters or tendrils or outliers will look like one of these. Any 3D point cloud *at all*--provided not all the points are coincident--can be described by one of these figures *as an initial point of departure* for identifying further clustering or patterning.

The intuition you develop from contemplating such configurations can be applied to higher dimensions, even though it is difficult or impossible to visualize those dimensions.

edited Nov 11 '14 at 16:53

answered Sep 15 '10 at 21:33

whuber ♦
192k   31   405   763

---

2   To add to this, when you have (near-)equal semiaxes (i.e. the ellipsoid has a (near-)circular slice), it indicates that the two pieces of data corresponding to those axes have (near-)dependency; one can talk about principal axes for an ellipse, but circles only have one radius. :) – J. M. is not a statistician Sep 16 '10 at 9:43

6   I would be more cautious here, J.M. First, just to clarify, by "near-dependency" you must mean "nearly independent." This would be true for a multinormal variate, but in many cases PCA is performed with data that are markedly non-normal. Indeed, the clustering analyses that follow some PCA calculations can be viewed as one way to assess a strong form of non-normality. Mathematically, circles *do* have principal axes, but they are just not uniquely determined: you can choose any orthogonal pair of radii as their principal axes. – whuber ♦ Sep 16 '10 at 14:11

1   Yes, sorry, I suppose "the principal axes of a circle are indeterminate" would have been a better way of putting it. – J. M. is not a statistician Sep 16 '10 at 16:13

2   Very nice interpretation! Trying to understand it better.. where in PCA math can one see that "PCA fits an ellipsoid to data"? – Kochede Nov 6 '13 at 2:27

4   @Kochede An ellipsoid is a contour of a quadratic form. The covariance matrix is a quadratic form. PCA identifies its axes and their lengths. – whuber ♦ Nov 6 '13 at 4:26

Hmm, here goes for a completely non-mathematical take on PCA...

Imagine you have just opened a cider shop. You have 50 varieties of cider and you want to work out how to allocate them onto shelves, so that similar-tasting ciders are put on the same shelf. There are lots of different tastes and textures in cider - sweetness, tartness, bitterness, yeastiness, fruitiness, clarity, fizziness etc etc. So what you need to do to put the bottles into categories is answer two questions:

1) What qualities are most important for identifying groups of ciders? e.g. does classifying based on sweetness make it easier to cluster your ciders into similar-tasting groups than classifying based on fruitiness?

2) Can we reduce our list of variables by combining some of them? e.g. is there actually a variable that is some combination of "yeastiness and clarity and fizziness" and which makes a really good scale for classifying varieties?

This is essentially what PCA does. Principal components are variables that usefully explain variation in a data set - in this case, that usefully differentiate between groups. Each principal component is one of your original explanatory variables, or a combination of some of your original explanatory variables.

answered Sep 15 '10 at 21:14

2   Okay: the Eigenvalue associated with each principal component tells you how much variation in the data set it explains (in my example, how clearly it separates your bottles into groups). They are usually expressed as a percentage of the total variation in the data set. As for the Eigenvectors, well, that's where as claws said I follow the output of an analysis like a machine ;) In my head, they are related to how you rotate Vince's mobile to its 'best' orientation, but this might not be the right way to think of them. – Freya Harrison Oct 27 '10 at 13:07

16   Eigenvectors are just the linear combinations of the original variables (in the simple or rotated factor space); they described how variables "contribute" to each factor axis. Basically, think of PCA as as way to construct new axes that point to the directions of maximal variance (in the original variable space), as expressed by the eigenvalue, and how variables contributions are weighted or linearly transformed in this new space. – chl ♦ Nov 23 '10 at 21:46

     what the covariance matrix of this problem would be like? what it tells us about the variables (sweetness, tartness, bitterness, yeastiness, fruitiness, clarity, fizziness etc etc)? – JustCurious Aug 9 '13 at 21:30

     This might not explain everything the OP asked for, but this is the only answer that I could actually use to explain PCA to my parents. – user86788 Nov 23 '15 at 22:24

I'd answer in "layman's terms" by saying that PCA aims to fit straight lines to the data points (everyone knows what a straight line is). We call these straight lines "principal components". There are as many principal components as there are variables. The first principal component is the best straight line you can fit to the data. The second principal component is the best straight line you can fit to the errors from the first principal component. The third principal component is the best straight line you can fit to the errors from the first and second principal components, etc., etc.

If someone asks what you mean by "best" or "errors", then this tells you they are not a "layman", so can go into a bit more technical details such as perpendicular errors, don't know where the error is in x- or y- direction, more than 2 or 3 dimensions, etc. Further if you avoid making reference to OLS regression (which the "layman" probably won't understand either) the explanation is easier.

The eigenvectors and eigenvalues are not needed concepts per se, rather they happened to be mathematical concepts that already existed. When you solve the mathematical problem of PCA, it ends up being equivalent to finding the eigenvalues and eigenvectors of the covariance matrix.

answered Sep 4 '11 at 23:18

probabilityislogic
18.4k   1   58   82

8   +1, this is truly in "layman's terms", and I know you could derive it *very* rigorously if you wanted to! – gung ♦ Jan 31 '12 at 3:53

2   The best answer so far, I'd say. And I use PCA a lot. – a11msp May 26 '14 at 21:56

2   Wow - this is really a great and simple explanation! Thank you! – Nick Feb 15 '15 at 19:20

Alright, I'll give this a try. A few months back I dug through a good amount of literature to find an intuitive explanation I could explain to a non-statistician. I found the derivations that use Lagrange multipliers the most intuitive.

Let's say we have high dimension data - say 30 measurements made on an insect. The bugs have different genotypes and slightly different physical features in some of these dimensions, but with such high dimension data it's hard to tell which insects belong to which group.

PCA is a technique to reduce dimension by:

1. Taking linear combinations of the original variables.
2. Each linear combination explains the most variance in the data it can.
3. Each linear combination is uncorrelated with the others

Or, in mathematical terms:

1. For $Y_j = a_j'x$ (linear combination for jth component)
2. For $k > j$, $V(Y_k) < V(Y_j)$ (first components explain more variation)
3. $a_k'a_j = 0$ (orthogonality)

Finding linear combinations that satisfy these constraints leads us to eigenvalues. Why?

I recommend checking out the book *An Introduction to Multivariate Data Analysis* for the full derivation (p. 50), but the basic idea is successive optimizations problems (maximizing variance) constrained such that a'a = 1 for coefficients a (to prevent the case when variance could be infinite) and constrained to make sure the coefficients are orthogonal.

I would never try to explain this to my grandmother, but if I had to talk generally about dimension reduction techniques, I'd point to this trivial projection example (not PCA). Suppose you have a Calder mobile that is very complex. Some points in 3-d space close to each other, others aren't. If we hung this mobile from the ceiling and shined light on it from one angle, we get a projection onto a lower dimension plane (a 2-d wall). Now, if this mobile is mainly wide in one direction, but skinny in the other direction, we can rotate it to get projections that differ in usefulness. Intuitively, a skinny shape in one dimension projected on a wall is less useful - all the shadows overlap and don't give us much information. However, if we rotate it so the light shines on the wide side, we get a better picture of the reduced dimension data - points are more spread out. This is often what we want. I think my grandmother could understand that :-)

edited Jan 16 '13 at 18:06	answered Sep 15 '10 at 21:07
Scortchi ♦	Vince
22.5k 7 43 149	**615** 6 7

---

6   That's very layman ;-) – mbq Sep 15 '10 at 21:24

2   It's a little mathy, but the best way to understand something is to derive it. – Vince Sep 16 '10 at 1:08

28   You have an exceptionally well-educated grandmother :-). – whuber ♦ Sep 16 '10 at 1:44

7   i like the explanation with the light shining on a 3-d structure – Neil McGuigan Jun 7 '11 at 18:40

   (+1) All are great answers but this is the one I would also give. – Digio Feb 25 '16 at 21:59

I can give you my own explanation/proof of the PCA, which I think is really simple and elegant, and doesn't require anything except basic knowledge of linear algebra. It came out pretty lengthy, because I wanted to write in simple accessible language.

Suppose we have some M samples of data from an n-dimensional space. Now we want to project this data on a few lines in the n-dimensional space, in a way that retains as much variance as possible (that means, the variance of the projected data should be as big compared to the variance of original data as possible).

Now, let's observe that if we translate (move) all the points by some vector β, the variance will remain the same, since moving all points by β will move their arithmetic mean by β as well, and variance is linearly proportional to $\sum_{i=1}^{M} \|x_i - \mu\|^2$. Hence we translate all the points by $-\mu$, so that their arithmetic mean becomes 0, for computational comfort. Let's denote the translated points as $x_i' = x_i - \mu$. Let's also observe, that the variance can be now expressed simply as $\sum_{i=1}^{M} \|x_i'\|^2$.

Now the choice of the line. We can describe any line as set of points that satisfy the equation $x = \alpha v + w$, for some vectors v, w. Note that if we move the line by some vector γ orthogonal to v, then all the projections on the line will also be moved by γ, hence the mean of the projections will be moved by γ, hence the variance of the projections will remain unchanged. That means we can move the line parallel to itself, and not change the variance of projections on this line. Again for convenience purposes let's limit ourselves to only the lines passing through the zero point (this means lines described by $x = \alpha v$).

Alright, now suppose we have a vector v that describes the direction of a line that is a possible candidate for the line we search for. We need to calculate variance of the projections on the line αv. What we will need are projection points and their mean. From linear algebra we know that in this simple case the projection of x_i' on αv is $\langle x_i, v \rangle / \|v\|_2$. Let's from now on limit ourselves to only unit vectors v. That means we can write the length of projection of point x_i' on v simply as $\langle x_i', v \rangle$.

In some of the previous answers someone said that PCA minimizes the sum of squares of distances from the chosen line. We can now see it's true, because sum of squares of projections plus sum of squares of distances from the chosen line is equal to the sum of squares of distances from point 0. By maximizing the sum of squares of projections, we minimize the sum of squares of distances and vice versa, but this was just a thoughtful digression, back to the proof now.

As for the mean of the projections, let's observe that v is part of some orthogonal basis of our space, and that if we project our data points on every vector of that basis, their sum will cancel out (it's like that because projecting on the vectors from the basis is like writing the data points in the new orthogonal basis). So the sum of all the projections on the vector v (let's call the sum S_v) and the sum of projections on other vectors from the basis (let's call it S_o) is 0, because it's the mean of the data points. But S_v is orthogonal to S_o! That means $S_o = S_v = 0$.

So the mean of our projections is 0. Well, that's convenient, because that means the variance is just the sum of squares of lengths of projections, or in symbols

Well well, suddenly the covariance matrix popped out. Let's denote it simply by X. It means we are now looking for a unit vector v that maximizes $v^T \cdot X \cdot v$, for some semi-positive definite matrix X.

Now, let's take the eigenvectors and eigenvalues of matrix X, and denote them by $e_1, e_2, \ldots, e_n$ and $\lambda_1, \ldots, \lambda_n$ respectively, such that $\lambda_1 \geq \lambda_2, \geq \lambda_3 \ldots$. If the values λ do not duplicate, eigenvectors form an orthonormal basis. If they do, we choose the eigenvectors in a way that they form an orthonormal basis.

Now let's calculate $v^T \cdot X \cdot v$ for an eigenvector e_i. We have

$$e_i^T \cdot X \cdot e_i = e_i^T \cdot (\lambda_i e_i) = \lambda_i (\|e_i\|_2)^2 = \lambda_i.$$

Pretty good, this gives us λ_1 for e_1. Now let's take an arbitrary vector v. Since eigenvectors form an orthonormal basis, we can write $v = \sum_{i=1}^{n} e_i \langle v, e_i \rangle$, and we have $\sum_{i=1}^{n} \langle v, e_i \rangle^2 = 1$. Let's denote $\beta_i = \langle v, e_i \rangle$.

Now let's count $v^T \cdot X \cdot v$. We rewrite v as a linear combination of e_i, and get:

$$\left(\sum_{i=1}^{n} \beta_i e_i\right)^T \cdot X \cdot \left(\sum_{i=1}^{n} \beta_i e_i\right) = \left(\sum_{i=1}^{n} \beta_i e_i\right) \cdot \left(\sum_{i=1}^{n} \lambda_i \beta_i e_i\right) = \sum_{i=1}^{n} \lambda_i (\beta_i)^2 (\|e_i\|_2)^2.$$

The last equation comes from the fact the eigenvectors where chosen to be pairwise orthogonal, so their dot products are zero. Now, because all eigenvectors are also of unit length, we can write $v^T \cdot X \cdot v = \sum_{i=1}^{n} \lambda_i \beta_i^2$, where β_i^2 are all positive, and sum to 1.

That means that the variance of the projection is a weighted mean of eigenvalues. Certainly, it is always less then the biggest eigenvalue, which is why it should be our choice of the first PCA vector.

Now suppose we want another vector. We should chose it from the space orthogonal to the already chosen one, that means the subspace $\mathrm{lin}(e_2, e_3, \ldots, e_n)$. By analogical inference we arrive at the conclusion, that the best vector to project on is e_2. And so on, and so on...

By the way, it should be now clear, why the variance retained can be expressed by $\sum_{i=1}^{k} \lambda_i / \sum_{i=1}^{n} \lambda_i$.

We should also justify the greedy choice of vectors. When we want to choose k vectors to project onto, it might not be the best idea to first choose the best vector, then the best from what remains and so on. I'd like to argue that in this case it is justified and makes no difference. Lets denote the k vector we wish to project onto by $v_1, \ldots, v_k$. Also, let's assume the vectors are pairwise orthogonal. As we already know, the total variance of the projections on those vectors can be expressed by

$$\sum_{j=1}^{k} \sum_{i=1}^{n} \lambda_i \beta_{ij}^2 = \sum_{i=1}^{n} \lambda_i \gamma_i$$

where $\gamma_i = \sum_{j=1}^{k} \beta_{ij}^2$.

Now, let's write e_i in some orthonormal basis that includes $v_1, \ldots, v_k$. Let's denote the rest of the basis as $u_1, \ldots, u_{n-k}$. We can see that $e_i = \sum_{j=1}^{k} \beta_{ij} v_j + \sum_{j=1}^{n-k} \theta_j \langle e_i, u_j \rangle$. Because $\|e_i\|_2 = 1$, we have $\sum_{j=1}^{k} \beta_{ij}^2 + \sum_{j=1}^{n-k} \theta_j^2 = 1$, and hence $\gamma_i \leq 1$ for all i.

Now we have a similar case to one vector only, we now know that the total variance of projections is $\sum_{i=1}^{n} \lambda_i \gamma_i$ with $\gamma_i \leq 1$ and $\sum_{i=1}^{n} \gamma_i = k$. This is yet another weighted mean, and is certainly no more than $\sum_{i=1}^{k} \lambda_i$ which corresponds to projecting on k eigenvectors corresponding to biggest eigenvalues.

edited Jun 22 '17 at 21:52   answered Aug 4 '12 at 3:54

sjm.majewski
2,651   1   12   22

+1 Very nice answer! Haven't yet read it completely, but your answer is the kind of I've been looking for. All the steps explained =) – jjepsuomi Jun 20 '13 at 6:57

6   Show me a Layman who know basic linear algebra, and I'll show you an mathematics undergraduate. – probabilityislogic Apr 15 '14 at 15:18

From linear algebra we know that in this simple case the projection of x_i' *on* αv *is* $\langle x_i, v \rangle / \|v\|^2$ (5th paragraph). Shouldn't it be $\langle x_i, v \rangle / \|v\|$? In other words, the scalar projection? – Antoni Parellada Jun 22 '17 at 1:39

Antoni Parellada: you're right, the 2 should be in lower index not the upper one. I have corrected it. – sjm.majewski Jun 22

Trying to be non-technical... Imagine you have a multivariate data, a multidimensional cloud of points. When you compute covariance matrix of those you actually (a) center the cloud, i.e. put the origin as the multidimensional mean, the coordinate system axes now cross in the centre of the cloud, (b) encrypt the information about the shape of the cloud and how it is oriented in the space by means of variance-covariance entries. So, most of the important info about the shape of the data as a whole is stored in the covariance matrix.

Then you do eigen-decomposition of that martrix and obtain the list of eigenvalues and the corresponding number of eigenvectors. Now, the 1st *principal component* is the new, latent variable which can be displayed as the axis going through the origin and oriented along the direction of the maximal variance (thickness) of the cloud. The variance along this axis, i.e. the variance of the coordinates of all points on it, *is* the first eigenvalue, and the orientation of the axis in space referenced to the original axes (the variables) is defined by the 1st eigenvector: its entries are the cosines between it and those original axes. The aforementioned coordinates of data points on the 1st component are the 1st principal component values, or component scores; they are computed as the product of (centered) data matrix and the eigenvector.

"After" the 1st pr. component got measured it is, to say, "removed" from the cloud with all the variance it accounted for, and the dimensionality of the cloud drops by one. Next, everything is repeated with the second eigenvalue and the second eigenvector - the 2nd pr. component is being recorded, and then "removed". Etc.

So, once again: eigenvectors are direction cosines for principal components, while eigenvalues are the magnitude (the variance) in the principal components. Sum of all eigenvalues is equal to the sum of variances which are on the diagonal of the variance-covariance matrix. If you transfer the "magnitudinal" information stored in eigenvalues over to eigenvectors to add it to the "orientational" information stored therein you get what is called principal component *loadings*; these loadings - because they carry both types of information - are the covariances between the original variables and the principal components.

Later P.S. I want especially to stress twice here the terminologic difference between **eigenvectors** and **loadings**. Many people and some packages (including some of `R`) flippantly use the two terms interchangeably. It is a bad practice because the objects and their meanings are different. Eigenvectors are the direction cosines, the angle of the orthogonal "rotation" which PCA amounts to. Loadings are eigenvectors inoculated with the information about the variability or magnitude of the rotated data. The loadings are the association coefficients between the components and the variables and they are directly comparable with the association coefficients computed between the variables - covariances, correlations or other scalar products, on which you base your PCA. Both eigenvectors and loadings are similar in respect that they serve regressional coefficients in predicting the variables by the components (not vice versa![1]). Eigenvectors are the coefficients to predict variables by raw component scores. Loadings are the coefficients to predict variables by scaled (normalized) component scores (no wonder: loadings have precipitated information on the variability, consequently, components used must be deprived of it). One more reason not to mix eigenvectors and loadings is that some other dimensionality reduction techiques besides PCA - such as some forms of Factor analysis - compute loadings directly, bypassing eigenvectors. Eigenvectors are the product of eigen-decomposition or singular-value decomposition; some forms of factor analysis do not use these decompositions and arrive at loadings other way around. Finally, it is loadings, not eigenvectors, by which you interpret the components or factors (if you need to interpret them). Loading is about a contribution of component into a variable: in PCA (or factor analysis) component/factor loads itself onto variable, not vice versa. In a comprehensive PCA results one should report both eigenvectors and loadings, as shown e.g. here or here.

See also about loadings vs eigenvectors.

[1] Since eigenvector matrix in PCA is orthonormal and its inverse is its transpose, we may say that those same eigenvectors are also the coefficients to back predict the components by the variables. It is not so for loadings, though.

edited Apr 13 '17 at 12:44	answered Sep 4 '12 at 8:11
Community ♦	ttnphns
1	35.9k 11 123 300

@amoeba, I don't insist and you may use any terminology you're used to. I clearly explicated why I think the terms "loadings" and "eigenvectors" are better to keep separate. I follow classic tradition, such as in Harman. Modern Factor analysis, if only I remember the tradition correctly. – ttnphns Apr 4 '14 at 12:12

(Cont.) Anyway, you know yourself that the term "loadings", albeit being really quite dubious, is nevetherless not mixed up with "eigenvector" in other multivariate analyses, such as discriminant analysis, for example. Once again, as I put it, in PCA loadings 1) incorporate info about the magnitude of variation; 2) Are the covariances/correlations, and are therefore used for interpretation. Eigenvector values - are not. – ttnphns Apr 4 '14 at 12:15

2   +1 I've been reading your posts on PCA and other related issues, and learned a lot. – Antoni Parellada Mar 12 '16 at 23:26

If you have a bunch of variables on a bunch of subjects and you want to reduce it to a smaller number of variables on those same subjects, while losing as little information as possible, then PCA is one tool to do this.

It differs from factor analysis, although they often give similar results, in that FA tries to recover a small number of latent variables from a larger number of observed variables that are believed to be related to the latent variables.
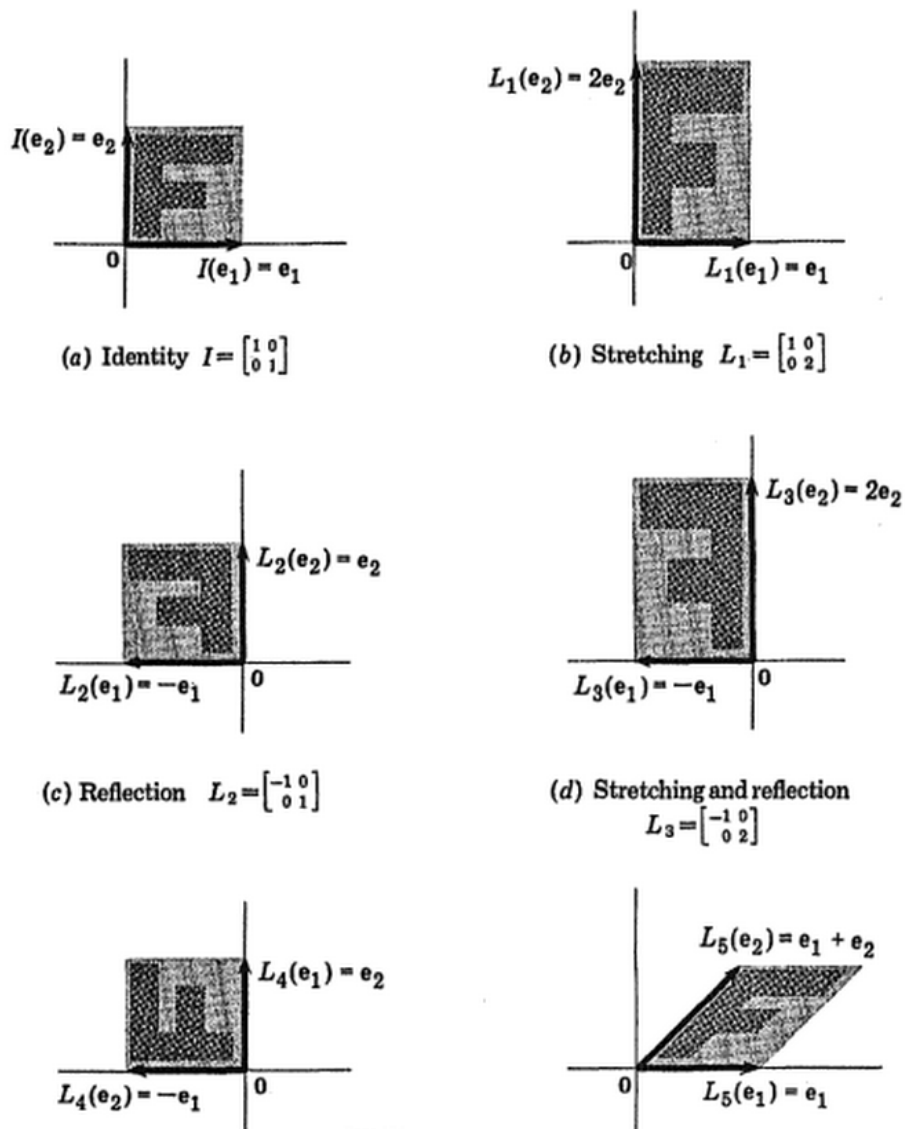
Hey Peter! Good to see you here. This is a really good, simple, no math answer. – JD Long Sep 16 '10 at 19:40
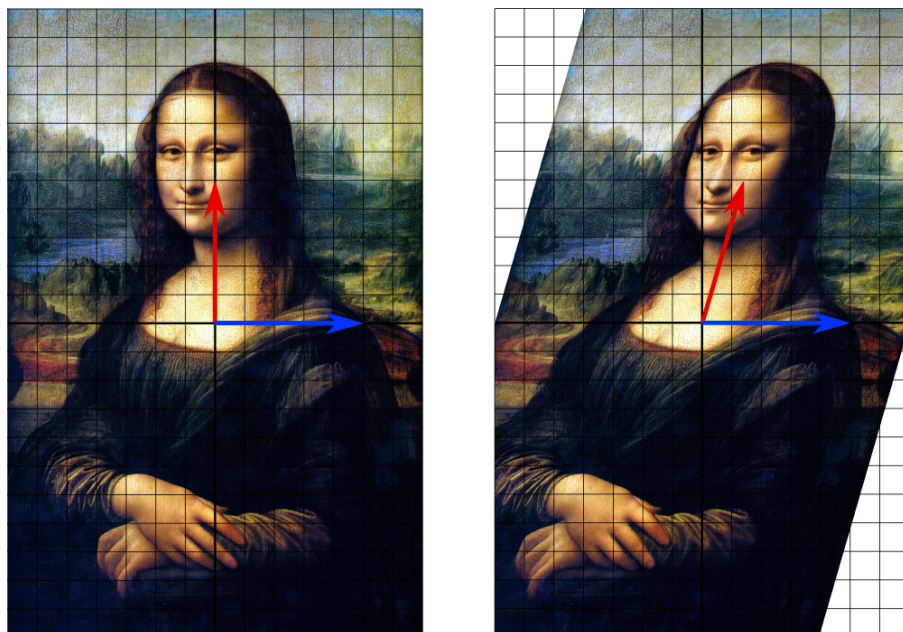
3   +1 for mentioning FA, which no one else seems to discuss, and which some people's explanations seem to blend towards. – gung ♦ Jan 31 '12 at 3:52

Seems to be no difference in the goals of PCA and FA - both aim to rotate so you can see the most important factors (latent vectors, or eigendimensions or singular vectors or whatever). But FA seems to be not an algorithm but a family of related techniques (to each other and SVD and PCA) with correspondingly ill-defined aims (which is to say diverse and mutually inconsistent, so different variants 'optimise' different things). – David M W Powers Jan 11 '14 at 2:45

It's easiest to do the maths in 2-D.

Every matrix corresponds to a linear transformation. Linear transformations can be visualised by taking a memorable figure on the plane and seeing how that figure is distorted by the linear transform:

(a) Identity $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

(b) Stretching $L_1 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$

(c) Reflection $L_2 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$

(d) Stretching and reflection $L_3 = \begin{bmatrix} -1 & 0 \\ 0 & 2 \end{bmatrix}$

(blue stayed the same, so that direction is an eigenvector of `shear`.)

- **Eigenvalues** are how much the stay-the-same vectors grow or shrink. (blue stayed the same size so the eigenvalue would be $\times 1$.)

- **PCA** rotates your axes to "line up" better with your data. PCA football
  http://weigend.com/files/teaching/stanford/2008/stanford2008.wikispaces.com/file/view/pca_example.gif PCA uses the eigenvectors of the covariance matrix to figure out how you should rotate the data. Because rotation is a kind of linear transformation, your new dimensions will be sums of the old ones, like $\langle 1 \rangle = 23\% \cdot [1] + 46\% \cdot [2] + 39\% \cdot [3]$.

The reason people who work with real data are interested in eigenvectors and linear transformations is that in different contexts, "linear" ($f(a \cdot x + b \cdot y) = a \cdot f(x) + b \cdot f(y)$) can cover really interesting stuff. For example think what that property means if $+$ and $\cdot$ are given new meanings, or if a and b come from some interesting field, or x and y from some interesting space. For example:
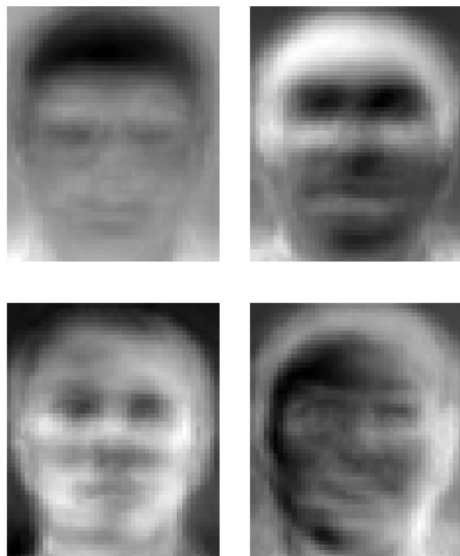
PCA itself is another example, the one most familiar to statisticians. Some of the other answers like Freya's give real-world applications of PCA.

† I find it totally surprising that something as simple as "rotation" could do so many things in different
$$\text{similar how?}$$
areas, like lining up products for a recommender system $\longleftarrow\!-\!-\!-\!-\!\longrightarrow$ explaining geopolitical
conflict. But maybe it's not so surprising if you think about physics, where choosing a better basis (e.g.

> 1   Actually it is kind of coincidental that rotations are linear and so a convenient way of describing what's going on for non-geometric data. The coincidence relates to the quadratic nature of both Cartesian/Euclidean space and the Central Limit Theorem/Gaussians. Viz. sigmas add up quadratically like orthogonal dimensions, which is where our ND rotational/orthogonal terminology originates by analogy with 2D and 3D space. – David M W Powers Apr 15 '14 at 14:06
>
> @DavidMWPowers Interesting. I am thinking about rotations from a linear-algebra standpoint. – isomorphismes Apr 15 '14 at 18:46

After the excellent post by JD Long in this thread, I looked for a simple example, and the R code necessary to produce the PCA and then go back to the original data. It gave me some first-hand geometric intuition, and I want to share what I got. The dataset and code can be directly copied and pasted into R form Github.

I used a data set that I found online on semiconductors here, and I trimmed it to just two dimensions - "atomic number" and "melting point" - to facilitate plotting.

> As a caveat the idea is purely illustrative of the computational process: PCA is used to reduce more than two variables to a few derived principal components, or to identify collinearity also in the case of multiple features. So it wouldn't find much application in the case of two variables, nor would there be a need to calculate eigenvectors of correlation matrices as pointed out by @amoeba.

Further, I truncated the observations from 44 to 15 to ease the task of tracking individual points. The ultimate result was a skeleton data frame (`dat1`):

```
compounds   atomic.no     melting.point
AIN         10            498.0
AIP         14            625.0
AIAs        23            1011.5
...         ...           ...
```

The "compounds" column indicate the chemical constitution of the semiconductor, and plays the role of row name.

This can be reproduced as follows (ready to copy and paste on R console):

```
dat             <- read.csv(url("http://rinterested.github.io/datasets/semiconductors"))
colnames(dat)[2] <- "atomic.no"
dat1            <- subset(dat[1:15,1:3])
row.names(dat1)  <- dat1$compounds
dat1            <- dat1[,-1]
```

The data were then scaled:

```
X <- apply(dat1, 2, function(x) (x - mean(x)) / sd(x))
# This centers data points around the mean and standardizes by dividing by SD.
# It is the equivalent to `X <- scale(dat1, center = T, scale = T)`
```

The linear algebra steps followed:

```
C <- cov(X)                                # Covariance matrix (centered data)
```

$$\begin{bmatrix} & \text{at\_no} & \text{melt\_p} \\ \text{at\_no} & 1 & 0.296 \\ \text{melt\_p} & 0.296 & 1 \end{bmatrix}$$

The correlation function `cor(dat1)` gives the same output on the non-scaled data as the function `cov(X)` on the scaled data.

```
lambda        <- eigen(C)$values            # Eigenvalues
lambda_matrix <- diag(2)*eigen(C)$values    # Eigenvalues matrix
```

$$\begin{bmatrix} \lambda_{PC1} & \lambda_{PC2} \\ 1.296422 & 0 \\ 0 & 0.7035783 \end{bmatrix}$$

```
e_vectors     <- eigen(C)$vectors           # Eigenvectors
```

Since the first eigenvector initially returns as $\sim [-0.7, -0.7]$ we choose to change it to $[0.7, 0.7]$ to make it consistent with built-in formulas through:

```
e_vectors[,1] = - e_vectors[,1]; colnames(e_vectors) <- c("PC1","PC2")
```

The resultant eigenvalues were 1.2964217 and 0.7035783. Under less minimalistic conditions, this result would have helped decide which eigenvectors to include (largest eigenvalues). For instance, the relative contribution of the first eigenvalue is 64.8%: `eigen(C)$values[1]/sum(eigen(C)$values) * 100` , meaning that it accounts for $\sim 65\%$ of the variability in the data. The variability in the direction of the second eigenvector is 35.2%. This is typically shown on a scree plot depicting the value of the eigenvalues:
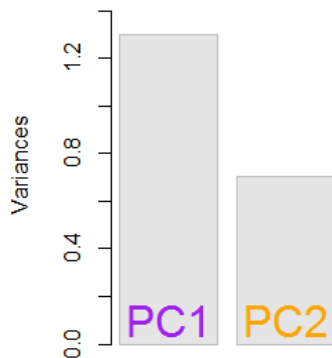
We'll include both eigenvectors given the small size of this toy data set example, understanding that excluding one of the eigenvectors would result in dimensionality reduction - the idea behind PCA.

The **score matrix** was determined as the matrix multiplication of the **scaled data** (`x`) by the **matrix of eigenvectors (or "rotations")**:

```
score_matrix <-  X %*% e_vectors
# Identical to the often found operation: t(t(e_vectors) %*% t(X))
```

The concept entails a linear *combination of each entry* (row / subject / observation / superconductor in this case) of the centered (and in this case scaled) data weighted by the *rows of each eigenvector*, so that in each of the final columns of the score matrix, we'll find a contribution from each variable (column) of the data (the entire `x`), BUT only the corresponding eigenvector will have taken part in the computation (i.e. the first eigenvector $[0.7, 0.7]^T$ will contribute to $\mathrm{PC}\,1$ (Principal Component 1) and $[0.7, -0.7]^T$ to $\mathrm{PC}\,2$, as in:

Score Matrix Generation
Starts with Centered Data and Matrix of e-Vectors of Cov M:
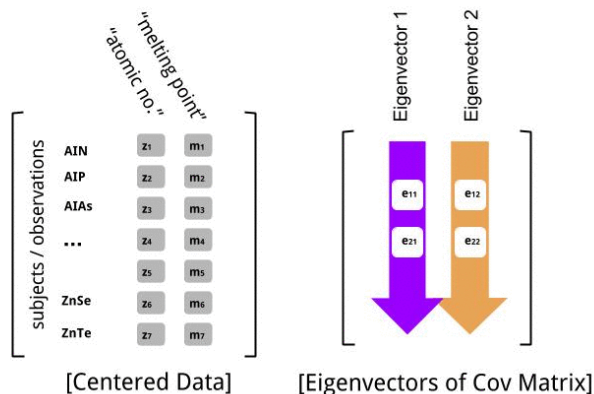
eigenvector $[0.7, -0.7]$ will change the sign of the melting point values in the linear combinations that produce PC2, whereas the effect of the first eigenvector will be consistently positive:
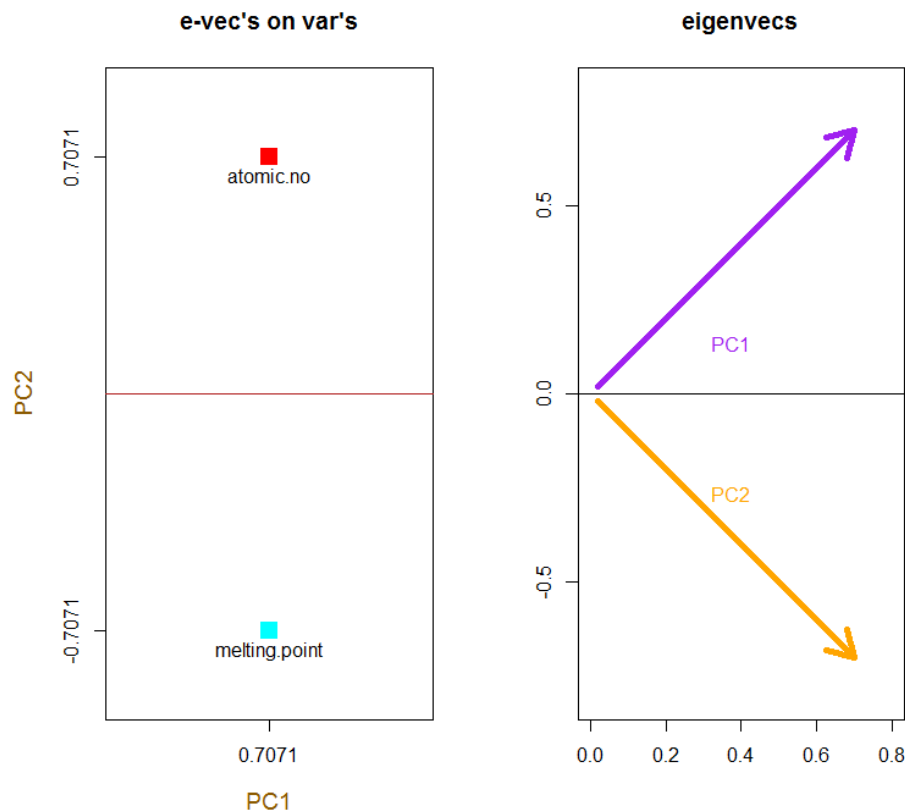
The eigenvectors are scaled to 1:

```
> apply(e_vectors, 2, function(x) sum(x^2))
PC1 PC2
  1   1
```

whereas the (**loadings**) are the eigenvectors scaled by the eigenvalues (despite the confusing terminology in the in-built R functions displayed below). Consequently, the loadings can be calculated as:

```
> e_vectors           %*% lambda_matrix
          [,1]      [,2]
[1,] 0.9167086  0.497505
[2,] 0.9167086 -0.497505

> prcomp(X)$rotation %*% diag(princomp(covmat = C)$sd^2)
                    [,1]      [,2]
atomic.no      0.9167086  0.497505
melting.point  0.9167086 -0.497505
```

It is interesting to note that the rotated data cloud (the score plot) will have variance along each component (PC) equal to the eigenvalues:

```
> apply(score_matrix, 2, function(x) var(x))
         PC1        PC2
53829.7896    110.8414
> lambda
[1] 53829.7896    110.8414
```

Utilizing the built-in functions the results can be replicated:

```
# For the SCORE MATRIX:
  prcomp(X)$x
# or...
  princomp(X)$scores # The signs of the PC 1 column will be reversed.

# and for EIGENVECTOR MATRIX:
```

```
# or...
  princomp(covmat = C)$sd^2
```

Alternatively, the singular value decomposition $(U\Sigma V^T)$ method can be applied to manually calculate PCA; in fact, this is the method used in `prcomp()`. The steps can be spelled out as:

```
svd_scaled_dat <-svd(scale(dat1))
eigen_vectors <- svd_scaled_dat$v
eigen_values <- (svd_scaled_dat$d/sqrt(nrow(dat1) - 1))^2
scores<-scale(dat1) %*% eigen_vectors
```

The result is shown below, with first, the distances from the individual points to the first eigenvector, and on a second plot, the orthogonal distances to the second eigenvector:
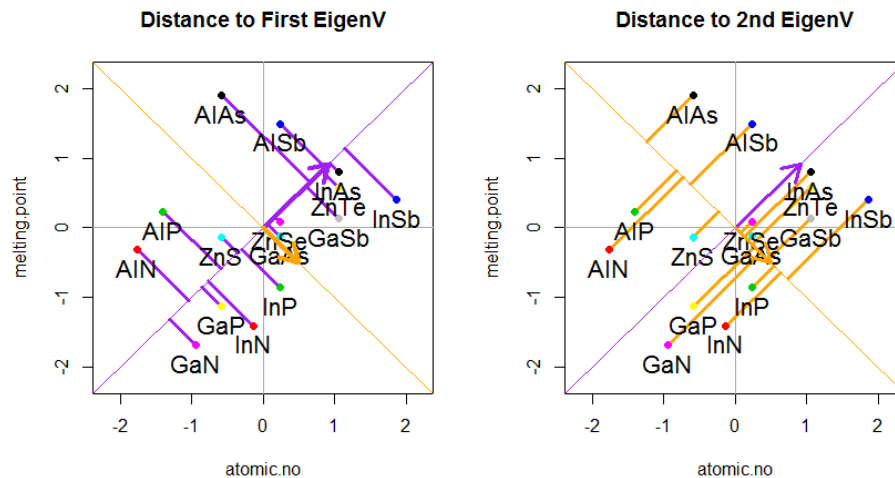

If instead we plotted the values of the score matrix (PC1 and PC2) - no longer "melting.point" and "atomic.no", but really a change of basis of the point coordinates with the eigenvectors as basis, these distances would be preserved, but would naturally become perpendicular to the xy axis:
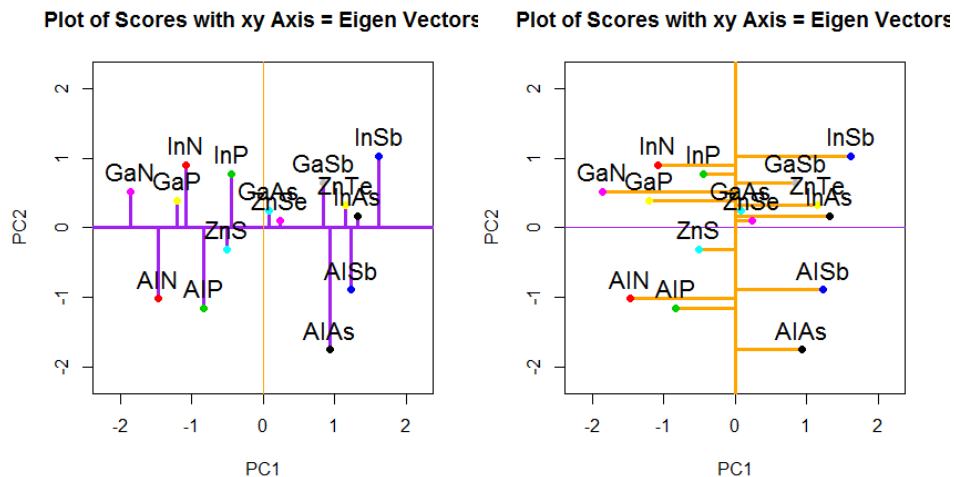
The trick was now to **recover the original data**. The points had been transformed through a simple matrix multiplication by the eigenvectors. Now the data was rotated back by multiplying by the **inverse of the matrix of eigenvectors** with a resultant marked change in the location of the data points. For instance, notice the change in pink dot "GaN" in the left upper quadrant (black circle in the left plot, below), returning to its initial position in the left lower quadrant (black circle in the right plot, below).

Now we finally had the original data restored in this "de-rotated" matrix:
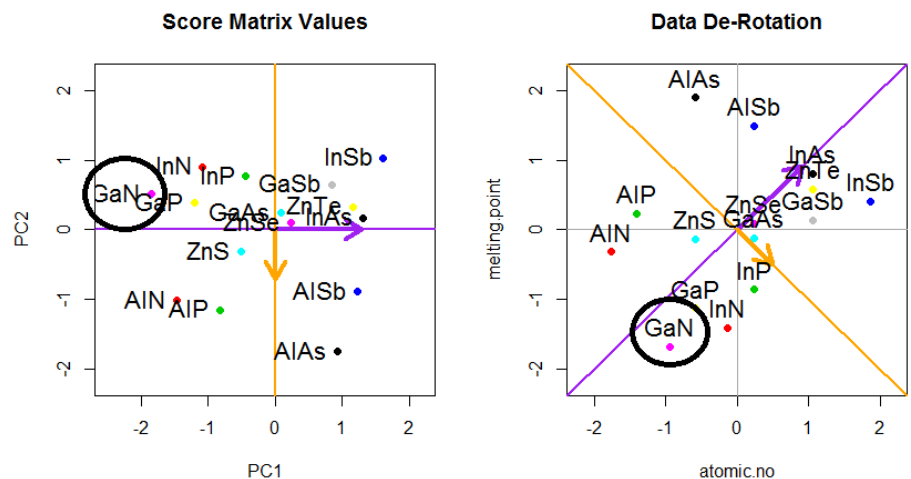
Beyond the change of coordinates of rotation of the data in PCA, the results must be interpreted, and this process tends to involve a `biplot`, on which the data points are plotted with respect to the new eigenvector coordinates, and the original variables are now superimposed as vectors. It is interesting to note the equivalence in the position of the points between the plots in the second row of rotation graphs above ("Scores with xy Axis = Eigenvectors") (to the left in the plots that follow), and the `biplot` (to the right):
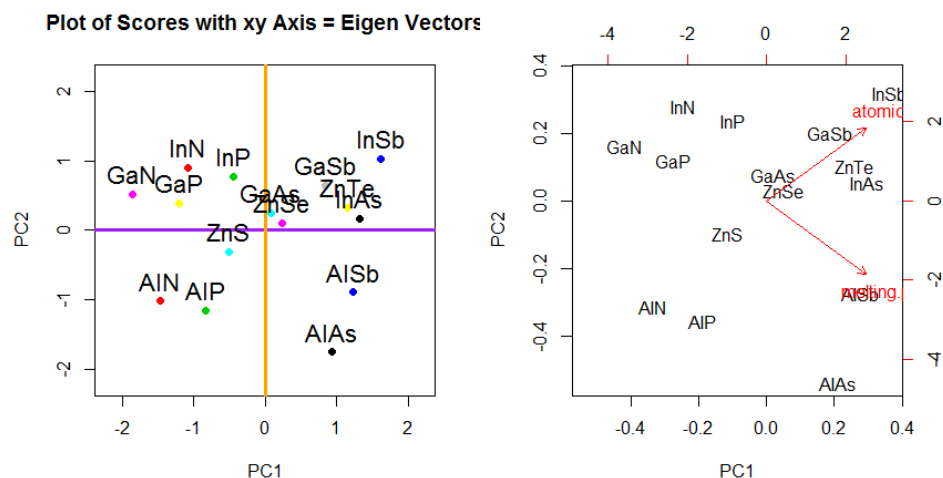
The superimposition of the original variables as red arrows offers a path to the interpretation of `PC1` as a vector in the direction (or with a positive correlation) with both `atomic no` and `melting point`; and of `PC2` as a component along increasing values of `atomic no` but negatively correlated with `melting point`, consistent with the values of the eigenvectors:

```
PCA$rotation
                    PC1        PC2
atomic.no     0.7071068  0.7071068
melting.point 0.7071068 -0.7071068
```

This interactive tutorial by Victor Powell gives immediate feedback as to the changes in the eigenvectors as the data cloud is modified.

+1 for the effort and in particular for the animation! But one should keep in mind that PCA on the correlation matrix of two variables is a bit of a special case because *all* correlation matrices of two variables have identical eigenvectors: one of them will always be [0.7 0.7] (0.7 being a 1/sqrt(2)). This is not the case for covariance matrices, or for correlation matrices in higher dimensions. – amoeba Feb 4 '16 at 14:41

From someone who has used PCA a lot (and tried to explain it to a few people as well) here's an example from my own field of neuroscience.

When we're recording from a person's scalp we do it with 64 electrodes. So, in effect we have 64 numbers in a list that represent the voltage given off by the scalp. Now since we record with microsecond precision, if we have a 1-hour experiment (often they are 4 hours) then that gives us 1e6 * 60^2 == 3,600,000,000 time points at which a voltage was recorded at each electrode so that now we have a 3,600,000,000 x 64 matrix. Since a major assumption of PCA is that your variables are correlated, it is a great technique to reduce this ridiculous amount of data to an amount that is tractable. As has been said numerous times already, the eigenvalues represent the amount of variance explained by the variables (columns). In this case an eigenvalue represents the variance in the voltage at a particular point in time contributed by a particular electrode. So now we can say, "Oh, well electrode x at time point y is what we should focus on for further analysis because that is where the most change is happening". Hope this helps. Loving those regression plots!

edited Feb 20 at 19:13   answered Jan 5 '11 at 21:11

Phillip Cloud
603   6   20

A small error in the calculation: `10e6 * 60^3` should be `10e6 * 60^2` . – Paul Aug 1 '17 at 20:05

I might be a bad person to answer this because I'm the proverbial grandmother who has had the concept explained to me and not much more, but here goes:

Suppose you have a population. A large portion of the population is dropping dead of heart attacks. You are trying to figure out what causes the heart attacks.

You have two pieces of data: height and weight.

Now, it's clear that there's SOME relationship between weight and heart attacks, but the correlation isn't really strong. There are some heavy people who have a lot of heart attacks, but some don't.

Now, you do a PCA, and it tells you that weight divided by height ('body mass') is a much more likely predictor of heart attacks then either weight or height, because, lo and behold, the "reality" is that it's body mass that causes the heart attacks.

Essentially, you do PCA because you are measuring a bunch of things and you don't really know if those are really the principal components or if there's some deeper underlying component that you didn't measure.

[Please feel free to edit this if it's completely off base. I really don't understand the concept any more deeply than this].

edited Jun 20 '13 at 10:05   community wiki
2 revs, 2 users 97%
Joel Spolsky

---

1   Welcome to the stats site @Joel! If you get a chance, you might also contribute to the discussion on our proposed distributed StackExchange data analysis project: stats.stackexchange.com/questions/2512/…. – Shane Sep 16 '10 at 2:11

5   Excellent example, but technically PCA can't find the body mass explanation as it can only find linear explanations, that is weighted sums of the original variables. However, if you take logs of your input variables, the ratio becomes a difference, and if it is the right explanation, PCA will be able to find it. – David M W Powers Jan 11 '14 at 2:50

This answer gives an intuitive and not-mathematical interpretation:

The PCA will give you a set of orthogonal vectors within a high-dimensional point cloud. The order of the vectors is determined by the information conveyed after projecting all points onto the vectors.

In different words: The first principal component vector will tell you the most about the point cloud after projecting all points onto the vector. This is an intuitve interpretation of course.

Look at this ellipsoid (follow link for a 3D model):

PCA applied to an ellipsoidically shaped point cloud
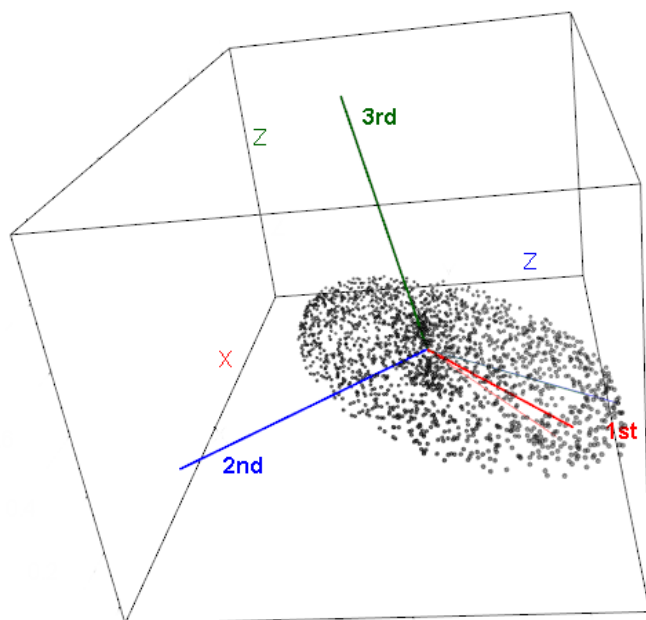
more information: www.joyofdata.de/blog/illustration-of-principal-component-analysis-pca

If you would have to choose one vector forming a one-dimensional sub-space onto which the points of the ellipsoids points will be projected. Which one would you choose because it conveys the most information about the original set in 3 dimensions?

I guess the red one along the longest axis. And this is actually the calculated 1st principal component! Which one next - I would choose the blue one along the next longest axis.

Typically you want to project a set of points from a high-dimensional space onto a two dimensional plane or into a three dimensional space.
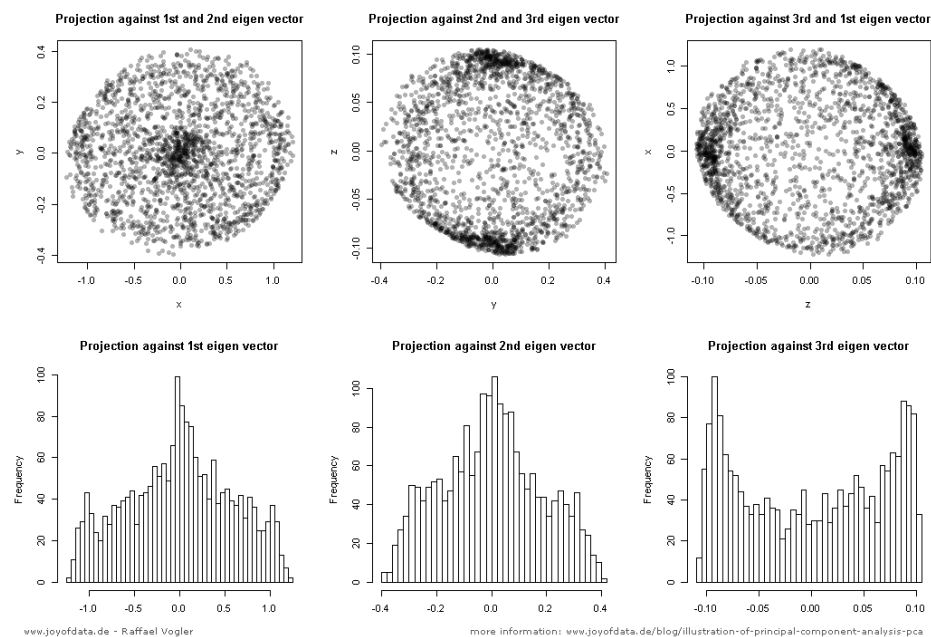
http://www.joyofdata.de/blog/illustration-of-principal-component-analysis-pca/

Although there are myriad examples given to provide an intuitive understanding of PCA, that fact can almost make it more difficult to grasp at the outset, at least it was for me.

"What was the one thing about PCA that all these different examples from different disciplines have in common??"

What helped me intuitively understand were a couple of math parallels I found, since it's apparent the maths is the easy part for you, although this doesn't help explain it to your grandmother...

Think of a regularization problem, trying to get

$$||XB - Y|| = 0$$

Or in English, break down your data Y into two other matrices which will somehow shed light on the data? If those two matrices work well, then the error between them and Y shouldn't be too much.

PCA gives you a useful factorizaton of Y, for all the reasons other people have said. It breaks the matrix of data you have, Y, down into two other useful matrices. In this case, X would be a matrix where the columns are first k PCs you kept, and B is a matrix giving you a recipe to reconstruct the columns of matrix Y using the columns of X. B is the first k rows of S, and all of V transpose. The eigenvalues on the diagonal of S basically weights which PCs are most important. That is how the math explicitly tells you which PCs are the most important: they are each weighted by their eigenvalues. Then, the matrix V^{T} tells the PCs how to combine.

I think people gave many intuitive examples, so I just wanted to share that. Seeing that helped me understand how it works. There are a world of interesting algorithms and methods which do similar things as PCA. Sparse coding is a subfield of machine learning which is all about factoring matrix A into two other useful and interesting ones that reflect patterns in A.

edited Nov 27 '14 at 12:18   answered Sep 11 '12 at 8:24
Salvador Dali   **bill_e**
546   7   16   **1,443**   1   12   25

Here is a math answer: the first principal component is the longest dimension of the data. Look at it and ask: where is the data widest? That's the first component. The next component is the perpendicular. So a cigar of data has a length and a width. It makes sense for anything that is sort of oblong.

answered Mar 20 '13 at 21:03
Peter Waksman
89   1   1

6   Unfortunately, the correctness of this answer depends on how the vague expression "longest" is interpreted. Many natural and relevant interpretations, such as the *diameter*, would be wrong. – whuber ♦ Mar 20 '13 at 21:56

PCA actually works pretty well with different types of natural way to measure the dimension/size. You just need to replace the covariance matrix with a matrix to measure "dimension" in any direction (the matrix just needs to be positive defined, or symmetrical.) This is just like QuickSort works for different ordering operator, but you will get different results for different ordering operators. – James Ll Feb 21 '14 at 0:50

The way I understand principal components is this: Data with multiple variables (height, weight, age, temperature, wavelength, percent survival, etc) can be presented in three dimensions to plot relatedness.

Now if you wanted to somehow make sense of "3D data", you might want to know which 2D planes (cross-sections) of this 3D data contain the most information for a given suite of variables. These 2D planes are the principal components, which contain a proportion of each variable.

Think of principal components as variables themselves, with composite characteristics from the original variables (this new variable could be described as being part weight, part height, part age, etc). When you plot one principal component (X) against another (Y), what you're doing is building a 2D map that can geometrically describe correlations between original variables. Now the useful part: since each subject (observation) being compared is associated with values for each variable, the subjects (observations) are also found somewhere on this X Y map. Their location is based on the relative contributions of each underlying variable (i.e. one observation may be heavily affected by age and temperature, while another one may be more affected by height and weight). This map graphically shows us the similarities and differences between subjects and explains these similarities/differences in terms of which variables are characterizing them the most.

Here's one for Grandma:

In our town there are streets going north and south, some going east and west, and even some going northwest and southeast, some NE to SW. One day a guy measures all the traffic on all the streets, he finds that the most traffic is going diagonally, from northwest to southeast, the second biggest is perpendicular to this going northeast to southwest and all the rest is fairly small. So he draws a big square and puts a big line left to right and says that is the NW to SE, then draws another line vertically up and down through the middle. He says that's the second most crowded direction for traffic (NE to SW). The rest is small so it can be ignored.

The left right line is the first eigenvector and the up down line is the second eigenvector. The total number of cars going left and right are the first eigenvalue and those going up and down are the second eigenvalue.

answered Nov 6 '13 at 0:22

BajaBob
97 1 1

---

1 This analogy appears to break down under examination. What if the largest and second largest traffic direction are not orthogonal? How does your analogy help us understand a PCA in such a case? – whuber ♦ Nov 6 '13 at 4:20

I guess grandma understand what orthogonal means? Yes, some flaws there but it's a start. I think it's great that there have been so many answers here. – BajaBob Nov 7 '13 at 1:32

2 Whether or not "grandma" understands a post, it needs to be reasonably clear and correct. Your analogy does not appear to accomplish either of those aims. That may be because I do not understand the analogy: I cannot connect it to what PCA is or does. Perhaps you could clarify how the analogy works so that other readers do not become as mystified as I am. – whuber ♦ Nov 7 '13 at 17:34

The fact that they are not orthogonal means you need ICA or FA not PCA. If grandma was watching Star Trek (yes she's that generation) when they show the disabled ship at an angle - PCA would tend to recover the reference plane relevant to the scale and view (the galactic plane or the ship's axes). – David M W Powers Apr 15 '14 at 13:47

-1. I agree with @whuber that this analogy does not work. What is supposed to be the data here, what covariance matrix are these "eigenvectors" of? I don't understand it at all. – amoeba Mar 6 '15 at 23:41

I view PCA as a geometric tool. If you are given a bunch of points in 3-space which are pretty much all on a straight line, and you want to figure out the equation of that line, you get it via PCA (take the first component). If you have a bunch of points in 3-space which are mostly planar, and want to discover the equation of that plane, do it via PCA (take the least significant component vector and that should be normal to the plane).

answered Sep 16 '10 at 5:15

shabbychef
7,000 7 35 85

Why so eigenvalues/eigenvectors ?

When doing PCA, you want to compute some orthogonal basis by maximizing the projected variance on each basis vector.

Having computed previous basis vectors, you want the next one to be:

- orthogonal to the previous
- norm 1
- maximizing projected variance, i.e with maximal covariance norm

This is a constrained optimization problem, and the Lagrange multipliers (here's for the geometric intuition, see wikipedia page) tell you that the gradients of the objective (projected variance) and the constraint (unit norm) should be "parallel" at the optimium.

This is the same as saying that the next basis vector should be an eigenvector of the covariance matrix. The best choice at each step is to pick the one with the largest eigenvalue among the remaining ones.

edited Jan 15 '11 at 12:31   answered Jan 15 '11 at 12:25
                                                              user2792

I'll give a non-mathy response and a more detailed birds-eye view of the motivation-through-math in the second part.

Non-Mathy:

The non-math explanation is that PCA helps for high dimensional data by letting you see in which directions your data has the most variance. These directions are the **principal components**. Once you have this information you can then, in some cases, decide to use the principal components as the meaningful variables themselves, and vastly reduce the dimensionality of your data by only keeping the principal components with the most variance (*explanatory power*).

For example, suppose you give out a political polling questionnaire with 30 questions, each can be given a response of 1 (*strongly disagree*) through 5 (*strongly agree*). You get tons of responses and now you have 30-dimensional data and you can't make heads or tails out of it. Then in desperation you think to run PCA and discover the 90% of your variance comes from one direction, and that direction does not correspond to any of your axis. After further inspection of the data you then conclude that this new hybrid axis corresponds to the political left-right spectrum i.e. democrat/republican spectrum, and go on to look at the more subtle aspects in the data.

Mathy:

It sometimes helps to zoom out and look at the mathematical motivation to shed some light on the meaning.

There is a special family of matrices which can be transformed into **diagonal** matrices simply by changing your coordinate axis. Naturally, they are called the **diagonalizeable** matrices and elegantly enough, the new coordinate axis that are needed to do this are indeed the eigenvectors.

As it turns out the **covariance matrix** are symmetric and will always be *diagonalizeable*! In this case the eigenvectors are called the **principal components** and when you write out the covariance matrix in eigenvector coordinates, the diagonal entries (the only ones left) correspond to the variance in the direction of your eigenvectors. This allows us to know which directions have the most variance. Moreover since the covariance matrix is diagonal in these coordinates, you have cleverly eliminated all correlation between your variables.

As is common in practical applications, we assume that our variables are normally distributed and so its quite natural to try and change our coordinates to see the simplest picture. By knowing your principal components and their respective eigenvalues (variance) you'll be able to reduce the dimensionality of your data if needed and also have a quick general summary of where the variation in your data lies.

But at the end of the day, the root of all this desirability comes from the fact that diagonal matrices are *way* easier to deal with in comparison to their messier, more general cousins.

edited Jul 15 '13 at 22:50

answered Jul 15 '13 at 19:37

Christian Bueno
278   2   7

---

2   Thank you for your contribution. It seems to address an unnecessarily narrow interpretation of PCA, however. (1) PCA has been fruitfully applied to highly non-Gaussian datasets. (2) PCA is not a formal parametric procedure; it perhaps is better to think of it as exploratory in spirit. (3) *All* covariance matrices, of any kind of multivariate distribution or data, are diagonalizable. Neither Gaussianity (Normality) nor non-degeneracy are requirements. (Symmetry of the matrix and having real components guarantee diagonalizability.) – whuber ♦ Jul 15 '13 at 21:04

I must admit, I'm embarrassed to have forgotten, but good point about covariance matrices being diagonalizable in general. I'm going to edit to reflect that. Also, could you elaborate on point (2)? I'm not familiar with the difference between parametric or non-parametric procedures. – Christian Bueno Jul 15 '13 at 22:50

Basically PCA finds new variables which are linear combinations of the original variables such that in the new space, the data has fewer dimensions. Think of a data set consisting of the points in 3 dimensions on the surface of a flat plate held up at an angle. In the original x, y, z axes you need 3 dimensions to represent the data, but with the right linear transformation, you only need 2.

Basically what @Joel said, but only linear combinations of the input variables.

answered Sep 16 '10 at 4:10

Shlomo Argamon
69   1

large sized model to a very small sized model.

For example, instead of constructing first the full sized bridge and then carrying out experiments and tests on it, it is possible to use EVs to create a very small sized bridge where all the factors/quantities will be reduced by the same margin and moreover the actual result of tests and stress related tests carried out on it can be calculated and enlarged appropriately as needed for the original model. **In a way EVs help to create abstracts of the original**.

To me, this explaination had profound meaning to what I was trying to do! Hope it helps you too!

edited Sep 11 '12 at 13:57   answered Sep 11 '12 at 7:28
Andy W   Rorschach
13.1k   2   45   155   **177**   1   2

-1. Perhaps I did not fully appreciate the analogy, but it looks pretty misleading to me. PCA does indeed (in a way) allow to "convert" a "large" model to a "small" model, but it does so by reducing dimensionality of the dataset. But how is the small bridge of lower dimensionality than the large one?! They are both 3D, aren't they. – amoeba Mar 6 '15 at 23:28

@amoeba : this extract is out of a paper I read, these are not exactly my words. I havn't studied this topic since a long time now and I have lost trace. – Rorschach Mar 7 '15 at 5:38

Imagine grandma has just taken her first photos and movies on the digital camera you gave her for Christmas, unfortunately she drops her right hand as she pushes down on the button for photos, and she shakes quite a bit during the movies too. She notices that the people, trees, fences, buildings, doorways, furniture, etc. aren't straight up and down, aren't vertical, and that the floor, the ground, the sea, the horizon isn't well horizontal, and well the movies are rather shaky as well. She asks if you can you help her fix them, all 3000 holiday photos and about 100 videos at home and beach (she's Australian), opening presents, walking in the country. She's got this photo software that allows you to do that she says. You tell her that that would take days, and won't work on the videos anyway, but you know techniques called PCA and ICA that might help. You explain that your research actually involves just this kind of rotation of data into the natural dimensions, that these techniques find the most important directions in the data, the photo in this case, and rotate so the most important one is horizontal, the second one is vertical (and it can even go on for more dimensions we can't imagine very well, although time is also a dimension in the movies).

--

Technical Aside. In fact, you could probably earn your PhD doing this for her, and there is an important paper by Bell and Sejnowski (1997) about independent components of images corresponding to edges. To relate this to PCA: ICA uses PCA or SVD as a first step to reduce the dimensionality and initial approximations, but then improves them that takes into account not only second order error (SSE) like PCA, but high order errors - if it's true ICA, all higher orders, although many algorithms confine themselves to 3rd or 4th. The low order PCA components do tend to be influenced strongly by the horizontals and verticals. Dealing with camera motion for the movies can also make use of PCA/ICA. Both for the 2D photos and the 2½D movies you need a couple of representational tricks to achieve this.

Another application you could explain to grandma is eigenfaces - higher order eigenvectors can approximate the '7 basic emotions' (the average face for each of them and the 'scaled rotation' or linear combination to do that averaging), but often we find components that are sex and race related, and some might distinguish individuals or individual features (glasses, beard, etc.). This is what happens if you have few photos of any one individual and many emotions/expressions, but you get a different bias if you have many faces with neutral expressions. Using ICA instead of PCA doesn't really seem to help much for basic emotions, but Bartlett and Sejnowsiki (1997) showed it found useful features for face recognition.

edited Jan 11 '14 at 5:26   answered Jan 11 '14 at 3:43
David M W Powers
749   6   9

---

1   I appreciate the effort to communicate with examples and by analogy. The use of images, though, is unfortunate because of the high likelihood grandma will not understand that your sense of "rotate" has little to do with actually rotating the axes of an *image*, nor is she likely to understand that you are using "dimension" in an abstract sense in which the photos have millions of dimensions and not just two. – whuber ♦ Jan 11 '14 at 15:02

Yes, you need to represent it as a point cloud, as with the images in other answers. Preprocessing with some form of edge detection and/or thresholding would be likely part of the "tricks" I mentioned. But to operate on a complex photo requires a PhD's worth of tricks. – David M W Powers Jan 12 '14 at 8:58

@whuber I have actually used PCA (well SVD) to find these rotations in doing stereo image calibration! It definitely is the same sense of rotate. – David M W Powers Apr 15 '14 at 13:49

tool that is used to address the real issue at hand which is a wrong coordinate system. I'll explain.

Let's start with a line. What is a line? It's a one dimensional object. So, you need only one dimension to move from one point to another. On a plane though you attach two coordinate any point of a line. That is because with respect to a line itself the coordinate system is chosen arbitrarily. The coordinate system, I would argue, does not reflect the inner one dimensional nature of the line. If only I would always put the origin of my Cartesian coordinate system on the line, and turned it so that its x-axis was on the line, then I would not need y-axis anymore! All my points are on one axis, because a line is a one dimensional object.

That's where PCA explanations should start. The eigen problem is a tool that does the rotation which I described, and de-meaning of variables puts the origin onto the line. PCA helps reveal true dimensions of the data **so long the relationships between the variables are linear**.

answered Feb 20 at 19:51

Aksakal
35k   3   43   96

Remember that an eigenvector is a vector whose transform is parallel to the same input vector. Thus an eigenvector with a high eigenvalue means that the eigenvector has a high degree of 'parallelity' to the data, meaning that you can represent the data with this vector only and expect a low error in the new representation. If you pick additional eigenvectors with lower eigenvalues, you will be able to represent more details of the data because you'll be representing other 'parallelities' - which are not as prominent as the first one because of lower eigenvalues.

answered Dec 8 '15 at 16:17

felipeduque
564   3   15

Perhaps late in these analyses is the implicit assumption that data from group I is different from group II and one is trying to find which component is likely to be the major contributing factor to the difference.

Performing a PCA analysis that results in identical ellipsoids for 2 different sets then tells you that the two sets are not different by any of the parameters you have measured.

answered May 31 '14 at 12:26

jcourtright
7

2   I wonder if you are thinking of MANOVA. If you ran two sepa4ate PCAs, you would only show that the correlation al structure was similar. – gung ♦ May 31 '14 at 13:48

1   -1. This is a completely misleading answer, for the reason outlined by @gung. Also, it does not even attempt to explain what PCA is. – amoeba Mar 6 '15 at 23:16

---

protected by whuber ♦ Nov 9 '14 at 17:11

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?