Quantitative Understanding in Biology Principal Components Analysis

Introduction

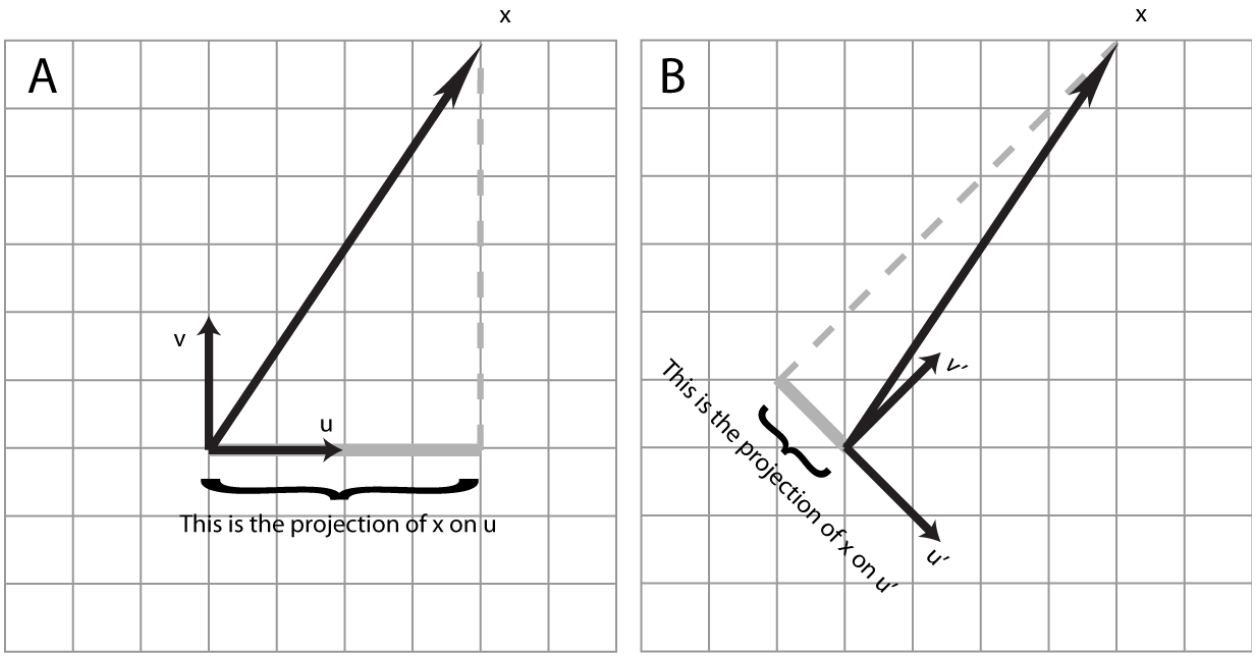
Throughout this course we have seen examples of complex mathematical phenomena being represented as linear combinations of simpler phenomena. For example, the solution to a set of ordinary differential equations is expressed as a linear combination of exponential terms, with the weights given to each term determined by the eigenvectors and the initial conditions of the problem.

Linear combinations are especially useful because they allow us to apply the principle of superposition. This means we can decompose a complex system into a number of simpler systems. If we can understand the simpler systems, we can understand the complex system as a sum of the simpler ones. This idea is pervasive throughout mathematics, and we will be using this idea again today. But before we venture into new territory, we'll formalize some probably already familiar notions about this idea of linear combinations.

Projections of Vectors

When we are working in 2D Cartesian space, we represent points as a pair of numbers, such as (2, 3). This is so natural that we often don't give it much thought, but in fact we are representing our vector as a linear combination of simpler entities. When we write (2, 3), at some level this is just a pair of numbers. Implicit in this representation, though, is the idea that the vector of interest is 2 parts $\vec{u}$ and three parts $\vec{v}$, where $\vec{u}$ and $\vec{v}$ are unit vectors in the x and y directions, respectively. So when we see (2, 3), we think of $2\vec{u} + 3\vec{v}$. Hopefully, this does not come as a shock to anyone.

One way of thinking about arbitrary vectors and their relationship to $\vec{u}$ and $\vec{v}$ is in terms of *projections*. A projection is the mathematical equivalent of casting a shadow. If we think about the vector (2, 3), its projection on the x-axis (i.e., the shadow it would cast on the x-axis) is of length two. Similarly, its shadow cast on the y-axis is of length three. Note that to 'cast a shadow', we 'shine light' perpendicular to the axis upon which we want the shadow to cast. This is represented graphically in panel A of the figure below.



Although the choice of $\vec{u}$ and $\vec{v}$ is quite natural, it is somewhat arbitrary. We might have chosen $\vec{u}'$ and $\vec{v}'$, as shown in panel B. In this case, the projection of our vector on $\vec{u}'$ is negative and relatively small in magnitude, whereas the projection of our vector on $\vec{v}'$ is relatively large. We see that we can represent a vector of interest in terms of linear combinations of $\vec{u}'$ and $\vec{v}'$. You might want to do this if you were, say, solving a problem regarding an object sitting on a 45˚ incline, so the coefficient of $\vec{u}'$ represents its position on the surface and the coefficient of $\vec{v}'$ represents its 'altitude' (memories of physics problems should be resurfacing now). Of course, if you want to do this formally, we will need to be a bit more quantitative.

The mathematical way to compute a projection is to take the inner product, also known as the dot product, of the vector of interest with the so-called basis vector. In the example in the figure, we have…

$$\vec{x} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \quad \vec{u}' = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} \quad \vec{x} \cdot \vec{u}' = \frac{2}{\sqrt{2}} - \frac{3}{\sqrt{2}} = -\frac{1}{\sqrt{2}}$$

$$\vec{x} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \quad \vec{v}' = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \quad \vec{x} \cdot \vec{v}' = \frac{2}{\sqrt{2}} + \frac{3}{\sqrt{2}} = \frac{5}{\sqrt{2}}$$

Basis Sets in Cartesian Space

The set of vectors we use in linear combinations to express an arbitrary vector is called a *basis set*. A good basis set has three important properties:

- members of the basis set are orthogonal
- members of the basis are normalized
- some linear combination of the members of the basis set can represent any arbitrary object in the space of interest

Let's review each of these properties in turn. We'll begin with the property of orthogonality. This means that the projection of each element of the basis set on all of the others should be zero. For the classic Cartesian basis set, we have…

$$\vec{u} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \vec{v} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \vec{u} \cdot \vec{v} = (1)(0) + (0)(1) = 0$$

…and for our rotated coordinate system we have…

$$\vec{u}' = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} \quad \vec{v}' = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \quad \vec{u}' \cdot \vec{v}' = \left(\frac{1}{\sqrt{2}}\right)\left(\frac{1}{\sqrt{2}}\right) - \left(\frac{1}{\sqrt{2}}\right)\left(\frac{1}{\sqrt{2}}\right) = 0$$

We see that both basis sets are orthogonal. Note that the eigenvectors of an arbitrary matrix are not necessarily orthogonal. This does not mean you can't use them as a basis set, it is just that they lack a certain convenience.

Next we consider the property of normality. In a normalized basis set, the projection of each element of the basis set on itself will be unity. Again, we consider the classic Cartesian basis set…

$$\vec{u} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \vec{v} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \vec{u} \cdot \vec{u} = (1)(1) + (0)(0) = 1 \quad \vec{v} \cdot \vec{v} = (0)(0) + (1)(1) = 1$$

…and our rotated system…

$$\vec{u}' = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} \quad \vec{v}' = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \quad \vec{u}' \cdot \vec{u}' = \left(\frac{1}{\sqrt{2}}\right)\left(\frac{1}{\sqrt{2}}\right) + \left(-\frac{1}{\sqrt{2}}\right)\left(-\frac{1}{\sqrt{2}}\right) = \frac{1}{2} + \frac{1}{2} = 1$$

$$\vec{v}' \cdot \vec{v}' = \left(\frac{1}{\sqrt{2}}\right)\left(\frac{1}{\sqrt{2}}\right) + \left(\frac{1}{\sqrt{2}}\right)\left(\frac{1}{\sqrt{2}}\right) = \frac{1}{2} + \frac{1}{2} = 1$$

Again, both basis sets are normalized. Normalizing a basis set is usually pretty easy, as all you have to do is multiply each element of the set by a constant so the inner products work out to unity. For example, if

we had chosen $\vec{u}'$ = (1, 1) and $\vec{v}' = (1, -1)$, we would have found that $\vec{u}' \cdot \vec{u}'$ = 2; and we would have corrected this by multiplying by $1/\sqrt{2}$ (and similarly for $\vec{v}'$).

The last property is completeness. It is easy to see that as long as the two basis vectors we are considering are linearly independent they will be sufficient to represent any 2D vector. However, if we were interested in 3D space, then two vectors wouldn't cut it. The projection of a 3D vector onto two 2D vectors gives us the best possible representation given the incomplete basis set, but it is not quite good enough for a full representation of the data in the original 3D vector. We lose information in such a projection.

Your hand casting a shadow on a wall is an example of an incomplete projection. Every point that makes up your hand is mapped to a 2D space (the surface upon which your shadow is cast). Given a shadow, you cannot reproduce a whole hand. You can choose different incomplete basis sets by moving the light source (or alternatively rotating your hand); you will get different projections, but none will be complete.

Basis sets that satisfy all three conditions are called *complete orthonormal basis sets*. We should note that the terminology we are using here is that commonly used by chemists. A mathematician would say that a basis is, by definition, complete. While the terminology may change from domain to domain, the ideas don't.

Principal Components Analysis

Sometimes using an incomplete basis set can be advantageous. When dealing with large datasets, we often seek to reduce the amount of data we deal with. Say you've collected three variables per observation in an experiment, and you're looking to reduce the amount of data you deal with. You could choose to ignore one of these variables (say the third: the z-value). This is equivalent to projecting your 3D (x,y,z) dataset onto the 2D (x,y) plane. Of course, you could eliminate any of the other variables, by projecting onto the (x,z) or (y,z) planes.

Once you start thinking of reducing your data as projecting high-dimensional vectors onto a lower dimensional plane, it becomes natural to think about projecting onto planes that don't correspond to the coordinate axes. Maybe, instead of dropping one or another variable altogether, we can find a 2D plane upon which we project our 3D data that results in a minimal loss of information. This is the essence of principal components analysis, or PCA.

To understand how PCA works, we need to recall the concepts of variance and correlation. Recall that the variance of a sample is given by

$$s^2 = \frac{\sum(x_i - \bar{x})^2}{n - 1}$$

We can define the covariance between two variables, x and y, as…

$$cov(x,y) = \frac{\sum\{(x_i - \bar{x})(y_i - \bar{y})\}}{n-1}$$

This is a lot like the Pearson correlation coefficient (Pearson invented PCA). Recall that when x varies with y, this expression will tend to accumulate positive terms. When x and y are independent, the covariance will tend to zero, and when x and y are anti-correlated, a negative quantity will result. Note that the variance of a variable is just the covariance of that variable with itself.

In higher-dimensional spaces, we can compute the covariance of every variable with every other variable. These values can be arranged in a square matrix, the size of which will be the number of dimensions of the original data. The diagonal terms will be the variances of the individual variables. The off diagonal terms will be near zero if the underlying variables are uncorrelated, and will be large if there is significant correlation.

We can demonstrate this idea in Matlab. We begin be creating a dataset, a, with two independent (uncorrelated) values of 500 observations…

```
>> n=500;
>> a(:,1) = normrnd(0,1,n,1);
>> a(:,2) = normrnd(0,1,n,1);
>> plot(a(:,1),a(:,2),'.')
>> cov(a)

ans =

    1.0255   -0.0057
   -0.0057    1.0072
```

Now, we synthesize a new dataset that has significant correlation. Note the values in the second column are an average of the value in the first column and a random component.

```
>> b(:,1) = normrnd(0,1,n,1);
>> b(:,2) = 0.5*b(:,1) + 0.5*normrnd(0,1,n,1);
>> plot(b(:,1),b(:,2),'.')
>> cov(b)

ans =

    0.9598    0.5038
    0.5038    0.5300
```

We see that the covariance matrix has strong off-diagonal terms.

Now for the magic of PCA. We want to know which line we can project our data onto to preserve the maximum variation in our data. It turns out that the eigenvectors and the eigenvalues of the covariance matrix give us the answer!

```
>> [V,D] = eig(cov(b))
```



```
V =

    0.5512   -0.8344
   -0.8344   -0.5512


D =

    0.1972         0
         0    1.2927
```

This tells us that the first principal component (the one with the largest eigenvector) is in the direction (0.83, 0.55). We can add that vector to our plot:

```
>> plot(b(:,1),b(:,2),'.')
>> hold on
>> V1 = V(:,2)

V1 =

   -0.8344
   -0.5512

>> plot([-V1(1), V1(1)],[-V1(2), V1(2)], 'k', , 'linewidth', 3)
```

Projecting all of our points onto this line will give us the projection with the largest amount of variation. In the context of data reduction, this means that if you were forced for describe each point in our data set by only one number instead of two, the best job you could do would be to describe it as the position along the first principal component.

Plotting the second principal component follows similar mechanics…

```
>> V2 = V(:,1);
>> plot([-V2(1), V2(1)], [-V2(2), V2(2)], 'k', 'linewidth', 3)
```

You can see here that the second principal component is orthogonal to the first. This is a general property of principal components that extends to higher dimensions; they always orthogonal to each other.

In the two dimensional case, the second principal component is trivially determined by the first component and the property of orthogonality. In higher dimensions, the second principal component turns out to be vector that is both orthogonal to the first component and has the highest variance when the original data is projected onto it. Again, if you were forced to express all the points in a three-dimension dataset with just two numbers each, the best job you could do would be to given the coordinates of those points along the first two principal components.

All of these ideas extend to higher dimensional data, and are most commonly applied to very high dimensional data. For example, in high-throughput gene expression studies, you often obtain data in spaces with 10,000 to 40,000 dimensions (the levels of expression of each gene measured).

Matlab will do all of this work, and the translation to PC coordinates, in one command (including ordering the eigenvalues)…

```
>> [pcs, trans, evs] = princomp(b);
```

The first result contains the principal components, the second contains the transformed values, and the third contains the eigenvalues (in order).

Interpreting the PCA Results

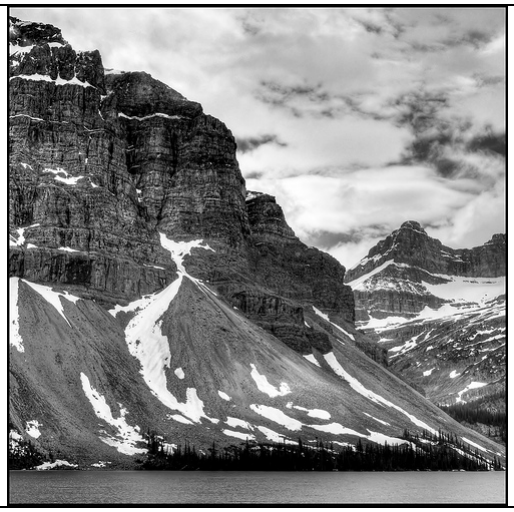
We mentioned that PCA is often used as a tool for data reduction. After applying PCA to a dataset, we typically only consider the few principal components in our subsequent analysis. In the case of gene expression studies, we may reduce our 30,000 dimensional data down to a projection of that data onto the first, say, hundred principal components.

One question that should naturally occur to you then is how to decide how many principal components to keep. Is 50 enough? Or maybe one needs to keep 200 components? It turns out that the eigenvalues of the covariance matrix can help answer this question. One interesting property of the eigenvalues is that the fraction of variation in the original data that is embedded in any particular principal component is simply the ratio of the eigenvalue associated with that component divided by the sum of all of the eigenvalues. In our example above, by retaining the first of the two principal components (really our only choice for data reduction), we would be preserving 1.2927/(1.2927+0.1972) = 87% of the variation in the data while retaining only 50% of the data.

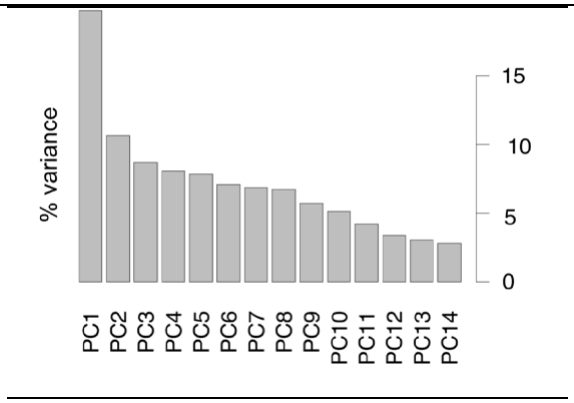
When dealing with real data sets that have more than two dimensions, there are often two practices that are carried out. For data sets with a handful of dimensions (say eight or twenty), one typically retains the first two or three PCs. This is usually motivated by simple expediency with regard to plotting. Since you can make plots in two or three dimensions, you can visually explore your data using common tools. The evaluation of how much variance is retained in the original dataset informs how well you can expect your downstream will perform. Ideally, you hope to capture around 90% of the variance in these first two or three components.

In very high dimensional datasets, you typically will need much many than two or three PCs to get close to capturing 90% of the variation. Often, your algorithm will be to retain as many PCs as needed to get the sum of the eigenvalues of the PCs you are keeping to be 90% of the sum of all of them.

An alternative is to make a so-called scree plot. This is simply a bar blot of each eigenvalue in descending order, and (hopefully) will be reminiscent of the geological feature after which it is named. Such a plot may help you evaluate where the point of diminishing returns lies. The hope here is that the first few PCs represent real (reduced dimension) structure in the data, and the others are just noise.



A scree (geological formation)
(CC) Evan Leeson

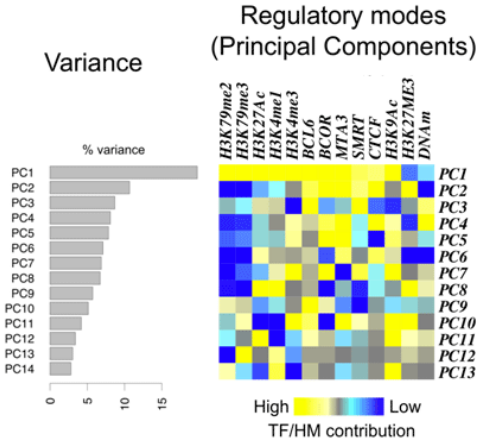


A scree plot (PCA analysis) from Dr. Elemento's lecture.

In addition to data reduction, analysis in principal components can sometimes by informative. Suppose, for example, you were studying heart disease and its relationship to obesity, and had collected the height and weight of a number of patients. Since weight tends to correlate with height, you might perform a PCA, and test to see if any of the principal components correlate with heart disease. (Typically one uses BMI which is proportional to (weight/height^2).

Further interpretation of PCA results

After performing PCA, it is very tempting to try to interpret the weights (or loadings) of the principal components. Since each principal component is a linear combination of the original variables, it can be of interest to see which variable contribute most to the most important principal components. For example, when looking at gene signatures, those genes that contribute most to the first few PCs may be considered 'more important' genes. We saw an example of interpreting the loadings of PCs in Dr. Elemento's lecture (the relevant figure is reproduced below).



While the PC loadings can sometimes be the source of useful hints about the underlying natural variables of a biological process, one "needs to be more than usually circumspect when interpreting" the loadings (Crawley, The R Book, 2007). Part of this derives from the fact that PCA results are sensitive to scaling (the lab in the section of the course explores this), and part of this may be that individual PCs may be very sensitive to noise in the data. When performing PCA on a computer, make sure you know what your program does in terms of scaling (as well as with the underlying numerics). In particular, you should be aware that R has both a `prcomp` and a `princomp` function, and scaling and mean centering is specified as options in the (preferred) `prcomp` implementation .

As scientists, we need as much help as we can to interpret our data, so to say that one should never look at or thing about loadings would be impractical. A good rule of thumb may be to be sure to treat any interpretations about loading to be a hypothesis that needs to be validated by a completely independent means. This is definitely one of those areas where you want to be conservative.

