Vectors and Loadings and Scores, Oh my!

Eli T. Brown

There have been repeated questions on a handful of things and there are some spots where I think clarification could be really helpful. In particular, there has been confusion about coefficients versus loadings, about matrix algebra and about eigenvectors. Here are some notes that I hope will dispel the confusion.

1 Loadings and Scores

Thanks to the fact that PCA and Factor Analysis are used in a variety of fields and implemented by a variety of packages, there is confusion about terms. In fact, as an example, even with something so basic as the fact that vectors are usually considered column vectors, MATLAB splits out eigenvectors as rows[1].

1.1 Latent Variables, Coefficients, and Loadings

The principal components or canonical variates are latent variables, i.e. they are variables but they are never measured directly (never observed). The latent variables' values can only be calculated from the real variables that combine to create them. Those "combinations" of real variables that form the latent ones are actual *linear combinations* (see Section 3 for an example). Specifically, the latent variables are a weighted sum of the real variables, with the weights being specified by the *coefficients* of the original variables. Conveniently, this is equivalent to the dot product of the vector of coefficients with a vector of sample data, also viewed as a projection of vector onto the other. These coefficients are the core part of the "model" that we learn with these algorithms. The *standardized* versions of these are normalized so that they are proportional to the standard deviation of the variable. That transformation makes it easier to compare the coefficients to each other.

One important thing about the linear combination is that it adds up a multiple of each variable's value and thus results in a single value. That formula gives the *score* of the latent variable. An actual score value applies to a single data point (a.k.a. instance, observation). The score is the value that latent variable would have for that observation, if it could have been sampled. Specifically, calculating it requires filling in the actual value from the observation for each variable in the linear combination equation. In PCA, there is sometimes another use of the word score. The variables' coefficients in the principal components are

[1] This maybe be because of what we see in Section 2.2.

sometimes called the scores, e.g. in SPSS, one of the outputs is the "component score coefficient matrix"

Since we can calculate a score value for each data point in each latent variable, we can then ask for the sample correlation between that variable and any of our others. These correlation values between the latent variables and real values are called *loadings*. The loadings are always correlations as opposed to coefficients.

Confusingly, with PCA (and Factor Analysis), multiple things get plotted. You can plot the variable relationships by their coefficients or by their loadings on the principal components. You can also plot the data points' scores or loadings on the PCs. We have used the scores of data points on principal components as our scatterplot axes for the outputs of PCA.

1.2 Covariance and Correlation

Remember that correlation is just a normalized covariance. If it's been a while since you've thought about this core statistics fact, it's helpful to remind start from the beginning and remember how covariance and correlation are calculated. While sample variance measures the average of the square differences from the mean of a sample ($E[(\mathbf{x}-\bar{\mathbf{x}})^2]$), covariance does that for two different variables rather than x with itself ($E[(\mathbf{x}-\bar{\mathbf{x}})(\mathbf{y}-\bar{\mathbf{y}})]$). As we walk through the sum to calculate the expected value, we will look at each sample, multiplying the extent to which $\mathbf{x}$ and $\mathbf{y}$ vary from their means. When the two vary together (are relatively high or low on the same samples), we will conclude they have strong covariance. The correlation will simply rescale by dividing by the standard deviations so that our values fall between -1 and 1.

2 Matrix Algebra

2.1 Why are we doing this again?

I had a couple people bring back the question of why we are working through so much math in lecture. I understand that not everyone will find that easy to understand, and you'll find that the most complicated parts of it are not needed for the homework or exams. The reason why it's important to present anyway is several fold:

- When you come across new methods, especially cutting edge methods you're considering using for a project, they will likely be presented with this mathematical formulation. This is the way new methods are discussed in the statistics and machine learning communities.
- In the same vein, if you someday decide you want to modify some algorithm to fit with some interesting data, it's important to have exposure to the details of these algorithms so you have a deeper idea of how to change them.

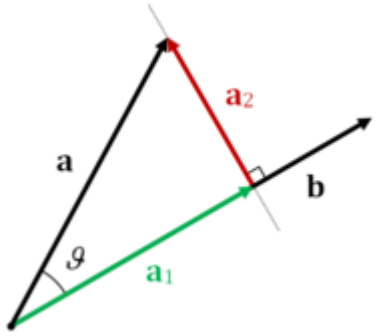

Fig. 1: This shows the dot product of **a** and **b**. The new value a_1 is the projection of **a** onto **b** and is calculated by their dot product. [image credit to Wikipedia]

We're mostly concerned with how to use them in this course, but learning how they work and how they're written in matrix notations will allow you to put them in other contexts that will arise.

2.2 Dot Products and Matrix Multiplication

If x and y are column vectors and we want to multiply them, so far in this course we're looking at the *dot product*, also called the *inner product*. This method for multiplying vectors produces a single number which shows how much they have in common, specifically the projection of one onto the other. This relationship is demonstrated in Figure 1.

The dot product of two vectors (length n) is the sum of the products of their entries.

$$\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^T \mathbf{y} = \sum_{i=1}^{n} x_i * y_i = x_1 * y_1 + x_2 * y_2 + \cdots + x_n * y_n \tag{1}$$

Note that the $\mathbf{x}^T\mathbf{y}$ notation assumes the two vectors are column vectors and is thus written out as matrix multiplication: 1xn * n x 1 will produce a 1x1 matrix, which is just a scalar.

We can take advantage of the matrix multiplication fact to get a whole bunch of vector projections at a time. After all, that is the point of PCA, to project a bunch of vectors (data) onto another set of vectors (principle components). If we line up all the data vectors (n vectors $\mathbf{x_i} \in \mathbb{R}^m$) we get a matrix

$$\mathbf{X} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ \mathbf{x_1} & \mathbf{x_2} & \cdots & \mathbf{x_n} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{2}$$

The PCA vectors come from eigenvectors of the covariance matrix, which is the matrix of the covariances of all pairs of variables and is thus m x m. There will be m eigenvectors, each length m. The matrix of these eigenvectors, which are then the coefficients for the principal component scores are (m vectors $\mathbf{u_i} \in \mathbb{R}^m$)

$$\mathbf{U} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ \mathbf{u_1} & \mathbf{u_2} & \cdots & \mathbf{u_m} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{3}$$

It's transpose is thus

$$\mathbf{U}^T \begin{bmatrix} \cdots & \mathbf{u_1} & \cdots \\ \cdots & \mathbf{u_2} & \cdots \\ \cdots & \vdots & \cdots \\ \cdots & \mathbf{u_m} & \cdots \end{bmatrix} \tag{4}$$

If we write instead of $\mathbf{u_i}^T \mathbf{x_i}$ for a single projection

$$\mathbf{U}^T\mathbf{X} = \begin{bmatrix} \cdots & \mathbf{u_1} & \cdots \\ \cdots & \mathbf{u_2} & \cdots \\ \cdots & \vdots & \cdots \\ \cdots & \mathbf{u_m} & \cdots \end{bmatrix} \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ \mathbf{x_1} & \mathbf{x_2} & \cdots & \mathbf{x_n} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{5}$$

m x m by m x n matrix multiplication

$$= \begin{bmatrix} \mathbf{u_1} \cdot \mathbf{x_1} & \cdots & \mathbf{u_1} \cdot \mathbf{x_n} \\ \vdots & \ddots & \vdots \\ \mathbf{u_m} \cdot \mathbf{x_1} & \cdots & \mathbf{u_m} \cdot \mathbf{x_n} \end{bmatrix} \tag{6}$$

m x n transformed matrix of data projected onto PCs

using fewer PCs makes the outer dimension of $\mathbf{U}$ smaller making the result ($< m$ x n) (7)

3 Eigenvectors

Eigenvectors are used a lot in this course and in many other applications. Getting a deeper, more intuitive understanding of them is useful. You don't need to finish a whole course in linear algebra to do so.

The eigenvector definition is deceptively simple. For an m x m matrix A (assuming it's square and full rank / non-singular), we have pairs of *eigenvalues* λ_i and corresponding *eigenvectors* $\mathbf{u}_i$ such that

$$A\mathbf{u}_i = \lambda\mathbf{u}_i \tag{8}$$

What this means is that for these special vectors, multiplication by this matrix only scales them, it does not rotate them. That is one way to think about what eigenvectors actually mean, but another one that resonates with

me is also more connected to PCA because it is about their use as a space into which data points can be projected.

Let's start with what a space is. We're really thinking about what's called a vector space in linear algebra and an accompanying *basis*. A basis is a set of vectors that can be used to express the whole vector space. By "express the whole vector space" is that you can write any item in that vector space as a linear combination of the basis vectors. For example, the space of all 3-dimensional vectors with real number values is called $\mathbb{R}^3$. The natural basis for that is the vectors

$$\left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\}$$

Writing a vector as a linear combination of those then looks like:

$$x \in \mathbb{R}^3 = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = a \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + b \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + c \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{9}$$

Here's what's great – the eigenvectors of an m x m matrix form a basis of the space of length m vectors, $\mathbb{R}^m$. This fact allows us to rewrite any vector in $\mathbb{R}^m$ as a linear combination of the eigenvectors. But if we then multiply that vector by the matrix, we see how the dominant eigenvector (the one with the highest eigenvalue) dominates. Getting the eigenvectors of the covariance matrix gives us the direction of most variance because the first eigenvector of the matrix dominates in this representation. This is not a proof, just a way of looking at it. The amount it dominates depends a lot on the eigenvalue spectrum of the matrix (same concept you see in a scree plot).
So let's write a vector as a linear combination of the eigenvectors $u_1, \ldots, u_m$:

$$\mathbf{x} \in \mathbb{R}^m = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = a_1\mathbf{u_1} + a_2\mathbf{u_2} + \cdots + a_m\mathbf{u_m} \tag{10}$$

Now, let's multiply that vector $\mathbf{x}$ by $\mathbf{A}$.

$$\begin{aligned} \mathbf{x} &= a_1\mathbf{u_1} + a_2\mathbf{u_2} + \cdots + a_m\mathbf{u_m} && (11) \\ \mathbf{Ax} &= \mathbf{A}a_1\mathbf{u_1} + \mathbf{A}a_2\mathbf{u_2} + \cdots + \mathbf{A}a_m\mathbf{u_m} && (12) \\ &= a_1\mathbf{Au_1} + a_2\mathbf{Au_2} + \cdots + a_m\mathbf{Au_m} \\ &\quad \text{now apply the eigenvector definition above} \\ &= a_1\lambda_1\mathbf{u_1} + a_2\lambda_2\mathbf{u_2} + \cdots + a_m\lambda_m\mathbf{u_m} && (13) \end{aligned}$$

And now we can see that when we break down the matrix behavior by its eigenvector parts, the dominating force is the first term, the dominant eigenvector. That is the essence of what an eigenvector does. It sums up the matrix. That's how it makes sense that we use them to reduce the dimensionality of our representation.