jPCA1 vs. jPCA2

John P. Cunningham

October 10, 2010

1 Preliminaries

We have been interested in matrices M, either unconstrained (that is, $M \in \mathbb{R}^{nxn}$), or where M is a skew-symmetric matrix, such that $M = -M^T$. We say that such matrices $M_{skew} \in \mathbb{S}^n$. Matrices $M \in \mathbb{R}^{n \times n}$ have n^2 variables, whereas matrices $M_{skew} \in \mathbb{S}^n$ have only $\frac{n(n-1)}{2}$ variables (since the diagonal is 0 by definition, the lower triangle defines a matrix entirely). Let's not worry about matrices and just think of these as vectors, where (using MATLAB notation) $m = M(:)$ and $m_{skew} = lowerTriangle(M_{skew})$. By this definition, m is a vector of length n^2 and m_{skew} is a vector of length $\frac{n(n-1)}{2}$. Ok great.

Now let's suppose there is a matrix $H \in \{-1, 0, 1\}^{n \times \frac{n(n-1)}{2}}$, namely a matrix that maps $\frac{n(n-1)}{2}$ elements to n^2 elements. The $\{-1, 0, 1\}$ part means that H can only put values from an argument $v \in \mathbb{R}^{\frac{n(n-1)}{2}}$ into the resulting n^2 vector (and perhaps flip the sign); H can not do any weighted mixing. Such a matrix exists and has some nice properties, but I won't bother you with them.

This matrix H is such that we can take $\frac{n(n-1)}{2}$ elements and turn that into n^2 elements, in particular we have $z = Hm_{skew}$. This is the familiar operation of taking $\frac{n(n-1)}{2}$ elements and making a full skew-symmetric matrix out of them (just in vector form). So too, if we take $y = H^T m$, this is the familiar operation of taking a full matrix M and getting the skew-symmetric part by doing $M - M^T$ (again, just in vector form instead of as matrices).

One more thing. In the stuff below, I'll use $d = dX(:)$ again to keep everything in vector notation. X then is a big block diagonal matrix with the original X repeated on the diagonals.

2 jPCA1

Ok so what. Well, with this definition, here's what jPCA1 is actually doing. Currently, we do (i)$M = X \backslash dX$, and then (ii)$M_{skew} = \frac{1}{2}(M - M^T)$. With the notation above, what that is actually doing is:

$$m_{skew} = \frac{1}{2} H^T (X^T X)^{-1} X^T d \tag{1}$$

You should read this equation as "(i) do least squares regression, and then (ii) get the skew-symmetric part." Mathematically, it is:

$$\begin{aligned} (i) \qquad M^* &= \underset{M \in \mathbb{R}^{n \times n}}{\operatorname{argmin}} \|dX - XM\|_F &(2) \\ (ii) \qquad M_{skew} &= \underset{K \in \mathbb{S}^n}{\operatorname{argmin}} \|M^* - K\|_F &(3) \end{aligned}$$

These two steps are exactly the same as (i)$M = X \backslash dX$, and then (ii)$M_{skew} = \frac{1}{2}(M - M^T)$. Proving the first is just least squares, but proving the second is a bit tricky (interesting though). I can give both proofs if you care, but regardless they are correct.

3 jPCA2

Now jPCA2 does directly what jPCA1 takes two (suboptimal) steps to do. Specifically, it does:

$$m_{skew} = (H^T X^T X H)^{-1} H^T X^T d \tag{4}$$

Compare this equation (Eq 4) with the above definition of jPCA1 in Equation 1. **jPCA1 and jPCA2 are different, but they are both pretty simple (no matter what the code may look like).** Importantly, the conditions for these two being the same are $X^T X = I$. That happens with orthonormal columns in X. By the way, with sample data just drawn as noise (as in a few of my examples), this is why jPCA1 and jPCA2 returned the same thing in the limit of infinite data.

Finally, mathematically, here's what jPCA2 is doing:

$$(i) \qquad M_{skew} = \underset{M \in \mathbb{S}^n}{\operatorname{argmin}} \|dX - XM\|_F \tag{5}$$