Abstract

Dimensionality reduction techniques such as PCA have become a cornerstone of visualizing and analyzing high dimensional data. Classic PCA can be viewed as a diagonalization of the $n \times n$ data covariance matrix, which produces a ranked order of orthogonal dimensions (where the ranking is determined by the size of the eigenvalues). By selecting a subset of those dimensions, one can produce linear projections of the data that represent the largest components of data variance. However, the data covariance is but one choice of summary matrices that can be used to create sets of linear projections. When the data in question are time series (that is, temporal structure evolving through the n dimensional space), the same diagonalization procedure can be used on a linear description of the dynamics (instead of the data covariance) to obtain a different low dimensional projection. This 'dynamical PCA' method produces linear projections that represent the largest eigenvalues of the linear dynamical system as it evolves over time. Linear dynamical systems capture both scaling and rotational aspects of the data (and using these building blocks, one gets familiar features of linear transformations such as shear, projection, reflection, etc.). Dynamical PCA makes no distinction between the scaling and rotational aspects of the data, seeking only the directions of largest and most consistent dynamical activity, and using these directions as the corresponding low dimensional projection of the data. However, in some settings, one may hypothesize the existence of certain types of dynamics, and thus an algorithm is desired that can verify the presence or absence of those dynamics. For example, it is common for neural circuits to generate oscillatory (or rotational) responses, and we wish to find projections of data that best capture these fundamental aspects of the neural response. Here we introduce jPCA, a variant that allows us to specifically extract projections of largest *rotational* dynamics. Using the theory of skew-symmetric matrices, the jPCA algorithm is a simple extension of fitting a linear dynamical system. It has a unique, closed-form solution that can be quickly and easily computed. Importantly, as with PCA and dynamical PCA, jPCA produces a simple set of projection vectors - orthogonal linear directions in the n dimensional space - and so the interpretation of the jPCA projections is precisely the same as with PCA or any other simple linear projection of the data. We motivate and present the details of the method, and we discuss extensions and point to its relevance in extracting rotational structure from electrophysiological data recorded from primate motor cortex.

Dynamical PCA: fitting simple linear dynamical systems

We consider high dimensional time series data $\mathbf{x}(t) = [x_1(t), ..., x_n(t)]$. We can represent this data as a matrix $X \in I\!R^{T \times n}$, where T is the number of time points in the time series, and n is the dimensionality of the data at any time point. The simplest dynamical system that we can fit to that data is a time-invariant linear dynamical system, which has the form $\dot{\mathbf{x}}(t) = \mathbf{x}(t)M$ for any matrix $M \in I\!R^{n \times n}$. Solving for such an M, given our data X, reduces to a simple least squares problem. We can write $\dot{X} = XM$, and then the least squares solution solves the problem $M^* = \text{argmin}_{M \in I\!R^{n \times n}} ||\dot{X} - XM||_F$ (where the subscript F denotes the Frobenius norm). This is often written and solved in simple closed form as $M^* = (X^T X)^{-1} X^T \dot{X}$ (or in MATLAB notation, $M^* = X \backslash \dot{X}$).

Importantly, this dynamics matrix M^* is a valid summary matrix describing the data. We could have also ignored the dynamical relationship and used the data covariance $\Sigma = X^T X$ (assuming X is mean-centered) as another valid $n \times n$ matrix summarizing the data. If we diagonalize Σ and use the resulting linear basis to project our data to lower dimensions, we have traditional PCA. However, we can also diagonalize M^* to produce a valid set of linear projections: we call this approach 'dynamical PCA.' The important difference is that, while PCA ignores temporal information and extracts only those directions in n dimensional space that have the most variance, dynamical PCA extracts directions in the data that have consistent and strong linear dynamical properties. Linear dynamical systems capture both scaling and rotational aspects of the data (combinations of these produce familiar features of linear transformations: shear, projection, reflection, etc.). While dynamical PCA is interesting in its own right, we are here particularly interested in just the rotational aspects of the data, so we first must develop slightly more mathematical machinery.

We now introduce an equivalent but slightly more cumbersome notation for this problem, as it will become useful when solving over matrices M that have special structure. Note that in the problem $X \backslash \dot{X}$, the columns of M are solved *independently* of each other. If we write $M^* = X \backslash \dot{X}$ in the expanded least squares form $M^* = (X^T X)^{-1} X^T \dot{X}$, we see that each column of $\dot{X}$ determines the corresponding column of M independently of other columns. Thus, we can rewrite the original problem as a vector problem instead of a matrix problem. We introduce the vector $\mathbf{m} \in I\!R^{n^2}$, which is simply the matrix $M \in I\!R^{n \times n}$ mapped to a vector (again, using MATLAB notation, this is the familiar $\mathbf{m} = M(:)$ operation). We can then rewrite the least squares problems as $\mathbf{m}^* = \text{argmin}_{\mathbf{m} \in I\!R^{n^2}} ||\dot{\mathbf{x}} - \tilde{X}\mathbf{m}||_2$, where $\dot{\mathbf{x}} = \dot{X}(:)$, and $\tilde{X}$ is a block diagonal matrix with the matrix X repeated on the n diagonal blocks.

These two forms of the least squares problem give identical solutions (it is just a formatting choice: do we write the solution as a vector $\mathbf{m}$ or a matrix M). They describe the dynamics of the data X as a simple linear transformation. One can then diagonalize that matrix as in PCA, and the original data X can be projected down into a lower dimensional space $k << n$ (often $k = 2$ for visualization purposes). As a concrete example, in the motor cortex we may record tens to hundreds of neurons, but we do not believe those neurons are all independent of one another. Using dynamical PCA allows us to view and analyze temporal evolution of neural activity in a lower dimensional space where consistent dynamical activity unfolds.

A subset of linear dynamical systems

Solving for $M \in \mathbb{R}^{n \times n}$ (or vectors $\mathbf{m} \in \mathbb{R}^{n^2}$) as above produces matrices that represent the linear dynamical system best describing the time series data X. General linear dynamical systems, and thus dynamical PCA, make no distinction between the scaling and rotational aspects of the data. Here, instead, we are interested not in general linear dynamical systems, but specifically only the *rotational* aspects of linear dynamical systems. Thus, we must consider how these two building blocks (expansion/contraction and rotation) arise. Every linear transformation M is some mixture of a symmetric matrix and a skew-symmetric matrix, which we write $M = M_{symm} + M_{skew}$. The symmetric part is, by analogy to functions, the "even" part of the matrix, and is defined as $M_{symm} = \frac{1}{2}(M + M^T)$. Such matrices satisfy $M_{symm} = M_{symm}^T$. Accordingly, the skew-symmetric matrix (odd part of the matrix, also called anti-symmetric) is $M_{skew} = \frac{1}{2}(M - M^T)$. Such matrices satisfy $M_{skew} = -M_{skew}^T$. Adding M_{symm} and M_{skew} returns the matrix M, so indeed these are a general description of any matrix M. Additionally, from linear algebra we know that the symmetric component M_{symm} has purely real eigenvalues and thus describes only expansions and contractions of data. So too, the skew-symmetric component has purely imaginary eigenvalues (in complex conjugate pairs) and describes rotations in the data [**?**, **?**].

If we are interested only in rotational linear dynamical systems, we should solve the original least squares problem ($\dot{X} = XM$) not over all matrices M (that describe any linear dynamical system), but instead we should solve the original problem constrained only to the set of rotational linear systems. We call this set of skew-symmetric matrices $\mathbb{S}^{n \times n}$, and we are interested in solutions to the least squares problem of the form $M_{skew} \in \mathbb{S}^{n \times n}$.

jPCA: fitting rotational linear dynamical systems

Let us first rewrite our constrained problem using the original terminology. Whereas previously we sought to solve $M^* = \text{argmin}_{M \in \mathbb{R}^{n \times n}} ||\dot{X} - XM||_F$, here we want to solve $M^* = \text{argmin}_{M \in \mathbb{S}^{n \times n}} ||\dot{X} - XM||_F$. Now we rewrite the problem in the vector notation introduced above, as that will allow us to naturally incorporate the constraint that M^* be a member of $\mathbb{S}^{n \times n}$. We previously considered $\mathbf{m}^* = \text{argmin}_{\mathbf{m} \in \mathbb{R}^{n^2}} ||\dot{\mathbf{x}} - \tilde{X}\mathbf{m}||_2$. However, for our set of skew-symmetric matrices $\mathbb{S}^{n \times n}$, by virtue of the constraint $M_{skew} = -M_{skew}^T$, these matrices only have $\frac{1}{2}n(n-1)$ free parameters (not the n^2 of general matrices $\mathbb{R}^{n \times n}$).

The key step is to note that we can represent skew-symmetric matrices as a vector of $\frac{1}{2}n(n-1)$ free parameters: we call these vectors $\mathbf{k} \in \mathbb{R}^{\frac{1}{2}n(n-1)}$. Further, we can specify a linear map from these vectors onto the space of our vectors $\mathbf{m} \in \mathbb{R}^{n \times n}$. This matrix, which we call H, simply places each of the elements of $\mathbf{k}$ into two indices in $\mathbf{m}$. We can consider $\mathbf{k}$ to be the lower triangle of a skew-symmetric matrix. Suppose we have an element of $\mathbf{k}$, k_r, that corresponds to the (i, j) element of the matrix for $i > j$ (the lower triangle). H then takes this element k_r and places: (1) the element k_r in the element of $\mathbf{m}$ corresponding to the (i, j) index of a matrix M, and (2) the negated element $-k_r$ in the element of $\mathbf{m}$ corresponding to the (j, i) index of M. By this construction we define a matrix H that maps from $\mathbb{R}^{\frac{1}{2}n(n-1)}$ (the number of free parameters in a skew-symmetric matrix) to $\mathbb{R}^{n^2}$. This process is described visually in Figure **??**.

Finally, we return to our problem of interest. Since, by our construction, the quantity $H\mathbf{k}$ is a vector in $\mathbb{R}^{n^2}$ and is equivalent to the set of all skew-symmetric matrices $\mathbb{S}^{n \times n}$, our original problem has the equivalence:

$$M^* = \underset{M \in \mathbb{S}^{n \times n}}{\text{argmin}} ||\dot{X} - XM||_F \quad \Longleftrightarrow \quad \mathbf{k}^* = \underset{\mathbf{k} \in \mathbb{R}^{\frac{1}{2}n(n-1)}}{\text{argmin}} ||\dot{\mathbf{x}} - \tilde{X}H\mathbf{k}||_2. \tag{1}$$

This form can then simply be solved in closed form by grouping H with $\tilde{X}$ and solving $\mathbf{k}^* = (\tilde{X}H)\backslash\dot{x}$. Since $\mathbf{k}^*$ is a vector of length $\frac{1}{2}n(n-1)$ that defines a skew-symmetric matrix M_{skew}^*(via H), and since skew-symmetric matrices describe rotational dynamics, this solution is the matrix defining the best-fitting rotational linear dynamical system.

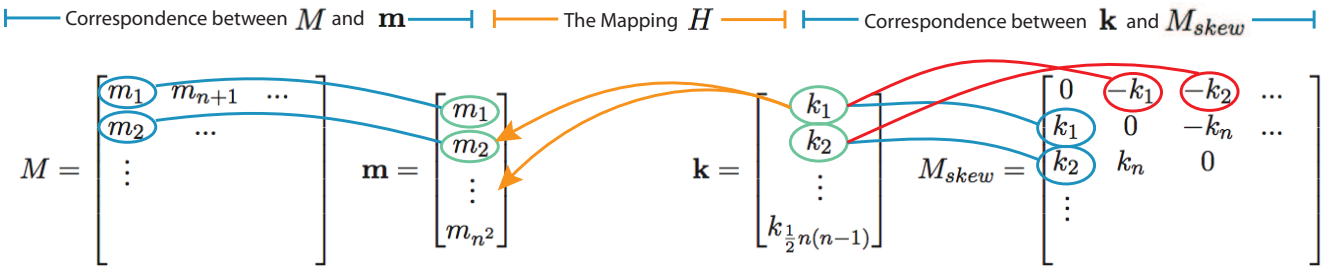

Figure 1: This figure shows the correspondence between M and $\mathbf{m}$, which maps one element of the vector to one element of the matrix (left). At right, we see the correspondence between $\mathbf{k}$ and the matrix M_{skew}, which maps one element of the vector to two elements of the matrix (and the diagonal is 0 by definition). The matrix H then (center) maps from the vector $\mathbf{k}$ to $\mathbf{m}$ and thus provides a correspondence between skew-symmetric and general matrices.

With this matrix M^*_{skew}, we can then diagonalize as before to project the data to lower dimensions. Crucially, just like regular PCA and dynamical PCA, jPCA produces a ranked set of orthonormal vectors that we can use to project our data. Again, the only difference is that, whereas in regular PCA we are interested in directions of highest variance, jPCA allows us to solve for highest frequency and consistency in *rotational* linear dynamical systems. By connection to the imaginary eigenvalues of M_{skew} (a property of skew-symmetric matrices), we call this algorithm jPCA. The jPCs are the plane in n-dimensional space that best isolates rotations.

Implementation, extensions, and results

In terms of implementation, we first note that the orthogonal vectors produced by the diagonalization of M_{skew} come in complex conjugate pairs (these vectors form a unitary matrix). Thus, to find any pair of real-valued projection vectors $\{\mathbf{u}_{i,1}, \mathbf{u}_{i,2}\}$, each complex conjugate pair of vectors $\{\mathbf{v}_{i,1}, \mathbf{v}_{i,2}\}$ is combined as $\mathbf{u}_{i,1} = \mathbf{v}_{i,1} + \mathbf{v}^*_{i,2}$ and $\mathbf{u}_{i,2} = j(\mathbf{v}_{i,1} - \mathbf{v}^*_{i,2})$. These $\mathbf{u}$ vectors can be used exactly as in traditional PCA. As a second implementation note, representing big matrices H and $\tilde{X}$ can be computationally burdensome. Instead, we can avoid the explicit creation of these matrices by solving Equation **??** with a gradient based-method, which allows us to manipulate vectors and matrices in the most computationally convenient way as in Figure **??**. The result is an extremely fast and accurate method for solving the skew-symmetric least squares problem we describe above (only slightly slower than regular unconstrained least squares).

In terms of extensions, here we have focused on developing jPCA. Along the way we introduced dynamical PCA, which is a dimensionality reduction method worth exploration in its own right. Furthermore, our choice of solving over skew-symmetric matrices was appropriate for isolating rotational dynamics. However, if instead we are only interested in expansive/contractive structure (and specifically not in rotations), we can easily modify jPCA to instead solve over symmetric matrices only. This 'symmetric PCA' involves only modifying the mapping H to map from $\mathbb{R}^{\frac{1}{2}n(n+1)}$ to n^2, where the extra n degrees of freedom include the diagonal of the matrix M_{symm} (the diagonal of skew-symmetric matrices is 0 by definition). Additionally, in symmetric PCA all elements are mapped without sign change (no negation as in jPCA). Thus, in working towards jPCA, we have developed a class of linear dimensionality reduction methods: dynamical PCA, jPCA, and symmetric PCA. These add to the standard PCA toolkit, enabling researchers to explore various aspects of structured activity in their data, not simply the highest variance offered by traditional PCA. We believe exploring these and other extensions is an important direction of future work.

Finally, we noted previously that jPCA is motivated by a desire to see rotational dynamics in electrophysiological data from motor cortex. When applied to this data, jPCA successfully projects the responses of tens to hundreds of neurons onto a plane that captures significant underlying rotational structure. These results and their neuroscientific implications are discussed in Churchland**, Cunningham**, et al, COSYNE 2011.

Funding: UK EPSRC EP/H019472/1.

References

[1] G. Strang, *Linear Algebra and its Applications*, Harcourth Brace Jovanovich, 1988.

[2] D. G. Leunberger, *Introduction to Dynamic Systems*, Wiley, 1979.