

Multivariate Methods

These notes include portions of “Multivariate Methods” notes from 2008-2009

Overview

We begin with standard linear regression. The key point, not often made explicit, is that minimizing the squared error between model and fit has a specific probabilistic interpretation. Once this interpretation is recognized, it becomes apparent that it is sometimes appropriate to modify or refine it. This leads to variants of regression, including various forms of regularized regression, ridge regression, and logistic regression.

Linear regression can be viewed as fitting a model, but it can also be viewed as a form of dimension reduction, when the coordinates (the regressors) are known. Principal components analysis can be viewed as attempting to determine the regressors that provide the best dimensional reduction. When the regressors are allowed to vary, there is a symmetry between the “regressors” and the “model coefficients”: they play exactly the same role.

There is also an intrinsic ambiguity that can be anticipated from a geometric view of PCA: the “regressors” are not uniquely determined, what is uniquely determined is the space that they span. There are a variety of ways to attempt to identify privileged coordinates in that space. We consider two very different kinds of strategies. First, it may be known that the data come from several different groups, and one seeks to identify regressors for which the difference between the groups is maximally apparent. This leads to the Fisher Discriminant. A second is independent components analysis (ICA). ICA seeks to identify a set of regressors that are independent of each other in an information-theoretic sense. The symmetry between the regressors and the coefficients is lost.

PCA hinges on analysis of the covariances of the data, and then uses these to determine reduced representations. There are a number of related procedures that focus on the covariances themselves, or similar quantities (e.g., measures of similarity), including canonical correlation analysis, and multidimensional scaling.

Regression

The general setup for regression is as follows:

There are n observations, y_1, \dots, y_n , which we write as a column vector \vec{y} , or, equivalently, as a $n \times 1$ matrix Y .

There are p “regressors”, $\vec{x}_1, \dots, \vec{x}_p$, each of which is a column vector. Each regressor \vec{x}_j has one entry for each observation, so a typical regressor \vec{x}_j is a column $x_{1,j}, \dots, x_{n,j}$, and the set of p regressors forms a $n \times p$ matrix X .

We seek a set of p coefficients b_1, \dots, b_p , written as a column vector \vec{b} or a $p \times 1$ matrix B , that best accounts for the observations via the model

$$Y^{fit} = XB. \quad (1)$$

Standard Regression Examples

Fitting lines and polynomials

To recover the basic use of regression to fit data to a straight line: One could choose the first

regressor to be a constant, $\vec{x}_1 = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$ and the second regressor to be the stimulus values

themselves, $\vec{x}_2 = \begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix}$, so then eq. (1) correspond to $y^{fit}(s) = b_1 + b_2 s$. Similarly, fitting a

polynomial can be done with $\vec{x}_j = \begin{pmatrix} s_1^{j-1} \\ \vdots \\ s_n^{j-1} \end{pmatrix}$, leading to $y^{fit}(s) = b_1 + b_2 s + \dots + b_p s^{p-1}$.

Mapping receptive fields

Each data point corresponds to an instance in time, and the response variable corresponds to whether a spike occurs, or does not occur. Each regressor is an influence to be modeled: the stimulus intensity at a position in space and a prior time.

Analyzing functional imaging data

Each data point corresponds to an instance in time. Each regressor is an influence to be modeled, for example, a stimulus variable convolved with an assumed hemodynamic response, or a nuisance variable, such as a representation of the respiratory or cardiac pulsation.

Formal solution

In standard regression, one determines B by minimizing the squared error between Y^{fit} and Y . As is well-known, this has a formal solution that corresponds to projecting Y into the subspace spanned by the columns of X .

As we saw before (LTGR1011), this projection operator is $P_X = X(X^*X)^{-1}X^*$ (recall, X^* is the adjoint, i.e., the conjugate transpose). Thus, projecting Y into the subspace spanned by X yields $Y^{fit} = P_X Y = X(X^*X)^{-1}X^*Y$. From $Y^{fit} = XB$, it follows that $X(X^*X)^{-1}X^*Y = XB$, and, we have a formal solution,

$$B = (X^*X)^{-1}X^*Y. \quad (2)$$

For this reason, $(X^*X)^{-1}X^*$ is known as the “pseudo-inverse” of X .

(There’s a piece of fine print: we “divided” by X ; this can only be justified if the columns of X are linearly independent. Otherwise, $X(X^*X)^{-1}X^*Y = XB$ would be consistent with multiple solutions for B . This of course makes sense, since if the columns of X , i.e., the regressors, were linearly dependent, then one could find values of B for which $XB = 0$, and hence, there would be no hope of finding a unique solution to eq. (1)).

Note that the same formal solution works can be applied to multiple regression analyses in parallel, provided that they share the same regressors. In this case, each separate “regression problem” can be put in a separate column of Y , and the results can be read out by eq. (2) in separate columns of B .

Formal solution, alternate method

Let’s say we didn’t know how to project onto the columns of X . Instead, we could directly seek to find the elements of B that minimize the squared error between Y^{fit} and Y , namely,

$$R^2 = \sum_{i=1}^n |y_i - y_i^{fit}|^2 = \text{tr}\left((Y - Y^{fit})^*(Y - Y^{fit})\right). \quad (3)$$

We do this by setting derivatives with respect to the coefficients b_j to zero. Each partial derivative is:

$$\begin{aligned} \frac{\partial}{\partial b_j} R^2 &= \frac{\partial}{\partial b_j} \text{tr}\left((Y - Y^{fit})^*(Y - Y^{fit})\right) \\ &= \frac{\partial}{\partial b_j} \text{tr}\left(-(Y^{fit})^*Y - Y^*(Y^{fit}) + (Y^{fit})^*Y^{fit}\right) \\ &= \frac{\partial}{\partial b_j} \text{tr}\left(-B^*X^*Y - Y^*XB + B^*X^*XB\right) \end{aligned}$$

The second line follows because Y is independent of the b_j ’s. The final expression is the j th row of $-2X^*Y + 2X^*XB$. Therefore, setting each of these to 0 results in a of equations (one for each $j = 1, \dots, p$), which are summarized by

$-2X^*Y + 2X^*XB = 0$, which is equivalent to $X^*XB = X^*Y$, and to $B = (X^*X)^{-1}X^*Y$, which is eq. (2).

What is the rationale for the least-squares criterion?

The rationale for the least-squares criterion emerges from a simple Bayesian viewpoint.

Let's say that the *a priori* probability that a given set of model parameters B is correct is given by $p(B)$ (i.e., $p(B)$ expresses our initial guess about the model), and we want to refine our knowledge by determining $p(B|Y)$, i.e., the probability that the model parameters are B , given our observations Y .

We relate $p(B|Y)$ to $p(Y|B)$, the probability that model parameters B will yield the observations Y , by Bayes' Theorem, $p(B|Y) = \frac{p(Y|B)p(B)}{p(Y)}$.

Bayes Theorem follows from $p(B|Y)p(Y) = p(B,Y) = p(Y|B)p(B)$.

We don't know $p(Y)$ (the *a priori* probability of our observations), but this is not an issue, since it is independent of B . Thus, Bayes' Theorem can be recast as

$$p(B|Y) \propto p(Y|B)p(B). \quad (4)$$

The simplest situation is that we have no prior knowledge of B . So then, the most likely *a posteriori* B is the B for which $p(Y|B)$ is maximized. (Fine print: this is the limiting case of progressively flatter $p(B)$. If $p(B)$ is completely flat, then we can't normalize it.)

To calculate $p(Y|B)$, we need to include measurement noise in the model. Otherwise, we would simply conclude that $p(Y|B) = 0$ for every B , except if $Y^{fit} = Y$. So we add noise to the model (1). Our revised model is

$$Y = XB + W, \quad (5)$$

where W is a "noise" term, and is assumed to be independent of X and B . $p(Y|B)$ is therefore the probability of the noise, $W = Y - Y^{fit}$. Each noise value w_i is a difference between the fitted value $y_i^{fit} = \sum_{j=1}^p b_j x_{i,j}$, and the observed value, y_i .

Since $p(Y|B) = p(W)$, we focus on the distribution of the noise vector $\vec{w} = (w_1, \dots, w_n)$. If we don't know what it is, a reasonable assumption is that each value has a mean 0 and a variance V . We can determine the maximum-entropy distribution, which is

$$p(W) = \prod_{i=1}^n p(w_i) = \prod_{i=1}^n \left(\frac{1}{\sqrt{2\pi V}} \exp(-w_i^2 / 2V) \right) = (2\pi V)^{-n/2} \exp\left(-\frac{1}{2V} \vec{w}^* \vec{w}\right).$$

(The above equation is a product of independent Gaussians of mean 0 and variance V).

In sum: finding the most likely *a posteriori* model B means maximizing $p(Y | B)$, which in turn means maximizing $p(W)$, which in turn, means minimizing $\vec{w}^* \vec{w} = R^2$ (eq. (3)). Put another way,

$-\log p(B | Y) = K + \frac{1}{2V} R^2$, i.e., the negative log likelihood of the model B is proportional to the mean squared error, plus an arbitrary constant. So maximizing the likelihood is equivalent to minimizing the squared error.

Thus, via a maximum-entropy assumption on the error, we see that minimizing the squared error is equivalent to a Bayesian approach to model determination, with a limitingly flat prior.

Some variants

The above linkage is predicated on a very simple prior for the model (flat prior) and a simple model for the noise (independent, known variance). If we change either of these – and there is often reason to do so -- we get a useful variant of regression. For example, we may know that models with small coefficients are more likely than models with large coefficients. This leads to “ridge regression.” For example, we may know that the model is smooth, i.e., that models in which adjacent coefficients are very different are *a priori* unlikely. This leads to “regularized regression.” Or we may know that the noise is correlated. Or we may know that the response can only be 0 or 1 (in the case of spikes); in this case, the “noise” cannot be Gaussian. This leads to “logistic regression.”

Correlated noise

Let’s say that there’s reason to believe that the noise quantities are correlated – for example, that the covariance between two components w_i and w_j of $\vec{w} = (w_1, \dots, w_n)$ is c_{ij} . The maximum-

entropy distribution is a correlated Gaussian, and $p(W) = \sqrt{\det M} (2\pi)^{-n/2} \exp\left(-\frac{1}{2} \vec{w}^* M \vec{w}\right)$,

where $M = C^{-1}$, both self-adjoint matrices. Since the covariance between w_i and w_j is c_{ij} , we can write $\langle \vec{w} \vec{w}^* \rangle = C$.

Now, maximizing $p(Y | B) = p(W)$ means minimizing $\vec{w}^* M \vec{w}$, i.e., setting

$$\frac{\partial}{\partial b_j} \text{tr}\left((Y - Y^{\text{fit}})^* M (Y - Y^{\text{fit}})\right) = 0.$$

This leads to $\frac{\partial}{\partial b_j} \text{tr}\left(-B^* X^* M Y - Y^* M X B + B^* X^* M X B\right) = 0$, which is equivalent to

$X^* M X B = X^* M Y$, or,

$$B = \left(X^* M X\right)^{-1} X^* M Y. \quad (6)$$

This can be viewed as a transformed version of the standard regression problem. Say Z is a matrix for which $Z^*Z = M$. Z transforms the correlated-noise problem into one in which the noise is decorrelated. To see this, write $W' = ZW$; the noise values $\vec{w}' = Z\vec{w}$ have covariance $\langle \vec{w}'\vec{w}'^* \rangle = \langle Z\vec{w}\vec{w}^*Z^* \rangle = ZCZ^* = ZM^{-1}Z^* = Z(Z^*Z)^{-1}Z^* = I$.

Next, use Z to transform the regressors and the data to the new coordinates: $X' = ZX$, $Y' = ZY$.

Multiplying both sides of the original model $Y = XB + W$ by Z yields a model equation in the new coordinates, $Y' = X'B + W'$. Finding B in the new coordinates (in which the noise is an uncorrelated Gaussian) yields $B = (X'^*X')^{-1}X'^*Y'$, from eq. (2). With $X' = ZX$, $Y' = ZY$, this is equivalent to $B = ((ZX)^*ZX)^{-1}(ZX)^*(ZY) = (X^*Z^*ZX)^{-1}(X^*Z^*ZY) = (X^*MX)^{-1}(X^*MY)$, which is the same as eq. (6).

Note that we can always find a “whitening” transformation Z as follows. Each pair of rows of Z corresponds to a frequency ω , and the values in that row correspond to extracting the real and imaginary part of the Fourier transform at the frequency ω , followed by multiplication by $1/\sqrt{P_w(\omega)}$. Since (for a Gaussian noise) the Fourier estimates at different frequencies are independent (as are their real and imaginary parts), the transformed values are independent. Since we’ve multiplied these by $1/\sqrt{P_w(\omega)}$, where $P_w(\omega)$ is proportional to their variance, the resulting coordinates are equated for variance.

Nontrivial priors for the model

Another extension of standard regression is to place a prior on the model. This is a natural thing to do when the regressors are related to each other. If so, one expects that nearby regressors will have similar model coefficients. Put another way, one’s prior is that the model coefficients, the b_j ’s, are similar when their subscripts are similar. This can be formalized by stating that the *a priori* probability of a model, $p(B)$, is drawn from a multivariate but correlated Gaussian. That is, $\log p(B) = K - \frac{1}{2}\vec{b}^*M\vec{b}$, where M^{-1} is the matrix of covariances of the b_j ’s.

Now, instead of maximizing $p(Y|B)$ (and assuming that $p(B)$ is constant), we maximize $p(Y|B)p(B) = p(W)p(B)$. This leads to $-(X^*Y + X^*XB)/V + MB = 0$, where V is the noise variance, which has an explicit solution for B , $B = (X^*X/V + M)^{-1}X^*Y$

In a similar vein, one may want to assert *a priori* that the b_j ’s are small (but not exactly how small, nor that they are correlated). shape of the distribution $p(B)$, not its absolute size (i.e., one wants to specify how the parameters are correlated, but not their expected size). This is

equivalent to taking M to be an unknown multiple of the identity. This is “ridge regression.” It leads to $B = (X^*X + kI)^{-1} X^*Y$. One way to choose the parameter k is to try a range of values, and to see which ones do better in an out-of-sample test, i.e., cross-validation.

As a further variant in the same direction: there is no need to assume that the prior for the coefficients is Gaussian. “ L^q ” regression consists of taking $\log p(B) = K - a \sum_{j=1}^p |b_j|^q$. For $q = 1$ (or any $0 < q < 2$), this has the effect of finding models with a small number of large parameters, rather than a large number of small parameters (as typically happens with standard regression). The problem is that solutions of the regression now must be found by iterative means, rather than simply matrix inversion.

Principal Components Analysis

Principal components analysis can be thought of as, finding the best regressors that simultaneously account for multiple datasets. Each dataset is a column vector \vec{y}_k , consisting of the observations $y_{1,k}, \dots, y_{n,k}$; we write a set of these column vectors together as a $n \times k$ matrix Y . The p “regressors”, the column vectors $\vec{x}_1, \dots, \vec{x}_p$ that constitute a $n \times p$ matrix X , are unknown. We want to choose them so that they explain as much of the data as possible, i.e., that we can find an associated set of coefficients B for which $Y - XB$ is as small as possible. The idea is that if n or k (or both) are large but p is small, then the matrices X and B , which are $n \times p$ and $p \times k$, together express the dataset Y in a much more compact form. Geometrically, the columns of X define a p -dimensional subspace that accounts for the data.

Typically, one carries out PCA for a range of values of p , and then chooses a useful one based on considerations such as the fraction of the data explained, and the intended use (visualization, noise removal, etc.)

We note at the outset that we can’t hope to determine X uniquely: alternative X ’s whose columns span the same space will give an equivalent solution. Put another way, for any invertible $p \times p$ matrix T , $XB = XTT^{-1}B$, so $X' = XT$ and $B' = T^{-1}B$ can replace X and B .

One consequence is that we can always assume that the columns of X are orthonormal. The solution is still ambiguous (since in the above, T can still be taken to be a unitary matrix), but this turns out to be very helpful in finding the solutions.

Solution

Our goal is to minimize $R^2 = \text{tr}((Y - Y^{\text{fit}})^*(Y - Y^{\text{fit}}))$, where $Y^{\text{fit}} = XB$ and X is unknown.

Based on the formal solution for standard regression (eq. (2)), we know that $B = (X^*X)^{-1}X^*Y$. Since we can assume that the columns of X are orthonormal, it follows that $X^*X = I$, and that $B = X^*Y$, and $Y^{fit} = XX^*Y$. Thus,

$$\begin{aligned} R^2 &= \text{tr}\left((Y - XX^*Y)^*(Y - XX^*Y)\right) \\ &= \text{tr}\left(Y^*Y - Y^*XX^*Y - Y^*XX^*Y + Y^*XX^*XX^*Y\right), \\ &= \text{tr}\left(Y^*Y - Y^*XX^*Y\right) \end{aligned}$$

where we have again used $X^*X = I$ in the second equality. Thus, minimizing R^2 is equivalent to maximizing $\text{tr}(Y^*XX^*Y)$, subject to the constraint that $X^*X = I$. Note that because $\text{tr}(AB) = \text{tr}(BA)$, this is equivalent to maximizing $\text{tr}(YY^*XX^*)$, and also to maximizing $\text{tr}(XX^*YY^*)$.

At this point, we note that since YY^* is a symmetric matrix, it (typically) has a full set of eigenvectors and eigenvalues. These are the natural, data-driven coordinates for our problem. It therefore makes sense to write a potential solution for X in terms of these eigenvectors.

We will find that the columns of X must be the eigenvectors of YY^* . There are two ways to see this. The first is to write out a possible solution for X in terms of these eigenvectors. This leads to a coordinate-based calculation, which eventually yields the desired result (see pp. 10-12 of MVAR01-MVAR18, 2008-2009 notes for this strategy).

Alternatively, we could use the method of Lagrange Multipliers, which leads to the same conclusion, but in a much more systematic way.

Here, our constraints ($X^*X = I$) can be thought of as a matrix of constraints, one for each element of X^*X . Thus, the Lagrange term (a sum of unknown coefficients multiplied by each constraint) can be compactly written as $\text{tr}(\Lambda X^*X)$, for some $p \times p$ matrix Λ .

Thus, maximizing $\text{tr}(YY^*XX^*)$ subject to $X^*X = I$ is equivalent to maximizing

$F = \text{tr}(YY^*XX^*) - \text{tr}(\Lambda X^*X)$ without constraints on X , and choosing Λ so that $X^*X = I$ at the

maximum. To do this, we calculate $\frac{\partial F}{\partial x_{j,m}}$, and put the resulting $n \times p$ equations, $\frac{\partial F}{\partial x_{j,m}} = 0$,

into a matrix. This yields (see p. 16 of MVAR01-MVAR18, 2008-2009 notes for details)

$$YY^*X = X\Lambda. \tag{7}$$

Now it is obvious that if we choose Λ to be a diagonal matrix consisting of the eigenvalues $\lambda_1, \dots, \lambda_p$ of YY^* , then choosing the columns of X to be the associated eigenvectors satisfies eq. (7) (the definition of eigenvectors) and also satisfies the constraints, because the eigenvectors of a symmetric matrix are orthogonal.

Which eigenvectors and eigenvalues to choose? Making use of the fact that X satisfies eq. (7),

it follows that

$$\text{tr}(YY^*XX^*) = \text{tr}(X\Lambda X^*) = \text{tr}(X^*X\Lambda) = \text{tr}\Lambda = \sum_{i=1}^p \lambda_i.$$

So if one is to choose p eigenvalues, one should choose the p largest ones.

In sum, the best approximation (in the least-squares sense) of an $n \times k$ matrix Y by a product XB of an $n \times p$ matrix X and a $p \times k$ matrix B is to choose the columns of X to be the eigenvectors corresponding to the p largest eigenvalues of YY^* , and to choose $B = X^*Y$. The unexplained variance is the sum of the remaining eigenvalues of YY^* .

This is implemented by matlab's 'princomp.m'.

Note that once p is as large as n (or k , see below) all eigenvectors can be included in X , and all the variance is accounted for – and there has been no dimensional reduction.

Comment: In the Lagrange multiplier setup, there are constraints on the real and the imaginary parts of X^*X . This translates into requiring that the constraint matrix Λ is self-adjoint. We were able to ignore this here, because the above solution yielded a Λ that is a diagonal matrix of real values. But it does matter for Procrustes analysis, below.

A symmetry – an important practical issue

What if we approached the above problem but replaced Y by its transpose, a $p \times n$ matrix? We then would have arrived at a representation for $Y^* = QA$, where the columns of Q are the eigenvectors of Y^*Y . But if $Y^* = QA$, $Y = A^*Q^*$. So we could have solved the original problem, seeking solutions $Y = XB$ where the rows of B are orthonormal, rather than the columns of X .

We can't require that X has *orthonormal* columns and that B has *orthonormal* rows, because then $Y = XB$ would have to be dimensionless. But we can come close, via a slight reformulation.

In fact, our original solution almost does this: via eq. (7), it follows that

$BB^* = (X^*Y)(Y^*X) = X^*YY^*X = X^*X\Lambda = \Lambda$. So the rows of B from the original solution are orthogonal, and the rows of $\Lambda^{-1/2}B$ are *orthonormal*.

So now, with $Z = \Lambda^{-1/2}B$, we have $Y = XB = X\Lambda^{1/2}\Lambda^{-1/2}B = X\Lambda^{1/2}Z$, a decomposition in which the columns of X and the rows of Z are orthogonal. In this formulation,

$$YY^* = (X\Lambda^{1/2}Z)(X\Lambda^{1/2}Z)^* = X\Lambda^{1/2}ZZ^*\Lambda^{1/2}X^* = X\Lambda X^*, \text{ and}$$

$$Y^*Y = (X\Lambda^{1/2}Z)^*(X\Lambda^{1/2}Z) = Z^*\Lambda^{1/2}X^*X\Lambda^{1/2}Z = Z^*\Lambda Z.$$

This is implemented by Matlab's 'svd.m'.

Other than the insight that the “regressors” X and the “model” B play equivalent roles, there’s a practical side to this. The computational effort in calculating principal components is the determination of the eigenvectors. One can either seek X as the first p (column) eigenvectors of the $n \times n$ matrix YY^* and then find $Z = \Lambda^{-1/2}B = \Lambda^{-1/2}X^*Y$, or seek Z as the first p (row) eigenvectors of the $k \times k$ matrix Y^*Y , and then find $X = YZ^*\Lambda^{-1/2}$.

Note that the eigenvalues of YY^* and Y^*Y must match exactly, except that the larger matrix has a string of zero eigenvalues, following the eigenvalues of the smaller matrix.

These approaches are mathematically equivalent, but are very different computationally. The time for eigenvector calculations typically scale as the cube of the edge length of the matrix, and space requirements scale with the square of the edge length. Typically either $n \gg k$ or $k \gg n$ (e.g., one is the number of pixels in an image, and one is the number of time points). Doing the problem “right” can make orders of magnitude difference in the amount of time required.

Using the option ‘econ’ for Matlab’s princomp.m or svd.m ensures that Matlab takes advantage of this symmetry.

Interpreting and using PCA results

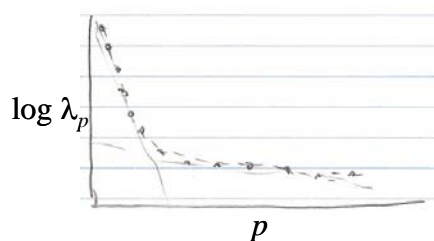
Two issues generically arise with PCA: how many components to keep, and, how to choose coordinates within the selected subspace. There is no general answer to either, but many strategies that can be useful.

How many components to keep?

Since the eigenvalues of YY^* (or Y^*Y) indicate $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p$ indicate the amount of variance explained by each component, this is the natural starting point.

One can choose a cutoff based on the null hypothesis that Y is random, and choose the eigenvalues that exceed those expected from a random matrix. (The latter can be calculated empirically by shuffling the data, *OR*, there are analytic expressions for the distribution of eigenvalues expected from random matrices, there’s a large literature on this.)

Or, one can use a “scree plot,” which often shows a breakpoint, and then choose the breakpoint.



How to choose coordinates within each subspace?

If the goal is simply data visualization, then the orthogonal coordinates identified by the PCA algorithm suffice. But these coordinates typically have nothing to do with the biology: if one imagines $Y = XB$ to represent how the sources X are mixed (by B) to produce the data Y , there's no reason to think that the sources are orthogonal. So it's reasonable to look a matrix T for which the new coordinates $X' = XT$ and $B' = T^{-1}B$ are a more useful representation of the data.

One approach is that one has some specific priors for the sources. For example (fMRI), a source could be the time series of a stimulus, or the latter convolved with a hemodynamic function. Or (EEG) a “source” could be the scalp weighting of a dipole. Then one can seek transformations T for which a column of X' is as close as possible (e.g., in the root-mean-squared sense) to the assumed source.

A more general approach is to look for sources (or mixing matrices) that are “sparse” – i.e., that the elements are either very large or close to zero. The basic procedure is known as the “varimax” rotation, for which the criterion is to find a unitary (rotation) matrix T for which the variances of the *squares* of the elements of B is extremized. Many variants of this basic idea exist (orthomax, quartimax, biquartimin – most implemented by Matlab's ‘rotatefactors.m’), that differ in how the criterion of sparseness is implemented, and whether non-unitary matrices T are allowed.

See also “Independent Components Analysis”, below.

Related procedures that focus on data values

The common denominator here is that in these problems, we seek a set of basis vectors that optimize a quantity that is both quadratic in the data, and quadratic in the coordinates of the basis vectors themselves. These quantities typically arise by considering squares of distances between the data points, in various combinations. The solution typically requires forming a covariance matrix or something like it, and then diagonalizing it.

“Regression” with model error

In ordinary regression, we assume that the regressors X are known exactly, and the data Y is associated with measurement noise. But often, both variables are subject to measurement error. As a simple case, take X and Y to be both univariate. Let's assume we have centered them (i.e., subtracted their respective means) and rescaled them to that they have similar measurement error associated with each point. We can still seek a linear relationship between them – but instead of one that accounts for Y in terms of X (and attributes all the error to X), one that recognizes the measurement error in both. That is, instead of projecting Y onto some multiple of X , we seek a line that minimizes the perpendicular distance from each point (x_i, y_i) to this line.

This is readily solved by PCA. The basic observation is that the sum of the squares of the distances of the points of X and Y from the origin can be decomposed into two quantities: the squared distances along the line (to be found), and the squared distances perpendicular to the line. Thus, minimizing the perpendicular distance to the line is equivalent to maximizing the variance explained by the line. So we simply apply PCA to the bivariate quantity $z = (x_i, y_i)$, viewed as a matrix of size $n \times 2$, where n is the number of observations. The line we seek is the larger of the two eigenvectors of $Z^T Z$. Equivalently, the line is orthogonal to the smallest eigenvector of $Z^T Z$.

The idea extends readily to multivariate quantities. If X has dimension p_x and Y has dimension p_y , then the hyperplane (of dimension $p_x + p_y - 1$) that accounts for as much as possible of the mutual relationship of X and Y is the hyperplane that is orthogonal to the smallest eigenvector of $Z^T Z$, which is a square matrix of size $p_x + p_y$.

In the multidimensional case, we can also seek a subspace of dimension greater than 1, for which the projections of X and Y onto the space agree as much as possible.

Procrustes Analysis

Related to this is the “Procrustes” procedure, which attempts to determine the similarities between two datasets, up to a change in coordinates. Let’s say you have an $n \times k$ dataset S (here, $k \gg n$), and a second dataset of the same size, T . Is there a transformation of the n coordinates that makes S look like T ? Put another way, to what extent do the k points in S and T describe the same shape?

To formalize this, we seek an orthogonal matrix R that minimizes $\text{tr}((S - RT)^*(S - RT))$.

Equivalently, minimize $\text{tr}((S - RT)^*(S - RT))$ for a general R , subject to the constraint $R^*R = I$. As with PCA, this sets up nicely as a Lagrange Multiplier problem, in which $F = \text{tr}((S - RT)^*(S - RT)) + \text{tr}(\Lambda R^*R)$ is minimized (with Λ self-adjoint, see comment above in the PCA section). Taking derivatives with respect to the elements of R leads to $ST^* = R^*\Lambda$.

There’s a classic solution: We write $Y = ST^*$, and determine the symmetric principal-components representation Y , namely, $Y = AQB^*$ where Q is a real diagonal matrix, and A and B are unitary (orthonormal) matrices (so $AA^* = BB^* = I$). Now, take $S = AB^*$. This is also a unitary matrix (since both A and B are). Λ is self-adjoint, since $ST^* = AQB^* = AB^*(BQB^*)$, which implies that $\Lambda = BQB^*$.

There are variants in which one allows R to include dilations, or to be non-orthogonal (closely related to “canonical correlations”).

Linear discriminant analysis, a.k.a. Fisher discriminant

Here, rather than try to find the best coordinates to represent a dataset, we seek the best coordinates to distinguish one subset from another. For example, the subsets may correspond to fMRI images under two conditions. The idea extends readily to more than two subsets.

Let's say there are a total of n samples of multivariate data X , with the samples tagged as belonging to two subsets, n_1 in the first subset and n_2 in the second. Say the two subsets have

means $\vec{\mu}^{[1]} = \frac{1}{n_1} \sum_{i=1}^{n_1} \vec{x}_i^{[1]}$, and $\vec{\mu}^{[2]} = \frac{1}{n_2} \sum_{i=1}^{n_2} \vec{x}_i^{[2]}$, and the global mean is $\vec{\mu} = \frac{n_1 \vec{\mu}^{[1]} + n_2 \vec{\mu}^{[2]}}{n_1 + n_2}$, all

row-vectors. We want to find a linear function of the coordinates that does the best job of separating these two clouds of data. That is, we want to discriminate these subsets by their projections onto a (row) vector \vec{v} . That means, we want to maximize the difference of the projections of the means, while, simultaneously minimizing the scatter *within* the groups, as projected onto \vec{v} .

The setup extends to C classes. The variance between the group means, after projection onto \vec{v} ,

is $V_{between} = \sum_{c=1}^C |\vec{v} \cdot (\vec{\mu}^{[c]} - \vec{\mu})|^2$. The variance within group j is $V_c = \frac{1}{n_c} \sum_{i=1}^{n_c} |\vec{v} \cdot (\vec{x}_i^{[c]} - \vec{\mu}^{[c]})|^2$, so the

total within-group variance is $V_{within} = \sum_{c=1}^C V_c$. We want to find directions that maximize the ratio

of $V_{between}$ to $V_{within} = \sum_{c=1}^C V_c$. It doesn't make sense to simply maximize $V_{between}$; we could do this

in an "empty" way just by magnifying \vec{v} . Simultaneously controlling V_{within} takes care of this, and ensures that we find we focus on the direction of \vec{v} , not its size.

To solve the problem, we could try a "brute-force" method of finding \vec{v} that maximizes the ratio $V_{between} / V_{within}$. Or, we could attempt to maximize $V_{between}$ subject to the constraint that V_{within} is constant. (The specific constant doesn't matter, since it just multiplies \vec{v} by a constant.)

The latter is more practical. Setting it up as a Lagrange Multiplier problem, our job is to minimize $V_{between} + \lambda V_{within}$. Each of the terms is quadratic in v , so derivatives will be linear. This leads to

$$\left(\sum_{c=1}^C (\vec{\mu}^{[c]} - \vec{\mu})^* (\vec{\mu}^{[c]} - \vec{\mu}) \right) \vec{v}^* = \lambda \left(\sum_{c=1}^C \frac{1}{n_c} \sum_{i=1}^{n_c} (\vec{x}_i^{[c]} - \vec{\mu}^{[c]})^* (\vec{x}_i^{[c]} - \vec{\mu}^{[c]}) \right) \vec{v}^*.$$

This is an equation of the form $Az = \lambda Bz$ (for $z = \vec{v}^*$), where A is the between-group covariance, and B is the within-group covariance. A has rank $C - 1$, since $\sum_{c=1}^C n_c (\bar{\mu}^{[c]} - \bar{\mu}) = 0$ (i.e., the weighted mean of the within-group means is the global mean.)

For the two-group case, it is easy to solve. In this case, A has rank 1, so Bz must be within the one-dimensional subspace in the range of A , namely, $(\bar{\mu}^{[1]} - \bar{\mu})^*$, which is necessarily a scalar multiple of $(\bar{\mu}^{[2]} - \bar{\mu})^*$ and also $(\bar{\mu}^{[2]} - \bar{\mu}^{[1]})^*$. Since Bz must be proportional to $(\bar{\mu}^{[2]} - \bar{\mu}^{[1]})^*$, it follows that $\vec{v}^* = z = B^{-1}(\bar{\mu}^{[2]} - \bar{\mu}^{[1]})^*$. More generally ($C > 2$), we seek eigenvalues of $B^{-1}A$, which are guaranteed to be in the span of the columns of A .

These solutions are known as “canonical variates;” they express the variables in which the classes are most cleanly discriminated.

Relation to optimal classifiers

While the criterion of finding a direction in which the ratio of the between-group to the within-group variance is maximized is reasonable, it is unclear whether it is, in any sense, “optimal.” With the additional hypothesis that the two groups are drawn from Gaussians of the same variance (but not necessarily uncorrelated), this turns out to be the case. Let’s say that the variances are V . Then, the probability that \vec{x} is drawn from class c with mean $\bar{\mu}^{[c]}$ is $P_c(\vec{x}) = \frac{1}{K(V)} \exp\left(-\frac{1}{2}(\vec{x} - \bar{\mu}^{[c]})V^{-1}(\vec{x} - \bar{\mu}^{[c]})^*\right)$. The optimal decision rule is to assign \vec{x} to class 1 if $P_1(\vec{x}) > P_2(\vec{x})$, and class 2 if $P_2(\vec{x}) > P_1(\vec{x})$. The log of the ratio of $P_1(\vec{x})/P_2(\vec{x})$ is a linear function of \vec{x} (plus offset), and it is the same linear function identified by the Fisher discriminant.

If the variances of the two groups are not identical, then the classifier includes terms that are quadratic in \vec{x} (i.e., squares and pairwise products of the coordinates), since they no longer cancel. This is known as the “quadratic discriminant.” One approach is to estimate the variances of each group and their means, and then, look at the terms in $\log(P_1(\vec{x})/P_2(\vec{x}))$. With sufficient data and Gaussian data, this procedure must yield the optimal classifier. A second approach is to replace each data vector \vec{x} by an augmented vector that has these additional terms, and then to find the linear discriminant. This is fast and robust, but it will yield a different classifier than the first approach, since in the augmented space, the clouds are no longer Gaussian.

(We are not addressing the issue of identifying the *optimal classifier from a finite sample* of each class; the above analysis assumes that the classes have known means and variances. When they are estimated from data, we have to remember that the “plug-in” estimate of the variance is biased. Debiasing the estimates of the variances of each class can matter if the classes are small and also very unequal in size.)

Related procedures that focus on covariances

In contrast, the common denominator here is that the starting point is already akin to a covariance matrix, i.e., we don't already have "coordinates" for the data – and we seek a simplified expression for the covariance matrix itself.

Analysis of covariance matrices

One motivation for this is to determine the number of noise sources (or, equivalently, "signals" that one cannot control), when only mixtures are observed. Let's say there are m independent noise sources, $X_j(t)$. Further, say that there are n observable quantities $Y_k(t)$, each of which is

an unknown mixture of the sources: $Y_k(t) = \sum_{j=1}^m a_{kj} X_j(t)$. We can measure a matrix of spectra and cross-spectra, $P_{Y_k, Y_r}(\omega)$; our previous work (LSBB1011, pgs 25-26) shows that

$P_{Y_k, Y_r}(\omega) = \sum_{j=1}^m a_{kj} \bar{a}_{rj} P_{X_j}(\omega)$. Restricting observations to a single frequency, can we determine the minimal number of sources m ? And how much of the variance can be explained with fewer noise sources?

Abstractly, the problem (for each frequency ω) is, given a matrix $C = AA^*$, find A . As is the case with PCA, this is underdetermined– since $AA^* = AUU^*A^*$ for any unitary matrix U . But we can find a solution in which the first row of A explains the most variance, the second explains the next-most, etc., by choosing the eigenvalues and eigenvectors of C . The the largest eigenvector of C can be thought of as quantifying the global correlation.

This does not tell us whether the solution at one frequency is consistent with the solution at a second frequency. For this to be the case, the eigenvectors at each each frequency need to be within the same linear span.

Multidimensional scaling

This is a very general procedure for data visualization and for comparison of multivariate datasets.

In contrast to the above setup, the data consist of a set of dissimilarities d_{ij} , which we can think of as distances between points in an as-yet-unknown space. For example, the d_{ij} could be the result of a survey of raters that are asked to compare stimuli i and j . But they also could be the vector-space distances between measurements \vec{y}_i and \vec{y}_j , i.e., $d_{ij} = |\vec{y}_i - \vec{y}_j|$ (but we are not given the vectors \vec{y}_i). Our problem is to find a representation of the d_{ij} as Euclidean distances. That is, we seek a set of vectors $\vec{x}_i = (x_{1,i}, \dots, x_{R,i})$, for which

$$d_{ij}^2 = |\vec{x}_i - \vec{x}_j|^2 = \sum_{r=1}^R |x_{i,r} - x_{j,r}|^2. \quad (8)$$

The embedding dimension R is not known, and we may want to choose a value of R for which eq. (8) is only approximately true. The distances and coordinates of the \vec{x}_i are assumed to be real numbers. We are of course only interested in solutions in which the embedding dimension R is substantially less than the number of data points, N .

We use a trick (due I think to Kruskal) to turn this problem into an eigenvalue problem. The first observation is that the solution (8) is non-unique in two ways. First, as with most of the above problems, it is ambiguous up to rotation – for any rotation matrix M , $|M\vec{x}_i - M\vec{x}_j| = |\vec{x}_i - \vec{x}_j|$. But also, we can add an arbitrary vector to each of the \vec{x}_i : $|(\vec{x}_i + \vec{b}) - (\vec{x}_j + \vec{b})| = |\vec{x}_i - \vec{x}_j|$. Because of this, we can restrict our search to a set of vectors \vec{x}_i whose mean is zero.

We next note that if eq. (8) holds and also that vectors $\sum_i \vec{x}_i = 0$, we can write an equation for the inner products $\vec{x}_i^T \vec{x}_j$ in terms of the d_{ij} . Beginning with

$$d_{ij}^2 = |\vec{x}_i - \vec{x}_j|^2 = (\vec{x}_i - \vec{x}_j)^T (\vec{x}_i - \vec{x}_j) = \vec{x}_i^T \vec{x}_i + \vec{x}_j^T \vec{x}_j - 2\vec{x}_i^T \vec{x}_j, \text{ we note that}$$

$$\frac{1}{N} \sum_{j=1}^N d_{ij}^2 = \frac{1}{N} \sum_{j=1}^N (\vec{x}_i^T \vec{x}_i + \vec{x}_j^T \vec{x}_j - 2\vec{x}_i^T \vec{x}_j) = \vec{x}_i^T \vec{x}_i + S, \text{ where } S = \frac{1}{N} \sum_{k=1}^N \vec{x}_k^T \vec{x}_k.$$

$$\text{Similarly, } \frac{1}{N} \sum_{i=1}^N d_{ij}^2 = \frac{1}{N} \sum_{i=1}^N (\vec{x}_i^T \vec{x}_i + \vec{x}_j^T \vec{x}_j - 2\vec{x}_i^T \vec{x}_j) = \vec{x}_j^T \vec{x}_j + S, \text{ and}$$

$$\frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N d_{ij}^2 = \frac{1}{N} \left(\sum_{j=1}^N (\vec{x}_j^T \vec{x}_j + S) \right) = 2S. \text{ So, if there are vectors } \vec{x}_i \text{ for which eq. (8) holds,}$$

then

$$\vec{x}_i^T \vec{x}_j = \frac{1}{2} \left(-d_{ij}^2 + \frac{1}{N} \sum_{i=1}^N d_{ij}^2 + \frac{1}{N} \sum_{j=1}^N d_{ij}^2 - \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N d_{ij}^2 \right). \quad (9)$$

We therefore write $G_{ij} = \frac{1}{2} \left(-d_{ij}^2 + \frac{1}{N} \sum_{i=1}^N d_{ij}^2 + \frac{1}{N} \sum_{j=1}^N d_{ij}^2 - \frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N d_{ij}^2 \right)$, which is entirely

determined by the given distances, and seek a set of vectors a set of vectors $\vec{x}_i = (x_{1,i}, \dots, x_{R,i})$ for

which $\vec{x}_i^T \vec{x}_j = G_{ij}$, i.e., $G_{ij} = \sum_{r=1}^R x_{r,i} x_{r,j}$. This is equivalent to the matrix equation $G = X^T X$,

where each vector \vec{x}_i forms a column of X .

This yields an immediate formal solution: we write G in terms of its normalized eigenvectors, $G = \sum_{i=1}^N \lambda_i \vec{v}_i \vec{v}_i^*$, and then take $\vec{x}_i = \sqrt{\lambda_i} \vec{v}_i$. The λ_i 's, which can be taken in descending order, indicate the importance of each coordinate in the representation (8).

This works fine provided that all the eigenvalues are non-negative (since we want to find real coordinates). But there is no guarantee that this is the case. The presence of negative values of the eigenvectors indicate that no Euclidean representation is possible, and that instead, the representation (8) must be generalized to

$$d_{ij}^2 = \sum_{r=1}^R \varepsilon_r |x_{i,r} - x_{j,r}|^2, \quad (10)$$

where $\varepsilon_r = +1$ along the Euclidean dimensions ($\lambda_r > 0$, $\vec{x}_i = \sqrt{\lambda_i} \vec{v}_i$), and $\varepsilon_r = -1$ along the non-Euclidean dimensions ($\lambda_r < 0$, $\vec{x}_i = \sqrt{-\lambda_i} \vec{v}_i$). The non-Euclidean dimensions can be considered to describe an intrinsic aspect of the geometry of the original data. Alternatively, if all that is desired is a representation of the rank order of the distances, it is always possible to “cure” the non-Euclidean-ness by replacing the original distances d_{ij} by some power of them, $(d_{ij})^a$. For a power a that is sufficiently close to 0, the non-Euclidean-ness goes away.

Recently, several important extensions of multidimensional scaling -- that embed the data into curved surfaces -- have been developed. Generally, these procedures result in a lower-dimensional representation than standard MDS, chiefly because they don't force the introduction of higher dimensions to deal with global properties. (For example, the distances between points on a sphere, considered as shortest-path distances on its surface, would require an arbitrarily large number of dimensions for an exact representation via standard MDS.)

Two key examples of this approach are the *isomap* procedure (Tenenbaum, J. B., Silva, V. d., and Langford, J. C., 2000, A global geometric framework for nonlinear dimensionality reduction, *Science*, 290, 2319-2323) and *geometric diffusion*, (Coifman, R.R., Lafon, S., Lee, A.B., Maggioni M., Nadler, B., Warner, F., and Zucker, S.W. (2005), Geometric Diffusions as a Tool for Harmonic Analysis and Structure Definition of Data, *PNAS* 102, 7426-7431.

Procedures that seem like simple extensions of PCA, but are not

For completeness, we mention here a few approaches that appear to be simple variants or extensions of PCA, but in fact, are much less straightforward to implement -- i.e., iterative solutions are necessary, and first-principles statistics are hard to come by.

As in standard PCA, we want to approximate the elements $y_{1,k}, \dots, y_{n,k}$ of a $n \times k$ matrix Y by a product XB , where X is $n \times p$ and B is $p \times k$. The standard PCA approach is to attempt to minimize the squared error $R^2 = \text{tr}((Y - Y^{fit})^*(Y - Y^{fit}))$, which, in coordinates is

$$R^2 = \sum_{i=1}^n \sum_{j=1}^k \left(y_{i,j} - \sum_{u=1}^p x_{i,u} b_{j,u} \right)^2. \quad (11)$$

One variant (“non-negative factorization”) is to require that all elements of B are positive. The columns of Y are now represented as a *positive* linear combination of the columns of X ; this allows the linear combination to be interpreted as a physical mixture. Generally, non-negative factorization leads to more components than standard PCA. In standard PCA, a representation based on $p = \min(n, k)$ can always be found that accounts, exactly, for Y . But a greater number of components may be required if the mixing matrix is required to be non-negative.

Another kind of extension relates to placing unequal weights on the observations in Y -- for example, Y may represent data points and some of them may be measured multiple times, or there is a noise model for Y that indicates that some data points are more trustworthy than others. Thus, we’d want to minimize

$$R^2 = \sum_{i=1}^n \sum_{j=1}^k w_{i,j} \left(y_{i,j} - \sum_{u=1}^p x_{i,u} b_{j,u} \right)^2, \quad (12)$$

for some known matrix W of weights. For generic matrices, this no longer reduces to an eigenvalue problem, and must be approached iteratively. This includes the important special case of “missing data” – in which all elements of W are 1, except for a few that are set to 0, to indicate complete unreliability.

Finally, one could imagine extending the PCA setup to include a third factor:

$y_{h,i,j}^{fit} = \sum_{u=1}^p z_{h,u} x_{i,u} b_{j,u}$. For example, we could attempt to fit EEG (or imaging) data at location i

and time j and task h , in terms of underlying mechanisms $u = 1, \dots, p$ -- with the notion that the size of the response is given by a sum over mechanisms, with the contribution of each mechanism depending on task (u), electrode location (x), and time (b). This is just the simplest example of such a decomposition; one might, for example, allow the timecourse to be task-

dependent, but not the geometric factor: $y_{h,i,j}^{fit} = \sum_{u=1}^p z_{h,u} x_{i,u} b_{h,j,u}$. The Procrustes procedure is

vaguely related, in that it attempts to compare PCA analyses across some “design” index (here, h), but it does not attempt to readjust the factors within each design.

Independent Components Analysis

This powerful procedure is typically attributed to Bell and Sejnowski (1994), but (perhaps because it is so useful), there is some debate. It is also called “blind source separation.”

Fundamentally, ICA can be thought of as a way to resolve the ambiguity that for any given PCA representation $Y = XB$, there are equally accurate representations $X' = XT$ and $B' = T^{-1}B$ for any square matrix T .

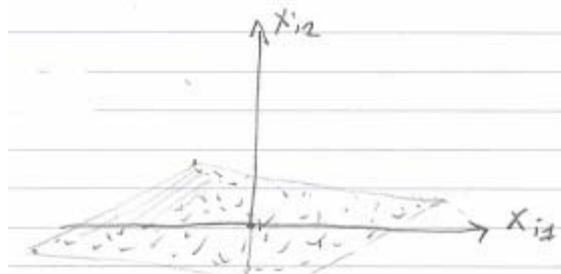
Like PCA, ICA does not make use of the order of the observations of Y . Y is just an array of numbers, and, whether these numbers represent pixels in an image, a time series of voltages, or scores on a behavioral inventory is irrelevant.

However, thinking of the columns of Y as a set of time series, corresponding to the sounds picked up by an array of microphones, is very helpful. In this view, ICA is a solution to the “cocktail party problem” -- i.e., how to decompose Y into a sum of p sources (X , an $n \times p$ array). Here, B , an $p \times k$ array, is a “mixing matrix”, and indicates how much of each source (one of the p columns of X) is picked up by each microphone (one of the k columns of Y). The decomposition may require more than $\min(n, k)$ sources.

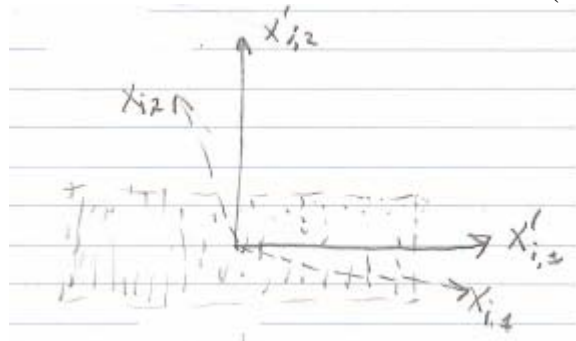
Basic strategy

The basic strategy is as follows. First, we guess a mixing matrix B . Then, as in PCA, we determine the best-fitting values for X , via projection: $X = YB^*(B^*B)^{-1}$. We then inspect the values $x_{i,u}$ of X . In the case of two components, this can be thought of as making a scatterplot of the $x_{i,u}$, with each component corresponding to a coordinate. That is, we plot the points $(x_{i,1}, x_{i,2})$, one point for each time step j .

At this point, several pictures could emerge. We could just get an amorphous, Gaussian cloud. But we could get something like this.



The key point is that the axis that explains the most variance (the $x_{i,1}$ -axis) is not the “simplest” axis. We can find a linear transformation (not necessarily a rotation) that simplifies the picture:



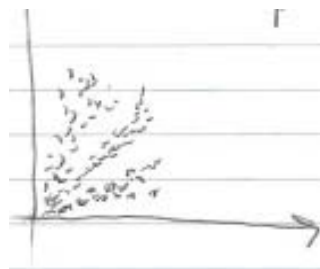
In the new coordinate system, the pairs of values are independent in the information-theoretic sense (not just uncorrelated): knowing the value of one gives no information about the other.

More formally, the shape of the cloud (in either coordinate system) can no longer be described solely by its second moments. (Recall, a Gaussian distribution is the maximum-entropy distribution for a given set of second moments. If the distribution is not Gaussian, then the higher moments cannot be guessed. So, any deviation from Gaussian-ness means that the higher moments are informative.) That is, ICA uses the higher moments to attempt to identify a special set of coordinates. In general, ICA attempts to find a new coordinate system in which the pairs of values are not just uncorrelated (a second-order statistic), but they are also as independent as possible; here $(x'_{i,1}, x'_{i,2})$ are in fact completely independent.

Another example makes the connection to the cocktail party problem more immediate. In this example, the scattergram has three “plumes;” every point lies along just one plume. In the cocktail party problem, this could correspond to three polite speakers, who never speak at the same time. There are two microphones, each represented by an axis. One speaker is closet to the abscissa-microphone, one to the ordinate microphone, and one is in between. Since most of the variance is along the diagonal, PCA would choose that axis as the first principal component. But it would fail to separate the sources.

However, there is a decomposition into three components, one pointing along each plume:

$(x_{i,1}, x_{i,2}) = \sum_{c=1}^3 q_{i,c} (v_1^{[c]}, v_2^{[c]})$. Each $(v_1^{[c]}, v_2^{[c]})$ points along a plume, and, at any given time (i), only one of the three values $q_{i,c}$ is nonzero. This yields the desired unmixing.



Note that there are more components (3) than there are dimensions (2).

In this decomposition, the values $q_{i,c}$ (for each c) are not completely independent, but the dependence is minimized. Specifically, if two of them are zero, there is no information about the third. (We are not proving that this achieves the minimum of the mutual information, but it does.)

Minimizing mutual information, and its consequences

Why does it make sense to minimize the mutual information of the projections onto the axes? The basic idea is that this is equivalent to an “unmixing”, in the following sense. We start with a

multivariate distribution, \vec{x}_i , described in terms of the vectors $\vec{q}_i = (q_{i,1}, \dots, q_{i,p})$, where

$(x_{i,1}, \dots, x_{i,k}) = \sum_{c=1}^p q_{i,c} (v_1^{[c]}, v_2^{[c]}, \dots, v_k^{[c]})$. The entropy of the distribution of the \vec{x}_i is fixed, but we

seek “plumes” v for which the distributions specified by each coordinate of \vec{q}_i is as independent as possible in the information-theoretic sense.

According to the definition of mutual information (the difference between the individual entropies and the joint entropy), it follows that

$$H(\vec{x}) = \sum_{c=1}^k H(q_{\bullet,c}) - I(\vec{q}). \quad (13)$$

That is, the the entropy of the multivariate distribution sampled by the \vec{x}_i is the the difference between the total entropy of each of the k components (the first term on the right), and the mutual information between these components, $I(\vec{q})$. Since $H(\vec{x})$ is constant (given a choice of B), eq. (13) states that minimizing the mutual information $I(\vec{q})$ is equivalent to minimizing the sum of the individual entropies, $H(q_{\bullet,c})$. Since a Gaussian is a distribution with the maximum entropy (given a criterion variance), this is a way of making the individual distributions $H(q_{\bullet,c})$ as *non*-Gaussian as possible. Finally, since mixing makes things Gaussian (via the Central Limit Theorem), finding non-Gaussian projections can therefore be thought of as *un*-mixing.

We also note that estimating entropy from empiric data has a number of pitfalls. For this reason, there are “flavors” of ICA in which, instead of minimum-entropy, some other measure of non-Gaussian-ness is used, such as maximal kurtosis.

Applicability

ICA has become extremely popular in analyzing multivariate datasets. There is broad agreement that it a very useful tool for removing noise components. This makes a lot of sense, since the central hypothesis of ICA – independence – is typically very closely met.

But it is unclear whether it is a useful tool for separating biological sources. Here, the central hypothesis may not be met: sources may not be independent, and, forcing a resolution into “independent” components may be misleading. As mentioned, ICA does not make use of the fact that the samples form a time series, and have a particular order. There are, however, extensions of ICA (and also, extensions of PCA) that make use of dynamics.

It is often useful to first apply PCA, and to choose the number of components liberally, and then to look within these components via ICA. Reducing dimensionality at the “front end” This makes the computations of ICA more rapid, and may eliminate some forms of noise.

In contrast to PCA, the number of components is not limited by the number of sensors or regions (k) or the number of timepoints (n). In principle, this may be an advantage. But in practice, it makes the problem of choosing the number of components even more difficult than in PCA. The

fundamental problem is that there is no principled way of comparing the entropy of distributions with different numbers of dimensions.