Multivariate Methods

These notes are modified from "Multivariate Methods" notes from 2020-2021 and 2022-2023.

Main changes in 2024-2025: material Bayesian justification for least-squares, material about ridge regression, LASSO and related. Main changes in 2022-2023: other variants of PCA, Procrustes analysis, MDS discussed. Main changes in 2020-2021: a brief description of Lagrange multipliers, revised PCA solution, removal of section on covariances, brief description of isomap and geometric diffusion, relationship of Fisher Discriminant and support vector machines, revised description of ICA.

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

There are p "regressors", $\vec{x}_1,\ldots,\vec{x}_p$, 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 $x_{1,j},\ldots x_{n,j}$, and the set of p regressors forms a $n\times p$ matrix X.

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

$$y_i^{fit}=\sum_{j=1}^{p}x_{ij}b_j\,, \tag{1}$$

in vector form

$$\vec{y}^{fit}=\sum_{j=1}^{p}b_j\vec{x}_j\,, \tag{2}$$

or, in matrix form,

$$Y^{fit}=XB\,. \tag{3}$$

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. (3) correspond to $y^{fit}(s)=b_1+b_2s$. 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 $y^{fit}(s)=b_1+b_2s+\ldots+b_ps^{p-1}$.

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.

Using a set of biomarkers to predict a behavior

Each data point corresponds to a patient, and the response variable is some measure of the behavior to be predicted. Each of the *P* regressors is the value of the biomarker for that patient. The coefficients $b_1,\ldots,b_p$ are the weightings of the biomarkers in the prediction of the behavior.

Formal solution

In standard regression, one determines B by minimizing the squared error between Y^{fit} and Y. Note that this corresponds to the notion that the regressors are known exactly, and the data are subject to error (either measurement error or model error). 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 (LTGR), 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,

$$B = (X^*X)^{-1}X^*Y. \tag{4}$$

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. (3)).

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. (4) 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,

$$R^2 = \sum_{i=1}^{n} \left| y_i - y_i^{fit} \right|^2 = \mathrm{tr}\left((Y - Y^{fit})^* (Y - Y^{fit}) \right). \tag{5}$$

The trace allows us to deal with multiple parallel regressions seamlessly. For a single regression, $Y - Y^{fit}$ is a single column, and the argument of the trace is a scalar.

We minimize R^2 by setting derivatives with respect to the coefficients b_j to zero. Each partial derivative is:

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

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 set of equations (one for each $j = 1, \ldots, p$), which are more compactly written as $-2X^*Y + 2X^*XB = 0$. This is equivalent to $X^*XB = X^*Y$, from which $B = (X^*X)^{-1}X^*Y$, which is eq. (4).

Why the least-squares criterion?

There's a pragmatic reason and a conceptual reason. The pragmatic reason is that it provides a simple, closed-form solution, and, relatedly, there is a unique minimum. Had we, for example, chosen to minimize the absolute value of the error, or the fourth-power of the error, this would not have been the case.

The conceptual reason for the least-squares criterion emerges from a 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. We want to use the data to refine our knowledge. That is, we want to dermine $p(B \mid Y)$, i.e., the probability that the model parameters are *B*, given our observations *Y*.

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

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

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

$$p(B \mid Y) \propto p(Y \mid B) p(B). \tag{6}$$

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 $p(Y \mid B)$ 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.)

The current set-up needs another ingredient to calculate $p(Y \mid B)$: a way to link the prediction error to a measure of the implausibility of the model parameters. As it stands, we would simply conclude that $p(Y \mid B) = 0$ for every B, except if $Y^{fit} = Y$. So a natural thing to do is to assume that the disparity between the model prediction $Y^{fit} = XB$ and the observations Y is due to the noise in the measurement. Thus, any model could be correct, but the likelihood that it is correct depends on the likelihood that the difference between model prediction and data could be explained by noise. So we add noise to the model (3). Our revised model is

$$Y = XB + W, \tag{7}$$

where W is a "noise" term.

Now we have to describe W. It is standard to assume that measurement noise is independent of X and B. $p(Y \mid B)$ 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 $p(Y \mid B) = p(W)$, we focus on the distribution of the noise vector $\vec{w} = (w_1, \ldots w_n)$. If we don't know what it is, a reasonable assumption is that each value has a mean 0 and a variance V, and that each is independent.

We still need to know how the noise is distributed. For several reasons (both computational practicality and principles), we assume that it is Gaussian. At least two principles here: the central limit theorem implies that if the noise is the net result of many largely-independent contributions that combine additively, the result will be Gaussian. The maximum-entropy property states that, given a specified variance, the Gaussian is the most random distribution possible. So:

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

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

The key observation is that maximizing $p(W)$ means minimizing $\vec{w}^* \vec{w}$, i.e., our least-squares criterion.

In sum: finding the most likely *a posteriori* model B means maximizing $p(Y \mid B)$, which in turn means maximizing $p(W)$, which in turn, means minimizing $\vec{w}^* \vec{w} = R^2$ (eq. (5)). 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.

Note also that changing the variance V does not change the best-fit values for B, but it does change how rapidly $p(Y\,|\,B)$ and $p(B\,|\,Y)$ decline as one moves away from the best-fit values.

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."

A few of these are considered in more detail below.

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 $\vec{w} = (w_1, \ldots w_n)$ 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 $\left\langle \vec{w}\vec{w}^* \right\rangle = C$.

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

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

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

$X^*MXB = X^*MY$, or,

$$B = \left(X^*MX\right)^{-1} X^*MY . \qquad (8)$$

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 $\left\langle \vec{w}'\vec{w}'^* \right\rangle = \left\langle Z\vec{w}\vec{w}^*Z^* \right\rangle = ZCZ^* = ZM^{-1}Z^* = Z\left(Z^*Z\right)^{-1}Z^* = I$.

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 $B = \left(X'^*X'\right)^{-1} X'^*Y'$, from eq. (4). With $X' = ZX$, $Y' = ZY$, this is equivalent to $B = \left((ZX)^* ZX\right)^{-1}(ZX)^*(ZY) = \left(X^*Z^*ZX\right)^{-1}\left(X^*Z^*ZY\right) = \left(X^*MX\right)^{-1}\left(X^*MY\right)$, which is the same as eq. (8).

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 $p(Y \mid B)$ (and assuming that $p(B)$ is independent of B , i.e., a flat prior), we allow $p(B)$ to vary and maximize $p(Y \mid B)p(B) = p(W)p(B)$. This leads to $(-X^*Y + X^*XB)/V + MB = 0$, where V is the (scalar) noise variance. It solves with $B = \left(X^*X + MV\right)^{-1} X^*Y$.

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). This is equivalent to taking M , the inverse of the covariance matrix, to be augmented by an unknown multiple of the identity. This is "ridge regression." So we are minimizing $\mathrm{tr}\left((Y - Y^{fit})^*(Y - Y^{fit})\right) + kB^*B$. This is also a quadratic function of B, so its derivative is a linar function, and it has an explicit solution. It leads to $B = \left(X^*X + kI\right)^{-1} X^*Y$ (see Homework). 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. Note that this approach will lead to explicit solutions for B even if the standard regression problem is underdetermined, i.e., even if X^*X is singular.

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 $\log p(B) = K - a\sum_{j=1}^{p}\left|b_j\right|^q$. 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. For $q=1$, This is LASSO.

LASSO and related methods are ways of putting a prior on the model parameters to favor solutions in which there are a small number of significant regression coefficients. This can be particularly helpful when the model formulation has a large number of regressors, but a simple model is desired. In LASSO and related, he quantity to be minimized is $\frac{1}{N}\mathrm{tr}\left((Y-Y^{fit})^*(Y-Y^{fit})\right)+\lambda\left(\sum\left|b_k\right|^q\right)^{1/q}$. The value of λ determines how strongly the tendency for "sparse" parameters is enforced, and, as in ridge regresssion, it is often determined by how well the model predicts out-of-sample data. LASSO and related methods also require an iterative approach.

Principal Components Analysis

Principal components analysis can be thought of as, finding a single set of regressors that best account for multiple datasets. Each dataset is a column vector $\vec{y}_k$, consisting of the observations $y_{1,k},\ldots,y_{n,k}$; we write a set of these column vectors together as a $n\times k$ matrix *Y*. The p "regressors", the column vectors $\vec{x}_1,\ldots,\vec{x}_p$ 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.

PCA Scenarios

EEG

A typical scenario is that the column vectors $\vec{y}_k$ correspond to the time series of voltages (at each of *n* time points, the length of each column) at *k* electrodes. PCA seeks to determine whether these time series can be accounted for by linear mixing of a smaller number of signals *p*. In the EEG scenario, such linear mixing will occur by volume-conduction spreading of the signals from intracranial generators – but (see below) PCA cannot hope to resolve these generators unambiguously.

Analyzing functional imaging data

One scenario is that the column vectors $\vec{y}_k$ correspond to the functional activation pattern observed in each of the *n* pixels (voxels), for the *k* behavioral paradigms. So PCA will determine whether these activation patterns all represent mixtures of a smaller number *p* of "fundamental" activation patterns. Alternatively, the column vectors $\vec{y}_k$ could correspond to the functional activation pattern observed in each subject, and the PCA analysis seeks to determine whether there are some common modes of activation that, when mixed together, account for the subject-to-subject variation.

Behavior/Psychometrics

This is the scenario in which PCA was originally developed. There is one column vector $\vec{y}_k$ for each of *k* subjects, and its entries correspond to a set of *n* behavioral measurements, e.g., scores on a battery of tests. PCA seeks to determine whether there are some underlying set of personality types, which, when mixed together in varying amounts, can account for each individual's behavioral data. Alternatively one could look at the "transpose" problem – where each of the column vectors corresponds to one of *k* behavioral tests, and the *n* entries corresponds to the scores of each subject. Here, PCA seeks to determine whether there is some small underlying set of psychological characteristics, and each test assays a different admixture of them.

In EEG, when the *k* columns are the *k* electrodes and *n* is the number of time points, typically $n >> k$. Similarly, in imaging, when the *k* columns are the behavioral paradigms (or subjects) and *n* is the number of pixels, then $n >> k$. But in the above psychometrics scenario, *n* and *k* may be comparable – and, as this example illustrates, interchanging rows and columns leads to a different viewpoint on the same data. Below we will see that there's a corresponding mathematical symmetry between the rows and columns, even though the relationship of PCA to regression might lead us to treat them differently.

Solution

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.

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

Based on the formal solution for standard regression (eq. (4)), 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,

$$R^2=\mathrm{tr}\left((Y-XX^*Y)^*(Y-XX^*Y)\right)$$
$$=\mathrm{tr}\left(Y^*Y-Y^*XX^*Y-Y^*XX^*Y+Y^*XX^*XX^*Y\right),$$
$$=\mathrm{tr}\left(Y^*Y-Y^*XX^*Y\right)$$

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

At this point, we note that since YY^* is a symmetric matrix, it (typically) has a full set of eigenvectors and eigenvalues, and these are orthogonal. 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, outlined as follows:

First solution. We express all matrices using the eigenvectors of YY^* as a basis in descending order of eigenvalues. So if YY^* has (column) eigenvectors $\vec{v}_j$ and eigenvalues $\lambda_1>\lambda_2>\cdots>\lambda_n$, YY^* is a nonzero only on the diagonal, which consists of the λ_i. Now say that in this basis, the *m*th column of *X*, written in terms of the eigenvectors of YY^*, is $\vec{x}_m=\sum a_{jm}\vec{v}_j$. In this basis, the matrix elements of XX^* are given by $\left(XX^*\right)_{jr}=\sum_m a_{jm}a_{rm}$. Given the diagonal structure of YY^*, the matrix elements of XX^*YY^* are therefore $\left(XX^*YY^*\right)_{jr}=\lambda_r\sum_m a_{jm}a_{rm}$. So the trace, which we are trying to maximize, is the sum of this expression over all elements with $j=r$:
$\mathrm{tr}\left(XX^*YY^*\right)=\sum_{m,r}\lambda_r a_{rm}{}^2$.
Now consider maximizing this trace, starting with the case of a matrix *X* with only one column. Since the a_{r1} are free to vary other than the constraint $\sum_r a_{r1}{}^2=1$ (i.e., the sole column of *X* has magnitude 1) , the trace is maximized when the coefficient corresponding to the largest eigenvalue is 1, i.e., $a_{1,1}=\pm 1$, and the remaining coefficients of $a_{j,1}$ are zero.

Now assume that *X* has two columns. The maximum is achieved if all of the mass of the first two columns can be concentrated in the first two rows, since these rows make the largest contribution to the trace (being multiplied by λ_1 or λ_2). So their span is all of the column vectors with zeros below the first two rows. Since any orthogonal basis within this subspace must yield an equal trace, the trace is maximized by choosing $a_{1,1} = \pm 1$ and $a_{2,2} = \pm 1$, with $a_{1,2} = a_{2,1} = 0$ for orthogonality.

So inductively, when expressed in the basis of eigenvectors of YY^*, the matrix that sequentially maximizes $\mathrm{tr}\left(XX^*YY^*\right) = \sum_{m,r} \lambda_r a_{rm}{}^2$ is a diagonal matrix of ± 1 -- and it follows that (other than sign-flips) the columns of *X* are the eigenvectors of YY^*, in descending order.

Alternatively, we could use the method of Lagrange Multipliers, which leads to the same conclusion, but in a coordinate-free way.

The Lagrange Multiplier method in a nutshell: We want to extremize some function $f(\vec{x})$, subject to constraints. Finding this extremum is equivalent an unconstrained problem, along with a new set of conditions: (i) find the unconstrained extremum of $F(\vec{x},\vec{\lambda})$, where $F(\vec{x},\vec{\lambda}) = f(\vec{x}) - \sum_k \lambda_k g_k(\vec{x})$; (ii) denote this unconstrained extremum as $\vec{x} = M(\vec{\lambda})$, and (iii) then find the values of the λ_k for which the constrains are satisfied, i.e., for which $g_k(M(\vec{\lambda})) = 0$. The basic idea is that at this point, the only way that $F(\vec{x},\vec{\lambda})$ can change is by changing the value of one or more of the constraints, or by changing $f(\vec{x})$. Since you can't change the constraints, an extremum for $F(\vec{x},\vec{\lambda})$ corresponds to an extremum for $f(\vec{x})$.

As an example (with one constraint): maximize $f(\vec{x}) = \vec{x}\bullet\vec{a}$ subject to the constraint $\vec{x}\bullet\vec{x} = 1$. Here, $F(\vec{x},\lambda) = \vec{x}\bullet\vec{a} - \lambda\vec{x}\bullet\vec{x}$. To extremize it, set $\frac{\partial}{\partial x_j} F(\vec{x},\lambda) = 0$:

$\frac{\partial}{\partial x_j} F(\vec{x},\lambda) = \frac{\partial}{\partial x_j} \vec{x}\bullet\vec{a} - \lambda\vec{x}\bullet\vec{x} = a_j - 2\lambda x_j$, so $\frac{\partial}{\partial x_j} F(\vec{x},\lambda) = 0$ requires that $x_j = \frac{1}{2\lambda} a_j$. To satisfy the constraint $\vec{x}\bullet\vec{x} = 1$, we need $\sum_j \left(\frac{a_j}{2\lambda}\right)^2 = 1$, i.e., $(2\lambda)^2 = \sum_j \left(a_j\right)^2$, so $x_j = \frac{a_j}{\sqrt{\sum_k a_k^2}}$.

Returning to the PCA problem: 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 $\mathrm{tr}(\Lambda X^*X)$, for some $p \times p$ matrix Λ.

Thus, maximizing $\operatorname{tr}\left(YY^*XX^*\right)$ subject to $X^*X = I$ is equivalent to maximizing $F = \operatorname{tr}\left(YY^*XX^*\right) - \operatorname{tr}(\Lambda X^*X)$ 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 ($\frac{\partial F}{\partial x_{j,m}} = 0$, for each $x_{j,m}$), , into a matrix. This yields (see p. 16 of MVAR01-MVAR18, 2008-2009 notes for details)

$$YY^*X = X\Lambda . \tag{9}$$

Now consider this equation if Λ is the diagonal matrix consisting of the eigenvalues $\lambda_1, \ldots \lambda_p$ of YY^*. The equation is solved if we choose the columns of X to be the associated eigenvectors normalized to a magnitude of 1: eq. (9) becomes the definition of eigenvectors, and the constraints are satisfied because the eigenvectors of YY^*, which is symmetric, are necessarily orthogonal.

Which eigenvectors and eigenvalues to choose to guarantee that we are at the global maximum? Making use of the fact that X satisfies eq. (9), it follows that

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

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.

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 ortho*normal* columns and that *B* has ortho*normal* 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. (9), 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 ortho*normal*.

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,
$YY^* = \left(X\Lambda^{1/2}Z\right)\left(X\Lambda^{1/2}Z\right)^* = X\Lambda^{1/2}ZZ^*\Lambda^{1/2}X^* = X\Lambda X^*$, and
$Y^*Y = \left(X\Lambda^{1/2}Z\right)^*\left(X\Lambda^{1/2}Z\right) = Z^*\Lambda^{1/2}X^*X\Lambda^{1/2}Z = Z^*\Lambda Z$.

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

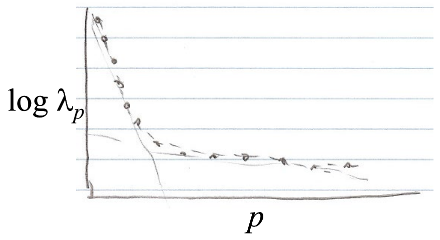
Two issues generically arise with PCA: how many components to keep, and, how to choose coordinates within the selected subspace. There is no general answer to either, but many strategies that can be useful. This is also one of the motivations for Independent Components Analysis, below.

How many components to keep?

Since the eigenvalues of YY^* (or Y^*Y) indicate the amount of variance explained by each component, and $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$, this ordering 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) matix T for which the variances of the *squares* of the elements of B is extremized. Many variants of this basic idea exist (orthomax, quartimax, biquaritimin – 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.

Demixed PCA

This approach is useful when there are built-in meanings associated with one set of variables – for example, neural data (time series) in an experiment in trials differ according to stimulus magnitude (drawn from a finite set) and the behavioral outcome (e.g., a decision, also drawn from a finite set). One factor (the time point) is treated in the standard way, but for the second factor (the trial type), one does not allow mixing between magnitude and behavioral outcome. See Kobak, …, Machens (2016), Demixed principal component analysis of neural population data, https://doi.org/10.7554/eLife.10989

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 measurement 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 $\mathrm{tr}\left((S-RT)^*(S-RT)\right)$. Equivalently, minimize $\mathrm{tr}\left((S-RT)^*(S-RT)\right)$ 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 $F = \mathrm{tr}\left((S-RT)^*(S-RT)\right) + \mathrm{tr}\left(\Lambda R^* R\right)$ 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 $R^* = AB^*$. This is also a unitary matrix (since both A and B are). Λ is self-adjoint, since $ST^* = AQB^* = AB^*(BQB^*)$, which implies that $\Lambda = BQB^*$. For successive approximations, include only the first h eigenvalues of Q,i.e.,its upper $h \times h$ block, and the first h columns of A and B^*.

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

Alignment of multiple datasets

One can also start with multiple datasets, T_i, each $n \times k$ matrices, and attempt to identify a set of transformations R_i such that the resulting transformed datasets $T_i' = R_i T_i$ are as similar to each other as possible. A natural notion of similarity is that the distance from the centroids of the corresponding points of the T_i' are as close as possible to the points themselves, in a mean-squared sense. There's a simple iterative algorithm to do this: first align each of the T_i ($i \geq 2$) with T_1, then find the centroids, then continue to align with the centroids and refining the centroids. However, a global minimum is not guaranteed.

There's a second way of framing the multiple-alignment problem: rather than compute the Euclidean distances to the centroid, compute the amount of rotation required to align each dataset with the centroid. And then minimize some function of these rotation angles, or of the Frobenius norms of the matrices $\log\left(R_i R_j^{-1}\right)$. See M. Moakher, Means and averaging in the group of rotations. SIAM J. Matrix Anal. Appl. 24(1), 1-16 (2002) and M. Moakher, A differential geometric approach to the geometric mean of symmetric positive-definite matrices. SIAM J. Matrix Anal. Appl. 26(3), 735-747 (2005) for more on this.

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 class means, after projection onto $\vec{v}$, is $V_{between}(\vec{v}) = \sum_{c=1}^{C}\left|\vec{v}\bullet\left(\vec{\mu}^{[c]} - \vec{\mu}\right)\right|^2$. The variance within class j is $V_c(\vec{v}) = \frac{1}{n_c - h}\sum_{i=1}^{n_c}\left|\vec{v}\bullet\left(\vec{x}_i^{[c]} - \vec{\mu}^{[c]}\right)\right|^2$ (where naively $h = 0$ but we should take $h = 1$ to get an unbiased estimate), so the total within-class variance is $V_{within}(\vec{v}) = \sum_{c=1}^{C} V_c(\vec{v})$. We want to find directions that maximize the ratio of $V_{between}(\vec{v})$ to $V_{within}(\vec{v})$. 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

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

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 at most $C - 1$, since $\sum_{c=1}^{C} n_c\left(\vec{\mu}^{[c]} - \vec{\mu}\right) = 0$ (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, $\left(\vec{\mu}^{[1]} - \vec{\mu}\right)^*$, which is necessarily a scalar multiple of $\left(\vec{\mu}^{[2]} - \vec{\mu}\right)^*$ and also $\left(\vec{\mu}^{[2]} - \vec{\mu}^{[1]}\right)^*$. Since Bz must be proportional to $\left(\vec{\mu}^{[2]} - \vec{\mu}^{[1]}\right)^*$, it follows that $\vec{v}^* = z = B^{-1}\left(\vec{\mu}^{[2]} - \vec{\mu}^{[1]}\right)^*$. 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 $\vec{\mu}^{[c]}$ is $P_c(\vec{x}) = \frac{1}{K(V)}\exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu}^{[c]})V^{-1}(\vec{x}-\vec{\mu}^{[c]})^*\right)$. 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). This is because the expression for each $P_c(\vec{x})$ has a different variance term, so the terms that are quadratic in $\vec{x}$ 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 $\log\left(P_1(\vec{x}) / P_2(\vec{x})\right)$. With 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 includes quadratic 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.

The above analysis assumes that the sample size is large enough so that the mean and variance of each class can be regarded as known. When sample size is small (and if mean and variance are not known a priori) then mean and variance must be estimated from the data. When doing so we have to remember that the "plug-in" estimate of the variance is biased. A related issue is the problem of overfitting which can be addressed to fitting the classifier to one subset of the data and then using it to classify the other portion.

Classifiers are of course a huge topic. Support vector machines, like the Fisher Discriminant, are linear classifiers. The basic difference is that the Fisher discriminant minimizes the overlap of the distributions (separation/variance), while a basic SVM minimizes the number of mis-classifications. More typically, SVM's minimize the number of severe mis-classifications, and add a graded penalty for errors that are near the boundary – the so-called "soft margin" or "hinge-loss" classifiers. Like the Fisher Discriminant, the parameters of an SVM can be found via an algorithm that reaches the global optimum.

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., $d_{ij} = \left|\vec{y}_i - \vec{y}_j\right|$ (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 $\vec{x}_i = (x_{1,i}, ..., x_{R,i})$, for which

$$d_{ij}^{\ 2} = \left|\vec{x}_i - \vec{x}_j\right|^2 = \sum_{r=1}^{R} \left|x_{i,r} - x_{j,r}\right|^2 . \qquad (10)$$

The embedding dimension *R* is not known, and we may want to choose a value of *R* for which eq. (10) 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 (10) 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*, $\left|M\vec{x}_i - M\vec{x}_j\right| = \left|\vec{x}_i - \vec{x}_j\right|$. But also, we can add an arbitrary vector to each of the $\vec{x}_i$: $\left|\left(\vec{x}_i + \vec{b}\right) - \left(\vec{x}_j + \vec{b}\right)\right| = \left|\vec{x}_i - \vec{x}_j\right|$. 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. (10) 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 $d_{ij}^{\ 2} = \left|\vec{x}_i - \vec{x}_j\right|^2 = \left(\vec{x}_i - \vec{x}_j\right)^T \left(\vec{x}_i - \vec{x}_j\right) = \vec{x}_i^{\,T}\vec{x}_i + \vec{x}_j^{\,T}\vec{x}_j - 2\vec{x}_i^{\,T}\vec{x}_j$, we note that

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

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

$\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}^{\ 2} = \frac{1}{N}\left(\sum_{j=1}^{N}\left(\vec{x}_j^{\,T}\vec{x}_j + S\right)\right) = 2S$. So, if there are vectors $\vec{x}_i$ for which eq. (10) holds, then

$$\vec{x}_i^T\vec{x}_j = \frac{1}{2}\left(-d_{ij}^{\ 2} + \frac{1}{N}\sum_{i=1}^{N} d_{ij}^{\ 2} + \frac{1}{N}\sum_{j=1}^{N} d_{ij}^{\ 2} - \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}^{\ 2}\right). \tag{11}$$

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 $\vec{x}_i = (x_{1,i},\ldots,x_{R,i})$ 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 (10).

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 (10) must be generalized to

$$d_{ij}^{\ 2} = \sum_{r=1}^{R} \varepsilon_r \left|x_{i,r} - x_{j,r}\right|^2, \tag{12}$$

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, $\left(d_{ij}\right)^a$. For a power *a* that is sufficiently close to 0, the non-Euclidean-ness goes away.

Non-Euclidean variants

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.)

There are two major classes of approaches. One class of approaches assumes a specific curved space, e.g, hyperbolic space. The space is translationally invariant, and curvature is uniform, so that the generalization from Euclidean space does not require introduction of many parameters. These methods both visualize, and characterize (via the global curvature) the dataset. See Zhou,…, Sharpee (2018) Hyperbolic geometry of the olfactory space. Science Advances doi: 10.1126/sciadv.aaq1458.

A second class of approaches does not assume any specific geometry at all; generally the goal is visualization of the data as clusters, rather than a quantitative characterization of the space.

Two examples of these approaches 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.

The isomap procedure illustrates how such a procedure can better capture the long-range distances, at the expense of losing some detail about the short-range distances. There are three steps. (1) Identify the K nearest neighbors of each data point. Create a graph whose nodes correspond to the data points, and whose links (edges) correspond to the K nearest neighbors of each data point. (2) Determine a global distance all point pairs, by counting how many links steps along this graph are needed to connect them. (3) Now embed the points using standard MDS. This procedure emphasizes getting the global distances correct, at the expense of local distances.

The geometric diffusion procedure is related. In its simplest form, it begins by (1) creating a graph based on nearest neighbors, or by setting a threshold based on a continuous-valued quantity, such as a correlation coefficient. (So each node may have a different number of connections.) Then, (2) instead of using multidimensional scaling to embed this graph, it considers a diffusion process, in which particles diffuse around the graph based on the connectivity of the nodes. As we will see later on in the Graph Theory section (or see prior year notes: GTM1819), this process is governed by a "graph Laplacian", one form of which is self-adjoint. (3) The first eigenvectors of the graph Laplacian identify coordinates for the data points that (in some sense) optimally embed the graph in a low-dimensional space. There is substantial further generality, as the connectivity of nodes can also be a continuous function, rather than just 0 or 1.

Other alternatives for visualizing high-dimensional data via projection into a 2- or 3-dimensinal domain are "t-SNE" and "UMAP" – see https://towardsdatascience.com/tsne-vs-umap-global-structure-4d8045acba17 for a gentle introduction. They make different compromises vis-à-vis a priority on veridical representation of local vs global distances. Neither approach has general guarantees of identifying a global optimum of its cost function, but this is not usually a practical concern.

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 $y_{1,k},\ldots,y_{n,k}$ of a $n\times k$ matrix *Y* by a product *XB*, were *X* is $n\times p$ and *B* is $p\times k$. The standard PCA approach is to attempt to minimize the squared error $R^2=\mathrm{tr}\left((Y-Y^{fit})^*(Y-Y^{fit})\right)$, which, in coordinates is

$$R^2=\sum_{i=1}^{n}\sum_{j=1}^{k}\left(y_{i,j}-\sum_{u=1}^{p}x_{i,u}b^*{}_{j,u}\right)^2. \tag{13}$$

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

$$R^2=\sum_{i=1}^{n}\sum_{j=1}^{k}w_{i,j}\left(y_{i,j}-\sum_{u=1}^{p}x_{i,u}b^*{}_{j,u}\right)^2, \tag{14}$$

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 $u=1,\ldots,p$ -- 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. There are numerical procedures for seeking such representations, but none of them carry guarantees as to unique solutions. See for example Feng et al., Robust block tensor principal component analysis. Signal Processing 166 (2020), 107271.

Note also that the Procrustes methods are also a kind of three-way PCA.

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*.

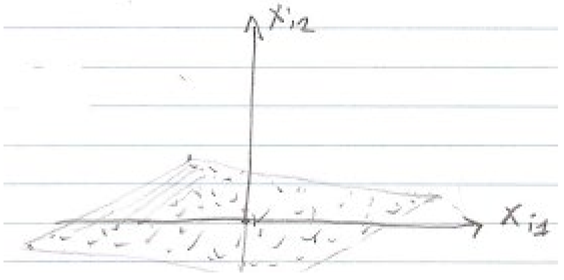
Like PCA, ICA does not make use of the order of the observations of *Y*. *Y* is just an array of numbers, and, whether these numbers represent pixels in an image, a time series of voltages, or scores on a behavioral inventory is irrelevant.

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.

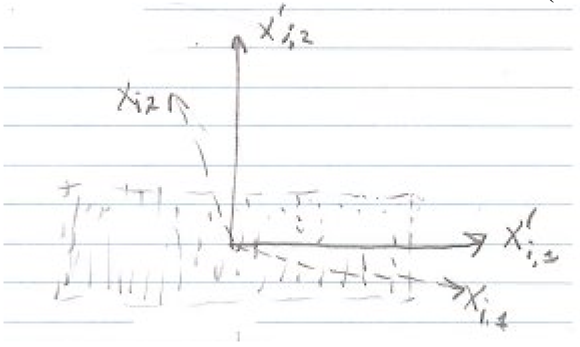
Basic strategy

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.

ICA makes headway because the shape of the cloud is more complex than a Gaussian blob. This is equivalent to saying a description of the data in terms of its second moments – which is what we need for PCA – is incomplete. If we knew that the cloud were Gaussian, then its second moments – i.e., its covariances – determines the Gaussian, and from the Gaussian, we could calculate the higher moments (third-order, fourth-order, etc.) But if the distribution is not Gaussian, then the higher moments cannot be computed from the Gaussian approximation. So one way of thinking about ICA is that ICA uses these higher moments to attempt to identify a special set of coordinates.

The more general framework is that ICA attempts to find components within the data that are not just just uncorrelated, but as independent as possible. Correlation (or lack of correlation) is a second-order statistic and can therefore be determined from the Gaussian approximation to the data, but independence, which is an information-theoretic concept, cannot. When a square is aligned with the coordinate axes, its projections onto the axes are independent. But if its diagonals point along the axes, the projections are not independent: knowing that one coordinate is extreme places bounds on the other coordinate.

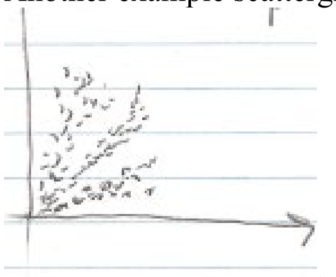
Brief detour into information theory, as it is the natural way of expressing independence. Recall that the entropy of a distribution $p(\vec{x})$ is defined by $H_X(P) = -\sum_{\vec{x}} p(\vec{x}) \log p(\vec{x})$. Similarly, a joint distribution $r(\vec{x},\vec{y})$ has an entropy defined by $H_{X,Y}(R) = -\sum_{\vec{x},\vec{y}} r(\vec{x},\vec{y}) \log r(\vec{x},\vec{y})$. Given this joint distribution, we can construct the marginal distributions for $\vec{x}$, $p(\vec{x}) = \sum_{\vec{y}} r(\vec{x},\vec{y})$, and for $\vec{y}$, $q(\vec{y}) = \sum_{\vec{x}} r(\vec{x},\vec{y})$. The information-theoretic measure of dependence, known as "mutual information", is the difference between the sum of the entropies of the two marginal distributions and the entropy of the joint distribution $r(\vec{x},\vec{y})$: $I_{X,Y}(R) = H_X(P) + H_Y(Q) - H_{X,Y}(R)$

The homework MVAR2021c shows mutual information is minimized when $r(\vec{x},\vec{y}) = p(\vec{x})q(\vec{y})$, It is straightforward algebra to show that under these circumstances $I_{X,Y}(R) = 0$:

$$\begin{aligned}
&H_X(P) + H_Y(Q) = -\sum_{\vec{x}} p(\vec{x}) \log p(\vec{x}) - \sum_{\vec{y}} q(\vec{y}) \log q(\vec{y}) \\
&= -\sum_{\vec{x},\vec{y}} r(\vec{x},\vec{y}) \log p(\vec{x}) - \sum_{\vec{x},\vec{y}} r(\vec{x},\vec{y}) \log q(\vec{y}) \\
&= -\sum_{\vec{x},\vec{y}} r(\vec{x},\vec{y}) \left(\log p(\vec{x}) + \log q(\vec{y}) \right) \\
&= -\sum_{\vec{x},\vec{y}} r(\vec{x},\vec{y}) \left(\log p(\vec{x}) q(\vec{y}) \right) = H_{X,Y}(R)
\end{aligned}$$

The bottom line is that any lack of independence between $\vec{x}$ and $\vec{y}$ is manifest by $I_{X,Y}(R) > 0$.

Another example scattergram shows the connection of ICA to the cocktail party problem:



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 closest 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 $\vec{q}_i = (q_{i,1}, \ldots, q_{i,p})$, where $(x_{i,1}, \ldots, x_{1,k}) = \sum_{c=1}^{p} q_{i,c}(v_1^{[c]}, v_2^{[c]}, ..., v_k^{[c]})$. 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.

The "mutual information" among several quantities (generalizing the idea of the mutual information between two quantitites, above) is the difference between the sum of their individual entropies, and the entropy of their joint distribution. Put another way, the entropy of a joint distribution is the difference between the sum of the entropies of its projections (to be found) and the mutual information between the projections. Applying this to the above setup:

$$H(\vec{x}) = \sum_{c=1}^{k} H(q_{\bullet,c}) - I(\vec{q}). \qquad (15)$$

That is, the entropy of the multivariate distribution sampled by the $\vec{x}_i$ is 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. (15) 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 corresponds to making the individual distributions $H(q_{\bullet,c})$ as *non*-Gaussian as possible. Another intuition makes use of the Central Limit Theorem: mixing distributions drives them towards Gaussian-ness, so un-mixing distributions should correspond to finding components that are as non-Gaussian as possible.

Many flavors of ICA

We also note that estimating entropy from empiric data is difficult. The basic reason is that in the definition of entropy, $H = -\sum p(\vec{x}) \log p(\vec{x})$, the probabilities $p(\vec{x})$ must be estimated from the data. Rare events contribute disproportionately because of the log term, but the probability of rare events is more difficult to estimate. Additionally, to use this plug-in estimate, one needs to choose a "bin size" in the above sum (which, really should be regarded as an integral). Large bins lead to blurring of estimates of the probability; small bins lead to inaccurate estimates because the number of samples are small.

This has led to the development of many variants of ICA in which, instead of minimum-entropy, some other measure of non-Gaussian-ness is used, such as maximal kurtosis.

Additionally, ICA algorithms can differ in terms of their search strategy.

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.