Are you normal? A new projection pursuit index to assess a sample against a multivariate null distribution.

June 24, 2024

1 Introduction

- Projection pursuit
- Elliptical distributions
- Explanations of application areas

2 Projecting a pD ellipse into 2D

Notation:

- $A_{p\times p}$ is the variance-covariance representing the normal population.
- $\mu_{p\times 1}$ vector of means of the normal population
- $P_{p\times 2}$ is the orthonormal basis on which the data is projected.
- $X_{n\times p}$ data matrix, let x represent a row, or abstractly a random p=dimensional vector.
- Equation of pD ellipse is $(\boldsymbol{x}-\mu)^\top A^{-1}(\boldsymbol{x}-\mu) = c^2$.
- Equation of projected ellipse is $(\boldsymbol{x}-\mu)^\top (P^\top AP)^{-1}(\boldsymbol{x}-\mu) = c^2$.
- $BB^\top = (P^\top AP)^{-1}$, so $B = (P^\top AP)^{-1/2}$
- The projected data is $\boldsymbol{y} = P\boldsymbol{x}$
- $B^{-1}\boldsymbol{y}$ are points that lie of the 2D ellipse
- c^2 is a quantile from a χ^2_p

3 Projection matrix onto a plane spanned by two vectors

For orthonormal p-dimensional vectors $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$, and a p-dimensional vector $\boldsymbol{x}$, the projection of $\boldsymbol{x}$ onto the plane spanned by $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ is given by:

$$(\boldsymbol{x} \cdot \boldsymbol{v_1})\boldsymbol{v_1} + (\boldsymbol{x} \cdot \boldsymbol{v_2})\boldsymbol{v_2} \tag{1}$$

Now, if we consider $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ to be our new basis vectors, we can write the projection with two coordinates as

$$\begin{bmatrix} — & \boldsymbol{v_1} & — \\ — & \boldsymbol{v_2} & — \end{bmatrix} \boldsymbol{x} = P\boldsymbol{x} \tag{2}$$

where P is the $2 \times p$ projection matrix onto the plane.

If we are given two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ that are not orthonormal, we can still construct an orthonormal projection matrix onto the plane they span. Let $\boldsymbol{m} = (\hat{\boldsymbol{u}} + \hat{\boldsymbol{v}})/|\hat{\boldsymbol{u}} + \hat{\boldsymbol{v}}|$ be the unit vector bisecting the angle between $\boldsymbol{u}$ and $\boldsymbol{v}$. Then $\boldsymbol{p} = (\hat{\boldsymbol{u}} - \hat{\boldsymbol{v}})/|\hat{\boldsymbol{u}} - \hat{\boldsymbol{v}}|$ is perpendicular to $\boldsymbol{m}$. Setting

$$P := \frac{1}{\sqrt{2}} \begin{bmatrix} — & \boldsymbol{m} + \boldsymbol{p} & — \\ — & \boldsymbol{m} - \boldsymbol{p} & — \end{bmatrix}, \tag{3}$$

we have an orthonormalised projection matrix spanning the same plane as $\boldsymbol{u}$ and $\boldsymbol{v}$.

4 Finding the projection of a hyperellipsoid onto a plane

Consider an ellipsoid of p-dimensional points $\boldsymbol{x}$ satisfying $\boldsymbol{x}^T A\boldsymbol{x} = c^2$, where A is a positive definite $p \times p$ matrix and c^2 is a constant dependent on the confidence interval. If the ellipsoid is constructed from a covariance matrix, A is the *inverse* of the given covariance matrix. Let L be the plane spanned by orthonormal vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. We aim to project this ellipsoid onto L. Let

$$P := \begin{bmatrix} — & \boldsymbol{u} & — \\ — & \boldsymbol{v} & — \end{bmatrix} \tag{4}$$

be the $2 \times p$ projection matrix onto L.

Let $f(\boldsymbol{x}) = \boldsymbol{x}^T A\boldsymbol{x}$. We want to find $\boldsymbol{x}$ for which $\nabla f(\boldsymbol{x})$ is parallel to L. These points, when projected onto L, will form the boundary of the projected ellipse.

Since $\nabla f(\boldsymbol{x}) = 2A\boldsymbol{x}$, we must have $A\boldsymbol{x}$ parallel to L. That is, there exists some 2×1 vector $\boldsymbol{s} = (s_1, s_2)$ such that $A\boldsymbol{x} = s_1\boldsymbol{u} + s_2\boldsymbol{v} = P^T\boldsymbol{s}$. We can set $\boldsymbol{x} = A^{-1}P^T\boldsymbol{s}$. For $\boldsymbol{x}$ to be a point on the ellipsoid, we require that $c^2 = \boldsymbol{x}^T A\boldsymbol{x}$; substituting out $\boldsymbol{x}$ gives

$$\begin{aligned} c^2 &= \boldsymbol{x}^T A\boldsymbol{x} && (5) \\ &= \boldsymbol{s}^T P(A^{-1})^T AA^{-1}P^T\boldsymbol{s} && (6) \\ &= \boldsymbol{s}^T PA^{-1}P^T\boldsymbol{s} && (7) \end{aligned}$$

where $(A^{-1})^T = A^{-1}$ by the symmetry of A. We see from this equation that $\boldsymbol{s}$ lies on an ellipse with 2×2 matrix $PA^{-1}P^T$.

Now, let $\bar{\boldsymbol{x}} := P\boldsymbol{x}$ be the projection of $\boldsymbol{x}$ onto L. For $\bar{\boldsymbol{x}}$ on the boundary of the projected ellipse, we have $\bar{\boldsymbol{x}} = P\boldsymbol{x} = PA^{-1}P^T\boldsymbol{s}$. Then, we can isolate $\boldsymbol{s} = (PA^{-1}P^T)^{-1}\bar{\boldsymbol{x}}$, and, substituting out $\boldsymbol{s}$, find

$$\begin{aligned} c^2 =& \boldsymbol{s}^T PA^{-1}P^T\boldsymbol{s} && (8)\\ =& \bar{\boldsymbol{x}}^T((PA^{-1}P^T)^{-1})^T PA^{-1}P^T(PA^{-1}P^T)^{-1}\bar{\boldsymbol{x}} && (9)\\ =& \bar{\boldsymbol{x}}^T(PA^{-1}P^T)^{-1}\bar{\boldsymbol{x}}. && (10)\end{aligned}$$

Again, $((PA^{-1}P^T)^{-1})^T = (PA^{-1}P^T)^{-1}$ by symmetry of A.

Therefore, the projection of our p-dimensional ellipsoid onto L is the ellipse with equation $\bar{\boldsymbol{x}}^T(PA^{-1}P^T)^{-1}\bar{\boldsymbol{x}} = c^2$.

In order to plot this ellipse, it can be useful to construct a sequence of points on its boundary. As $PA^{-1}P^T$ is positive definite, it can be deconstructed as $PA^{-1}P^T = BB^T$. Then our ellipse equation becomes

$$\begin{aligned} c^2 &= \bar{\boldsymbol{x}}^T(BB^T)^{-1}\bar{\boldsymbol{x}}\\ &= (B^{-1}\bar{\boldsymbol{x}})^T(B^{-1}\bar{\boldsymbol{x}}); \end{aligned}$$

that is, the points $B^{-1}\bar{\boldsymbol{x}}$ lie on the circle with radius c centred at the origin. As that circle is parameterised by $(c\cos(t), c\sin(t)),\ t \in [0, 2\pi)$, we can parameterise $\bar{\boldsymbol{x}}$ as

$$\bar{\boldsymbol{x}} = cB\begin{bmatrix}\cos(t)\\ \sin(t)\end{bmatrix} + P\boldsymbol{\mu}_E,\ t \in [0, 2\pi) \tag{11}$$

where $\boldsymbol{\mu}_E$ is the p-dimensional center of the hyperellipsoid. Plotting 50–100 such points gives a good resolution for the shape without taking too long to render.

The constant c depends on the confidence interval that the ellipse represents. In the application, users can choose to show projections of the hyperellipsoids corresponding to 1 s.d., 2 s.d., ..., 6 s.d. Let $\chi(t, p)$ denote the percentage point function of the chi-square distribution, with probability t and p degrees of freedom. Let $\Phi(z)$ be the cumulative distribution function of the standard, univariate normal distribution. For p-dimensional data, the ellipsoid corresponding to z standard deviations has

$$c^2 = \chi(2\Phi(z) - 1, p). \tag{12}$$

In p-dimensional space, we should then expect to have 68% of the data within 1 s.d., 95% of the data within 2 s.d., etc., matching a univariate normal distribution. After projection, however, more points may end up inside the ellipse. The only guarantee of the projected ellipse is if a point is *outside* an ellipse in the 2-dimensional projection, then the point is outside that ellipsoid in p-dimensional space. The converse is generally not true, but this is still an improvement on a slice ellipse, which cannot make either guarantee.

5 Derivation using tour notation

5.1 Notation

The main difference is that we consider row-vectors such x or mu are considered rows of an $(n \times p)$ data matrix, i.e. they are $(1 \times p)$ which is needed to work directly with projections as defined in the tour.

- $A_{p\times d}$ is the orthonormal basis defining the projection plane, for us $d = 2$
- $\mu_{1\times p}$ vector of means of the normal population
- $\Sigma_{p\times p}$ is the variance-covariance matrix for the normal population
- $X_{n\times p}$ data matrix, let x represent a row, or abstractly a random p=dimensional vector, but note that in our notation x is $(1 \times p)$

Our starting point is the p-dimensional ellipsoid defined by the variance-covariance matrix. For a selected size of the ellipsoid this is defined via the precision matrix, for points x on the surface we have

$$(x - \mu)\Sigma^{-1}(x - \mu)^T = c^2 \tag{13}$$

where c controls the size of the ellipsoid, in the special case of a sphere c would be the radius of the sphere. In our case of an ellipsoid defined via the variance-covariance matrix c measures the size with scale determined by the Malahanobis distance.

5.2 Projection of the ellipse

The projection of the ellipsoid onto the $d = 2$-dimensional plane defined via A is defining an ellipse. To find the outline of this ellipse consider that the projection should capture where the tangent hyperplane on the ellipsoid surface is normal on the projection plane. This is the point along the surface curvature where the direction is changing. This is equivalent to requiring that the gradient of the ellipsoid equation should be parallel to the projection plane. For simplicity here we assume $\mu = 0$.

We therefore are interested in the gradient of the function

$$f(x) = x\Sigma^{-1}x^T \tag{14}$$

which is

$$\nabla f(x) = 2x\Sigma^{-1} \tag{15}$$

and since we require that this is parallel to the plane defined via A it means that we can express the gradient in this basis:

$$x\Sigma^{-1} = sA^T \tag{16}$$

therefore we have

$$x = sA^T\Sigma \tag{17}$$

This is defining the second condition for x to define a point on the ellipse. The first condition was the requirement to be on the ellipsoid surface. Putting both conditions together by replacing x we get

$$sA^T\Sigma\Sigma^{-1}\Sigma^T As^T = c^2 \tag{18}$$

and using $\Sigma^T = \Sigma$ this gives

$$sA^T\Sigma As^T = c^2$$

(19)

Since s should be defined via a projection y of our p-dimensional x we rewrite it in those terms. From

$$y = xA = sA^T\Sigma A \tag{20}$$

we obtain

$$s = y(A^T\Sigma A)^{-1} \tag{21}$$

and substitute this into Eq. 5.2 to find

$$y(A^T\Sigma A)^{-1}A^T\Sigma A((A^T\Sigma A)^{-1})^T y^T = c^2 \tag{22}$$

again we can use the symmetry of A when simplifying the equation, and we arrive at our final equation for the ellipse outline

$$y(A^T\Sigma A)^{-1}y^T = c^2 \tag{23}$$

5.3 Drawing the ellipse

XXX not done, need to check this still

To draw the projected ellipse we use the decomposition of a real-valued positive definite matrix as

$$M = (A^T\Sigma A)^{-1} = BB^T \tag{24}$$

where we can define B as the square-root of M. This can be computed via the eigendecomposition of M, and taking the square-root of the eigenvalues. Using the decomposition we can rewrite equation (23) as

$$yBB^T y^T = c^2 \tag{25}$$

therefore yB defines a circle of radius c. Thus to find points on the ellipse we first sample points on a circle and then use B to find the corresponding points y.

6 A projection pursuit index using the projected ellipse

A shortfall of this orthogonal projection is that some interesting points, that are many standard deviations from $\boldsymbol{\mu}_E$ in p-dimensional space, can be projected inside the ellipse and not stand out. The proposed projection pursuit index aims to remedy this. Let W be a matrix where each column

is an 'interesting' p-dimensional point $\boldsymbol{w}$. Let F be a function on a projection matrix P, with parameter W, such that:

$$F(P;W) = \sum_{\boldsymbol{w}\in W} (\boldsymbol{w}-\boldsymbol{\mu}_E)^T P^T (PA^{-1}P^T)^{-1} P(\boldsymbol{w}-\boldsymbol{\mu}_E) \tag{26}$$

Notice that $P(\boldsymbol{w}-\boldsymbol{\mu}_E)$ is the 2D projection of the vector pointing from the centre of the ellipsoid to $\boldsymbol{w}$. Furthermore, $(PA^{-1}P^T)^{-1}$ is the 2D projection of the hyperellipsoid. That is, the summand of (26) gives the squared Mahalanobis distance of $P\boldsymbol{w}$ from $P\boldsymbol{\mu}_E$, with respect to the shape of the *projected* ellipse. The 'optimal' projection, for a given W, is the matrix P which maximises $F(P;W)$. A way of numerically finding P to do this is detailed in the appendix.

One way of selecting an interesting W is to focus on outliers. Specifically, W can be chosen as the set of points that are more than z standard deviations from $\boldsymbol{\mu}_E$ in p-dimensional space. In the python application, the value of z can be chosen by the user.

To make these changes more efficient, the data is sorted beforehand in order of increasing elliptical distance from $\boldsymbol{\mu}_E$; we also keep a list of these elliptical distances. When the user selects some z, we calculate a cutoff elliptical distance c from z by equation (12). We can binary search our list of elliptical distances for the index of this c. Then we get W by truncating our sorted data matrix at this index.

Alternatively, the user can select W using the lasso tool, specifying W to be the set of points whose projects land in the selected region.

7 Manual tour functionality

We have in our projection matrix p columns, where the i-th column is the 2D projection of the i-th basis vector of our high-dimensional space. When we drag the i-th basis vector to a position (A, B) in 2D space, we want to construct a new projection matrix with the i-th column equal to (A, B), or as close to it as possible.

Firstly, as (A, B) is the orthogonal projection of a unit vector, we must have $|(A, B)| \leq 1$. We normalise this vector if the user drags further out.

Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be the first and second rows of our new projection matrix P, so that the i-th component of $\boldsymbol{u}$ is A and the i-th component of $\boldsymbol{v}$ is B. Let $\boldsymbol{a}$ and $\boldsymbol{b}$ be $p-1$-dimensional vectors, obtained by removing the i-th component from $\boldsymbol{u}$ and $\boldsymbol{v}$ respectively. To have $\boldsymbol{u}$ and $\boldsymbol{v}$ perpendicular we require:

$$0 = \boldsymbol{u}\cdot\boldsymbol{v} = \boldsymbol{a}\cdot\boldsymbol{b} + AB$$

so that $\boldsymbol{a}\cdot\boldsymbol{b} = -AB$. Then, to have $\boldsymbol{u}$ and $\boldsymbol{v}$ normal, we require $|\boldsymbol{a}| = \sqrt{1-A^2}$, and $|\boldsymbol{b}| = \sqrt{1-B^2}$. Let θ be the angle between $\boldsymbol{a}$ and $\boldsymbol{b}$. Combining our previous requirements, we get

$$\cos(\theta) = \frac{\boldsymbol{a}\cdot\boldsymbol{b}}{|\boldsymbol{a}||\boldsymbol{b}|} = \frac{-AB}{\sqrt{(1-A^2)(1-B^2)}} \tag{27}$$

We can then construct $\boldsymbol{a}$ and $\boldsymbol{b}$ to be as close to the previous projection as possible, using their previous values $\boldsymbol{a}^*$ and $\boldsymbol{b}^*$. We set $\boldsymbol{p}$ to be the normal vector in the direction of $\boldsymbol{a}^* + \boldsymbol{b}^*$, and $\boldsymbol{m}$ to be the normal vector in the direction of $\boldsymbol{a}^* - \boldsymbol{b}^*$. We can find $\cos(\theta/2)$ and $\sin(\theta/2)$ using:

$$\cos(\theta/2) = \sqrt{\frac{1+\cos(\theta)}{2}}; \qquad \sin(\theta/2) = \sqrt{\frac{1-\cos(\theta)}{2}}$$

Then, set

$$\boldsymbol{a} = \sqrt{1-A^2}(\cos(\theta/2)\boldsymbol{p} + \sin(\theta/2)\boldsymbol{m})$$
$$\boldsymbol{b} = \sqrt{1-B^2}(\cos(\theta/2)\boldsymbol{p} - \sin(\theta/2)\boldsymbol{m})$$

It is then trivial to check, using (27), the orthonormality of $\boldsymbol{p}$ and $\boldsymbol{m}$, and the identity $\cos(\theta) = \cos^2(\theta/2) - \sin^2(\theta/2)$, that $\boldsymbol{a} \cdot \boldsymbol{b} = -AB$, $|\boldsymbol{a}| = \sqrt{1-A^2}$, and $|\boldsymbol{b}| = \sqrt{1-B^2}$. The corresponding vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ can then be stacked back into a new projection matrix.

Error handling. This method struggles when $\boldsymbol{a}^* + \boldsymbol{b}^*$ or $\boldsymbol{a}^* - \boldsymbol{b}^*$ turns out to be the zero vector. This is a frequent issue, occurring whenever the axes not being dragged are projected onto one line. In this case, the code sets the i-th components of $\boldsymbol{u}$ and $\boldsymbol{v}$ to A and B respectively, then reverts to the orthonormalisation method detailed in section 3.

8 Appendix: Numerically optimising the projection pursuit index

Let $C = A^{-1}$ be the covariance matrix of the data. For brevity we will suppose our dataset has been centred at the origin. As the matrix of the projected ellipse is $(PCP^T)^{-1}$, we need to find P to maximise

$$\sum_{\boldsymbol{w} \in W} \boldsymbol{w}^T P^T (PCP^T)^{-1} P \boldsymbol{w}.$$

Let the rows of P be $\boldsymbol{u}$ and $\boldsymbol{v}$. Suppose W has p rows with m columns, corresponding to m data points in p dimensions. We need a multivariate function f to maximise. Define $f : \mathbb{R}^{2n} \to \mathbb{R}$ as:

$$\begin{aligned}
f(\boldsymbol{u}, \boldsymbol{v}) :&= \sum_{\boldsymbol{w} \in W} \boldsymbol{w}^T \begin{bmatrix} | & | \\ \boldsymbol{u} & \boldsymbol{v} \\ | & | \end{bmatrix} \left(\begin{bmatrix} — & \boldsymbol{u} & — \\ — & \boldsymbol{v} & — \end{bmatrix} C \begin{bmatrix} | & | \\ \boldsymbol{u} & \boldsymbol{v} \\ | & | \end{bmatrix} \right)^{-1} \begin{bmatrix} — & \boldsymbol{u} & — \\ — & \boldsymbol{v} & — \end{bmatrix} \boldsymbol{w} \\
&= \sum_{\boldsymbol{w} \in W} \boldsymbol{w}^T \begin{bmatrix} \boldsymbol{u} \cdot \boldsymbol{w} & \boldsymbol{v} \cdot \boldsymbol{w} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^T C \boldsymbol{u} & \boldsymbol{u}^T C \boldsymbol{v} \\ \boldsymbol{v}^T C \boldsymbol{u} & \boldsymbol{v}^T C \boldsymbol{v} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{u} \cdot \boldsymbol{w} \\ \boldsymbol{v} \cdot \boldsymbol{w} \end{bmatrix} \boldsymbol{w} \\
&= \sum_{\boldsymbol{w} \in W} \frac{\begin{bmatrix} \boldsymbol{u} \cdot \boldsymbol{w} & \boldsymbol{v} \cdot \boldsymbol{w} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}^T C \boldsymbol{v} & -\boldsymbol{u}^T C \boldsymbol{v} \\ -\boldsymbol{v}^T C \boldsymbol{u} & \boldsymbol{u}^T C \boldsymbol{u} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \cdot \boldsymbol{w} \\ \boldsymbol{v} \cdot \boldsymbol{w} \end{bmatrix}}{(\boldsymbol{u}^T C \boldsymbol{u})(\boldsymbol{v}^T C \boldsymbol{v}) - (\boldsymbol{u}^T C \boldsymbol{v})^2} \\
&= \sum_{\boldsymbol{w} \in W} \frac{(\boldsymbol{u}^T C \boldsymbol{u})(\boldsymbol{v} \cdot \boldsymbol{w})^2 - 2(\boldsymbol{u}^T C \boldsymbol{v})(\boldsymbol{u} \cdot \boldsymbol{w})(\boldsymbol{v} \cdot \boldsymbol{w}) + (\boldsymbol{v}^T C \boldsymbol{v})(\boldsymbol{u} \cdot \boldsymbol{w})^2}{(\boldsymbol{u}^T C \boldsymbol{u})(\boldsymbol{v}^T C \boldsymbol{v}) - (\boldsymbol{u}^T C \boldsymbol{v})^2}
\end{aligned}$$

With $\odot$ denoting element-wise multiplication, let $\boldsymbol{g} : \mathbb{R}^{2n} \to \mathbb{R}^m$ be the function

$$\boldsymbol{g}(\boldsymbol{u}, \boldsymbol{v}) := (\boldsymbol{u}^T C \boldsymbol{u})(\boldsymbol{v}^T W \odot \boldsymbol{v}^T W) - 2(\boldsymbol{u}^T C \boldsymbol{v})(\boldsymbol{u}^T W \odot \boldsymbol{v}^T W) + (\boldsymbol{v}^T C \boldsymbol{v})(\boldsymbol{u}^T W \odot \boldsymbol{u}^T W)$$

and let $h : \mathbb{R}^{2n} \to \mathbb{R}$ be

$$h(\boldsymbol{u}, \boldsymbol{v}) := (\boldsymbol{u}^T C \boldsymbol{u})(\boldsymbol{v}^T C \boldsymbol{v}) - (\boldsymbol{u}^T C \boldsymbol{v})^2.$$

With $\boldsymbol{s}$ as the $1 \times m$ summation vector (vector of ones), we have $f = \boldsymbol{g}\boldsymbol{s}/h$. Then the vector $(d\boldsymbol{u}, d\boldsymbol{v})$ giving the direction of the steepest increase of f is

$$\begin{aligned}\nabla f(\boldsymbol{u},\boldsymbol{v}) &= (f_{\boldsymbol{u}}(\boldsymbol{u},\boldsymbol{v}), f_{\boldsymbol{v}}(\boldsymbol{u},\boldsymbol{v})) \\ &= (f_{\boldsymbol{u}}(\boldsymbol{u},\boldsymbol{v}), f_{\boldsymbol{u}}(\boldsymbol{v},\boldsymbol{u})) \quad \text{by symmetry of } f, \\ &= \left(\frac{h(\boldsymbol{u},\boldsymbol{v})\boldsymbol{g}_{\boldsymbol{u}}(\boldsymbol{u},\boldsymbol{v})\boldsymbol{s} - \boldsymbol{g}(\boldsymbol{u},\boldsymbol{v})\boldsymbol{s}h_{\boldsymbol{u}}(\boldsymbol{u},\boldsymbol{v})}{(h(\boldsymbol{u},\boldsymbol{v}))^2}, \frac{h(\boldsymbol{u},\boldsymbol{v})\boldsymbol{g}_{\boldsymbol{u}}(\boldsymbol{v},\boldsymbol{u})\boldsymbol{s} - \boldsymbol{g}(\boldsymbol{u},\boldsymbol{v})\boldsymbol{s}h_{\boldsymbol{u}}(\boldsymbol{v},\boldsymbol{u})}{(h(\boldsymbol{u},\boldsymbol{v}))^2}\right).\end{aligned}$$

Differentiating, we calculate $\boldsymbol{g}_{\boldsymbol{u}} : \mathbb{R}^{2n} \to \mathbb{R}^{n\times m}$ to be

$$\boldsymbol{g}_{\boldsymbol{u}}(\boldsymbol{u},\boldsymbol{v}) = 2(\boldsymbol{v}^T C\boldsymbol{v})(\boldsymbol{u}^T W)\odot W - 2C\boldsymbol{v}\otimes(\boldsymbol{u}^T W \odot \boldsymbol{v}^T W) - 2(\boldsymbol{u}^T C\boldsymbol{v})(\boldsymbol{v}^T W)\odot W + 2C\boldsymbol{u}\otimes(\boldsymbol{v}^T W \odot \boldsymbol{v}^T W)$$

and have $h_{\boldsymbol{u}} : \mathbb{R}^{2n} \to \mathbb{R}^n$ given by:

$$h_{\boldsymbol{u}}(\boldsymbol{u},\boldsymbol{v}) = 2(\boldsymbol{v}^T C\boldsymbol{v})C\boldsymbol{u} - 2(\boldsymbol{u}^T C\boldsymbol{v})C\boldsymbol{v}.$$

This gives us the means to calculate $\nabla f(\boldsymbol{u},\boldsymbol{v})$. From starting values of $\boldsymbol{u}$ and $\boldsymbol{v}$, the optimiser enters a loop; it calculates ∇f, scales it down to a size `step` (default 0.01), increments $\boldsymbol{u}$ and $\boldsymbol{v}$ by this vector, re-orthonormalises them, and recalculates f, keeping track of the increase Δf. When Δf falls below a given `tol` (default 0.00001), we exit the loop and return $\boldsymbol{u}$ and $\boldsymbol{v}$.

Finding the starting values. I wanted this algorithm to be deterministic, and independent of the order of the dimensions in the input file. The code attempts to start with a significant $\boldsymbol{u}$ and $\boldsymbol{v}$. It is possible to algebraically find a projection matrix P to maximise:

$$\sum_{\boldsymbol{w}\in W} \boldsymbol{w}^T P^T P A P^T P \boldsymbol{w}.$$

As A is a positive definite matrix, it has a Cholesky decomposition $A = BB^T$. Substituting this in, we are maximising

$$\begin{aligned}&\sum_{\boldsymbol{w}\in W} \boldsymbol{w}^T P^T P B B^T P^T P \boldsymbol{w} \\ =&\sum_{\boldsymbol{w}\in W} |B^T P^T P \boldsymbol{w}|^2\end{aligned}$$

Let $T := B^{-1}W$, with columns $\boldsymbol{t} = B^{-1}\boldsymbol{w}$. Let $K := PB$ be a $2 \times n$ projection matrix, so that $B^T P^T P\boldsymbol{w} = B^T P^T PB\boldsymbol{t} = K^T K\boldsymbol{t}$. Now we aim to find K to maximise

$$\sum_{\boldsymbol{t}\in T} |K^T K\boldsymbol{t}|^2.$$

That is, K is the best fitted plane to T, or equivalently the best *rank 2 approximation*. The method for finding this is well-documented. First, we modify T, setting $\hat{T} = (T - TS/m)^T$, where S is the $m \times m$ matrix of ones. That is, $\hat{T}$ is the translation of T to be centred at the origin.

Now we can calculate the singular value decomposition $\bar{T} = U\Sigma V^T$, where U is $m \times m$ orthonormal, Σ is $m \times n$ diagonal with entries decreasing along the diagonal, and V is $n \times n$ orthonormal. Then the first two rows of V^T give the best rank-2 approximation to $\bar{T}$ (see `https://www.cs.cmu.edu/~avrim/598/chap3only.pdf`); that is, they give the plane of best fit. These rows are already orthonormal, so we can set $K = V^T[:2]$.

Having found K, we can set $P = KB^{-1}$. We orthonormalise the rows of P to get our initial $\boldsymbol{u}$ and $\boldsymbol{v}$ and start our numerical optimising process.