Chapter 1

Independent Component Analysis(ICA)

ICA tries to find original signals $\hat{S}.$ from mixed signal X, called Blind Source Separation problem.

$$X = AS \qquad \rightarrow \qquad \hat{S} = WX$$

where

- $X \in \mathbb{R}^{\text{Nxp}}$
- $A, W \in \mathbb{R}^{\text{NxN}}$
- $S, \hat{S} \in \mathbb{R}^{\text{Nxp}}$
- N is no. sources
- p is no. observations in the source

One could say that ICA tries to find $W \approx A^{-1}$.

1.1 ICA vs PCA

The goal of PCA is to find directions that don't correlate with each other(orthogonal projections). However, ICA finds sources that are independent to each other. The right most figure of Fig 1.1 shows that even correlationC_{12}, C_{21} is zero, but x_1, x_2 are dependent to each other(knowing one of them can tell position of the other).

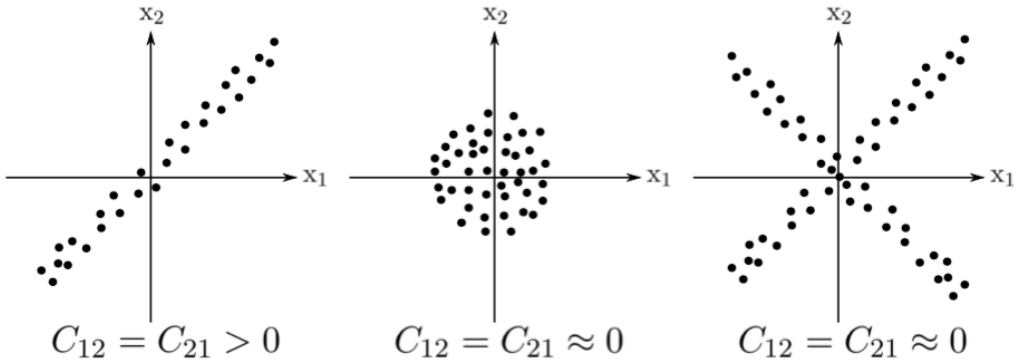

Figure 1.1: Decorrelation vs Indenpendent (Drawn from MI2 Slides)

1.2 ICA Limitations

1.2.1 Permutation of Sources

$\hat{\boldsymbol{s}}_i$ that found that ICA is not necessary the same as the original source $\boldsymbol{s}_i$.

1.2.2 Discovered Amplitude

1.2.3 Gaussian Sources

The underly theory behind ICA is Central Limit Theorem in which it implies that sum of random variables will approach Gaussian distribution. Thus, if the original sources come from Gaussian distribution, we won't have any clue to separate the sources.

1.3 ICA Approaches

An ICA algorithm can be derived using different cost functions using assumption that

$$\hat{P}_{\boldsymbol{s}}(\hat{\boldsymbol{s}}) = \prod_{i=1}^{N} \hat{P}_{s_i}(\hat{s}_i)$$

and $P_{\boldsymbol{s}}(\hat{\boldsymbol{s}}) = P_{\boldsymbol{s}}(W\boldsymbol{x})$

1.3.1 Informax

From Kullback-Leibler divergence, we know that

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$$

Thus, we can formulate the problem as

$$\min_{W} D_{KL}(P_{\boldsymbol{s}}(\hat{\boldsymbol{s}})||\hat{P}_{\boldsymbol{s}}(\hat{\boldsymbol{s}}))$$

However, deriving cost from D_{KL} directly would not work as it require probability density estimation. Hence, we shall use a transformation :

$$\hat{u}_i = \hat{f}(\hat{s}_i) \qquad \text{s.t.} \qquad \hat{P}_{u_i}(\hat{u}_i) = \text{const.}$$

From Conservation of Probability, we know that

$$\hat{P}_{u_i}(\hat{u}_i)d\hat{u}_i = \hat{P}_{s_i}(\hat{s}_i)d\hat{s}_i$$

Then

$$\begin{aligned}
\hat{P}_{u_i} &= \left|\frac{d\hat{s}_i}{d\hat{u}_i}\right| \hat{P}_{s_i}(\hat{s}_i) \\
&= \left|\frac{1}{\frac{d\hat{u}_i}{d\hat{s}_i}}\right| \hat{P}_{s_i}(\hat{s}_i) \\
&= \left|\frac{1}{\hat{f}'(\hat{s}_i)}\right| \hat{P}_{s_i}(\hat{s}_i) \\
&\stackrel{!}{=} 1
\end{aligned}$$

This implies that $\hat{f}(\hat{s}_i)$ is CDF of $\hat{P}_{s_i}(\hat{s}_i)$. From this knowledge, we can derive D_{KL} :

$$\begin{aligned}
D_{KL}(P_{\boldsymbol{s}}(\hat{\boldsymbol{s}})||\hat{P}_{\boldsymbol{s}}(\hat{\boldsymbol{s}})) &= \int_{-\infty}^{\infty} d\hat{\boldsymbol{s}} P_{\boldsymbol{s}}(\hat{\boldsymbol{s}}) \log \frac{P_{\boldsymbol{s}}(\hat{\boldsymbol{s}})}{\prod_{i=1}^{N} \hat{P}_{s_i}(\hat{s}_i)} \\
&= \int_{-\infty}^{\infty} d\hat{\boldsymbol{s}} P_{\boldsymbol{s}}(\hat{\boldsymbol{s}}) \log \frac{P_{\boldsymbol{s}}(\hat{\boldsymbol{s}})}{\prod_{i=1}^{N} \hat{P}_{s_i}(\hat{s}_i)} \frac{\prod_{i=1}^{N} 1/\hat{f}'}{\prod_{i=1}^{N} 1/\hat{f}'} \\
&= \int_{-\infty}^{\infty} d\hat{\boldsymbol{u}} P_{\boldsymbol{u}}(\hat{\boldsymbol{u}}) \log \frac{P_{\boldsymbol{u}}(\hat{\boldsymbol{u}})}{\prod_{i=1}^{N} \hat{P}_{u_i}(\hat{u}_i)} \\
&= \int_{-\infty}^{\infty} d\hat{\boldsymbol{u}} P_{\boldsymbol{u}}(\hat{\boldsymbol{u}}) \log P_{\boldsymbol{u}}(\hat{\boldsymbol{u}}) - \int_{-\infty}^{\infty} d\hat{\boldsymbol{u}} P_{\boldsymbol{u}}(\hat{\boldsymbol{u}}) \log \prod_{i=1}^{N} \hat{P}_{u_i}(\hat{u}_i)
\end{aligned}$$

Hence,

$$\begin{aligned}\min D_{KL}(P_{\boldsymbol{s}}(\hat{\boldsymbol{s}})||\hat{P}_{\boldsymbol{s}}(\hat{\boldsymbol{s}})) &\approx \min \int_{-\infty}^{\infty} d\hat{\boldsymbol{u}} P_{\boldsymbol{u}}(\hat{\boldsymbol{u}}) \log P_{\boldsymbol{u}}(\hat{\boldsymbol{u}}) \\ &\approx \min -H(u) \\ &\approx \max H(u) \qquad \text{(Informax Principle)}\end{aligned}$$

Learning via Neural Network and Empirical Risk Minimization

From Conservation of Probability, we know that

$$\begin{aligned}P(\hat{\boldsymbol{u}})d\hat{\boldsymbol{u}} &= P(\hat{\boldsymbol{s}})d\hat{\boldsymbol{s}} \\ &= P(\boldsymbol{x})d\boldsymbol{x}\end{aligned}$$

and,

$$\begin{aligned}\hat{\boldsymbol{u}} &= \hat{f}(\hat{\boldsymbol{s}}) \\ &= \hat{f}(W\boldsymbol{x})\end{aligned}$$

Thus,

$$\begin{aligned}P(\hat{\boldsymbol{u}}) &= P(\boldsymbol{x})d\boldsymbol{x} \\ &= \left|\frac{d\boldsymbol{x}}{d\hat{\boldsymbol{u}}}\right| P(\boldsymbol{x}) \\ &= \frac{1}{|M|} P(\boldsymbol{x})\end{aligned}$$

where $|M|$ is functional determinant and

$$\begin{aligned}|M| &= \left|\frac{\partial \hat{\boldsymbol{u}}}{\partial \boldsymbol{x}}\right| \\ &= |W| \prod_{l=1}^{N} \hat{f}'\Big(\sum_{k=1}^{N} w_{lk} x_k\Big)\end{aligned}$$

Hence,

$$\begin{aligned}H(\boldsymbol{u}) &= -\int d\hat{\boldsymbol{u}} P_{\boldsymbol{u}}(\hat{\boldsymbol{u}}) \log P_{\boldsymbol{u}}(\hat{\boldsymbol{u}}) \\ &= -\int d\boldsymbol{x} P_{\boldsymbol{x}}(\boldsymbol{x}) \log \frac{P_{\boldsymbol{x}}(\hat{\boldsymbol{x}})}{|M|}) \\ &= -\int d\boldsymbol{x} P_{\boldsymbol{x}}(\boldsymbol{x}) \log P_{\boldsymbol{x}} + \int d\boldsymbol{x} P_{\boldsymbol{x}}(\boldsymbol{x}) \log |M|\end{aligned}$$

Because $-\int d\boldsymbol{x} P_{\boldsymbol{x}}(\boldsymbol{x}) \log P_{\boldsymbol{x}}$ is a constant if we optimize over w, therefore, the cost function E^G is reduced to

$$\begin{aligned}E^G = & \int d\boldsymbol{x} P_{\boldsymbol{x}}(\boldsymbol{x}) \log |M| \\ &= \int d\boldsymbol{x} P_{\boldsymbol{x}}(\boldsymbol{x}) \log |W| + \int d\boldsymbol{x} P_{\boldsymbol{x}}(\boldsymbol{x}) \log \prod_{l=1}^{N} \hat{f}'\Big(\sum_{k=1}^{N} w_{lk} x_k\Big) \\ &= \log |W| + \int d\boldsymbol{x} P_{\boldsymbol{x}}(\boldsymbol{x}) \Bigg\{ \sum_{l=1}^{N} \log \hat{f}'\Big(\sum_{k=1}^{N} w_{lk} x_k\Big) \Bigg\}\end{aligned}$$

Using Empirical Risk Minimization, we convert expectation to average thus

$$\begin{aligned}E^T &= \log |W| + \frac{1}{p} \sum_{\alpha=1}^{p} \sum_{l=1}^{N} \log \hat{f}'\Big(\sum_{k=1}^{N} w_{lk} x_k^{(\alpha)}\Big) \\ &\stackrel{!}{=} \max_{W}\end{aligned}$$

We can learn W using gradient ascent :

$$W \leftarrow W + \eta \nabla_W E^T$$

For individual cost of each observation $e^{(\alpha)}$,

$$\frac{\partial e^{(\alpha)}}{\partial w_{ij}} = W^{-1}_{ji} + \frac{\hat{f}''\left(\sum_{k=1}^{N} w_{ik} x_k^{(\alpha)}\right)}{\hat{f}'\left(\sum_{k=1}^{N} w_{ik} x_k^{(\alpha)}\right)} x_j^{(\alpha)}$$

Or

$$\frac{\partial}{\partial W} e^{(\alpha)} = (W^{-1})^T + \psi(W x^{(\alpha)})(x^{(\alpha)})^T$$

One could observe that there is W^{-1} in the computation which causes performance.

Learning via Natural Gradient

Due to the fact that W space is not orthogonal coordinate, hence gradient doest point to the direction of steepest ascent. Thus, we need to transform the space back to comparable space before computing gradient. This yields

$$dZ = dW \cdot W^{-1}$$

Using Taylor expansion of $e^{(\alpha)}$ or e for short, around W, we have :

$$\begin{aligned} e(W + dW) &= e(W) + \nabla_W e^T dW \\ &= e(W) + \eta \nabla_W e^T z_w \end{aligned}$$

where $z_w = \eta dw$
To find the direction of steepest ascent, we have to

$$\nabla_W e^T z_w \stackrel{!}{=} \max \qquad \text{s.t.} \qquad (z_w \cdot W^{-1})^2 \stackrel{!}{=} 1$$

Using Lagrange multiplier we have

$$\mathcal{L}(z_w, \lambda) = \nabla_W e^T z_w - \lambda((z_w \cdot W^{-1})^2 - 1)$$

Taking derivative wrt to z_w and set to zero, we have

$$\begin{aligned} 0 &= \nabla_W e - 2\lambda(z_w \cdot W^{-1})(W^{-1})^T \\ \nabla_W e &= 2\lambda(z_w \cdot W^{-1})(W^{-1})^T \\ z_w &= \frac{1}{2\lambda} \nabla_W e W^T W \end{aligned}$$

Hence the direction for **Natural Gradient** is

$$\Delta W = \eta \nabla_W e W^T W$$

where $\nabla_W e$ is original gradient.
Putting things together, we have

$$\Delta W = \eta \left(\mathbb{I} + \psi(Wx)(Wx)^T \right) W$$

Commonly, people use the sigmoid function as CDF :

$$f(x) = \frac{1}{1 + \exp(-x)} \qquad \rightarrow \qquad \psi(x) = \frac{f''(x)}{f'(x)} = 1 - 2f(x)$$

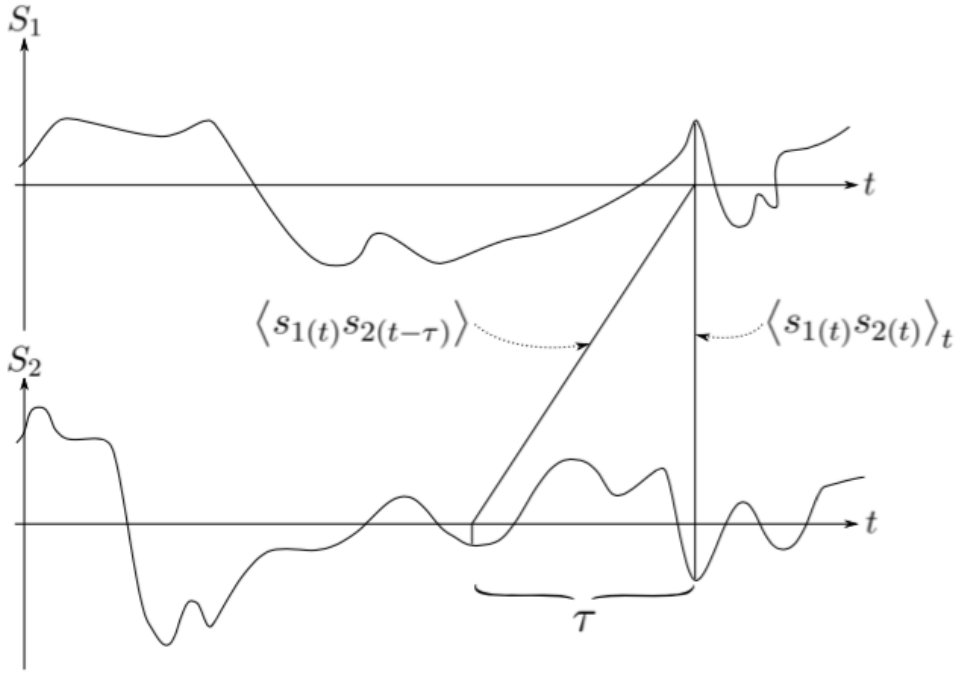

Figure 1.2: Autocorrelation with time τ shifted (Drawn from MI2 Slides)

1.3.2 Second-order Approach

Due to the fact that, higher moment measurements are sensitive to noise, especially when we have small set of samples, hence it might yield unreliable result. Moreover, signals, such as EEG, fMRI data, might correlate to each other in temporal pattern. Because most of the time we know the length of the signal, we can exploit this fact by using autocorrelation.

Formally, the motivation of 2nd order approach is to find unmixing matrix W that cross-correlation varnishes.

Source Separation with 2 shifts

1. Whitening data

$$\boldsymbol{x}_c = \boldsymbol{x} - \frac{1}{p} \sum_{\alpha} x^{(\alpha)}$$

Then compute eigen value decomposition of $C_x^{(0)}$ where $(C_x)_{ij} =< x_i, x_j >$. Hence,

$$\begin{aligned} \boldsymbol{u} &= \Lambda_0^{-1} \cdot E_0 \boldsymbol{x} \\ &= M_0 \boldsymbol{x} \end{aligned}$$

where Λ_0^{-1} is a diagonal of inverse eigen values and and E_0 is matrix of eigen vectors.

2. Orthogonal Transformation

Assume : $\boldsymbol{s} = B\boldsymbol{u}$. We know that

$$\begin{aligned} < s_i s_j > &= \delta_{ij} \\ &= \sum_{k,l=1}^{N} B_{ik} < u_k u_l > B_{l,j}^T \\ &= \sum_{k,l=1}^{N} B_{ik} B_{l,j}^T \qquad (< u_k u_l >= 1 \text{ from Whitening}) \end{aligned}$$

Hence,

$$BB^T = B^T B = \mathbb{I} \rightsquigarrow \textbf{Orthogonal Transformation}$$

3. Find a candidate of B using time-shifted cross-correlation matrix Apply eigen value decomposition on $C_u^{(\tau)}$ where
$$(C_u^{(\tau)})_{ij} =< u_i^{(t)} u_j^{(t-\tau)} >$$
This results in a matrix of eigen vectors E_τ whose $E_\tau^T E_\tau = \mathbb{I}$ Putthing thing together, we have
$$\hat{\boldsymbol{s}} = E_\tau \Lambda_0^{-1} E_0 \boldsymbol{x}$$

Noise robust algorithms in general case

Given a set of T zero mean observation : $\boldsymbol{x^{(t)}}$. We compute joint cross-correlation metric $C_x^{(\tau)}$ as :
$$(C_x^{(\tau)})_{ij} = \frac{1}{T} \sum_{t=0}^{T-1} x_i^{(t)} x_j^{(t-\tau)}$$

To find NxN matrix W that diagonalize $C^{(\tau)}, \tau = 0, 1, ...,$ we use sum of off-diagonal entries of $WC^{(\tau)}W^T$ as a cost function.

1. QDIAG Algorithm
$$E_W^T = \sum_\tau \alpha_\tau \sum_{i \neq j} \left(WC^{(\tau)}W^T \right)_{ij}^2$$
where α_τ is a weighting factor and the optmization problem is $E^T \stackrel{!}{=} \min$ s.t.
$$\forall i, \quad \left(WC^{(0)}W^T \right)_{ii} = 1$$

2. FFDIAG Algorithm
$$E_W^T = \sum_\tau \sum_{i \neq j} \left(WC^{(\tau)}W^T \right)_{ij}^2$$
with a constraint that W is a non-singular matrix.

1.3.3 Maximize Non-Gaussianity

According to Central Limit Theorem, the sum of random variables is more Gaussian than the original sources. Hence, we can leverage this knowledge by finding a unmixing matrix that minimizes Gaussianity of the unmixed sources yielding interesting directions (projection pursuit). There are 2 ways to measure Gaussianity, namely Kurtosis (4th moment) and Negentropy.

Kurtosis
$$\text{kurt}(x) = \langle x^4 \rangle - 3 \underbrace{(\langle x^2 \rangle)^2}_{\substack{1 \text{ for} \\ \text{sphered data}}}$$

- $\text{kurt}(x) = 0$ for Gaussian distribution
- $\text{kurt}(x) > 0$ for **Super**-Gaussian : long-tail, e.g. Laplace Distribution
- $\text{kurt}(x) < 0$ for **Sub**-Gaussian : constant distribution.

For independent variable x and y we have :
$$\text{kurt}(x + y) = \text{kurt}(x) + \text{kurt}(y)$$
$$\text{kurt}(ax) = a^4 \text{kurt}(x)$$

Because,
$$\hat{\boldsymbol{s}} = W\boldsymbol{x} = WA\boldsymbol{s} = b^T \widetilde{A} \boldsymbol{s} = b^T \boldsymbol{u}$$

Hence, the optimization problem can be written as

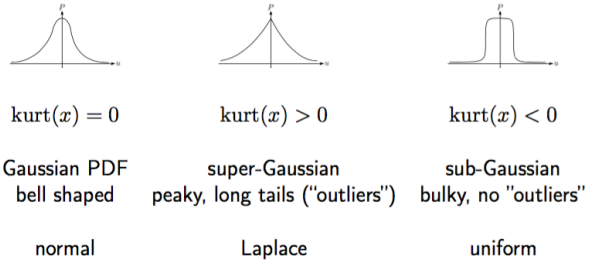

Figure 1.3: Distribution and Kurtosis (Drawn from MI2 Slides)

$$\max_{\boldsymbol{b}} |\text{kurt}(\boldsymbol{b}^T\boldsymbol{u})| \qquad \text{s.t.} \quad |\widetilde{A}^T\boldsymbol{b}| = |\boldsymbol{b}| = 1$$

We have :

$$\frac{\partial}{\partial \boldsymbol{b}}|\text{kurt}(\boldsymbol{b}^T\boldsymbol{u})| = 4\text{sgn}[\text{kurt}(\boldsymbol{b}^T\boldsymbol{u})]\Big(\langle \boldsymbol{u}(\boldsymbol{b}^T\boldsymbol{u})^3\rangle - 3\boldsymbol{b}\langle(\boldsymbol{b}^T\boldsymbol{u})^2\rangle\Big)$$
$$\stackrel{!}{=} 0 \quad \text{s.t.} \quad |\boldsymbol{b}| = 1$$

Due to the fact that

$$\begin{aligned} 3\boldsymbol{b}\langle(\boldsymbol{b}^T\boldsymbol{u})^2\rangle &= 3\boldsymbol{b}\langle((\boldsymbol{b}^T\boldsymbol{u})(\boldsymbol{b}^T\boldsymbol{u})^T\rangle \\ &= 3\boldsymbol{b}\langle((\boldsymbol{b}^T\boldsymbol{u})(\boldsymbol{u}^T\boldsymbol{b})\rangle \\ &= 3\boldsymbol{b}\boldsymbol{b}^T\langle\boldsymbol{u}\boldsymbol{u}^T\rangle\boldsymbol{b} \\ &= 3\boldsymbol{b}\boldsymbol{b}^T\mathbb{I}\boldsymbol{b} \\ &= 3\boldsymbol{b}|\boldsymbol{b}|^2 \\ &= 3\boldsymbol{b} \qquad (|\boldsymbol{b}|^2 = 1) \end{aligned}$$

which changes only direction of b, we can drop it and hence the problem is reduced to

$$4\text{sgn}[\text{kurt}(\boldsymbol{b}^T\boldsymbol{u})]\langle\boldsymbol{u}(\boldsymbol{b}^T\boldsymbol{u})^3\rangle \stackrel{!}{=} 0$$
$$\epsilon\text{sgn}[\text{kurt}(\boldsymbol{b}^T\boldsymbol{u})]\langle\boldsymbol{u}(\boldsymbol{b}^T\boldsymbol{u})^3\rangle \approx 0$$

Therefore for **batch** learning, we have

$$\Delta\boldsymbol{b} = \epsilon\text{sgn}[\text{kurt}(\boldsymbol{b}^T\boldsymbol{u})]\langle\boldsymbol{u}(\boldsymbol{b}^T\boldsymbol{u})^3\rangle$$
$$\boldsymbol{b} \leftarrow \boldsymbol{b}/|\boldsymbol{b}|$$

where kurt and $\langle\cdot\rangle$ are replaced with corresponding empirical method and b_0 is a arbitrary vector with unit length.

For **online** approach, we use running average of kurt γ where $\gamma_0 = 0$.

$$\Delta\boldsymbol{b} = \epsilon\text{sgn}[\gamma]\langle\boldsymbol{u}(\boldsymbol{b}^T\boldsymbol{u})^3\rangle$$
$$\Delta\gamma = \eta[(\boldsymbol{b}^T\boldsymbol{u})^4 - 3 - \gamma]$$
$$\boldsymbol{b} \leftarrow \boldsymbol{b}/|\boldsymbol{b}|$$

At the equilibrium point of gradient ascent, one could observes that

$$\boldsymbol{b} \propto \Delta\boldsymbol{b}$$
$$\rightsquigarrow \langle\boldsymbol{u}(\boldsymbol{b}^T\boldsymbol{u})^3\rangle - 3\boldsymbol{b}$$

This yields fixed-point algorithm (**Kurtosis-based fastICA**)

$$\boldsymbol{b} \leftarrow \langle\boldsymbol{u}(\boldsymbol{b}^T\boldsymbol{u})^3\rangle - 3\boldsymbol{b}$$
$$\boldsymbol{b} \leftarrow \boldsymbol{b}/|\boldsymbol{b}|$$

For whitened data $\boldsymbol{u}^{(\alpha)}, \alpha = 1, .., p$, we have

$$\boldsymbol{b} \leftarrow \frac{1}{p} \sum_{\alpha=1}^{p} \boldsymbol{u}^{(\alpha)} (\boldsymbol{b}^T \boldsymbol{u}^{(\alpha)})^3 - 3\boldsymbol{b}$$
$$\boldsymbol{b} \leftarrow \boldsymbol{b}/|\boldsymbol{b}|$$

One remark is that **kurtosis** tends to be very sensitive to outliers.

1.3.4 Negentropy