Chapter 1

Neural Network and Learning Algorithm

Chapter 2

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

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

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

2.1 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 2.1 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).

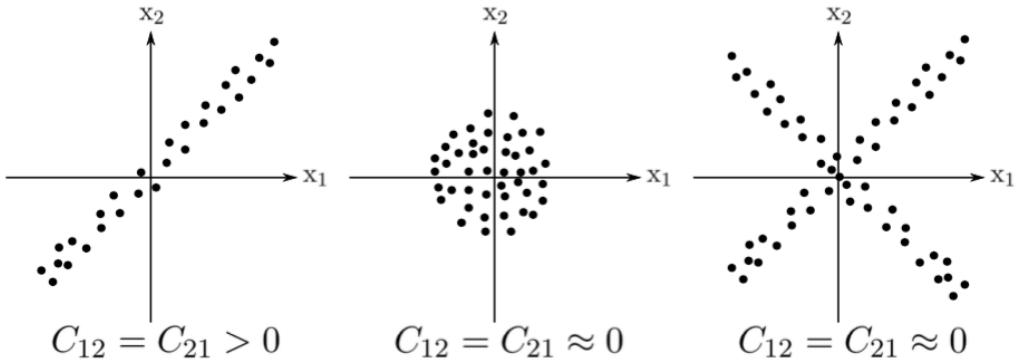

Figure 2.1: Decorrelation vs Indenpendent (Drawn from MI2 Slides)

2.2 ICA Limitations

2.2.1 Permutation of Sources

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

2.2.2 Discovered Amplitude

2.2.3 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.

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

2.3.1 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 :

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

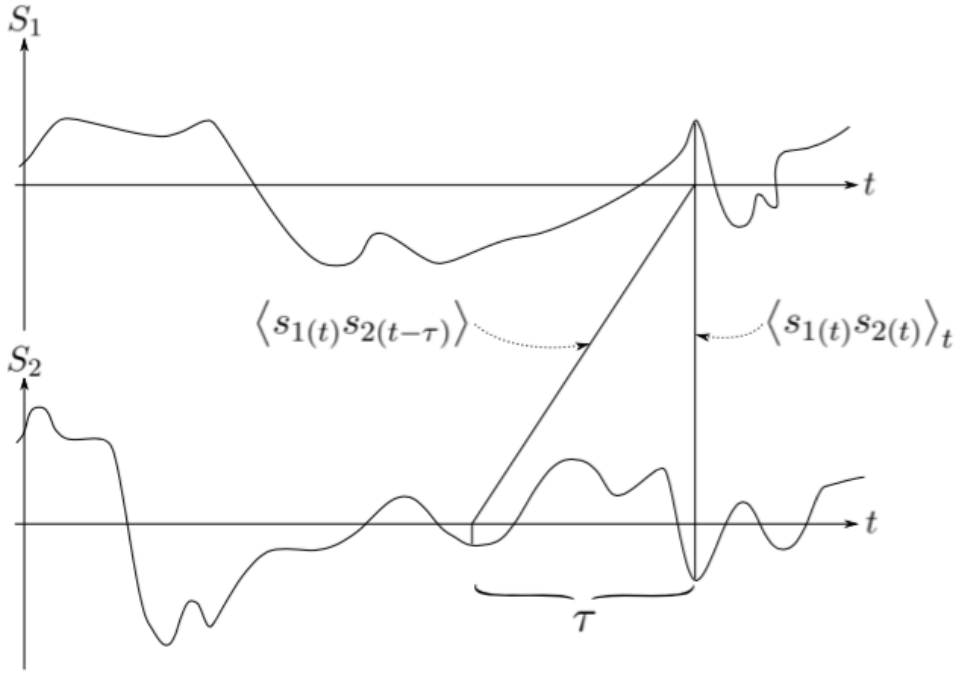

Figure 2.2: Autocorrelation with time τ shifted (Drawn from MI2 Slides)

2.3.2 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,

$$\begin{aligned} \boldsymbol{u} &= \Lambda_0^{-1/2} \cdot E_0 \boldsymbol{x} \\ &= M_0 \boldsymbol{x} \end{aligned}$$

where $\Lambda_0^{-1/2}$ 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.

2.3.3 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

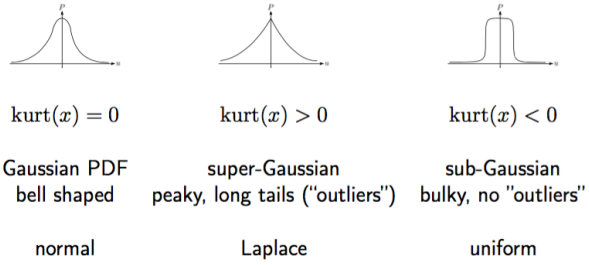

Figure 2.3: Distribution and Kurtosis (Drawn from MI2 Slides)

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

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

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

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

2.3.4 Negentropy

$$J(\hat{\boldsymbol{s}}) = H^{\text{Gauss}}_{(\hat{\boldsymbol{s}})} - H_{(\hat{\boldsymbol{s}})}$$

Properties of $J(\hat{\boldsymbol{s}})$, non-negative and scale-invariant : $J(\alpha\hat{\boldsymbol{s}}) = J(\hat{\boldsymbol{s}}), \forall\alpha \neq 0$

Computing $J(\hat{\boldsymbol{s}})$ directly requires estimation of density $p(\hat{\boldsymbol{s}})$. Hence, we shall use **Nonpolynomial moment** contrast function G:

$$J(\hat{\boldsymbol{s}}) \approx (\langle G(\hat{\boldsymbol{s}})\rangle_{\mathbb{P}_{(\boldsymbol{s})}} - \langle G(\hat{\boldsymbol{s}})\rangle_{\text{Gauss}})^2$$

Choices of Contrast Function

General Purpose

$$G_1(\hat{\boldsymbol{s}}) = \frac{1}{a}\log\cosh(a\hat{\boldsymbol{s}}) \quad G_1'(\hat{\boldsymbol{s}}) = \tanh(a\hat{\boldsymbol{s}}) \quad G_1''(\hat{\boldsymbol{s}}) = a(1-\tanh^2(a\hat{\boldsymbol{s}}))$$

Good for super Gaussian sources with many outliers

$$G_2(\hat{\boldsymbol{s}}) = -\exp\left(-\frac{\hat{\boldsymbol{s}}^2}{2}\right) \quad G_2'(\hat{\boldsymbol{s}}) = \hat{\boldsymbol{s}}\exp\left(-\frac{\hat{\boldsymbol{s}}^2}{2}\right) \quad G_2''(\hat{\boldsymbol{s}}) = (1-\hat{\boldsymbol{s}}^2)\exp\left(-\frac{\hat{\boldsymbol{s}}^2}{2}\right)$$

Good for sub-Gaussian sources with few outliers

$$G_3(\hat{\boldsymbol{s}}) = \frac{1}{4}\hat{\boldsymbol{s}}^4 \quad G_3'(\hat{\boldsymbol{s}}) = \hat{\boldsymbol{s}}^3 \quad G_3''(\hat{\boldsymbol{s}}) = 3\hat{\boldsymbol{s}}^2$$

The optimization problem can be written as :

$$J(\boldsymbol{b}^T\boldsymbol{u}) = \left(\langle G(\boldsymbol{b}^T\boldsymbol{u})\rangle_{\mathbb{P}_{(\boldsymbol{u})}} - \langle G(\hat{\boldsymbol{u}}_{\text{Gauss}})\rangle_{\mathcal{N}_{(0,1)}}\right)^2$$

Batch Learning

$$\begin{aligned}\Delta\boldsymbol{b} &= \epsilon\left\{\langle G(\boldsymbol{b}^T\boldsymbol{u})\rangle_{\mathbb{P}_{(\boldsymbol{u})}} - \langle G(\hat{\boldsymbol{u}}_{\text{Gauss}})\rangle_{\mathcal{N}_{(0,1)}}\right\}\langle\boldsymbol{u}G'(\boldsymbol{b}^T\boldsymbol{u})\rangle_{\mathbb{P}_{(u)}} \\ \boldsymbol{b} &\leftarrow \boldsymbol{b}/|\boldsymbol{b}|\end{aligned}$$

where $\boldsymbol{b}_0$ is a random unit vector.

Online Learning

$$\begin{aligned}\Delta\boldsymbol{b} &= \epsilon\gamma\langle\boldsymbol{u}G'(\boldsymbol{b}^T\boldsymbol{u})\rangle_{\mathbb{P}_{(u)}} \\ \boldsymbol{b} &\leftarrow \boldsymbol{b}/|\boldsymbol{b}| \\ \Delta\gamma &= \eta(\langle G(\boldsymbol{b}^T\boldsymbol{u})\rangle_{\mathbb{P}_{(\boldsymbol{u})}} - \langle G(\hat{\boldsymbol{u}}_{\text{Gauss}})\rangle_{\mathcal{N}_{(0,1)}} - \gamma)\end{aligned}$$

where $\gamma_0 = 0$.

Fixed point algorithm (fastICA)

$$\boldsymbol{b} = \langle \boldsymbol{u} G'(\boldsymbol{b}^T \boldsymbol{u}) \rangle - \langle G''(\boldsymbol{b}^T \boldsymbol{u}) \rangle \boldsymbol{b}$$
$$\boldsymbol{b} \leftarrow \boldsymbol{b}/|\boldsymbol{b}|$$

One Unit Neural Network Implementation

1. whiten data $\boldsymbol{u}^{(\alpha)} = \Lambda_0^{-1/2} \mathrm{E}^T x_{\mathrm{centered}}^{(\alpha)}$
2. choose a random unit vector $\boldsymbol{b}$
3. Loop

$$\boldsymbol{b} \leftarrow \frac{1}{p} \left\{ \sum_{\alpha=1}^{p} \boldsymbol{u}^{(\alpha)} G'(\boldsymbol{b}^T \boldsymbol{u}^{(\alpha)}) - \boldsymbol{b} \sum_{\alpha=1}^{p} \boldsymbol{u}^{(\alpha)} G''(\boldsymbol{b}^T \boldsymbol{u}^{(\alpha)}) \right\}$$
$$\boldsymbol{b} \leftarrow \boldsymbol{b}/|\boldsymbol{b}|$$

and $\boldsymbol{w}$ for original centered data is $\boldsymbol{w} = \mathrm{E}\Lambda^{-1/2}\boldsymbol{b}$.

Multiple Units Neural Network Implementation

1. whiten data $\boldsymbol{u}^{(\alpha)} = \Lambda_0^{-1/2} \mathrm{E}^T x_{\mathrm{centered}}^{(\alpha)}$
2. choose a random orthogonal matrix $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_N)$.
3. Outer Loop
4. Inner Loop for all i

$$\boldsymbol{b}_i \leftarrow \frac{1}{p} \left\{ \sum_{\alpha=1}^{p} \boldsymbol{u}^{(\alpha)} G'(\boldsymbol{b}_i^T \boldsymbol{u}^{(\alpha)}) - \boldsymbol{b}_i \sum_{\alpha=1}^{p} \boldsymbol{u}^{(\alpha)} G''(\boldsymbol{b}_i^T \boldsymbol{u}^{(\alpha)}) \right\}$$

5. End inner loop and do orthogonalization

$$B \leftarrow B/\max_i |\boldsymbol{b}_i|$$
$$B \leftarrow (BB^T)^{-\frac{1}{2}} B$$

and W for original centered data is $W = \mathrm{E}\Lambda^{-1/2}\boldsymbol{b}$.

Chapter 3

Stochastic Optimization

3.1 Simulated Annealing

The goal of Simulated Annealing(SA) is to find a state $\boldsymbol{s}$ that minimize a cost function $E_{T(\boldsymbol{s})}$ where $s_i \in \boldsymbol{s}$ is a discrete variable.

$$\boldsymbol{s}^* = \underset{\boldsymbol{s}}{\operatorname{argmin}} E_{T(\boldsymbol{s})}$$

Data: $\boldsymbol{s}_0, \tau, M, \beta_0$
for *annealing loop,* $t = 1, 2, ..$ **do**
  $\boldsymbol{s}_t = \boldsymbol{s}_{t-1}$
  for M*iterations* **do**
    $\boldsymbol{s}'$ = do local permutation from $\boldsymbol{s}_t$;
    $\Delta E = E_{\boldsymbol{s}'} - E_{\boldsymbol{s}_t}$;
    switch to $\boldsymbol{s}_t$ to $\boldsymbol{s}'$ with probability $W = \frac{1}{1+\exp(\beta_t \Delta E)}$
  end
  $\beta_{t+1} = \tau \beta_t \quad \tau \in [1.01 - ...]$
end

Algorithm 1: Simulated Annealing

where $\beta = 1/\tau$ and $M \approx 2000$. Ideally, we want $\beta_t = \ln t$ but this is too slow in practice.
Influence of T and β to transition probability W.

- $\downarrow T \uparrow \beta$: W changes **dramatically** when ΔE has a slight change.
- $\uparrow T \downarrow \beta$: W **barely** changes when ΔE has a huge change.

3.2 The Gibbs Distribution

If we fix T or β at a specific value, this results in a stochastic process with Markov property[1]. Hence $P_{(\boldsymbol{s})}$ converges to a stationary distribution as $t \to \infty$.

$$\lim_{t' \to \infty} \prod(\boldsymbol{s}, t') = P_{(\boldsymbol{s})}$$

With **Detailed Balance** assumption which states as follow :

$$P_{(\boldsymbol{s})} W_{\boldsymbol{s} \to \boldsymbol{s}'} = P_{(\boldsymbol{s}')} W_{\boldsymbol{s}' \to \boldsymbol{s}}$$

We can find $P_{(\boldsymbol{s})}$ analytically :

[1] what is this property?

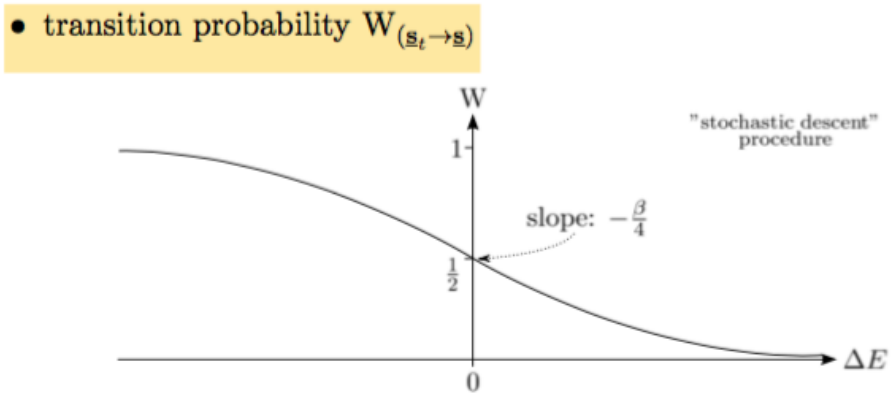

Figure 19: Transition probabilities

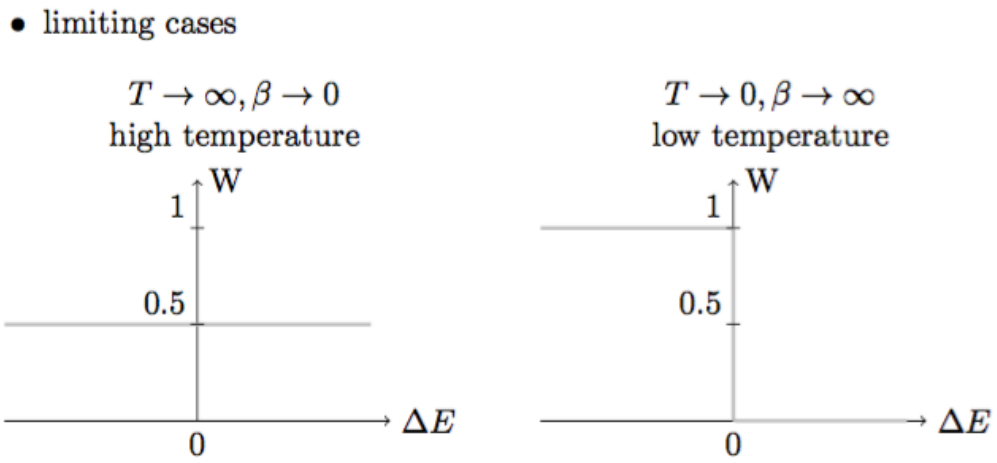

Figure 3.1: Transition Probability

$$\begin{aligned}\frac{P_{(\boldsymbol{s})}}{P_{(\boldsymbol{s}')}} &= \frac{W_{\boldsymbol{s}' \to \boldsymbol{s}}}{W_{\boldsymbol{s} \to \boldsymbol{s}'}} \\ &= \frac{1 + \exp\left(\beta(E_{\boldsymbol{s}'} - E_{\boldsymbol{s}})\right)}{1 + \exp\left(\beta(E_{\boldsymbol{s}} - E_{\boldsymbol{s}'})\right)} \\ &= \frac{1 + \exp\left(\beta \Delta E\right)}{1 + \exp\left(-\beta \Delta E\right)} \\ &= \exp\left(\beta \Delta E\right)\end{aligned}$$

To fulfill this condition, we need $P_{(\boldsymbol{s})}$ to be a Gibbs-Boltzmann Distribution

$$P_{(\boldsymbol{s})} = \frac{1}{Z} \exp\left(-\beta E_{\boldsymbol{s}}\right)$$

where $Z = \sum_{\boldsymbol{s}} \exp\left(-\beta E_{\boldsymbol{s}}\right)$.

3.3 Mean-field Annealing

As we know that $P_{(\boldsymbol{s})}$ has an analytic solution, we can exploit this knowledge to make the optimization faster, but estimate $P_{(\boldsymbol{s})}$ is difficult and its moment doesn't have analytic solution.

To alleviate the problem, we will choose a distribution $Q_{(\boldsymbol{s})}$ that approximates $P_{(\boldsymbol{s})}$. Formally, $Q_{(\boldsymbol{s})}$ is a factorizing distribution that E_Q is a linear combination of s_k.

$$\begin{aligned} Q_{\boldsymbol{s}} &= \frac{1}{Z_Q} \exp\left(-\beta E_Q \right) \\ &= \frac{1}{Z_Q} \exp\left(-\beta \sum_k e_k s_k \right) \end{aligned}$$

e_k is the mean field that parameterizes family of distributions. As $\beta \to \infty$, the first moment $\langle \boldsymbol{s} \rangle_Q$ well characterizes the distributions. For example, it will peak at the location of optimum value.

For factorized distribution,

$$Q_{(\boldsymbol{s})} = \prod_k Q_{(\boldsymbol{s}_k)}$$

and the moment of 2 independent variables $f_{(\boldsymbol{s}/s_l)}$ and $g_{(s_l)}$ of $\boldsymbol{s}$ can be computed as follows :

$$\begin{aligned} \langle f_{(\boldsymbol{s}/s_l)} g_{(s_l)} \rangle &= \frac{1}{Z_Q} \sum_{\boldsymbol{s}} f_{(\boldsymbol{s}/s_l)} g_{(s_l)} \exp\left(-\beta \sum_{s_k \in \boldsymbol{s}} e_k s_k \right) \\ &= \frac{1}{Z_Q} \left[f_{(\boldsymbol{s}/s_l)} g_{(s_{l,i})} \exp\left(-\beta \sum_{s_k \in \boldsymbol{s}/s_l} e_k s_k \right) \exp\left(-\beta e_l s_{l,i} \right) + \dots \right] \\ &= \frac{1}{Z_Q} \left[f_{(\boldsymbol{s}/s_l)} \exp\left(-\beta \sum_{s_k \in \boldsymbol{s}/s_l} e_k s_k \right) \left(\sum_{s_{l,i} \in s_l} g_{s_{l,i}} \exp\left(-\beta e_l s_{l,i} \right) \right) + \dots \right] \\ &= \frac{1}{Z_Q} \left[\sum_{\boldsymbol{s}/s_l} f_{(\boldsymbol{s}/s_l)} \exp\left(-\beta \sum_{s_k \in \boldsymbol{s}/s_l} e_k s_k \right) \right] \left[\sum_{s_{l,i} \in s_l} g_{(s_{l,i})} \exp\left(-\beta e_l s_{l,i} \right) \right] \\ &= \frac{1}{Z_Q} \left[\sum_{\boldsymbol{s}/s_l} f_{(\boldsymbol{s}/s_l)} \exp\left(-\beta \sum_{s_k \in \boldsymbol{s}/s_l} e_k s_k \right) \right] \frac{\sum_{s_{l,i} \in s_l} \exp\left(-\beta e_l s_{l,i} \right)}{\sum_{s_{l,i} \in s_l} \exp\left(-\beta e_l s_{l,i} \right)} \left[\sum_{s_{l,i} \in s_l} g_{(s_{l,i})} \exp\left(-\beta e_l s_{l,i} \right) \right] \\ &= \underbrace{\frac{1}{Z_Q} \left[\sum_{\boldsymbol{s}/s_l} f_{(\boldsymbol{s}/s_l)} \exp\left(-\beta \sum_{s_k \in \boldsymbol{s}} e_k s_k \right) \right]}_{\langle f_{(\boldsymbol{s}/s_l)} \rangle_Q} \underbrace{\frac{1}{\sum_{s_{l,i} \in s_l} \exp\left(-\beta e_l s_{l,i} \right)} \left[\sum_{s_{l,i} \in s_l} g_{(s_{l,i})} \exp\left(-\beta e_l s_{l,i} \right) \right]}_{\langle g_{(s_l)} \rangle_Q} \end{aligned}$$

Therefore,

$$\langle s_l \rangle_Q = \frac{\sum_{s_{l,i} \in s_l} s_{l,i} \exp\left(-\beta e_l s_{l,i} \right)}{\sum_{s_{l,i} \in s_l} \exp\left(-\beta e_l s_{l,i} \right)}$$

The mean-field approximation

- True Distribution : $P_{(\boldsymbol{s})} = \frac{1}{Z} \exp\left(-\beta E_p \right)$
- Approximation Distribution : $Q_{(\boldsymbol{s})} = \frac{1}{Z} \exp\left(-\beta \sum_k e_k s_k \right)$ and e_k are mean fields which we shall find.

To find e_k, we minimize Kullback-Leibler Divergence as follow :

$$\begin{aligned} D_{KL}(Q||P) &= \sum_{\boldsymbol{s}} Q_{(\boldsymbol{s})} \ln \frac{Q_{(\boldsymbol{s})}}{P_{(\boldsymbol{s})}} \\ &\stackrel{!}{=} \min \end{aligned}$$

To find e_l, we do $\frac{\partial D_{KL}}{\partial e_l} \stackrel{!}{=} 0$.

$$
\begin{aligned}
D_{KL} &= \sum_{\boldsymbol{s}} Q_{(\boldsymbol{s})} \ln Q_{(\boldsymbol{s})} - Q_{(\boldsymbol{s})} \ln P_{(\boldsymbol{s})} \\
&= \sum_{\boldsymbol{s}} Q_{(\boldsymbol{s})} \ln \frac{1}{Z_Q} \exp\left(-\beta E_Q\right) - Q_{(\boldsymbol{s})} \ln \frac{1}{Z_P} \exp\left(-\beta E_P\right) \\
&= \sum_{\boldsymbol{s}} Q_{(\boldsymbol{s})} \Big(- \beta E_Q - \ln Z_Q \Big) - Q_{(\boldsymbol{s})} \Big(- \beta E_P - \ln Z_P \Big) \\
&= \sum_{\boldsymbol{s}} -\beta E_Q Q_{(\boldsymbol{s})} - Q_{(\boldsymbol{s})} \ln Z_Q + \beta E_P Q_{(\boldsymbol{s})} + Q_{(\boldsymbol{s})} \ln Z_P \\
&= \sum_{\boldsymbol{s}} -\beta E_Q Q_{(\boldsymbol{s})} - \sum_{\boldsymbol{s}} Q_{(\boldsymbol{s})} \ln Z_Q + \sum_{\boldsymbol{s}} \beta E_P Q_{(\boldsymbol{s})} + \sum_{\boldsymbol{s}} Q_{(\boldsymbol{s})} \ln Z_P \\
&= \sum_{\boldsymbol{s}} -\beta E_Q Q_{(\boldsymbol{s})} - \ln Z_Q + \sum_{\boldsymbol{s}} \beta E_P Q_{(\boldsymbol{s})} + \ln Z_P \\
&= \sum_{\boldsymbol{s}} -\beta E_Q Q_{(\boldsymbol{s})} - \ln Z_Q + \beta \langle E_P \rangle_Q + \ln Z_P
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
\frac{\partial}{\partial e_l} D_{KL} &= \frac{\partial}{\partial e_l} \sum_{\boldsymbol{s}} -\beta E_Q Q_{(\boldsymbol{s})} - \frac{\partial}{\partial e_l} \ln Z_Q + \beta \frac{\partial}{\partial e_l} \langle E_P \rangle_Q + \frac{\partial}{\partial e_l} \ln Z_P \\
&= -\beta \sum_{\boldsymbol{s}} \frac{\partial}{\partial e_l} E_Q Q_{(\boldsymbol{s})} - \frac{1}{Z_Q} \frac{\partial}{\partial e_l} Z_Q + \beta \frac{\partial}{\partial e_l} \langle E_P \rangle_Q \\
&= -\beta \sum_{\boldsymbol{s}} \left[E_Q \frac{\partial}{\partial e_l} Q_{(\boldsymbol{s})} + Q_{(\boldsymbol{s})} \frac{\partial}{\partial e_l} E_Q \right] - \frac{1}{Z_Q} \frac{\partial}{\partial e_l} \sum_{\boldsymbol{s}} \exp\Big(- \beta \sum_k e_k s_k \Big) + \beta \frac{\partial}{\partial e_l} \langle E_P \rangle_Q \\
&= -\beta \sum_{\boldsymbol{s}} \left[\sum_k e_k s_k \frac{\partial}{\partial e_l} Q_{(\boldsymbol{s})} + Q_{(\boldsymbol{s})} s_l \right] - \frac{1}{Z_Q} \sum_{\boldsymbol{s}} \exp\Big(- \beta \sum_k e_k s_k \Big) (-\beta s_l) + \beta \frac{\partial}{\partial e_l} \langle E_P \rangle_Q \\
&= -\beta \sum_{\boldsymbol{s}} \left[\sum_k e_k s_k \frac{\partial}{\partial e_l} Q_{(\boldsymbol{s})} \right] - \beta \left[\sum_{\boldsymbol{s}} Q_{(\boldsymbol{s})} s_l \right] + \frac{\beta}{Z_Q} \sum_{\boldsymbol{s}} s_l \exp\Big(- \beta \sum_k e_k s_k \Big) + \beta \frac{\partial}{\partial e_l} \langle E_P \rangle_Q \\
&= -\beta \sum_k \left[e_k \frac{\partial}{\partial e_l} \sum_s s_k Q_{(\boldsymbol{s})} \right] - \beta \left[\sum_{\boldsymbol{s}} Q_{(\boldsymbol{s})} s_l \right] + \beta \sum_{\boldsymbol{s}} Q_{\boldsymbol{s}} s_l + \beta \frac{\partial}{\partial e_l} \langle E_P \rangle_Q \\
&= -\beta \sum_k \left[e_k \frac{\partial}{\partial e_l} \langle s_k \rangle_Q \right] + \beta \frac{\partial}{\partial e_l} \langle E_P \rangle_Q \\
&\stackrel{!}{=} 0
\end{aligned}
$$

Hence,

$$
\sum_k \left[e_k \frac{\partial}{\partial e_l} \langle s_k \rangle_Q \right] = \frac{\partial}{\partial e_l} \langle E_P \rangle_Q
$$

Due to the independent assumption that $s_i \perp s_j$, yields

$$
\frac{\partial}{\partial e_l} \langle E_P \rangle_Q - e_l \frac{\partial}{\partial e_l} \langle s_l \rangle_Q = 0
$$

Mean Field for Ising Model

$\boldsymbol{s} = \{s_i\}_{i=1,..,N}$ where $s_i \in \mathcal{S} = -1, 1$ and

$$
E_{(\boldsymbol{s})} = -\frac{1}{2} \sum_{i,j=1, i \neq j}^{N} W_{ij} s_i s_j
$$

where W is a real symmetric matric with zero diagonal entries.

Algorithm 6: Mean Field Annealing

initialization: $\langle \underline{\mathbf{s}} \rangle_0, \beta_0, t = 0$
begin Annealing loop
  repeat
    calculate mean-fields: $e_k, \quad k = 1, \ldots, N$
    calculate moments: $\langle s_k \rangle_Q, \quad k = 1, \ldots, N$
  until $|e_k^{\text{old}} - e_k^{\text{new}}| < \varepsilon$
  increase β
end

$\Rightarrow$ inner loop: fixed-point iteration for the mean-fields e_k

$\Rightarrow$ the moments $\langle s_k \rangle$ for not too small temperatures are in general not from the discrete set $\mathcal{S}$ but range continuously between the elements of $\mathcal{S}$

$\Rightarrow$ however: $\beta \to \infty (T \to 0) : \langle s_k \rangle \to s_k^*$
because $P_{(\underline{\mathbf{s}})}$ becomes singular at the state $\underline{\mathbf{s}}^*$ of minimal cost!

$\Rightarrow$ deterministic (fast) rather than stochastic (slow) optimization method (given that mean-field equations can be easily evaluated)

Figure 3.2: Drawn from MI2 course note

1. The first moment $\langle s_l \rangle_Q$ can be computed follows :

$$\begin{aligned}
\langle s_l \rangle_Q &= \frac{\sum_{s_{l,i} \in s_l} s_{l,i} \exp\left(- \beta e_l s_{l,i} \right)}{\sum_{s_{l,i} \in s_l} \exp\left(- \beta e_l s_{l,i} \right)} \\
&= \frac{\exp(-\beta e_l) - \exp(\beta e_l)}{\exp(-\beta e_l) + \exp(\beta e_l)} \\
&= \tanh(-\beta e_l)
\end{aligned}$$

2. e_l is computed by

$$\begin{aligned}
0 &= \frac{\partial}{\partial e_l} \langle E_P \rangle_Q - e_l \frac{\partial}{\partial e_l} \langle s_l \rangle_Q \\
&= \frac{\partial}{\partial e_l} \left\langle -\frac{1}{2} \sum_{i,j=1, i \neq j}^{N} W_{ij} s_i s_j \right\rangle_Q - e_l \frac{\partial}{\partial e_l} \langle s_l \rangle_Q \\
&= -\frac{1}{2} \frac{\partial}{\partial e_l} \sum_{i,j=1, i \neq j}^{N} W_{ij} \langle s_i \rangle_Q \langle s_j \rangle_Q - e_l \frac{\partial}{\partial e_l} \langle s_l \rangle_Q \\
&= \sum_{i=1, i \neq l}^{N} W_{il} \langle s_i \rangle_Q \frac{\partial}{\partial e_l} \langle s_l \rangle_Q - e_l \frac{\partial}{\partial e_l} \langle s_l \rangle_Q
\end{aligned}$$

Hence,

$$e_l = \sum_{i=1, i \neq l}^{N} W_{il} \langle s_i \rangle_Q$$

Chapter 4

Clustering and Embedding

Instead of finding interesting projections from data, we can use clustering algorithms to find groups or categories of the data. Each cluster is represented by a prototype $\boldsymbol{w}$.

Given set of $\boldsymbol{x}^{(\alpha)}, \alpha = 1, \ldots, p; \boldsymbol{x}^{(\alpha)} \in \mathbb{R}^N$, we want to assign them into M clusters where the indicator

$$m_q^{(\alpha)} = \begin{cases} 1, & \text{if } \boldsymbol{x}^{(\alpha)} \in \text{cluster } q \\ 0, & \text{otherwise} \end{cases}$$

and $\sum_q m_q^{(\alpha)} = 1$.

4.1 K-Means Clustering

4.1.1 Cost function

$$E^T_{[\{m_q^{(\alpha)}\},\{\boldsymbol{w}_q\}]} = \frac{1}{p} \sum_{\alpha,q} m_q^{(\alpha)} \left(\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q \right)^2$$

- Cluster center is defined by continuous variable
- Cluster assign is binary
- Dissimilarity is measured by Euclidean distance.

Data: $\boldsymbol{w}_q = \langle \boldsymbol{x} \rangle + \boldsymbol{\eta}_q$ where $\boldsymbol{\eta}_q$ is a random vector
while *not converge* **do**
  $\forall \alpha \in p : m_q^{(\alpha)} = 1 \left\{ q = \underset{\gamma}{\operatorname{argmin}} \left(\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_\gamma \right)^2 \right\}$
  $\forall q \in M : \boldsymbol{w}_q = \frac{\sum_\alpha m_q^{(\alpha)} \boldsymbol{x}^{(\alpha)}}{\sum_\alpha m_q^{(\alpha)}}$
end

Algorithm 2: Batch K-means

- E^T is monotonic decreasing.
- $\boldsymbol{w}_q$ is center of mass $\rightarrow$ E^T is total variance.

Data: $\boldsymbol{w}_q = \langle \boldsymbol{x} \rangle + \boldsymbol{\eta}_q$ where $\boldsymbol{\eta}_q$ is a random vector
Data: $\epsilon \in [0, 1]$
while *not converge* **do**
  Select a point $\boldsymbol{x}^{(\alpha)}$
  $q \leftarrow \underset{\gamma}{\operatorname{argmin}} \left(\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_\gamma \right)^2$
  $\boldsymbol{w}_q \leftarrow \boldsymbol{w}_q + \epsilon(\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q)$
end

Algorithm 3: Online K-means

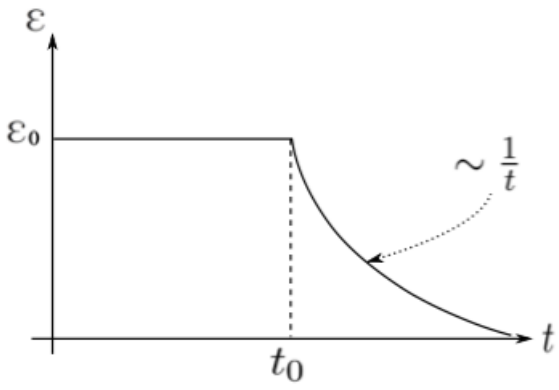

Figure 4.1: ϵ Annealing Schedule

4.1.2 Number of prototypes

- $E^T \propto \frac{1}{M}$
- $E^T = 0$ when $M = p$

Data: $\boldsymbol{w}_q = \frac{1}{p}\sum_\alpha \boldsymbol{x}^{(\alpha)}$
Data: $(E^T)^* =$ some constant
Data: $M = 1$
while $E^T > (E^T)^*$ **do**

$$q \leftarrow \underset{\gamma \in M}{\operatorname{argmax}} \frac{\sum_\alpha m_\gamma^{(\alpha)} (\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q)^2}{\sum_\alpha m_\gamma^{(\alpha)}}$$

$\boldsymbol{w}_{M+1} \leftarrow \boldsymbol{w}_q + n_q$
$M \leftarrow M + 1$
do regular K-Means algorithm with M prototypes
end

Algorithm 4: Iterative refinement

4.1.3 Validation Measure

Given $Y = (y_1, ..., y_p), y_i \in \{1, .., M\}$ solution of the clustering algorithm. We can measure the solutions Y_1, Y_2 by

$$d = \frac{1}{|Y_1|} \sum_\alpha \mathbf{1}\{Y_{1,\alpha} \neq Y_{2,\alpha}\}$$

Algorithm 4: Validation measure

begin for each $M \in \{M_{min}, \ldots, M_{max}\}$
  begin loop for r splits of data
    Split data $\mathbf{X}$ into $\mathbf{X}_1^{(i)}$ and $\mathbf{X}_2^{(i)}$ and find corresponding clustering solutions $\mathbf{Y}_1^{(i)}$ and $\mathbf{Y}_2^{(i)}$
    Compute dissimilarity $d_i := \frac{1}{|\mathbf{X}_2|}\sum_\alpha \mathbf{1}\left\{\mathbf{Y_2}^{(\alpha)} \neq \phi_1[\mathbf{X}_2^{(\alpha)}]\right\}$
    Where ϕ_1 denotes the label of the nearest prototype taking into account permutations
  end
  Compute average dissimilarity: $\hat{S}_{clustering} = \frac{1}{r}\sum_r d_i$

  Sample s random clustering assignments and compute estimate average of the distances to estimate $\hat{S}_{random}$

  Calculate stability index: $\bar{S}_M = \frac{\hat{S}_{clustering}}{\hat{S}_{random}}$
end
Return $\hat{M} = \text{argmin}_M(\bar{S}_M)$

Figure 4.2: Validation measure

4.2 Pairwise Clustering

distance matrix $\{d_{\alpha\alpha'}\}$ represents pairwise data and it is symmetric and its on diagonal elements are zero.

4.2.1 Cost Function

$$
\begin{aligned}
E_{[\{m_q^{(\alpha)}\}]} &= \frac{1}{2p}\sum_q \frac{\sum_{\alpha\alpha'} m_q^{(\alpha)} m_q^{(\alpha')} (\boldsymbol{x}^{(\alpha)} - \boldsymbol{x}^{(\alpha')})^2}{\sum_\alpha m_q^{(\alpha)}} \\
&= \frac{1}{2p}\sum_q \frac{\sum_{\alpha\alpha'} m_q^{(\alpha)} m_q^{(\alpha')} \Big((\boldsymbol{x}^{(\alpha)})^2 - 2(\boldsymbol{x}^{(\alpha)})^T \boldsymbol{x}^{(\alpha')} + (\boldsymbol{x}^{(\alpha')})^2 \Big)}{\sum_\alpha m_q^{(\alpha)}} \\
&= \frac{1}{2p}\sum_q \frac{\sum_{\alpha\alpha'} m_q^{(\alpha)} m_q^{(\alpha')} (\boldsymbol{x}^{(\alpha)})^2}{\sum_\alpha m_q^{(\alpha)}} - 2\frac{\sum_{\alpha\alpha'} m_q^{(\alpha)} m_q^{(\alpha')} \boldsymbol{x}^{(\alpha)} \boldsymbol{x}^{(\alpha')}}{\sum_\alpha m_q^{(\alpha)}} + \frac{\sum_{\alpha\alpha'} m_q^{(\alpha)} m_q^{(\alpha')} (\boldsymbol{x}^{(\alpha')})^2}{\sum_\alpha m_q^{(\alpha)}} \\
&= \frac{1}{2p}\sum_q \frac{\sum_\alpha m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^2 \sum_{\alpha'} m_q^{(\alpha')}}{\sum_\alpha m_q^{(\alpha)}} - 2\frac{\sum_\alpha m_q^{(\alpha)} \boldsymbol{x}^{(\alpha)} \sum_{\alpha'} m_q^{(\alpha')} \boldsymbol{x}^{(\alpha')}}{\sum_\alpha m_q^{(\alpha)}} + \frac{\sum_\alpha m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^2 \sum_{\alpha'} m_q^{(\alpha')}}{\sum_\alpha m_q^{(\alpha)}} \\
&= \frac{1}{2p}\sum_q \sum_\alpha m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^2 - 2\sum_\alpha m_q^{(\alpha)} x^{(\alpha)} \frac{\sum_{\alpha'} m_q^{(\alpha')} \boldsymbol{x}^{(\alpha')}}{\sum_{\alpha'} m_q^{(\alpha')}} + \sum_\alpha m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^2 \\
&= \frac{1}{p}\sum_q \sum_\alpha m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^2 - \sum_\alpha m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^T \boldsymbol{w}_q \\
&= \frac{1}{p}\left\{ \sum_{q,\alpha} m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^2 - \sum_q \sum_\alpha m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^T \boldsymbol{w}_q \right\} \\
&= \frac{1}{p}\left\{ \sum_{q,\alpha} m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^2 - \sum_q \sum_\alpha m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^T \boldsymbol{w}_q - \sum_q \sum_\alpha m_q^{(\alpha)} \boldsymbol{w}_q^T \boldsymbol{w}_q + \sum_q \sum_\alpha m_q^{(\alpha)} \boldsymbol{w}_q^T \boldsymbol{w}_q \right\} \\
&= \frac{1}{p}\left\{ \sum_{q,\alpha} m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^2 - \sum_q \sum_\alpha m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^T \boldsymbol{w}_q - \sum_q \sum_\alpha m_q^{(\alpha)} \frac{\sum_{\alpha'} m_q^{(\alpha')} (\boldsymbol{x}^{(\alpha')})^T}{\sum_{\alpha'} m_q^{(\alpha')}} \boldsymbol{w}_q + \sum_q \sum_\alpha m_q^{(\alpha)} \boldsymbol{w}_q^T \boldsymbol{w}_q \right\} \\
&= \frac{1}{p}\left\{ \sum_{q,\alpha} m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^2 - \sum_q \sum_\alpha m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^T \boldsymbol{w}_q - \sum_q \frac{\sum_\alpha m_q^{(\alpha)}}{\sum_{\alpha'} m_q^{(\alpha')}} \sum_{\alpha'} m_q^{(\alpha')} (\boldsymbol{x}^{(\alpha')})^T \boldsymbol{w}_q + \sum_q \sum_\alpha m_q^{(\alpha)} \boldsymbol{w}_q^T \boldsymbol{w}_q \right\} \\
&= \frac{1}{p}\left\{ \sum_{q,\alpha} m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^2 - \sum_q \sum_\alpha m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^T \boldsymbol{w}_q - \sum_q \sum_{\alpha'} m_q^{(\alpha')} (\boldsymbol{x}^{(\alpha')})^T \boldsymbol{w}_q + \sum_q \sum_\alpha m_q^{(\alpha)} \boldsymbol{w}_q^T \boldsymbol{w}_q \right\} \\
&= \frac{1}{p}\left\{ \sum_{q,\alpha} m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^2 - 2\sum_q \sum_\alpha m_q^{(\alpha)} (\boldsymbol{x}^{(\alpha)})^T \boldsymbol{w}_q + \sum_q \sum_\alpha m_q^{(\alpha)} \boldsymbol{w}_q^T \boldsymbol{w}_q \right\} \\
&= \frac{1}{p}\sum_{q,\alpha} m_q^{(\alpha)} \left\{ (\boldsymbol{x}^{(\alpha)})^2 - 2(\boldsymbol{x}^{(\alpha)})^T \boldsymbol{w}_q + \boldsymbol{w}_q^T \boldsymbol{w}_q \right\} \\
&= \frac{1}{p}\sum_{q,\alpha} m_q^{(\alpha)} \left\{ \boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q \right\}^2 \\
&\implies \text{K-means cost function}
\end{aligned}
$$

4.2.2 Mean-field approximation for pairwise-clustering

Define

- $\mathcal{M}$: Cartesian product of all assignments for $\forall\alpha$

- $\mathcal{M}_\gamma = \mathcal{M}/\{\boldsymbol{m}^{(\gamma)}\}$

From Gibbs Distribution, we know that

$$\begin{aligned}
\langle m_q^{(\gamma)} \rangle &= \frac{1}{\sum_{\mathcal{M}} \exp\left(-\beta \sum_{r,\delta} m_r^{(\delta)} e_r^{(\delta)} \right)} \sum_{\mathcal{M}} m_q^{(\gamma)} \exp\left(-\beta \sum_{r,\delta} m_r^{(\delta)} e_r^{(\delta)} \right) \\
&= \frac{\left[\sum_{\mathcal{M}_\gamma} \exp\left(-\beta \sum_{r,\delta \neq \gamma} m_r^{(\delta)} e_r^{(\delta)} \right) \right] \left[\sum_{\{\boldsymbol{m}^{(\gamma)}\}} m_q^{(\gamma)} \exp\left(-\beta \sum_r m_r^{(\gamma)} e_r^{(\gamma)} \right) \right]}{\sum_{\mathcal{M}} \exp\left(-\beta \sum_{r,\delta} m_r^{(\delta)} e_r^{(\delta)} \right)} \\
&= \frac{\left[\sum_{\mathcal{M}_\gamma} \exp\left(-\beta \sum_{r,\delta \neq \gamma} m_r^{(\delta)} e_r^{(\delta)} \right) \right] \left[\sum_{\{\boldsymbol{m}^{(\gamma)}\}} m_q^{(\gamma)} \exp\left(-\beta \sum_r m_r^{(\gamma)} e_r^{(\gamma)} \right) \right]}{\left[\sum_{\mathcal{M}_\gamma} \exp\left(-\beta \sum_{r,\delta \neq \gamma} m_r^{(\delta)} e_r^{(\delta)} \right) \right] \left[\sum_{\{\boldsymbol{m}^{(\gamma)}\}} \exp\left(-\beta \sum_r m_r^{(\gamma)} e_r^{(\gamma)} \right) \right]} \\
&= \frac{\sum_{\{\boldsymbol{m}^{(\gamma)}\}} m_q^{(\gamma)} \exp\left(-\beta \sum_r m_r^{(\gamma)} e_r^{(\gamma)} \right)}{\sum_{\{\boldsymbol{m}^{(\gamma)}\}} \exp\left(-\beta \sum_r m_r^{(\gamma)} e_r^{(\gamma)} \right)} \\
&= \frac{\exp\left(-\beta e_q^{(\gamma)} \right)}{\sum_{\{\boldsymbol{m}^{(\gamma)}\}} \exp\left(-\beta \sum_r m_r^{(\gamma)} e_r^{(\gamma)} \right)} && \text{(only remain when } m_q^{(\alpha)} = 1\text{)} \\
&= \frac{\exp\left(-\beta e_q^{(\gamma)} \right)}{\sum_{\{\boldsymbol{m}^{(\gamma)}\}} \exp\left(-\beta e_r^{(\gamma)} \right)} && \text{(only remain when } m_r^{(\alpha)} = 1\text{)} \\
&= \frac{\exp\left(-\beta e_q^{(\gamma)} \right)}{\sum_r \exp\left(-\beta e_r^{(\gamma)} \right)} && (r \in \boldsymbol{m}^{(\gamma)})
\end{aligned}$$

$\approx$ soft-max of the mean fields

$$e_q^{(\alpha)} = \underbrace{\frac{2}{p} \frac{1}{\sum_\gamma \langle m_q^{(\gamma)} \rangle_Q} \sum_\delta \langle m_q^{(\delta)} \rangle_Q \underbrace{\left\{ d_{\delta\alpha} - \frac{1}{2} \frac{1}{\sum_\gamma \langle m_q^{(\gamma)} \rangle_Q} \sum_\varepsilon \langle m_q^{(\varepsilon)} \rangle_Q d_{\varepsilon\delta} \right\}}_{\substack{\text{distance between data objects } \alpha \text{ and } \delta, \\ \text{corrected by the average distance between} \\ \text{objects of the cluster, to which } \delta \text{ belongs (here: } q\text{)}}}}_{\substack{\text{average corrected distance between data objects} \\ \alpha \text{ and all objects } \delta \text{ of cluster } q}} \tag{4.27}$$

interpretation of the "mean fields":

$\Rightarrow$ "metric visualization" of the $e_q^{(\alpha)}$:

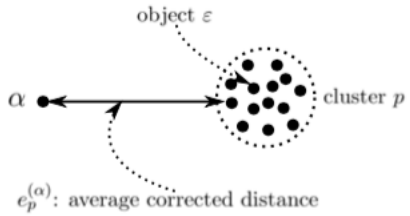

Figure 4.3:

Algorithm 10: soft k-means clustering for general distances

Initialization:
- choose number M of partitions (max no)
- choose initial (β_0) and final (β_f) values of the noise parameter
- choose annealing factor η and convergence criterion θ
- initialize mean-fields $e_q^{(\alpha)}$ with random numbers $\in [0,1]$
- set $\beta \leftarrow \beta_0$

while $\beta < \beta_f$ **do** annealing

  repeatEM fixpoint iteration

    compute assignment probabilities

$$\langle m_q^{(\alpha)} \rangle_Q = \frac{\exp\{-\beta (e_q^{(\alpha)})_{\text{old}}\}}{\sum_r \exp\{-\beta (e_r^{(\alpha)})_{\text{old}}\}} \text{ for all } q, \alpha$$

    compute new mean-fields

$$(e_q^{(\alpha)})_{\text{new}} = \frac{2}{M} \frac{1}{\sum_\gamma \langle m_q^{(\gamma)} \rangle_Q} \sum_\delta \langle m_q^{(\delta)} \rangle_Q \cdot \left\{ d_{\delta\alpha} - \frac{1}{2} \frac{1}{\sum_\gamma \langle m_q^{(\gamma)} \rangle_Q} \sum_\varepsilon \langle m_q^{(\varepsilon)} \rangle_Q d_{\varepsilon\delta} \right\} \text{ for all } q, \alpha$$

  until $|(e_q^{(\alpha)})_{\text{new}} - (e_q^{(\alpha)})_{\text{old}}| < \theta$ *for all* q, α

  $\beta \leftarrow \eta \cdot \beta$

end

Figure 4.4:

4.2.3 Missing data

Sometimes, we don't have access to all $d_{q\alpha}$ pairs. We can derive as follow :

$$\bar{d}_{q\alpha} = \frac{\sum_\gamma \langle m_q^{(\gamma)} \rangle_Q d_{\gamma\alpha}}{\sum_\gamma \langle m_q^{(\gamma)} \rangle_Q}$$

$$e_q^{(\alpha)} = \left\{ \bar{d}_{q\alpha} - \frac{1}{2} \sum_\delta \langle m_q^{(\delta)} \rangle_Q \bar{d}_{\gamma\delta} \right\}$$

Therefore, we can compute $\bar{d}_{\gamma\delta}$ using only measured data that we have.

4.2.4 Soft K-means for Euclidean distances

$$d_{\alpha\alpha'} = \frac{1}{2} \left(\boldsymbol{x}^{(\alpha)} - \boldsymbol{x}^{(\alpha')} \right)^2$$

$$e_q^{(\alpha)} = \frac{1}{2} \left(\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q \right)^2$$

and

$$\boldsymbol{w}_q = \frac{\sum_\gamma \langle m_q^{(\gamma)} \rangle_Q \boldsymbol{x}^{(\gamma)}}{\sum_\gamma \langle m_q^{(\gamma)} \rangle_Q}$$

For online version, we compute $\langle m_q^{(\alpha)} \rangle_Q, \forall q \in M$ and then update $\boldsymbol{w}_q$ with the rule :

$$\boldsymbol{w}_q \leftarrow \boldsymbol{w}_q + \eta \langle m_q^{(\alpha)} \rangle_Q \left(\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q \right)$$

4.2.5 Influence of β to cluster structure

Average cost is

$$\langle E \rangle = \frac{1}{Z} \sum_{\{m_p^{(\alpha)}\}} E \exp(-\beta E)$$

$\uparrow \beta$ means $\downarrow$ average cost, $\downarrow$ cluster size and $\uparrow$ hierachical clustering.

Algorithm 11: soft k-means clustering for Euclidean distances

Initialization:

- choose no. M of partitions (max no)
- choose initial (β_0) and final (β_f) values of the noise parameter
- initialize prototypes: $\underline{\mathbf{w}}_q = \frac{1}{p}\sum_{\alpha}\underline{\mathbf{x}}^{(\alpha)} + \underline{\eta}_q$ (small random vector)
- choose annealing factor η
- choose convergence criterion θ
- $\beta \leftarrow \beta_0$

while $\beta < \beta_f$ *(annealing)* **do**

  repeat EM

    compute assignment probabilities

$$\langle m_q^{(\alpha)} \rangle_Q = \frac{\exp\left\{-\frac{\beta}{2}\left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_q^{\text{old}}\right)^2\right\}}{\sum_r \exp\left\{-\frac{\beta}{2}\left(\underline{\mathbf{x}}^{(\alpha)} - \underline{\mathbf{w}}_r^{\text{old}}\right)^2\right\}} \quad \text{for all } \alpha, q$$

    compute new prototypes

$$\underline{\mathbf{w}}_q^{\text{new}} = \underbrace{\frac{\sum_{\alpha}\langle m_q^{(\alpha)}\rangle_Q \underline{\mathbf{x}}^{(\alpha)}}{\sum_{\alpha}\langle m_q^{(\alpha)}\rangle_Q}}_{\substack{\text{center of mass of the data points} \\ \text{which belong to cluster } q \text{ - weighted} \\ \text{by assignment probability}}} \quad \text{for all } q$$

  until $\left|\underline{\mathbf{w}}_q^{\text{new}} - \underline{\mathbf{w}}_q^{\text{old}}\right| < \theta$ *for all* q

  $\beta \leftarrow \eta\beta$

end

Figure 4.5: Soft K-means with Euclidean distance

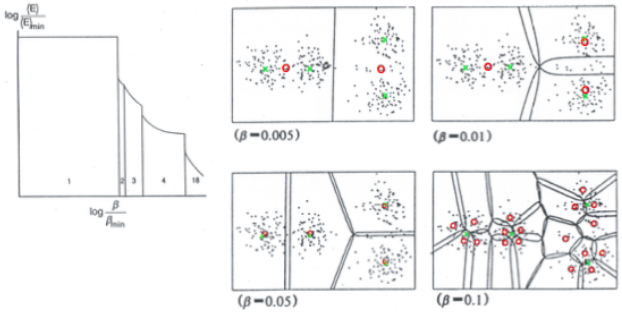

Figure 4.6: Influence of β to cluster structure

4.3 Self-Organizing Maps(SOM)

SOM is a clustering algorithm that is based on similarity, e.g. Euclidean distance. It's a low-dimensional embedding technique that preverves neighborhood.

Data: choose of no. partitions (clusters) M
Data: choose of annealing schedule ϵ and σ
for $\alpha = shuffle(\{1, \ldots, p\})$ **do**

  Find the closest prototype : $p = \underset{r}{\text{argmin}}|\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_r|$

  $\forall q$: update $\boldsymbol{w}_q$ respect to :

$$h_{qp} = \exp\left(-\frac{(p-q)^2}{2\sigma^2}\right) \qquad \text{(Neighborhood function)}$$

$$\Delta\boldsymbol{w}_q = \epsilon h_{qp}(\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q)$$

end

Algorithm 5: On-line learning for SOM

Noting that, if $\sigma = 0$ the algorithm becomes an standard on-line K-means.

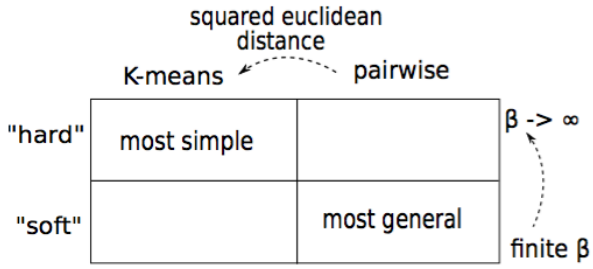

Figure 4.7: Summary of clustering algorithms

4.3.1 Neighborhood function h_{qp}

$$h_{qp} = \exp\left(-\frac{(p-q)^2}{2\sigma^2}\right)$$

- $\uparrow \sigma$ gives us smooth cost function $\implies$ lower-cost optimization but results might not well represent the data.
- $\downarrow \sigma$ gives us rough cost function $\implies$ higher-cost optimization but has more representation capabilities.

4.3.2 Self-organizing maps for pairwise data

- For each pair of data, we compute dissimilarity $d_{\alpha\alpha'}$.
- Construct M clusters with geometrical structure (1d line, 2d grid)
- $m_q^{(\alpha)}$ binary assignment variable (normalize to 1 for each α)

$$m_q^{(\alpha)} = 1\{\boldsymbol{x}^{(\alpha)} \text{ belongs to } \boldsymbol{w}_q\}$$

Cost Function

$$E_{[\{m_q^{(\alpha)}\}]} = \frac{1}{p}\sum_r \frac{\sum_{\alpha,\alpha'}\left(\sum_q h_{rq} m_q^{(\alpha)}\right)\left(\sum_q h_{rq} m_q^{(\alpha')}\right) d_{\alpha\alpha'}}{\sum_\alpha \left(\sum_q h_{rq} m_q^{(\alpha)}\right)}$$
$$\stackrel{!}{=} \min$$

This cost function is similar to pairwise clustering but $\boldsymbol{w}_q$ is replaced by $\sum_q h_{rq} m_q^{(\alpha)}$.

From Figure 4.8, if we replace h_{qp} with Direc's $\delta_{qp} = 1\{q = p\}$, the computation becomes standard pairwise clustering. This is called "Kohonen-approximation", hence no need σ for annealing.

Euclidean distances

$$d_{\alpha\alpha'} = \frac{1}{2}\left(x^{(\alpha)} - x^{(\alpha')}\right)^2$$

$$\begin{aligned} E_{[\{m_q^{(\alpha)}\}]} &= \frac{1}{p}\sum_{q,\alpha}\left(\sum_p h_{qp} m_p^{(\alpha)}\right)(\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q)^2 \\ &= \frac{1}{p}\sum_{p,\alpha} m_p^{(\alpha)}\left(\sum_q h_{qp}\right)(\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q)^2 \end{aligned}$$

$$\boldsymbol{w}_q = \frac{\sum_{\alpha'}\left(\sum_p h_{qp} m_p^{(\alpha')} \boldsymbol{x}^{(\alpha')}\right)}{\sum_{\alpha'}\left(\sum_p h_{qp} m_p^{(\alpha')}\right)}$$

Initialization:
- choose no. M of partitions, initial (β_0) and final (β_f) noise parameters, annealing factor η, width σ of neighborhood function $h_{\underline{\mathbf{sq}}}$, tolerance θ
- initialize mean-fields $e_{\underline{\mathbf{q}}}^{(\alpha)}$: random numbers $\in [0,1]$

$\beta \leftarrow \beta_0$
while $\beta < \beta_f$ **do** annealing
  repeat EM
    compute assignment probabilities $\langle m_{\underline{\mathbf{q}}}^{(\alpha)} \rangle_Q = \frac{\exp\left\{-\beta\left(e_{\underline{\mathbf{q}}}^{(\alpha)}\right)_{\text{old}}\right\}}{\sum_{\underline{\mathbf{r}}} \exp\left\{-\beta\left(e_{\underline{\mathbf{r}}}^{(\alpha)}\right)_{\text{old}}\right\}} \quad \forall \underline{\mathbf{q}}, \alpha$
    compute new mean-fields

$$\left(e_{\underline{\mathbf{q}}}^{(\alpha)}\right)_{\text{new}} = \frac{1}{p} \sum_{\underline{\mathbf{s}}} h_{\underline{\mathbf{sq}}} \left[\frac{1}{\sum_{\gamma} \left(\sum_{\underline{\mathbf{r}}} h_{\underline{\mathbf{sr}}} \langle m_{\underline{\mathbf{r}}}^{(\gamma)} \rangle_Q \right)} \sum_{\delta} \left(\sum_{\underline{\mathbf{r}}} h_{\underline{\mathbf{sr}}} \langle m_{\underline{\mathbf{r}}}^{(\delta)} \rangle_Q \right) \right.$$

$$\left. \cdot \left\{ d_{\delta\alpha} - \frac{1}{2} \frac{1}{\sum_{\gamma} \left(\sum_{\underline{\mathbf{r}}} h_{\underline{\mathbf{sr}}} \langle m_{\underline{\mathbf{r}}}^{(\gamma)} \rangle_Q \right)} \sum_{\varepsilon} \left(\sum_{\underline{\mathbf{r}}} h_{\underline{\mathbf{sr}}} \langle m_{\underline{\mathbf{r}}}^{(\varepsilon)} \rangle_Q \right) d_{\varepsilon\delta} \right\} \right] \quad \forall \underline{\mathbf{q}}, \alpha$$

  until $\left|\left(e_{\underline{\mathbf{q}}}^{(\alpha)}\right)_{\text{new}} - \left(e_{\underline{\mathbf{q}}}^{(\alpha)}\right)_{\text{old}}\right| < \theta \;\; \forall \underline{\mathbf{q}}, \alpha$
  $\beta \leftarrow \eta\beta$
end

Figure 4.8: SOM for Pairwise data with mean-field approximation

$\boldsymbol{w}_q$ is now a center of mass of the cluster weighted by neighborhood function h_{qp}.

Data: choose of no. partitions (clusters) M
Data: choose of annealing schedule ϵ
for $\alpha = shuffle(\{1, \ldots, p\})$ **do**
  Find the closest prototype : $p = \underset{r}{\operatorname{argmin}} \sum_q h_{rq} (\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q)^2$
  $\forall q$: update $\boldsymbol{w}_q$ respect to :

$$h_{qp} = \exp\left(-\frac{(p-q)^2}{2\sigma^2} \right) \qquad \text{(Neighborhood function)}$$

$$\Delta \boldsymbol{w}_q = \epsilon h_{qp} (\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q)$$

end

Algorithm 6: On-line learning for SOM with Euclidean distances

4.4 Locally Linear Embedding

Locally linear embedding (LLE) seeks a lower-dimensional projection of the data which preserves distances within local neighborhoods - quoted from scikit-learn[1].

Given data $\boldsymbol{x}^{(\alpha)} \in \mathbb{R}^N, \alpha = \{1, \ldots, p\}$.

1. Find K-nearest neighbors of $\boldsymbol{x}^{(\alpha)}$

2. Calculate reconstruction weights W of $\boldsymbol{x}^{(\alpha)}$ from its K-neighbors.

3. Obtain embedding $\boldsymbol{u}^{(\alpha)} \in \mathbb{R}^M, M \ll N$ from W and its neighbors' embedding.

4.4.1 Find K-nearest neighbors

Traditional K-nearest neighbors approach has $O(Np^2)$ complexity, but k-d tree reduces the complexity to $O(Np \log p)$.

[1] http://scikit-learn.org/stable/modules/manifold.html#locally-linear-embedding

4.4.2 Calculate reconstruction weights

We want to find W that minimizes reconstruction cost

$$\begin{aligned} E(\boldsymbol{W}) &= \sum_{\alpha=1}^{p} \left(\boldsymbol{x}^{(\alpha)} - \sum_{\beta=1}^{p} W_{\alpha\beta} \boldsymbol{x}^{(\beta)} \right)^2 \\ &\stackrel{!}{=} \min \end{aligned}$$

s.t.

$$\begin{aligned} W_{\alpha\beta} &= 0 \text{ if } \boldsymbol{x}^{(\beta)} \notin \mathrm{KNN}(\boldsymbol{x}^{(\alpha)}) \\ \sum_{\beta} W_{\alpha\beta} &= 1 \end{aligned}$$

Nothing that $\boldsymbol{W}$ is asymmetric sparse matrix that have up to K nonempty elements for each row. It is also invariant to linear transformations e.g. rotation, scaling and translation.

For each $\boldsymbol{x}^{(\alpha)}$,

- compute $\boldsymbol{C}^{(\alpha)} \in \mathbb{R}^{K,K}$

  $$C_{ij}^{(\alpha)} = (\boldsymbol{x}^{(\alpha)} - \boldsymbol{x}^{(\beta_i)})^T (\boldsymbol{x}^{(\alpha)} - \boldsymbol{x}^{(\beta_j)})$$

  where $\boldsymbol{x}^{(\beta_i)}, \boldsymbol{x}^{(\beta_j)} \in \mathrm{KNN}(\boldsymbol{x}^{(\alpha)})$

- solve linear system

  $$\boldsymbol{C}^{(\alpha)} \widetilde{\boldsymbol{w}}^{(\alpha)} = (1, \ldots, 1)^T$$

- rescale weight

  $$W_{\alpha\beta_i} = \frac{\widetilde{\boldsymbol{w}}_i^{(\alpha)}}{\sum_{j=1}^{K} \widetilde{\boldsymbol{w}}_j^{(\alpha)}}$$

4.4.3 Obtain Embedding

We want to find embedding $\forall \alpha \in \{1, \ldots, p\}, \boldsymbol{u}^{(\alpha)} \in \mathbb{R}^M$ s.t.

$$\begin{aligned} F(\boldsymbol{U}) &= \sum_{\alpha=1}^{p} \left(\boldsymbol{u}^{(\alpha)} - \sum_{\beta=1}^{p} W_{\alpha\beta} \boldsymbol{u}^{(\beta)} \right) \\ &= \sum_{\alpha,\beta=1}^{p} g_{\alpha\beta} (\boldsymbol{u}^{(\alpha)})^T \boldsymbol{u}^{(\beta)} \\ &\stackrel{!}{=} \min \end{aligned}$$

s.t.

$$\sum_{\alpha=1}^{p} \boldsymbol{u}^{(\alpha)} = 0 \qquad \text{(remove translation)}$$

$$\frac{1}{p} \sum_{\alpha=1}^{p} (\boldsymbol{u}^{(\alpha)})^T \boldsymbol{u}^{(\alpha)} = I \qquad \text{(prevent trivial solutions e.g. } \boldsymbol{u}^{(\alpha)} = 0 \text{)}$$

where $g_{\alpha\beta} = \delta_{\alpha\beta} - \boldsymbol{W}_{\alpha\beta} - \boldsymbol{W}_{\beta\alpha} + \sum_{\gamma=1}^{p} \boldsymbol{W}_{\gamma\alpha} \boldsymbol{W}_{\gamma\beta}$.

Equivalently,

$$\begin{aligned} \boldsymbol{G} &\in \mathbb{R}^{p,p} \\ &= \{g_{\alpha\beta}\} \ , \forall \alpha, \beta = \{1, \ldots, p\} \\ &= (I - W^T)(I - W) \end{aligned}$$

The solution to $F(\boldsymbol{U})$ is to compute eigen value decomposition of $\boldsymbol{G}$ with $M + 1$ from the lowest eigen values. We discard eigen vector e_p whose eigen value is zero.

$$\boldsymbol{U} = \begin{pmatrix} e_{p-M}^T \\ \cdots \\ e_{p-1}^T \end{pmatrix} = \left(u^{(1)}, \ldots, u^{(p)} \right) \in \mathbb{R}^{M,p}$$

parameters: K, M

1. find K nearest neighbors $\mathrm{KNN}(\underline{\mathbf{x}}^{(\alpha)}) = \left\{\beta_1^{(\alpha)}, \ldots, \beta_K^{(\alpha)}\right\}\ \forall \alpha = 1, \ldots, p$
2. calculate (locally invariant) reconstruction weights $\underline{\mathbf{W}}$:

$$\underline{\mathbf{C}}^{(\alpha)} \underline{\widetilde{\mathbf{w}}}^{(\alpha)} = (1, \ldots, 1)^T, \quad \forall \alpha = 1, \ldots, p$$

$$\mathrm{W}_{\alpha\beta_j^{(\alpha)}} = \frac{\widetilde{\mathrm{w}}_j^{(\alpha)}}{\sum_{k=1}^{K} \widetilde{\mathrm{w}}_k^{(\alpha)}}$$

3. calculate the embedding coordinates $\underline{\mathbf{U}}$:
   compute the $M+1$ eigenvectors $\left(\underline{\mathbf{e}}_p, \ldots, \underline{\mathbf{e}}_{p-M}\right)$ of $\underline{\mathbf{G}}$
   with the smallest eigenvalues

$$g_{\alpha\beta} = \delta_{\alpha\beta} - \mathrm{W}_{\alpha\beta} - \mathrm{W}_{\beta\alpha} + \sum_{\gamma=1}^{p} \mathrm{W}_{\gamma\alpha} \mathrm{W}_{\gamma\beta}$$

$$\underline{\mathbf{G}} \cdot \underline{\mathbf{e}}_j = \lambda_j \underline{\mathbf{e}}_j \qquad \underline{\mathbf{U}} = \begin{pmatrix} \underline{\mathbf{e}}_{p-M}^T \\ \vdots \\ \underline{\mathbf{e}}_{p-1}^T \end{pmatrix} = \left(\underline{\mathbf{u}}^{(1)}, \ldots, \underline{\mathbf{u}}^{(p)}\right)$$

Figure 4.9: Locally Linear Embedding Algorithm

Chapter 5

Probability Density Estimation

Density Estimation is important because if $P(x)$ is known, we can easily compute moments of x.

There are 2 approaches to estimate density, namely

1. **Nonparametric** : histogram-based or kernel-based
2. **Parametric** : Gaussian densities

5.1 Nonparametric Methods

5.1.1 Gliding Histogram

This approach finds $\hat{P}(\boldsymbol{x})$ by counting number of points in volume centered by $\boldsymbol{x}$.

$$H(\boldsymbol{u}) = \begin{cases} 1, & |u_j| < \frac{1}{2} \ \forall j \in 1, \dots, n \\ 0, & \text{otherwise} \end{cases}$$

where n is no. of dimensions of $\boldsymbol{u}$.

Given observations $\boldsymbol{x}^{(\alpha)} \in \mathbb{R}^n, \alpha \in 1, \dots, p$. We can derive :

$$\hat{P}(\boldsymbol{x}) = \frac{1}{h^n} \cdot \frac{1}{p} \sum_{\alpha=1}^{p} H\left(\frac{\boldsymbol{x} - \boldsymbol{x}^{(\alpha)}}{h} \right)$$

where h is width of the counting volume and $\frac{1}{h^n}$ plays a normalization role.

One remark of Gliding Histogram is the **discontinuity** of the estimated density.

5.1.2 Gaussian Kernel

Instead of relying on count of observations in volume (Histogram Kernel), we can apply a kernel computation, for example Gaussian Kernel.

$$\begin{aligned} H(\boldsymbol{u}) &= \frac{1}{(2\pi)^{\frac{n}{2}}} \exp\left(-\frac{\boldsymbol{u}^2}{2} \right) \\ \hat{P}(\boldsymbol{x}) &= \frac{1}{h^n} \cdot \frac{1}{p} \sum_{\alpha=1}^{p} H\left(\frac{\boldsymbol{x} - \boldsymbol{x}^{(\alpha)}}{h} \right) \\ &= \frac{1}{h^n} \cdot \frac{1}{p} \sum_{\alpha=1}^{p} \frac{1}{(2\pi)^{\frac{n}{2}}} \exp\left(-\frac{(\boldsymbol{x} - \boldsymbol{x}^{(\alpha)})^2}{2h^2} \right) \\ &= \frac{1}{p} \sum_{\alpha=1}^{p} \frac{1}{(2\pi h^2)^{\frac{n}{2}}} \exp\left(-\frac{(\boldsymbol{x} - \boldsymbol{x}^{(\alpha)})^2}{2h^2} \right) \end{aligned}$$

where h is width of the kernel.

$$\text{smoothness of estimated density} \propto h$$

small $h \rightarrow$ non smooth estimation (kind of overfitting)
large $h \rightarrow$ smooth estimation (could be underfitting)

5.2 Parametric Methods

We assume that observations $x^{(\alpha)} \in \mathbb{R}^n, \alpha = 1, \ldots, p$ are generated from a model. The goal here is to find a parameterized model via $\boldsymbol{w}$ parameters ($\hat{P}(\boldsymbol{x}; \boldsymbol{w})$) s.t.

$$P(\boldsymbol{x}) = \hat{P}(\boldsymbol{x}; \boldsymbol{w})$$

Using Kullback-Leibler Divergence

$$\begin{aligned} D_{KL}\Big[P(\boldsymbol{x}), \hat{P}(\boldsymbol{x}; \boldsymbol{w})\Big] &= \int d\boldsymbol{x} P(\boldsymbol{x}) \ln \frac{P(\boldsymbol{x})}{\hat{P}(\boldsymbol{x}; \boldsymbol{w})} \\ &\stackrel{!}{=} \min \end{aligned}$$

Nothing that $D_{KL} \geq 0$ and $D_{KL} = 0$ iff $P(\boldsymbol{x}) = \hat{P}(\boldsymbol{x}; \boldsymbol{w})$.
Therefore,

$$\begin{aligned} \boldsymbol{w}^* &= \underset{\boldsymbol{w}}{\operatorname{argmin}} \Big\{ \int d\boldsymbol{x} P(\boldsymbol{x}) \ln \frac{P(\boldsymbol{x})}{\hat{P}(\boldsymbol{x}; \boldsymbol{w})} \Big\} \\ &= \underset{\boldsymbol{w}}{\operatorname{argmin}} \Big\{ \int d\boldsymbol{x} P(\boldsymbol{x}) \ln P(\boldsymbol{x}) - \int d\boldsymbol{x} P(\boldsymbol{x}) \ln \hat{P}(\boldsymbol{x}; \boldsymbol{w}) \Big\} \\ &= \underset{\boldsymbol{w}}{\operatorname{argmin}} \Big\{ - \int d\boldsymbol{x} P(\boldsymbol{x}) \ln \hat{P}(\boldsymbol{x}; \boldsymbol{w}) \Big\} \\ &= \underset{\boldsymbol{w}}{\operatorname{argmin}} \Big\{ E^G \Big\} \end{aligned}$$

Using Empirical Risk Minimization (ERM), we can approximate expectation E^G using empirical average. As a result, the training cost E^T is

$$\begin{aligned} E^T &= -\frac{1}{p} \sum_{\alpha=1}^{p} \ln \hat{P}(\boldsymbol{x}^{(\alpha)}; \boldsymbol{w}) \\ &= \frac{1}{p} \sum_{\alpha=1}^{p} - \ln \hat{P}(\boldsymbol{x}^{(\alpha)}; \boldsymbol{w}) \\ &= \frac{1}{p} \sum_{\alpha=1}^{p} e_{[\boldsymbol{w}]}^{(\alpha)} \\ &\stackrel{!}{=} \min \end{aligned}$$

Using SGD, we can update $\boldsymbol{w}$ as follow

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} E^T \qquad \text{(batch-learning)}$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} e_{[\boldsymbol{w}]}^{(\alpha)} \qquad \text{(online-learning)}$$

There are several approches to validate quality of model, e.g. test-set, cross-validation set. Overfitting, E^T low but E^G high.

Another way to derive this learning procedure is to look at **likelihood function**. The likelihood of a model is the probability that the model generates the given observations with iid assumption.

$$\begin{aligned} \hat{P}(\{\boldsymbol{x}^{(\alpha)}\}; \boldsymbol{w}) &= \prod_{\alpha=1}^{p} \hat{P}(\boldsymbol{x}^{(\alpha)}; \boldsymbol{w}) \\ &\stackrel{!}{=} \max \end{aligned}$$

$$\begin{aligned}
\boldsymbol{w}^* &= \underset{\boldsymbol{w}}{\operatorname{argmax}}\left\{ \prod_{\alpha=1}^{p} \hat{P}(\boldsymbol{x}^{(\alpha)}; \boldsymbol{w}) \right\} \\
&= \underset{\boldsymbol{w}}{\operatorname{argmax}}\left\{ \ln \prod_{\alpha=1}^{p} \hat{P}(\boldsymbol{x}^{(\alpha)}; \boldsymbol{w}) \right\} \\
&= \underset{\boldsymbol{w}}{\operatorname{argmax}}\left\{ \sum_{\alpha=1}^{p} \ln \hat{P}(\boldsymbol{x}^{(\alpha)}; \boldsymbol{w}) \right\} \\
&= \underset{\boldsymbol{w}}{\operatorname{argmin}}\left\{ -\sum_{\alpha=1}^{p} \ln \hat{P}(\boldsymbol{x}^{(\alpha)}; \boldsymbol{w}) \right\}
\end{aligned}$$

5.2.1 Multivariate Gaussian

$$\hat{P}\big(\{\boldsymbol{x}^{(\alpha)}\}; \boldsymbol{\mu}, \boldsymbol{\Sigma}\big) = \left(\frac{1}{\sqrt{(2\pi)^n \det \boldsymbol{\Sigma}}} \right)^p \cdot \prod_{\alpha=1}^{p} \exp\left(-\frac{1}{2} \big(\boldsymbol{x}^{(\alpha)} - \boldsymbol{\mu}\big)^T \boldsymbol{\Sigma}^{-1} \big(\boldsymbol{x}^{(\alpha)} - \boldsymbol{\mu}\big) \right)$$

Hence,

$$\begin{aligned}
E^T_{[\boldsymbol{\mu},\boldsymbol{\Sigma}]} &= -\ln \hat{P}\big(\{\boldsymbol{x}^{(\alpha)}\}; \boldsymbol{\mu}, \boldsymbol{\Sigma}\big) \\
&= -\ln \left(\frac{1}{\sqrt{(2\pi)^n \det \boldsymbol{\Sigma}}} \right)^p - \ln \prod_{\alpha=1}^{p} \exp\left(-\frac{1}{2} \big(\boldsymbol{x}^{(\alpha)} - \boldsymbol{\mu}\big)^T \boldsymbol{\Sigma}^{-1} \big(\boldsymbol{x}^{(\alpha)} - \boldsymbol{\mu}\big) \right) \\
&= -\ln \Big((2\pi)^n \det \boldsymbol{\Sigma} \Big)^{-\frac{p}{2}} - \sum_{\alpha=1}^{p} \left(-\frac{1}{2} \big(\boldsymbol{x}^{(\alpha)} - \boldsymbol{\mu}\big)^T \boldsymbol{\Sigma}^{-1} \big(\boldsymbol{x}^{(\alpha)} - \boldsymbol{\mu}\big) \right) \\
&= \frac{p}{2} \ln \Big((2\pi)^n \det \boldsymbol{\Sigma} \Big) + \frac{1}{2} \sum_{\alpha=1}^{p} \left(\big(\boldsymbol{x}^{(\alpha)} - \boldsymbol{\mu}\big)^T \boldsymbol{\Sigma}^{-1} \big(\boldsymbol{x}^{(\alpha)} - \boldsymbol{\mu}\big) \right) \\
&= \frac{pn}{2} \ln(2\pi) + \frac{p}{2} \ln(\det \boldsymbol{\Sigma}) + \frac{1}{2} \sum_{\alpha=1}^{p} \left(\big(\boldsymbol{x}^{(\alpha)} - \boldsymbol{\mu}\big)^T \boldsymbol{\Sigma}^{-1} \big(\boldsymbol{x}^{(\alpha)} - \boldsymbol{\mu}\big) \right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial E^T_{[\boldsymbol{\mu},\boldsymbol{\Sigma}]}}{\partial \boldsymbol{\mu}} = 0 \Rightarrow \boldsymbol{\mu}^* &= \frac{1}{p} \sum_{\alpha=1}^{p} \boldsymbol{x}^{(\alpha)} \\
\frac{\partial E^T_{[\boldsymbol{\mu},\boldsymbol{\Sigma}]}}{\partial \boldsymbol{\Sigma}} = 0 \Rightarrow \boldsymbol{\Sigma}^* &= \frac{1}{p} \sum_{\alpha=1}^{p} (\boldsymbol{x}^{(\alpha)} - \boldsymbol{\mu}^*)(\boldsymbol{x}^{(\alpha)} - \boldsymbol{\mu}^*)^T
\end{aligned}$$

Noting that $\boldsymbol{\mu}^*$ is unbiased but $\boldsymbol{\Sigma}^*$ is biased estimator.

5.2.2 Gaussian Mixture Model

Here we assume that data is generated from various sources. Formally,

$$P(\boldsymbol{x}) = \sum_{q=1}^{M} P(\boldsymbol{x}|q) P(q)$$

where

1. $\boldsymbol{x} \in \mathbb{R}^d$.
2. $P(q)$ is a mixture parameter that component q creates a data point

3. $P(\boldsymbol{x}|q)$ is probability that $\boldsymbol{x}$ is created by component q.

   There are several choices of model that we can use for $P(\boldsymbol{x}|q)$. One popular choice is a Gaussian.

$$\begin{aligned} P(\boldsymbol{x}|q) &= \mathcal{N}(\boldsymbol{x}; \boldsymbol{w}_q, \sigma_q^2) \\ &= \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(- \frac{(\boldsymbol{x} - \boldsymbol{w}_q)^2}{2\sigma_q^2} \right) \end{aligned}$$

Hence, parameters to determine for this kind of mixture model are $P(q), \boldsymbol{w}_q, \sigma_q^2$.
Using principle of maximum likelihood, we have

$$\begin{aligned} P(\{\boldsymbol{x}^{(\alpha)}\}) &= \prod_{\alpha=1}^{p} \sum_{q=1}^{M} P(\boldsymbol{x}|q) P(q) \\ &= \prod_{\alpha=1}^{p} \sum_{q=1}^{M} \mathcal{N}(\boldsymbol{x}^{(\alpha)}; \boldsymbol{w}_q, \sigma_q^2) P(q) \\ &\stackrel{!}{=} \max \end{aligned}$$

Hence,

$$\begin{aligned} \{\boldsymbol{w}, \boldsymbol{\sigma}\} &= \underset{\{\boldsymbol{w},\boldsymbol{\sigma}\}}{\operatorname{argmax}} \prod_{\alpha=1}^{p} \sum_{q=1}^{M} \mathcal{N}(\boldsymbol{x}^{(\alpha)}; \boldsymbol{w}_q, \sigma_q^2) P(q) \\ &= \underset{\{\boldsymbol{w},\boldsymbol{\sigma}\}}{\operatorname{argmax}} \sum_{\alpha=1}^{p} \ln \sum_{q=1}^{M} \mathcal{N}(\boldsymbol{x}^{(\alpha)}; \boldsymbol{w}_q, \sigma_q^2) P(q) \\ &= -\underset{\{\boldsymbol{w},\boldsymbol{\sigma}\}}{\operatorname{argmin}} \sum_{\alpha=1}^{p} \ln \sum_{q=1}^{M} \mathcal{N}(\boldsymbol{x}^{(\alpha)}; \boldsymbol{w}_q, \sigma_q^2) P(q) \end{aligned}$$

Relation of Soft-Clustering method

Assumptions:

1. We have M Gaussian components.

2. $\sigma_q^2 = \sigma^2 := \frac{1}{\beta}$ for all components.

3. Constant mixture parameter $P(q) = \frac{1}{M}$.

We can compute posterior that $\boldsymbol{x}$ is created by component q via :

$$\begin{aligned} P(q|\boldsymbol{x}) &= \frac{P(\boldsymbol{x}|q)P(q)}{P(\boldsymbol{x})} \\ &= \frac{\frac{1}{M} \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{w}_q, \sigma_q^2)}{\sum_{\gamma=1}^{M} \frac{1}{M} \cdot \mathcal{N}(\boldsymbol{x}; \boldsymbol{w}_\gamma, \sigma_\gamma^2)} \\ &= \frac{\mathcal{N}(\boldsymbol{x}; \boldsymbol{w}_q, \sigma_q^2)}{\sum_{\gamma=1}^{M} \mathcal{N}(\boldsymbol{x}; \boldsymbol{w}_\gamma, \sigma_\gamma^2)} \end{aligned}$$

Hence, the objective function is reduced to

$$
\begin{aligned}
&- \operatorname{argmin} \sum_{\alpha=1}^{p} \ln \sum_{q=1}^{M} \mathcal{N}(\boldsymbol{x}^{(\alpha)}; \boldsymbol{w}_q, \sigma_q^2) P(q) \\
&- \operatorname{argmin} \sum_{\alpha=1}^{p} \ln \frac{1}{M} \sum_{q=1}^{M} \mathcal{N}(\boldsymbol{x}^{(\alpha)}; \boldsymbol{w}_q, \sigma_q^2) \\
&- \operatorname{argmin} \sum_{\alpha=1}^{p} \ln \left\{ \frac{1}{M} \sum_{q=1}^{M} \left(\frac{\beta}{2\pi} \right)^{d/2} \exp\left(- \frac{\beta(\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q)^2}{2} \right) \right\} \\
&- \operatorname{argmin} \sum_{\alpha=1}^{p} \ln \left\{ \frac{1}{M} \left(\frac{\beta}{2\pi} \right)^{d/2} \sum_{q=1}^{M} \exp\left(- \frac{\beta(\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q)^2}{2} \right) \right\} \\
&- \operatorname{argmin} \sum_{\alpha=1}^{p} \ln \left\{ \sum_{q=1}^{M} \exp\left(- \frac{\beta(\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q)^2}{2} \right) \right\} + \text{const}(\beta, M)
\end{aligned}
$$

Given

$$
E^T = - \sum_{\alpha=1}^{p} \ln \left\{ \sum_{q=1}^{M} \exp\left(- \frac{\beta}{2} (\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q)^2 \right) \right\}
$$

We have :

$$
\begin{aligned}
\frac{\partial E^T}{\partial \boldsymbol{w}_r} &= - \sum_{\alpha=1}^{p} \ln \left\{ \sum_{q=1}^{M} \exp\left(- \frac{\beta}{2} (\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q)^2 \right) \right\} \\
&= - \sum_{\alpha=1}^{p} \frac{\exp\left(- \frac{\beta}{2} (\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_r)^2 \right)}{\sum_{q=1}^{M} \exp\left(- \frac{\beta}{2} (\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q)^2 \right)} \beta(\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_r) \\
&= - \sum_{\alpha=1}^{p} P(r|\boldsymbol{x}^{(\alpha)}) \beta(\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_r) \\
&\stackrel{!}{=} 0
\end{aligned}
$$

Hence,

$$
\boldsymbol{w}_r = \frac{\sum_{\alpha=1}^{p} P(r|\boldsymbol{x}^{(\alpha)}) \boldsymbol{x}^{(\alpha)}}{\sum_{\alpha=1}^{p} P(r|\boldsymbol{x}^{(\alpha)})}
$$

New interpretation of Soft-Clustering

1. All clusters have the same width $\sigma^2 = \frac{1}{\beta}$.
2. and same number of data points $P(q) = \frac{1}{M}$.

Mixture Models (an extension of soft-clustering)

1. each cluster has different width σ_q^2. and number of points $P(q)$.

Optimization of Gaussian Mixture Model

We know that

$$
E^T = - \sum_{\alpha=1}^{p} \ln \left(\sum_{q=1}^{M} P(\boldsymbol{x}^{(\alpha)}|q) P(q) \right) \stackrel{!}{=} \min
$$

$$
\begin{aligned}
\frac{\partial}{\partial \boldsymbol{w}_r} P(\boldsymbol{x}^{(\alpha)}|r) &= \frac{(\boldsymbol{x} - \boldsymbol{w}_r)}{\sigma^2} P(\boldsymbol{x}^{(\alpha)}|r) \\
\frac{\partial}{\partial \sigma_r} P(\boldsymbol{x}^{(\alpha)}|r) &= \left\{ - \frac{d}{\sigma_r} + \frac{(\boldsymbol{x} - \boldsymbol{w}_r)^2}{\sigma_r^3} \right\} P(\boldsymbol{x}^{(\alpha)}|r)
\end{aligned}
$$

$$\begin{aligned}
\frac{\partial E^T}{\partial \boldsymbol{w}_r} &= -\sum_{\alpha=1}^{p} \frac{\partial}{\partial \boldsymbol{w}_r} \ln\Big(\sum_{q=1}^{M} P(\boldsymbol{x}^{(\alpha)}|q)P(q)\Big) \\
&= -\sum_{\alpha=1}^{p} \frac{P(r)\frac{\partial}{\partial \boldsymbol{w}_r} P(\boldsymbol{x}^{(\alpha)}|r)}{\sum_{q=1}^{M} P(\boldsymbol{x}^{(\alpha)}|q)P(q)} \\
&= -\sum_{\alpha=1}^{p} \frac{P(r)\frac{(\boldsymbol{x}-\boldsymbol{w}_r)}{\sigma^2} P(\boldsymbol{x}^{(\alpha)}|r)}{\sum_{q=1}^{M} P(\boldsymbol{x}^{(\alpha)}|q)P(q)} \\
&= -\sum_{\alpha=1}^{p} \frac{(\boldsymbol{x}-\boldsymbol{w}_r)}{\sigma^2} \frac{P(r)P(\boldsymbol{x}^{(\alpha)}|r)}{P(\boldsymbol{x}^{(\alpha)})} \\
&= -\sum_{\alpha=1}^{p} \frac{(\boldsymbol{x}-\boldsymbol{w}_r)}{\sigma^2} P(r|\boldsymbol{x}^{(\alpha)}) \\
&\stackrel{!}{=} 0
\end{aligned}$$

Hence,

$$\boldsymbol{w}_r = \frac{\sum_{\alpha=1}^{p} P(r|\boldsymbol{x}^{(\alpha)})\boldsymbol{x}^{(\alpha)}}{\sum_{\alpha=1}^{p} P(r|\boldsymbol{x}^{(\alpha)})}$$

Similarly,

$$\begin{aligned}
\frac{\partial E^T}{\partial \sigma_r} &= -\sum_{\alpha=1}^{p} \frac{\partial}{\partial \sigma_r} \ln\Big(\sum_{q=1}^{M} P(\boldsymbol{x}^{(\alpha)}|q)P(q)\Big) \\
&= -\sum_{\alpha=1}^{p} \frac{P(r)\frac{\partial}{\partial \sigma_r} P(\boldsymbol{x}^{(\alpha)}|r)}{\sum_{q=1}^{M} P(\boldsymbol{x}^{(\alpha)}|q)P(q)} \\
&= -\sum_{\alpha=1}^{p} \frac{P(r)\Big\{-\frac{d}{\sigma_r} + \frac{(\boldsymbol{x}-\boldsymbol{w}_r)^2}{\sigma_r^3}\Big\} P(\boldsymbol{x}^{(\alpha)}|r)}{\sum_{q=1}^{M} P(\boldsymbol{x}^{(\alpha)}|q)P(q)} \\
&= -\sum_{\alpha=1}^{p} \frac{\Big\{-\frac{d}{\sigma_r} + \frac{(\boldsymbol{x}-\boldsymbol{w}_r)^2}{\sigma_r^3}\Big\} P(\boldsymbol{x}^{(\alpha)}|r)P(r)}{P(\boldsymbol{x}^{(\alpha)})} \\
&= -\sum_{\alpha=1}^{p} \Big\{-\frac{d}{\sigma_r} + \frac{(\boldsymbol{x}-\boldsymbol{w}_r)^2}{\sigma_r^3}\Big\} P(r|\boldsymbol{x}^{(\alpha)}) \\
&= -\frac{1}{\sigma_r}\sum_{\alpha=1}^{p} \Big\{-d + \frac{(\boldsymbol{x}-\boldsymbol{w}_r)^2}{\sigma_r^2}\Big\} P(r|\boldsymbol{x}^{(\alpha)}) \\
&= -\frac{1}{\sigma_r^3}\sum_{\alpha=1}^{p} \Big\{-d\sigma_r^2 P(r|\boldsymbol{x}^{(\alpha)}) + (\boldsymbol{x}-\boldsymbol{w}_r)^2 P(r|\boldsymbol{x}^{(\alpha)})\Big\} \\
&\stackrel{!}{=} 0
\end{aligned}$$

Hence,

$$\sigma_r^2 = \frac{1}{d}\frac{\sum_{\alpha=1}^{p}(\boldsymbol{x}-\boldsymbol{w}_r)^2 P(r|\boldsymbol{x}^{(\alpha)})}{\sum_{\alpha=1}^{p} P(r|\boldsymbol{x}^{(\alpha)})}$$

For $P(q)$,we use Lagrange Multiplier as follow:

$$\frac{\partial}{\partial P(r)}\Big\{E^T + \lambda\Big(\sum_{q=1}^{M} P(q) - 1\Big)\Big\} \stackrel{!}{=} 0$$

$$\begin{aligned}
\frac{\partial}{\partial P(r)}\Big\{E^T + \lambda\Big(\sum_{q=1}^{M} P(q) - 1\Big)\Big\} &= \frac{\partial}{\partial P(r)}\Big\{ -\sum_{\alpha=1}^{p} \ln \sum_{q=1}^{M} P(\boldsymbol{x}^{(\alpha)}|q)P(q) + \lambda\Big(\sum_{q=1}^{M} P(q) - 1\Big)\Big\} \\
&= -\sum_{\alpha=1}^{p} \frac{P(\boldsymbol{x}^{(\alpha)}|r)}{\sum_{q=1}^{M} P(\boldsymbol{x}^{(\alpha)}|q)P(q)} + \lambda \\
&= -\sum_{\alpha=1}^{p} \frac{P(\boldsymbol{x}^{(\alpha)}|r)}{P(\boldsymbol{x}^{(\alpha)})} + \lambda \\
&= -\sum_{\alpha=1}^{p} \frac{P(\boldsymbol{x}^{(\alpha)}, r)}{P(r)P(\boldsymbol{x}^{(\alpha)})} + \lambda \\
&= -\sum_{\alpha=1}^{p} \frac{P(r|\boldsymbol{x}^{(\alpha)})}{P(r)} + \lambda
\end{aligned}$$

Hence,

$$P(r) = \frac{1}{\lambda}\sum_{\alpha=1}^{p} P(r|\boldsymbol{x}^{(\alpha)})$$

and because

$$\begin{aligned}
1 &= \sum_{q=1}^{M} P(q) \\
&= \sum_{q=1}^{M} \frac{1}{\lambda}\sum_{\alpha=1}^{p} P(q|\boldsymbol{x}^{(\alpha)}) \\
&= \frac{1}{\lambda}\sum_{\alpha=1}^{p}\sum_{q=1}^{M} P(q|\boldsymbol{x}^{(\alpha)}) \\
&= \frac{1}{\lambda}\sum_{\alpha=1}^{p} 1 \\
&= \frac{1}{\lambda}p
\end{aligned}$$

Therefore,

$$P(r) = \frac{1}{p}\sum_{\alpha=1}^{p} P(r|\boldsymbol{x}^{(\alpha)})$$

Expectation Maximization Algorithm

Data: choose of no. Gaussian (clusters) M
Data: Initialize $P(q) = \frac{1}{M}$
Data: Initialize $\boldsymbol{w}_q = \frac{1}{p}\sum_{\alpha=1}^{p} \boldsymbol{x}^{(\alpha)} + \eta_q$, $\sigma_q^2 = \frac{1}{p}\sum_{\alpha=1}^{p}(\boldsymbol{x}^{(\alpha)} - \boldsymbol{\mu})^2 + \epsilon_q$
, η_q, ϵ_q is random vectors.
while *until converage* **do**
E-step : $\forall q = \{1, ..., M\}$

$$P(q|\boldsymbol{x}^{(\alpha)}) \leftarrow \frac{P(\boldsymbol{x}^{(\alpha)}|q)P(q)}{\sum_{r=1}^{M} P(\boldsymbol{x}^{(\alpha)}|r)P(r)}$$

M-step : $\forall q = \{1, ..., M\}$

$$\boldsymbol{w}_q^{new} \leftarrow \frac{\sum_{\alpha=1}^{p} P(q|\boldsymbol{x}^{(\alpha)})\boldsymbol{x}^{(\alpha)}}{\sum_{\alpha=1}^{p} P(q|\boldsymbol{x}^{(\alpha)})}$$

$$(\sigma_q^2)^{new} \leftarrow \frac{\sum_{\alpha=1}^{p} P(q|\boldsymbol{x}^{(\alpha)})(\boldsymbol{x}^{(\alpha)} - \boldsymbol{w}_q)^2}{\sum_{\alpha=1}^{p} P(q|\boldsymbol{x}^{(\alpha)})}$$

$$P(q) \leftarrow \frac{1}{p}\sum_{\alpha=1}^{p} P(q|\boldsymbol{x}^{(\alpha)})$$

end

Algorithm 7: EM Algorithm

Proof of EM algorithm

Given $\boldsymbol{x}^{(\alpha)} \in \mathbb{R}^N, \alpha \in \{1, ..., p\}$ and $\boldsymbol{m}^{(\alpha)} = \{m_1^{(\alpha)}, ..., m_M^{(\alpha)}\}^T \in \{0, 1\}^M$ where $m_q^{(\alpha)} = 1$ if component q generates $\boldsymbol{x}^{(\alpha)}$, otherwise 0. We call these $\boldsymbol{m}^{(\alpha)}$ latent variables.

Latent variables and maximum likelihood,

$$\begin{aligned} P(\{\boldsymbol{x}^{(\alpha)}\}|\boldsymbol{\theta}) &= \prod_{\alpha=1}^{p} P(\boldsymbol{x}^{(\alpha)}|\boldsymbol{\theta}) \\ &= \prod_{\alpha=1}^{p} P(\boldsymbol{x}^{(\alpha)}, \boldsymbol{m}^{(\alpha)}) \\ &\stackrel{!}{=} \max \end{aligned}$$

We know that :

$$0 \le P(q) \le 1 \text{ and } \sum_{q=1}^{M} P(q) = 1$$

$$\begin{aligned} P(\boldsymbol{x}|\boldsymbol{\theta}) &= \sum_{q=1}^{M} P(q)\mathcal{N}(\boldsymbol{x}|\boldsymbol{w}_q, \sigma_q^2) \\ &= \sum_{\boldsymbol{m}} P(\boldsymbol{x}, \boldsymbol{m}) \\ &= \sum_{\boldsymbol{m}} P(\boldsymbol{x}|\boldsymbol{m})P(\boldsymbol{m}) \end{aligned}$$

and the prior distribution :

$$P(\boldsymbol{m}) = \prod_{q=1}^{M} P(q)^{m_q}$$

similarly,

$$P(\boldsymbol{x}|\boldsymbol{m}) = \prod_{q=1}^{M} \mathcal{N}(\boldsymbol{x}|\boldsymbol{w}_q, \sigma_q^2)^{m_q}$$

Hence,

$$P(\boldsymbol{x}, \boldsymbol{m}) = \prod_{q=1}^{M} P(q)^{m_q} \mathcal{N}(\boldsymbol{x}|\boldsymbol{w}_q, \sigma_q^2)^{m_q}$$

Therefore,

$$\begin{aligned}
\boldsymbol{\theta}^* &= \underset{\{P(q)\},\{\boldsymbol{w}_q\},\{\sigma_q^2\}}{\operatorname{argmax}} \prod_{\alpha=1}^{p} \prod_{q=1}^{M} P(q)^{m_q^{(\alpha)}} \mathcal{N}(\boldsymbol{x}^{(\alpha)}|\boldsymbol{w}_q, \sigma_q^2)^{m_q^{(\alpha)}} \\
&= \operatorname{argmax} \sum_{\alpha=1}^{p} \sum_{q=1}^{M} \ln P(q)^{m_q^{(\alpha)}} \mathcal{N}(\boldsymbol{x}^{(\alpha)}|\boldsymbol{w}_q, \sigma_q^2)^{m_q^{(\alpha)}} \\
&= \operatorname{argmax} \sum_{\alpha=1}^{p} \sum_{q=1}^{M} m_q^{(\alpha)} \Big(\ln P(q) + \ln \mathcal{N}(\boldsymbol{x}^{(\alpha)}|\boldsymbol{w}_q, \sigma_q^2) \Big) \\
&= \operatorname{argmin} - \sum_{\alpha=1}^{p} \sum_{q=1}^{M} m_q^{(\alpha)} \Big(\ln P(q) + \ln \mathcal{N}(\boldsymbol{x}^{(\alpha)}|\boldsymbol{w}_q, \sigma_q^2) \Big)
\end{aligned}$$

Posterior distribution :

$$P(\boldsymbol{m}|\boldsymbol{x}) = \frac{P(\boldsymbol{x}, \boldsymbol{m})}{P(\boldsymbol{x})} = \frac{\prod_{q=1}^{M} [P(q)\mathcal{N}(\boldsymbol{x}|\boldsymbol{w}_q, \sigma_q^2)]^{m_q}}{\sum_{q=1}^{M} P(q)\mathcal{N}(\boldsymbol{x}|\boldsymbol{w}_q, \sigma_q^2)}$$

Hence, posterior distribution of all hidden varibles

$$P(\{\boldsymbol{m}^{(\alpha)}\}|\{\boldsymbol{x}^{(\alpha)}\}, \boldsymbol{\theta}) = \prod_{\alpha=1}^{p} \frac{\prod_{q=1}^{M} [P(q)\mathcal{N}(\boldsymbol{x}^{(\alpha)}|\boldsymbol{w}_q^{(\alpha)}, \sigma_q^2)]^{m_q}}{\sum_{q=1}^{M} P(q)\mathcal{N}(\boldsymbol{x}^{(\alpha)}|\boldsymbol{w}_q^{(\alpha)}, \sigma_q^2)}$$

Therefore,

$$\langle m_q^{(\alpha)} \rangle_{P(\{\boldsymbol{m}^{(\alpha)}\}|\{\boldsymbol{x}^{(\alpha)}\}, \boldsymbol{\theta})} = P(q|\boldsymbol{x}^{(\alpha)})$$