Co-clustering deep latent block model

Theo Millot

April to Septembre 2025

1 Variable

Notation	Description	Dimension
π	Prior line cluster probability	$[0,1]^L$
τ	Prior columns cluster probability	$[0,1]^Q$
r	Cluster memberships of rows	$[\![0,1]\!]^L$
c	Cluster memberships of columns	$[\![0,1]\!]^Q$
X	Rows latent matrix	$\mathbb{R}^{M*D}$
Y	Columns latent matrix	$\mathbb{R}^{P*D}$
A	Observed adjacency matrix	$[0,1]^{M*P}$
μ_l	Rows cluster mean	$\mathbb{R}^{L*D}$
σ_l^2	Rows cluster variance	$\mathbb{R}^L$
m_q	Columns cluster mean	$\mathbb{R}^{Q*D}$
s_q^2	Columns cluster variance	$\mathbb{R}^Q$
$\alpha;\beta$	Parameters of the decoding neural network	$\mathbb{R}$
L	Number of rows cluster	$\mathbb{N}$
Q	Number of columns cluster	$\mathbb{N}$
D	Dimension of latent space	$\mathbb{N}$
M	Number of rows	$\mathbb{N}$
P	Number of columns	$\mathbb{N}$
i	Index of rows	$[\![0,M]\!]$
j	Index of columns	$[\![0,P]\!]$
l	Index of rows cluster	$[\![0,L]\!]$
q	Index of columns cluster	$[\![0,Q]\!]$
θ	Parameters set	...
ϕ	Parameters of the encoding neural network for rows	$\mathbb{R}^{nM}$
ψ	Parameters of the encoding neural network for columns	$\mathbb{R}^{nP}$
γ	Variational probability of cluster membership for rows	$[0,1]^{M*L}$
δ	Variational probability of cluster membership for columns	$[0,1]^{P*Q}$

2 Generative model

As we are using LPM, we assume that each node has an unknown position in a latent space. The probability of a link between two points depends on their position in the latent space. First, π (resp. τ) is the probability vector of each row (resp. columns) cluster such as : $\sum_{l=1}^{L}\pi_l = 1$ and $\sum_{q=1}^{Q}\tau_q = 1$

From these we can assign each node to its cluster :

$r_i \sim \mathcal{M}(1;\pi)$ and $c_j \sim \mathcal{M}(1;\tau)$

Then, we can generate the latent position conditionally to the assigned cluster independently for each node

$X_i|(r_{i,l}=1) \sim \mathcal{N}(\mu_l;\sigma_l^2 I_D)$ and $Y_j|(c_{j,q}=1) \sim \mathcal{N}(m_q;s_q^2 I_D)$

The probability is modeled through a Bernoulli random variable conditionally to the latent position :

$A_{i,j}|X_i,Y_j \sim \mathcal{B}(f_\alpha(X_i,Y_j))$
with f_α is the parameters of the decoding neural network such as $f_\alpha(X_i,Y_j) = \sigma(\alpha + ||X_i - Y_j||^2)$ where σ is the logistic sigmoid function.

3 Inference

3.1 VAE inference

We denote by $\theta = \{\pi,\tau,\mu_k,\sigma_k,m_q,s_q,\alpha,\phi,\psi\}$ the set of parameters to optimize.
We first want to maximize the integrated log-likelihood :
$\log(\text{P}(\text{A} \mid \theta)) = \log \int\limits_{\mathcal{X}}\int\limits_{\mathcal{Y}}\sum\limits_{r}\sum\limits_{c} P(A,X,Y,r,c \mid \theta)\,dY\,dX$
Unfortunately, this is untractable, and we have to do with variational inference to approximate it.

$\log(\text{P}(\text{A} \mid \theta)) = \mathcal{L}(\text{q(X, Y, r, c)} \; ; \theta) + D_{KL}(\text{q(X, Y, r, c)} \;||\; \text{P(X, Y, r, c} \mid \text{A}, \theta))$

where D_{KL} denotes the Kullback-Leibler divergence between the true distribution (P(X, Y, r, c | A, θ)), which is commonly unknown, and the variational distribution (q(X, Y, r, c)).
In order to be fully-tractable, we assume to fully factorize q(X,Y,r,c) (mean-field assumption).
$q(X,Y,r,c) = q(X)q(Y)q(r)q(c) = \prod\limits_{i=1}^{M} q(X_i)q(r_i) \prod\limits_{j=1}^{P} q(Y_j)q(c_j)$

Moreover, we assume :

$q(X_i) = \mathcal{N}(X_i; \tilde{\mu}_\phi(\bar{A})_i; \tilde{\sigma}^2_\phi(\bar{A})_i)$
$q(Y_j) = \mathcal{N}(Y_j; \tilde{m}_\psi(\bar{A})_j; \tilde{s}^2_\psi(\bar{A})_j)$
where $\tilde{\mu}_\phi(.)$ (resp. $\tilde{m}_\psi$) is the function that calculates the mean of the normalized adjacency matrix $\bar{A}$ for the latent rows (resp. columns). $\bar{A} = D_1^{-\frac{1}{2}} A D_2^{-\frac{1}{2}}$ where D_1 (resp. D_2) is a diagonal rows (resp. columns) degree matrix of A. The calculations of $\tilde{\mu}_\phi(.)$ and $\tilde{\sigma}_\phi(.)$ are obtained using two graph neural networks such as :

$\tilde{\mu}_\phi(\bar{A}) = GNN_{\tilde{\mu}}(\bar{A}) = \bar{A} ReLU(\bar{A}^T \phi^{(1)}_\mu) \phi^{(2)}_\mu$
$\tilde{\sigma}_\phi(\bar{A}) = GNN_{\tilde{\sigma}}(\bar{A}) = f_{act}(\bar{A} ReLU(\bar{A}^T \phi^{(1)}_\sigma) \phi^{(2)}_\sigma)$

Here $f_{act}(x) = max(-3, x)$
Note that the two GNN can share the first layer ie $\phi^{(1)}_\mu = \phi^{(1)}_\sigma$

On the same principle, the calculations of $\tilde{m}_\psi(.)$ and $\tilde{s}_\psi(.)$ are obtained using two graph neural networks such as
$\tilde{m}_\psi(\bar{A}) = GNN_{\tilde{m}}(\bar{A}) = \bar{A}^T ReLU(\bar{A} \psi^{(1)}_m) \psi^{(2)}_m$
$\tilde{s}_\psi(\bar{A}) = GNN_{\tilde{s}}(\bar{A}) = f_{act}(\bar{A}^T ReLU(\bar{A} \psi^{(1)}_s) \psi^{(2)}_s)$

Note that the two GNN can share the first layer ie $\psi^{(1)}_m = \psi^{(1)}_s$

As we are in variational clustering probabilities, we also have
$q(r_i) = \prod_{i=1}^{M} \mathcal{M}(r_i; 1, \gamma_i)$ and $q(c_j) = \prod_{j=1}^{P} \mathcal{M}(c_j; 1, \delta_i)$
where γ_i (resp. δ_j) the variational probability for each row (resp. column) cluster for the i-th individuals (resp. j-th).

3.2 ELBO

As the Kullback-Leibler divergence is not computable to maximize the variational log-likelihood, we focus on the first term of the sum. We know that the Kullback-Leibler divergence will be a positive term; in other words, the first term can be seen as a lower bound. We now focus on maximizing the evidence lower bound (ELBO).

$$
\begin{aligned}
\mathcal{L}(q(X,Y,r,c);\theta) &= \int_{\mathcal{X}} \int_{\mathcal{Y}} \sum_r \sum_c q(X,Y,r,c) log(\tfrac{P(A,X,Y,r,c|\theta)}{q(X,Y,r,c)})\, dY\, dX \\
&= \int_{\mathcal{X}} \int_{\mathcal{Y}} \sum_r \sum_c q(X,Y,r,c) log(\tfrac{P(A|X,Y,\alpha)P(X|r,\mu_k,\sigma_k)P(Y|c,m_q,s_q)P(r|\pi)P(c|\tau)}{q(X,Y,r,c)})\, dY\, dX \\
&= \mathbb{E}[log(P(A|X,Y,\alpha))] + \mathbb{E}[log(\tfrac{P(X|r,\mu_k,\sigma_k)}{q(X)})] + \mathbb{E}[log(\tfrac{P(Y|c,m_q,s_q)}{q(Y)})] \\
&+ \mathbb{E}[log(\tfrac{P(r|\pi)}{q(r)})] + \mathbb{E}[log(\tfrac{P(c|\tau)}{q(c)})] \\
&= \mathbb{E}\left[\sum_{i=1}^{M} \sum_{j=1}^{P} A_{ij} log(\eta_{ij}) + (1 - A_{ij}) log(1 - \eta_{ij})\right] \\
&- \sum_{i=1}^{M} \sum_{l=1}^{L} \gamma_{ik} D_{KL}(\mathcal{N}(X_i; \tilde{\mu}_\phi(\bar{A})_i; \tilde{\sigma}^2_\phi(\bar{A})_i I_d) || \mathcal{N}(X_i; \mu_l; \sigma^2_l I_D)) \\
&- \sum_{j=1}^{P} \sum_{q=1}^{Q} \delta_{jq} D_{KL}(\mathcal{N}(Y_j; \tilde{m}_\psi(\bar{A})_j; \tilde{s}^2_\psi(\bar{A})_j I_D) || \mathcal{N}(Y_j; m_q; s^2_q I_D)) \\
&+ \sum_{i=1}^{M} \sum_{l=1}^{L} \gamma_{il} log(\tfrac{\pi_l}{\gamma_{il}}) \\
&+ \sum_{j=1}^{P} \sum_{q=1}^{Q} \delta_{jq} log(\tfrac{\tau_q}{\delta_{jq}})
\end{aligned}
$$

where $\eta_{ij} = f_\alpha(X_i, Y_j)$.

The first term of the ELBO, the reconstruction term, can be estimated by drawing Monte Carlo samples from the posterior distribution.
For the following, we write $D_{KL}(\mathcal{N}(X_i; \tilde{\mu}_\phi(\bar{A})_i; \tilde{\sigma}^2_\phi(\bar{A})_i I_d || \mathcal{N}(X_i; \mu_l; \sigma^2_l I_d))$ as D^{il}_{KL} and $D_{KL}(\mathcal{N}(Y_j; \tilde{m}_\psi(\bar{A})_j; \tilde{s}^2_\psi(\bar{A})_j I_d)|| \mathcal{N}(Y_j; m_q; s^2_q I_d))$ as D^{jq}_{KL}.

Note that $D^{il}_{KL} = \frac{1}{2}\left(log(\frac{\sigma^2_l}{\tilde{\sigma}^2_\phi(\bar{A})_i})^D - D + D * \frac{\tilde{\sigma}^2_\phi(\bar{A})_i}{\sigma^2_l} + \frac{1}{\sigma^2_l}||\mu_l - \tilde{\mu}_\phi(\bar{A})_i||^2\right)$
And $D^{jq}_{KL} = \frac{1}{2}\left(log(\frac{s^2_q}{\tilde{s}^2_\psi(\bar{A})_j})^D - D + D * \frac{\tilde{s}^2_\psi(\bar{A})_j}{s^2_q} + \frac{1}{s^2_q}||m_q - \tilde{m}_\psi(\bar{A})_j||^2\right)$

Each of the previous formulas denotes the Kullback-Leibler divergence between the variational distribution and the prior distribution for the rows and columns.

3.3 Parameters Optimization

3.3.1 Explicit Optimization

With the parameters ϕ, ψ and α fixed, we can explicitly optimize the ELBO with respect to the parameters $\gamma; \delta; \pi; \tau; \mu_l; \sigma_l; m_q; s_q$ and obtain the following updates parameters :

$$\hat{\gamma}_{il} = \frac{\pi_l e^{-D_{KL}^{il}}}{\sum_{k=1}^{L} \pi_k e^{-D_{KL}^{ik}}}$$

$$\hat{\delta}_{jq} = \frac{\tau_q e^{-D_{KL}^{jq}}}{\sum_{n=1}^{Q} \tau_n e^{-D_{KL}^{jn}}}$$

$$\hat{\pi}_l = \frac{\sum_{i=1}^{M} \gamma_{il}}{M}$$

$$\hat{\tau}_q = \frac{\sum_{j=1}^{P} \delta_{jq}}{P}$$

$$\hat{\mu}_l = \frac{\sum_{i=1}^{M} \gamma_{il} \tilde{\mu}_\phi(\bar{A})_i}{\sum_{i=1}^{M} \gamma_{il}}$$

$$\hat{m}_q = \frac{\sum_{j=1}^{P} \delta_{jq} \tilde{m}_\psi(\bar{A})_j}{\sum_{j=1}^{P} \delta_{jq}}$$

$$\hat{\sigma}_l^2 = \frac{\sum_{i=1}^{M} \gamma_{il} (D\tilde{\sigma}_\phi^2(\bar{A})_i + ||\mu_l - \tilde{\mu}_\phi(\bar{A})_i||^2)}{D \sum_{i=1}^{M} \gamma_{il}}$$

$$\hat{s}_q^2 = \frac{\sum_{j=1}^{P} \delta_{jq} (D\tilde{s}_\psi^2(\bar{A})_j + ||m_q - \tilde{m}_\psi(\bar{A})_j||^2)}{D \sum_{j=1}^{P} \delta_{jq}}$$

3.3.2 Implicit Optimization

The encoders parameters ϕ, ψ as well as the decoder parameters α will be optimized by performing a stochastic gradient descent on the model.

3.3.3 Algorithm

First, we are doing a pre-training phase in order to get an initialization of the parameters. We are training without the parameters relative to the clusters $(\pi, \tau, \mu_k, \sigma_k, m_q, s_q)$ and we estimate them at the end with a k-means run. Then, we calculate through the encoders $\tilde{\mu}_\phi(\bar{A})_i, \tilde{\sigma}_\phi^2(\bar{A})_i, \tilde{m}_\psi(\bar{A})_j$ and $\tilde{s}_\psi^2(\bar{A})_j$. Next, we update the γ and δ parameters to update the $\pi, \tau, \mu_l, \sigma_l, m_q; s_q$ parameters. When this is done, we reconstruct A' and calculate the loss. Finally, we update ϕ, ψ and α according to the stochastic gradient descent on the loss.

Algorithm 1: Co-clustering estimation

Input: Adjacency matrix A
Output: Reconstructed adjacency matrix A', rows cluster probability matrix γ and columns cluster probability matrix δ

pretrain_model = pretrain(A, pre_epoch) ;
while *ELBO increases* **do**
  $\tilde{\mu}_\phi(\bar{A}), \tilde{\sigma}_\phi^2(\bar{A}), \tilde{m}_\psi(\bar{A}), \tilde{s}_\psi^2(\bar{A}) = \mathrm{GNN}(\bar{A})$;
  X, Y = $\mathcal{N}(\tilde{\mu}_\phi(\bar{A}), \tilde{\sigma}_\phi^2(\bar{A})), \mathcal{N}(\tilde{m}_\psi(\bar{A}), \tilde{s}_\psi^2(\bar{A}))$;
  A' = LPMdecoder(X,Y);
  update γ and *delta*;
  update $\pi; \tau; \mu_l; \sigma_l; m_q; s_q$;
  calculate the ELBO;
  update ϕ ; ψ and α with gradient descent;
end

3.4 Model Architecture

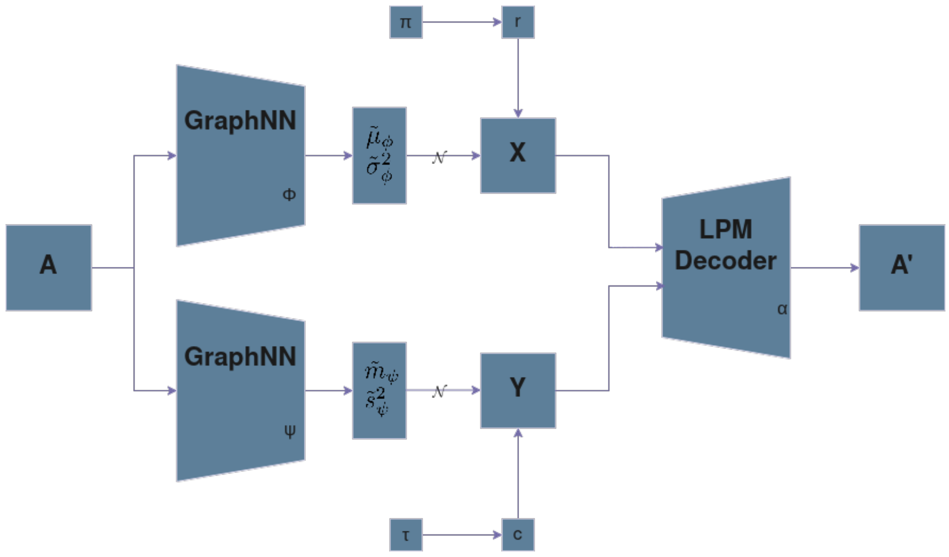

Figure 1: Deep architecture of the model