1 Hyperbolic random graphs

Joint probability of the model:

$$\begin{aligned} p(\boldsymbol{A}, \boldsymbol{r}, \boldsymbol{\phi}, R, T, \alpha) &= p(\boldsymbol{r}, \boldsymbol{\phi}, R, T, \alpha) p(\boldsymbol{A} | \boldsymbol{r}, \boldsymbol{\phi}, R, T, \alpha) \\ &= p(\boldsymbol{r}, \boldsymbol{\phi} | R, T, \alpha) p(\boldsymbol{A} | \boldsymbol{r}, \boldsymbol{\phi}, R, T, \alpha) p(R) p(T) p(\alpha) \end{aligned} \tag{1}$$

The joint probability of hyperbolic coordinates is given by

$$\begin{aligned} p(\boldsymbol{r}, \boldsymbol{\phi} | R, T, \alpha) &= \frac{\alpha \sinh(\alpha \boldsymbol{r})}{2\pi(\cosh(\alpha R) - 1)} \\ &= \prod_{i=1}^{N} \frac{\alpha \sinh(\alpha r_i)}{2\pi(\cosh(\alpha R) - 1)} \end{aligned} \tag{2}$$

The log probability of the edges is given by

$$\log p(\boldsymbol{A} | \boldsymbol{r}, \boldsymbol{\phi}, R, T, \alpha) = \sum_{i,j} \Big(A_{ij} \log\big(p(dist(i,j))\big) + (1 - A_{ij}) \log\big(1 - p(dist(i,j))\big) \Big) \tag{3}$$

For each two nodes the probability of an edge depends on the hyperbolic distance:

$$p\big(dist(u,v)\big) = \left(1 + \exp\left(\frac{1}{2T}\big(dist(u,v) - R\big)\right)\right)^{-1} \tag{4}$$

And the hyperbolic distance between nodes i and j is defined as

$$dist(i,j) = \cosh^{-1}\big(\cosh(r_i)\cosh(r_j) - \sinh(r_i)\sinh(r_j)\cos(\phi_i - \phi_j)\big) \tag{5}$$

The variational distribution will be

$$q(\boldsymbol{r}, \boldsymbol{\phi}, R, T, \alpha) = \prod_i q(r_i, \phi_i) q(R) q(T) q(\alpha) \tag{6}$$

2 ELBO

$$\begin{aligned} &\mathbb{E}_{q(\boldsymbol{r}, \boldsymbol{\phi}, R, T, \alpha)}[\log p(\boldsymbol{A}, \boldsymbol{r}, \boldsymbol{\phi}, R, T, \alpha)] + H(q(\boldsymbol{r}, \boldsymbol{\phi}, R, T, \alpha)) \\ &= \mathbb{E}_{q(R,T,\alpha)}\left[\mathbb{E}_{q(\boldsymbol{r}, \boldsymbol{\phi})}\left[\log p(\boldsymbol{A} | \boldsymbol{r}, \boldsymbol{\phi}, R, T, \alpha) + \log p(\boldsymbol{r}, \boldsymbol{\phi} | R, T, \alpha) + \log p(R) + \log p(T) + \log p(\alpha) \;\middle|\; R, T, \alpha\right]\right] \\ &\quad + H(q(\boldsymbol{r}, \boldsymbol{\phi}, R, T, \alpha)) \\ &= \mathbb{E}_{q(R,T,\alpha)}\left[\mathbb{E}_{q(\boldsymbol{r}, \boldsymbol{\phi})}\left[\sum_{i,j}\Big(A_{ij}\log\big(p(dist(i,j))\big) + (1 - A_{ij})\log\big(1 - p(dist(i,j))\big)\Big) \;\middle|\; R, T, \alpha\right]\right. \\ &\quad \left. + \sum_i \mathbb{E}_{q(r_i, \phi_i)}\left[\log(\sinh(\alpha r_i)) + \log(\alpha) - \log(2\pi) - \log(\cosh(\alpha R) - 1) \;\middle|\; R, T, \alpha\right]\right] \\ &\quad + \mathbb{E}_{q(R)}[\log p(R)] + \mathbb{E}_{q(T)}[\log p(T)] + \mathbb{E}_{q(\alpha)}[\log p(\alpha)] \\ &\quad - \sum_i \mathbb{E}_{q(r_i, \phi_i)}[\log q(r_i, \phi_i)] - \mathbb{E}_{q(R)}[\log q(R)] - \mathbb{E}_{q(T)}[\log q(T)] - \mathbb{E}_{q(\alpha)}[\log q(\alpha)] \end{aligned} \tag{7}$$

3 ELBO for a minibatch

$$
\begin{aligned}
\sum_{\substack{ij\\ i\neq j}} \Bigg[& \frac{1}{N(N-1)} \mathbb{E}_{q(R)} \Big[\log p(R) - \log q(R) \Big] \\
& + \frac{1}{N(N-1)} \mathbb{E}_{q(T)} \Big[\log p(T) - \log q(T) \Big] \\
& + \frac{1}{N(N-1)} \mathbb{E}_{q(\alpha)} \Big[\log p(\alpha) - \log q(\alpha) \Big] \\
& - \frac{1}{N-1} \Bigg(\mathbb{E}_{q(r_i)} \Big[\log q(r_i) \Big] + \mathbb{E}_{q(\phi_i)} \Big[\log q(\phi_i) \Big] \Bigg) \\
& + \frac{1}{N-1} \Bigg(\mathbb{E}_{q(\alpha)} \big[\log(\alpha) \big] + \mathbb{E}_{q(R,\alpha)} \big[\log\big(\cosh(\alpha R) - 1\big) \big] - \log(2\pi) + \mathbb{E}_{q(r_i,\alpha)} \big[\log(\sinh(\alpha r_i)) \big] \Bigg) \\
& + \mathbb{E}_{q(R,T,\alpha)} \Bigg[\mathbb{E}_{q(r_i,\phi_i)} \Bigg[A_{ij} \log\big(p(dist(i,j))\big) + (1 - A_{ij}) \log\big(1 - p(dist(i,j))\big) \,\Bigg|\, R, T, \alpha \Bigg] \Bigg] \Bigg]
\end{aligned}
\tag{8}
$$

3.1 Algorithm implementation

Considering $R \geq 0$, $\alpha \geq 0$, $0 < T \leq 1$, $0 \leq r_i \leq R$ and $0 \leq \phi_i \leq 2\pi$, we assume that

$$
\begin{aligned}
\boldsymbol{R} &\sim \texttt{Gamma}(c_R, s_R) \\
\boldsymbol{\alpha} &\sim \texttt{Gamma}(c_\alpha, s_\alpha) \\
\boldsymbol{T} &\sim \texttt{Beta}(a_T, b_T)
\end{aligned}
$$

where c_R and c_α are shapes (concentrations), s_R and s_α are scales of `Gamma` distributions.

Further, we choose for $q(r_i)$ a `Radius`(μ, σ, R) distribution, which is a `Normal`(μ, σ^2) distribution mapped to a constrained space [0, 1] using *sigmoid* function and then scaled on the interval $[0, R]$. This transformation in Pytorch is rather simple using the functionality of `torch.distributions.transformed_distribution` and `torch.distributions.transforms`. In this way the Jacobian correction is calculated automaticaly by the framework.

The choose of $q(\phi_i)$ is more complicated because it should be a spherical two-dimensional (in Cartesian coordinates) distribution. Ideally, we would take a wrapped normal but calculating its Jacobian troublesome, so we adapted a $\texttt{vonMises} - \texttt{Fisher}$ distribution from [1]. This implementation generally utilize rejection sampling for multi-dimensional distributions but for three-dimensional case a transformation of a uniform distribution is used. Encountering some problems with rejection sampling, which can be very slow, we chosen a three-dimensional variant projected on a two-dimensional plane.

Numerical stability

For numerical stability we used `log1mexp()` function defined in [2]. This allows us to rewrite hyperbolic functions as

$$
\begin{aligned}
\log\big(\sinh(\alpha r)\big) &= \log(1/2) - \log(e^{\alpha r}) + \log(e^{2\alpha r} - 1) \\
&= \log(1/2) - \alpha r + 2\alpha r + \log(1 - e^{-2\alpha r}) \\
&= \log(1/2) + \alpha r + log1mexp(2\alpha r)
\end{aligned}
\tag{9}
$$

and

$$
\begin{aligned}
\log\big(\cosh(\alpha R) - 1\big) &= \log(1/2) - \log(e^{\alpha R}) + 2\log(e^{\alpha R} - 1) \\
&= \log(1/2) - \alpha R + 2\alpha R + \log(1 - e^{-\alpha R}) \\
&= \log(1/2) + \alpha R + 2 log1mexp(\alpha R)
\end{aligned}
\tag{10}
$$

Then from (2)

$$\begin{aligned}\log\big(p(r_i,\phi_i|R,T,\alpha)\big) &= \log\left(\frac{\alpha\sinh(\alpha r_i)}{2\pi(\cosh(\alpha R)-1)}\right)\\ &= \log(\alpha)-\log(2\pi)+\alpha(r-R)+log1mexp(2\alpha r)-2\cdot log1mexp(\alpha R)\end{aligned} \tag{11}$$

Approximation for $\cosh^{-1}$

The inverted cosh operation not only requires additional computation time but also cause some numerical instability.

Consider the hyperbolic distance

$$dist(i,j)=\cosh^{-1}(D)$$

where

$$D(i,j)=\cosh(r_i)\cosh(r_j)-\sinh(r_i)\sinh(r_j)\cos(\phi_i-\phi_j)$$

Observe that for $y \gg 1$:

$$\cosh^{-1}(y) := \log(y+\sqrt{y^2-1}) \approx \log(2y) \tag{12}$$

That allows us to approximate the distance as $dist(i,j) \approx \log(2D)$. It will bound the true distance from below and will be quite precise for large distances. Observe however that if we plugin the approximation into the shifted sigmoid, appears in a region, where the sigmoid is near constant anyhow:

$$\begin{aligned}p(dist(i,j))^{-1} &= 1+\exp\left(\frac{\cosh^{-1}(D)}{2T}\right)\cdot\exp\left(\frac{-R}{2T}\right)\\ &\approx 1+\big(2D\big)^{1/(2T)}\cdot\exp\left(\frac{-R}{2T}\right)\end{aligned} \tag{13}$$

This solution is also not perfect. During the initialization or the further learning process T can take a very small values, which can push the distance to an infinity and produce NaN values. The growth of D can have the same consequences as well as $D=0$, that is why we have to clamp D. Moreover, clamping $p(dist(i,j))$ in the interval $(0,1)$ is also practical.

The result optimization metric

$$\begin{aligned}ELBO_{HRG}(\Lambda) = &-\frac{L}{N^2}D_{KL}\left(q(R)||p(R)\right)\\ &-\frac{L}{N^2}D_{KL}\left(q(T)||p(T)\right)\\ &-\frac{L}{N^2}D_{KL}\left(q(\alpha)||p(\alpha)\right)\\ &+\sum_{(i,j)\in\Lambda}\mathbb{E}_{q(R,T,\alpha)}\left[\mathbb{E}_{q(r_i,\phi_i)}\left[A_{ij}\log\big(p(dist(i,j))\big)+(1-A_{ij})\log\big(1-p(dist(i,j))\big)\;\Big|\;R,T,\alpha\right]\right]\\ &+\frac{L}{N^3}\sum_{(i,j)\in\Lambda}\mathbb{E}_{q(r_i,\alpha)}\big[\alpha(r_i-R)+log1mexp(2\alpha r_i)\big]\\ &+\frac{L}{N^2}\quad\mathbb{E}_{q(R,\alpha)}\big[\log(\alpha)-\log(2\pi)-2\cdot log1mexp(\alpha R)\big]\\ &-\frac{L}{N^3}\sum_{(i,j)\in\Lambda}\left(\mathbb{E}_{q(r_i)}\Big[\log q(r_i)\Big]+\mathbb{E}_{q(\phi_i)}\Big[\log q(\phi_i)\Big]\right)\end{aligned} \tag{14}$$

References

[1] TR Davidson, L Falorsi, N De Cao, T Kipf, JM Tomczak. (2018). Hyperspherical variational Auto-Encoders. arXiv:1804.00891
`https://github.com/nicola-decao/s-vae-pytorch`

[2] Mächler, Martin. (2015). Accurately Computing $\log(1 - exp(-|a|))$ Assessed by the Rmpfr package. 10.13140/RG.2.2.11834.70084.