The S^1 x S^1 Latent Model for Bipartite Networks: Properties and Parameters Inference

Francesco Barbara[a]

[a]Mathematical Institute, University of Oxford, Radcliffe Observatory Quarter, Woodstock Road, Oxford OX2 6GG, UK

Bipartite networks possess several properties of interest that latent geometry frameworks for standard networks are not able to capture. In this work, we first briefly introduce the features commonly observed in real-world bipartite graphs (power-law degree distributions, high number of common neighbours and bipartite clustering coefficients) and their corresponding mathematical definitions. We then review a powerful latent-geometry framework specific to bipartite networks and proceed to show, through numerical simulations, that this model maintains several of the aforementioned properties and give original insights on its limitations. Finally, we propose a novel Montecarlo-based approach for the inference of the hidden parameters underlying the latent geometry.

Bipartite Networks | Latent Geometry | Nested Montecarlo Estimators

Real-world bipartite networks exhibit a number of shared properties of interest (1). These include:

Heterogeneity in the degree distribution. Often the degree distribution $P(d)$ is well approximated by a power-law (2):

$$P(d) \sim d^{-\gamma} \quad \gamma > 2$$

High number of common neighbours. Let A and B represent the two node partitions of a bipartite network with edge-list E. For two nodes i, $j \in A$, the number of common neighbours m_{ij} is defined as:

$$m_{ij} = |\{l \in B : (i,l) \in E, (j,l) \in E\}|$$

In real-world networks, the set $\{m_{ij} : i,, j \in A\}$ also tends to exhibit a fat-tailed distribution (this means that there are nodes in the same partition which tend to share several neighbours) which cannot be explained by the heterogeneity in $P(d)$ alone (3) (4).

4-loops and high bipartite clustering coefficients. Analogies with the mechanism of triadic closure (5) and the consequent high number of triangles in standard networks, can be observed in the context of bipartite graphs.
A 4-loop is defined as a set of four nodes i, $j \in A$ and k, $l \in B$ such that (i,l), (i,k), (j,l), $(j,k) \in E$.
As a consequence of the large number of common neighbours, bipartite graphs are also rich in 4-loops.

A measure aimed at quantifying the density of 4-loops around a specific node is the bipartite clustering coefficient (6):

For a node $i \in A$ let $N(i) \subseteq B$ be the set of neighbours of i.

$$c(i) = \frac{\sum_{j \neq l \in N(i)} |N(j) \cap N(l)|}{\sum_{j \neq l \in N(i)} |N(j) \cup N(l)|} = \frac{\sum_{j \neq l \in N(i)} (m_{jl} - 1)}{\sum_{j \neq l \in N(i)} (k_j + k_l - m_{jl} - 1)}$$

where k_j and k_l are the degrees of node j and l.
$c(i)$ can be thought of as representing a 'normalised density' of 4-loops containing node i.

For a given partition A the mean clustering coefficient can be defined as:

$$\bar{c}_A = \frac{1}{|A|} \sum_{i \in A} c(i)$$

Models of bipartite networks. A common approach is to model a bipartite graph as an AB random geometric graph (10). In such case the two partitions are modelled as two independent Poisson processes in a Euclidean space with connection between two points happening when their distance is below a specified threshold.
A second approach, the $\mathbb{S}^1 \times \mathbb{S}^1$ model (11), uses the idea of mapping nodes to points in a latent geometry and equipping them with hidden variables. The probability of two nodes forming a connection depends on their geometric distance and the values of their hidden variables.

1. Formalism of the $\mathbb{S}^1 \times \mathbb{S}^1$ model

Assume the two partitions of the network (which we shall refer to as top and bottom partition) have N and M nodes respectively. We associate each partition with a circumference of radius R.

Each node $i \in \{1, ..., N\}$ belonging to the top partition has a latent representation (θ_i, κ_i) where:
$\theta_i \overset{\text{i.i.d.}}{\sim} Unif[0, 2\pi]$ encodes the position of node i on the circumference (notably $(R\,cos(\theta_i),\ R\,sin(\theta_i))$) and
$\kappa_i \overset{\text{i.i.d.}}{\sim} p(\kappa)$ is a hidden variable drawn from a more general distribution (sensible choices of $p(\kappa)$ and the corresponding properties induced on the graph are discussed in later sections).

Similarly, each node j in the bottom partition has a latent representation (ϕ_j, λ_j) with $\phi_j \overset{\text{i.i.d.}}{\sim} Unif[0, 2\pi]$ and $\lambda_j \overset{\text{i.i.d.}}{\sim} p(\lambda)$.

Intuitively, nodes i and j which are closer on the circumference (i.e. with similar values of θ_i and ϕ_j) are more likely

Significance Statement

Bipartite networks are able to model several phenomena such as recommender systems (7) and scientific collaboration networks (8). Such networks are often studied by projecting them onto one of the two bipartite-components (9), resulting in significant loss of structure. Recently, some ad-hoc latent geometries for the description of bipartite networks have been proposed. We review in detail one of such models, analyse its properties and, by drawing from the field of nested Montecarlo estimators, we propose a novel approach for inference of some of the latent parameters of the model.

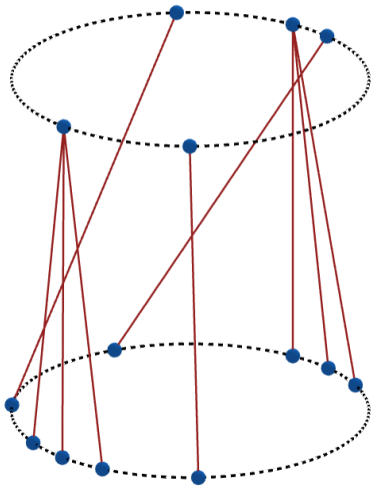

Fig. 1. Syntethic bipartite graph generated according to the $\mathbb{S}^1 \times \mathbb{S}^1$ model

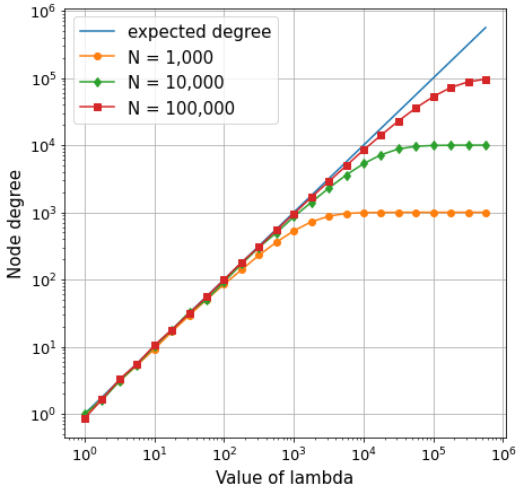

Fig. 2. Each point is the average degree of a node with hidden variable λ. Parameters of the $\mathbb{S}^1 \times \mathbb{S}^1$ model were chosen as following:
$p(\kappa) \sim \kappa^{-3}$ is power-law distributed (with $\kappa_0 = 3.5$ picked arbitrarily, and power-law parameter chosen greater than 2 to guarantee finiteness of $\bar{k}$).
$M = 100$, $R = 1$, $\mu = \frac{\pi R}{N\,(\pi/\beta)\,csc(\pi/\beta)\bar{\kappa}}$.
Three different values of N were chosen; 1,000 graphs were simulated and the average degree of bottom nodes was computed (for N = 100,000 only 100 graphs were simulated due to limited computing resources).

to be connected by an edge; moreover degree heterogeneity is induced by varying values of the hidden variables κ_i and λ_j, with higher values of the corresponding hidden variable making a node likelier to form connections (**Fig. 1.**).

Formally the connection probability r_{ij} between a top node i and a bottom node j is given by:

$$r_{ij} = r\left(\frac{d(\theta_i, \phi_j)}{d_c(\kappa_i, \lambda_j)}\right)$$

$$\text{where } r(x) = (1 + x^{\beta})^{-1} \text{ with } \beta \in (1, \infty)$$

$$d(\theta_i, \phi_j) = R(\pi - |\pi - |\theta_i - \phi_j||)$$

$$d_c(\kappa_i, \lambda_j) = \mu\kappa_i\lambda_j$$

We can see that the above intuitions are all satisfied by such formulation: $r(x) \to 0$ as $x \to \infty$ and $r(x) \to 1$ as $x \to 0$ (x can be thought of as a notion of distance, when two nodes have a high distance in the latent geometry they are less likely to be connected in the network), and $d(\theta_i, \phi_j)$ measures the 'angle difference' between the nodes.

2. Mean degree in the $\mathbb{S}^1 \times \mathbb{S}^1$ model

For a bottom node with given hidden variable λ and angle ϕ, its expected degree $\bar{l}(\lambda, \phi)$ under the model can be written as a double integral by first conditioning on the hidden value κ of a top node and its angle θ, and then calculating the connection probability between two nodes with fixed representations (θ, κ) and (ϕ, λ):

$$\bar{l}(\lambda, \phi) = N \int_{\kappa} p(\kappa) \int_{\theta} \frac{1}{2\pi} r\left(\frac{d(\theta, \phi)}{\kappa\lambda}\right) d\theta d\kappa$$

Because of angular symmetry, the expression must be independent of ϕ. Even though such double integral cannot generally be computed analytically, (12) showed that for $N \gg \lambda$ (which is generally the case for large networks, or small sparsely connected graphs) the following approximation holds:

$$\bar{l}(\lambda) \approx \frac{\mu\, N\, (\pi/\beta)\, csc(\pi/\beta)}{\pi R} \cdot \bar{\kappa}\, \lambda$$

$$\text{where } \bar{\kappa} = \mathbb{E}\left[\kappa\right] = \int_{\kappa} p(\kappa) \cdot \kappa d\kappa$$

$$\implies \bar{l} := \mathbb{E}_{p(\lambda)}\left[\bar{l}(\lambda)\right] \approx \frac{\mu\, N\, (\pi/\beta)\, csc(\pi/\beta)}{\pi R} \cdot \bar{\kappa}\, \bar{\lambda}$$

By symmetry, an analogous approximation can be obtained for the expected degree of a top node.

Numerical simulations for the mean degree approximation. We verify to which extent such approximation holds through a number of simulations (see **Fig. 2.**):
For each value of λ we simulate 1,000 graphs where each bottom node j has the hidden value fixed to $\lambda_j = \lambda$ and we then take the average of the observed degrees. Moreover, to make the graph more interpretable, as suggested in (12), for each value of λ we choose a corresponding scaling parameter $\mu = \frac{\pi R}{N\,(\pi/\beta)\,csc(\pi/\beta)\bar{\kappa}}$ so that the expected degree approximation simplifies to $\bar{l}(\lambda) \approx \lambda$.

Clearly, $\bar{l}(\lambda)$ is bounded by N, as any bottom node can form at most N connections with the top partition. As anticipated, the difference between the above approximation and the empirical results grows as λ increases (extremely high values of λ imply that the distance proxy is approximately zero, leading to a connection probability $r_{ij} \approx 1$ and to bipartite graphs that are almost complete). The approximation, nevertheless, shows excellent performance for more moderate values of λ (which is where most real-world networks belong).

3. Degree distribution in the $\mathbb{S}^1 \times \mathbb{S}^1$ model

Despite being a quantity of interest, the mean degree $\bar{l}$ of a graph does not reveal much about the overall degree distribution of the network.

For a node with fixed hidden variable λ, let $g(l|\lambda)$ be its degree distribution. (13) showed that, in the case of sparse

bipartite networks, $g(l|\lambda) \approx e^{-\lambda}\,\lambda^{l}\,/\,l!$, so its degree distribution is approximately Poisson distributed (for the full proof we refer to Section III.A of (13)).

Integrating over λ, we obtain that the degree distribution $P(l)$ can be approximated as:

$$P(l) = \int_{\lambda} p(\lambda)\, g(l|\lambda)\, d\lambda \approx \int_{\lambda} p(\lambda) e^{-\lambda}\, \frac{\lambda^{l}}{l!}\, d\lambda$$

Power-law distributed hidden variables produce power-law degree distributions. A very natural research question to ask, given this approximation, is how to pick a latent distribution $p(\lambda)$ such that a synthetic bipartite network generated according to the $\mathbb{S}^1 \times \mathbb{S}^1$ has a specific degree distribution.

We now proceed to show that, under the above approximation, choosing $p(\lambda) \sim \lambda^{-\gamma}$ results in a degree distribution which is approximately a power-law with the same parameter $P(l) \sim l^{-\gamma}$.

Let $p(\lambda) = (\gamma-1)\lambda_0^{\gamma-1}\lambda^{-\gamma}$; using the above approximation:

$$P(l) \approx \int_{\lambda=\lambda_0}^{\infty} (\gamma-1)\,\lambda_0^{\gamma-1}\,\lambda^{-\gamma}\, e^{-\lambda}\, \frac{\lambda^{l}}{l!}\, d\lambda$$

$$= \frac{(\gamma-1)\,\lambda_0^{\gamma-1}\Gamma(l-\gamma+1)}{\Gamma(l+1)} \int_{\lambda=\lambda_0}^{\infty} \frac{e^{-\lambda}\lambda^{(l-\gamma)}}{\Gamma(l-\gamma+1)}\, d\lambda$$

We can see that the integrand is the density of a $Gamma(l-\gamma+1, 1)$. For small values of λ_0 (which is generally the case), the integral will be approximately equal to 1, giving:

$$P(l) \approx \frac{(\gamma-1)\,\lambda_0^{\gamma-1}\Gamma(l-\gamma+1)}{\Gamma(l+1)} \approx (\gamma-1)\,\lambda_0^{\gamma-1}\,l^{-\gamma}$$

Experiments for the power-law degree distribution approximation. We now proceed to test numerically how well such approximation manages to describe the structure of networks generated using the $\mathbb{S}^1 \times \mathbb{S}^1$ model.

To this purpose, we generated a network with $M = 10,000$ bottom nodes and $N = 1,000$ top nodes, where $p(\lambda) \sim \lambda^{-2.5}$. We then compared the observed degree distribution with the approximated degree distribution obtained from $P(l) \approx (\gamma-1)\,\lambda_0^{\gamma-1}\,l^{-\gamma}$.

It can be observed in **Fig. 3.** that the behaviour around the mean degree in the log plot is approximately linear (meaning that the observed degrees are also power-law distributed), and that the coefficients for the two lines are roughly the same (implying that the power-law coefficient $\gamma = 2.5$ is empirically observed).

Despite these good properties, it can be observed that the range of node degrees in the approximated model is smaller than in the true model (where we see nodes with degrees around $exp(6) \approx 400$ and nodes with fewer than 3 connections)

Our interpretation of such finding is that, while approximating the gamma integral with 1 is not problematic for large values of l (as most mass lies above the threshold λ_0), it is not a good approximation for medium-to-small values of l, causing more mass to be allocated to such points. As a consequence, when the approximate density is normalised, high values of l are assigned values in the p.m.f. much lower than what they should be, resulting in a thin-tailed distribution (it can be observed in **Fig. 3.** that no nodes with degree higher than $exp(5)$ are present in the approximated model). Moreover, using the Poisson approximation proposed in (13) implies that the node degrees are narrowly distributed around their hidden parameter (i.e. the mean of the Poisson distribution in this case), bringing additional justification for the fact that the approximation model has a more concentrated distribution.

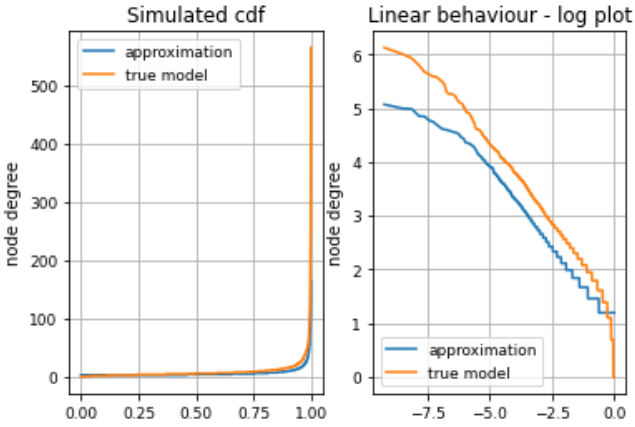

Fig. 3. For the true model a graph with the following parameters of the $\mathbb{S}^1 \times \mathbb{S}^1$ model was simulated:
$N = 1,000$; $M = 10,000$; $\beta = 2.5$; $R = 1$; $\mu = \frac{\pi\, R}{N\,(\pi/\beta)\,csc(\pi/\beta)\bar{\kappa}}$.
$p(\kappa) \sim \kappa^{-2.5}$; $\kappa_0 = 3.3$, hence $\bar{\kappa} = 10$.
$p(\lambda) \sim \lambda^{-2.5}$; $\lambda_0 = 3.3$, hence $\bar{\lambda} = 10$.
Its bottom-degree distribution was compared to a degree distribution of 10,000 samples drawn from the approximate model $P(l) \approx (\gamma-1)\,\lambda_0^{\gamma-1}\,l^{-\gamma}$, with $\gamma = 2.5$.
Left plot: The cumulative distributions of the node degrees for the two models are plotted. We plot $\{(p_i, d_i)\}_{i=1}^{10,000}$ where d_i is the degree of bottom node i, and p_i is the value of the empirical cdf evaluated at d_i
Right plot: In order to make the linear behaviour appear, we plot $\{(log(1-p_i), log(d_i))\}_{i=1}^{10,000}$. In the case of a power-law distribution this would result in a line with slope equal to $-\gamma+1$.

4. Bipartite clustering coefficients and common neighbours distributions

In addition to heterogeneity in the degree distribution, two other common properties of real-world bipartite networks are a high number of common neighbours, and high bipartite clustering coefficients (the formal definitions for both quantities were given in the introductory section).

We now aim to show that the $\mathbb{S}^1 \times \mathbb{S}^1$ model possesses both these properties.

To this purpose, we simulate a $\mathbb{S}^1 \times \mathbb{S}^1$ network with $N = M = 1,000$ nodes for each partition. We then generate a second graph with the same degree distribution using a bipartite variant of the configuration model.

For each pair of nodes in the bottom partition (approximately 1 million), we calculate the number of common neighbours in both networks. **Fig. 4.** shows that the $\mathbb{S}^1 \times \mathbb{S}^1$ model clearly exhibits a higher number of common neighbours.

In **Fig. 5.** we instead calculate the clustering coefficient for all bottom nodes, which results in a neat separation of such distribution for the two models.

These results can be intuitively explained by the fact that bottom nodes lying close together on the circumference, will tend to connect with the same top nodes, giving rise to such properties.

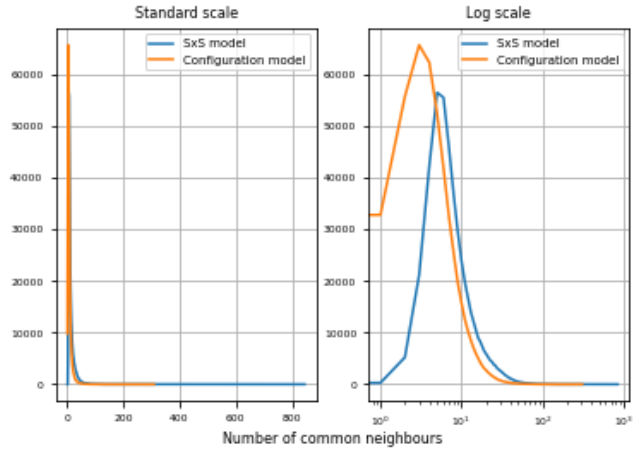

Fig. 4. A graph with the following parameters of the $\mathbb{S}^1 \times \mathbb{S}^1$ model was simulated: $N = 1,000$; $M = 1,000$; $\beta = 2.5$; $R = 0.1$; $\mu = \frac{\pi\, 10\, R}{N\,(\pi/\beta)\,csc(\pi/\beta)\bar{\kappa}}$.
$p(\kappa) \sim \kappa^{-2.5}$; $\kappa_0 = 3.3$; $\bar{\kappa} = 10$.
$p(\lambda) \sim \lambda^{-2.5}$; $\lambda_0 = 3.3$; $\bar{\lambda} = 10$.
The distribution of the number of common neighbours over all bottom node pairs is plotted for the two models on both a standard and a log scale.

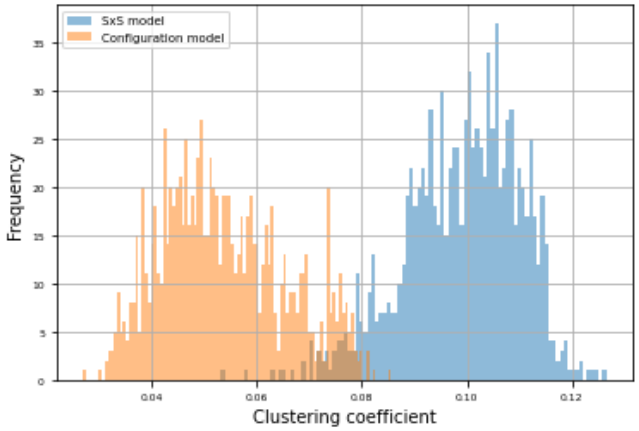

Fig. 5. A graph with the following parameters of the $\mathbb{S}^1 \times \mathbb{S}^1$ model was simulated: $N = 1,000$; $M = 1,000$; $\beta = 2.5$; $R = 0.1$; $\mu = \frac{\pi\, 10\, R}{N\,(\pi/\beta)\,csc(\pi/\beta)\bar{\kappa}}$.
$p(\kappa) \sim \kappa^{-2.5}$; $\kappa_0 = 3.3$; $\bar{\kappa} = 10$.
$p(\lambda) \sim \lambda^{-2.5}$; $\lambda_0 = 3.3$; $\bar{\lambda} = 10$.
The distributions of the clustering coefficients for the $\mathbb{S}^1 \times \mathbb{S}^1$ and Configuration model are plotted.

5. Point estimates for latent parameters

Another key feature of a good latent geometry model, is the possibility to infer the latent representation of a given network. For the $\mathbb{S}^1 \times \mathbb{S}^1$ model, little work has been done so far on the matter.

In this section we draw from recent advances in the theory of Nested Montecarlo Estimators to propose a novel method to estimate the angular distance between two nodes of the same partition.

Theoretical set-up. Suppose a $\mathbb{S}^1 \times \mathbb{S}^1$ is generated with known model parameters (β, R, μ, $p(\kappa)$, $p(\lambda)$).

As already shown in sections 2 and 3, a bottom node with hidden variable λ has a degree distribution which is approximately Poissonian with mean $\bar{l}(\lambda) \approx \frac{\mu\, N\,(\pi/\beta)\, csc(\pi/\beta)}{\pi R} \cdot \bar{\kappa}\,\lambda$. A simple but effective point estimate (which is present in (12)) for the hidden variable λ when degree l_{obs} is observed consists of simply inverting the above approximation to get:

$$\lambda_{est} = \frac{\pi R}{\mu\, N\,(\pi/\beta)\, csc(\pi/\beta)\,\bar{\kappa}} \cdot l_{obs} \qquad (Approx.\ 1)$$

An analogous point estimate for hidden variables κ_i can be obtained.
We are now left with the considerably more complex task of generating point estimates for the angle parameters $\{\theta_i\}_{i=1}^{N}$ and $\{\phi_j\}_{j=1}^{M}$.

Given two top nodes (κ_1, θ_1) and (κ_2, θ_2) with known parameters, the expected number of common neighbours can be written as:

$$\bar{m}(\kappa_1,\, \theta_1,\, \kappa_2,\, \theta_2) = \frac{M}{2\pi} \int_\lambda \int_\phi p(\lambda)\, \mathbb{P}(connection \,|\, \kappa_1, \theta_1, \lambda, \phi) \cdot$$
$$\cdot \mathbb{P}(connection \,|\, \kappa_2, \theta_2, \lambda, \phi)\, d\lambda\, d\phi$$

(the integrand can be easily seen to represent the probability that both nodes are connected to a specific bottom node with parameters (λ, ϕ)). By plugging-in our definition of connection probability r, (12) showed that such quantity can be expressed as:

$$\frac{\sqrt{\kappa_1 \kappa_2}}{I\bar{\lambda}} \int \lambda p(\lambda) \int_{-\infty}^{+\infty} r\left(\sqrt{\frac{\kappa_2}{\kappa_1}}|x|\right) r\left(\sqrt{\frac{\kappa_1}{\kappa_2}}|x - \Delta\tilde{\theta}_{12}|\right) dx\, d\lambda$$

where $r(x) = \frac{1}{1+x^\beta}$ as usual, and $\Delta\tilde{\theta}_{12} := NI\bar{\kappa}\Delta\theta_{12}$ with $\Delta\theta_{12} = \pi - |\pi - |\theta_1 - \theta_2||$ representing the usual angular distance.

The result makes sense from an intuitive level as its dependency on θ_1 and θ_2 is only through the distance between the two (which was expected because of angular symmetry). However, because

$$\int r(x)\, dx = \int \frac{1}{1+x^\beta} = x\ {}_2F_1(1; \frac{1}{b}; \frac{b+1}{b}; -x^b)$$

(${}_2F_1$ is the hypergeometric function), this means that $\bar{m}(\kappa_1,\, \theta_1,\, \kappa_2,\, \theta_2)$ cannot be easily approximated.

Montecarlo approximation. Nevertheless, if we fix $\beta = 2$ (which is a reasonable parameter choice - most of the work we read on the $\mathbb{S}^1 \times \mathbb{S}^1$ model had $\beta \in [2, 3]$ and our experiments used $\beta = 2.5$), we derived the following expression for $\bar{m}(\kappa_1,\, \kappa_2,\, \Delta\theta_{12})$:

$$\frac{\kappa_1 \pi}{I\bar{\lambda}} \int \lambda p(\lambda) \int_{-\infty}^{+\infty} \left(\frac{1}{\pi} \cdot \frac{1}{1+x^2}\right) r\left(\frac{\kappa_1}{\kappa_2}x - \sqrt{\frac{\kappa_1}{\kappa_2}}\Delta\tilde{\theta}_{12}\right) dx\, d\lambda$$

This quantity can be rewritten as a nested expectation:

$$\bar{m}(\kappa_1,\, \kappa_2,\, \Delta\theta_{12}) = \frac{\kappa_1 \pi}{I\bar{\lambda}} \mathbb{E}_\lambda \left[\lambda\, \mathbb{E}_x \left[r\left(\frac{\kappa_1}{\kappa_2}x - \sqrt{\frac{\kappa_1}{\kappa_2}}\Delta\tilde{\theta}_{12}\right)\right]\right]$$

where $\lambda \sim p(\lambda)$ and $x \sim p(x)$ with $p(x) = \frac{1}{\pi}\frac{1}{1+x^2}$

We can proceed to approximate the above expectation with a Nested Montecarlo estimator $\bar{m}^{MC}(\kappa_1, \kappa_2, \Delta\theta_{12})$ defined as:

$$\frac{\kappa_1 \pi}{I\bar{\lambda}} \frac{1}{n} \sum_{i=1}^{n} \left(\lambda_i \frac{1}{m} \sum_{j=1}^{m} r \left(\frac{\kappa_1}{\kappa_2} x_j - \sqrt{\frac{\kappa_1}{\kappa_2}} \Delta\tilde{\theta}_{12} \right) \right)$$

where $\lambda_i \overset{\text{i.i.d.}}{\sim} p(\lambda)$ and $x_j \overset{\text{i.i.d.}}{\sim} p(x)$.
This expression satisfies the conditions of the work of Hong and Juneja (14), ensuring that if n and m are chosen such that $n \propto m^2$, then given computational resources $T = n \cdot m$, the expected squared error is of order $T^{-2/3}$:

$$\mathbb{E}\left[\left(\bar{m}(\kappa_1, \kappa_2, \Delta\theta_{12}) - \bar{m}^{MC}(\kappa_1, \kappa_2, \Delta\theta_{12}) \right)^2 \right] = O\left(\frac{1}{T^{2/3}} \right)$$

Inferring angular distances between nodes. Let κ_1 and κ_2 be point estimates for the latent variables obtained by (*Approx.* 1). A good point estimate for the angular distance $\Delta\theta_{12}$ is:

$$\Delta\theta_{12}^{est} = arg\,min_{\Delta\theta_{12}} \left(\bar{m}(\kappa_1, \kappa_2, \Delta\theta_{12}) - m_{12}^{obs} \right)^2$$

This means that we aim to choose the value of $\Delta\theta_{12} \in [0, \pi]$ that produces the mean number of common neighbours that is the closest to the observed number of common neighbours between the two nodes.

Despite being now able to approximate $\bar{m}(\kappa_1, \kappa_2, \Delta\theta_{12})$, we still do not have a method to solve the outer optimisation problem (i.e. the choice of $\Delta\theta_{12}$ itself).

A key property of $\bar{m}(\kappa_1, \kappa_2, \Delta\theta_{12})$ is that it is decreasing with respect to $\Delta\theta_{12}$ (intuitively, as the distance increases, the number of common neighbours goes down). This allows us to solve the optimisation part of the problem by using simple binary search.

Algorithm 1 Estimation of $\Delta\theta_{12}$

Inputs: Hidden variables estimates κ_1, κ_2; model parameters ($p(\lambda)$, $p(\kappa)$, μ, R), total number of samples T for each NMC approximation, number of iterations, observed m_{12}
Set $n = T^{2/3}$, $m = T^{1/3}$
Initialise $\Delta\theta_{min} = 0$, $\Delta\theta_{max} = \pi$
for $i \in \{1, ..., iterations\}$ **do**
    $\Delta\theta_{est} = (\Delta\theta_{min} + \Delta\theta_{max})/2$
    $m_{est} = NestedMC(\Delta\theta_{est}, n, m, parameters = \{\kappa_1, \kappa_2, model\,parameters\})$
    if $m_{est} \geq m_{12}$ **then** $\Delta\theta_{min} = \Delta\theta_{est}$
    else $\Delta\theta_{max} = \Delta\theta_{est}$
Return $\Delta\theta_{est} = (\Delta\theta_{min} + \Delta\theta_{max})/2$

In order to test the performance of this algorithm, we simulated 250 pairs of top nodes (the choice of simulating pairs, rather than a full network, was made to guarantee independence of the pairs, but the process roughly corresponds to simulating a network and then estimating distances between all top pairs).

For each pair, we simulated a full bottom partition with $M = 1,000$ and connections between the two nodes and the bottom partition were created according to the $\mathbb{S}^1 \times \mathbb{S}^1$ model.

We chose $R = 0.1$ to make the graph highly connected (the parameters choice is analogous to **Fig. 4.**). In fact, choosing $R = 1$ gives a sparsely connected network where most nodes have no neighbours in common. In such case, it is clear to see that our algorithm outputs π as a point estimate for $\Delta\theta_{12}$ (because the lowest possible expected number of common neighbours is achieved when two points are diametrically opposite on the circumference).
To avoid trivial estimates, node pairs with no common neighbours were not considered.

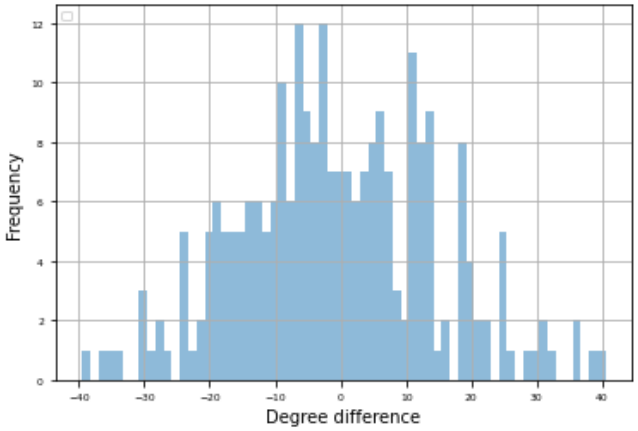

Fig. 6. Values of $\Delta\theta_{est} - \Delta\theta_{real}$ for 250 node pairs (expressed in degrees). The following parameters of the $\mathbb{S}^1 \times \mathbb{S}^1$ model were used: $M = 1,000$; $\beta = 2$; $R = 0.1$; $\mu = \frac{\pi\, 10\, R}{N\,(\pi/\beta)\,csc(\pi/\beta)\bar{\kappa}}$. $p(\kappa) \sim \kappa^{-2.5}$; $\kappa_0 = 3.3$, $\bar{\kappa} = 10$, $p(\lambda) \sim \lambda^{-2.5}$; $\lambda_0 = 3.3$, $\bar{\lambda} = 10$.

We can see in **Fig. 6.** that $\Delta\theta_{est} - \Delta\theta_{real}$ (expressed in degrees, rather than radians for interpretability) are generally between -20 and 15 degrees (we performed 8 iterations to allow a granularity of roughly 1 degree), which represents a satisfying result, especially considered the lack of available research on the matter.
The distribution appears to be biased towards the left. We think this is due to having removed pairs with $m_{ij} = 0$ (which produced a few high outliers of $\Delta\theta_{est} - \Delta\theta_{real}$ for the reason mentioned above).

Conclusion and future research directions. We believe that our proposed method for estimating angular distances is a good starting point; however, several pieces are still missing to make full inference possible (i.e. given a real network we would ideally like to return its full inferred latent representation $(\kappa_i, \theta_i)_{i \leq N}$, $(\lambda_j, \phi_j)_{j \leq M}$ and its model parameters $\{\mu, R, \beta, p(\lambda), p(\kappa)\}$).

In Section 3, we showed how a power-law distribution for the hidden variables induced a power-law distribution on the node degrees, therefore a sensible way of inferring $p(\lambda)$ and $p(\kappa)$ might already be at our disposal for this specific case; but estimation of μ, R and β remains an open problem.
Moreover, a final question of interest, which was not examined in this paper, is how to output angle estimates $\{\theta_i\}_{i=1}^{N}$ given $n(n-1)/2$ estimated angular distances $\{\Delta\theta_{ij}\}$.
A possible idea would be a greedy approach where you start by placing the first node at $\theta_1 = 0$ and then for each of the following node j you pick θ_j by taking the average over the set $\{\theta_i + \Delta\theta_{ij}\}_{i<j}$ for nodes i that have already been assigned a hidden variable. Several more robust algorithms and constructions for this purpose might be within reach.

1. MEJ Newman, *Networks: an introduction*. (Oxford University Press, Oxford; New York), (2010).
2. M Latapy, C Magnien, N Del Vecchio, Basic notions for the analysis of large affiliation networks/bipartite graphs. *arXiv preprint cond-mat/0611631* (2006).
3. E Burgos, et al., Two classes of bipartite networks: Nested biological and social systems. *Phys. review. E, Stat. nonlinear, soft matter physics* **78**, 046113 (2008).
4. A Iamnitchi, M Ripeanu, I Foster, Small-world file-sharing communities in *IEEE INFOCOM 2004*. (IEEE), Vol. 2, pp. 952–963 (2004).
5. T Opsahl, Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. *Soc. networks* **35**, 159–167 (2013).
6. P Zhang, et al., Clustering coefficient and community structure of bipartite networks. *Phys. A: Stat. Mech. its Appl.* **387**, 6869–6875 (2008).
7. J Bennett, S Lanning, The netflix prize. (2007).
8. MEJ Newman, The structure of scientific collaboration networks. *Proc. Natl. Acad. Sci.* **98**, 404–409 (2001).
9. S Banerjee, M Jenamani, DK Pratihar, Properties of a projected network of a bipartite network in *2017 International Conference on Communication and Signal Processing (ICCSP)*. (IEEE), pp. 0143–0147 (2017).
10. J Dall, M Christensen, Random geometric graphs. *Phys. review E* **66**, 016121 (2002).
11. MÁ Serrano, M Boguná, F Sagués, Uncovering the hidden geometry behind metabolic networks. *Mol. biosystems* **8**, 843–850 (2012).
12. M Kitsak, F Papadopoulos, D Krioukov, Latent geometry of bipartite networks. *Phys. Rev. E* **95**, 032309 (2017).
13. M Kitsak, D Krioukov, Hidden variables in bipartite networks. *Phys. Rev. E* **84**, 026114 (2011).
14. LJ Hong, S Juneja, Estimating the mean of a non-linear function of conditional expectation. *Proc. 2009 Winter Simul. Conf. (WSC)* pp. 1223–1236 (2009).

A

Code. Code for simulating a synthetic $\mathbb{S}^1 \times \mathbb{S}^1$ network. Full code for the experiments is provided in an attached *.py* file.

```
def sample_power_law(gamma = 2, kappa_0 = 1, sample_size = 10):
    '''Given the parameters of a power-law distribution it returns samples drawn from it'''
    #sampling independent uniform random variables
    uniform_samples = np.random.uniform(size = sample_size)

    #applying inversion to obtain i.i.d.samples from a power-law distribution
    samples = kappa_0 * ( uniform_samples ** (1 / (1 - gamma)) )
    return samples

def distance(theta, phi, R):
    '''Returns the angular distance introduced in the paper'''
    return R*(np.pi - abs(np.pi - abs(theta - phi)))

def distance_scale(kappa, Lambda, mu):
    '''Returns the scaling value dependent of the hidden parameters'''
    return mu*kappa*Lambda

def connection_power_law(x, beta_parameter):
    '''Returns r(x) defined in the paper, i.e. the connection probability'''
    return 1/(1+x**beta_parameter)

class LatentBipartiteGraph():
    '''Class for bipartite graphs generated according to the S^1xS^1 model'''
    def __init__(self, N, M, connection_parameter, kappa_distr = ('power_law', 2.5, 1), \
                 lambda_distr = ('power_law', 2.5, 1), kappa = None, Lambda = None, R = 1, mu = 1):

        thetas = 2*np.pi * np.random.uniform(size = N)

        if kappa != None:
            kappas = np.repeat(kappa, N)

        else:
            if kappa_distr[0] == 'power_law':
                kappas = sample_power_law(gamma = kappa_distr[1], kappa_0 = kappa_distr[2], \
                                          sample_size = N)
            elif kappa_distr[0] == 'poisson':
                kappas = np.repeat(kappa_distr[1], N)
            else:
                raise ValueError('Kappa distribution not supported')


        phis = 2*np.pi * np.random.uniform(size = M)

        if Lambda != None:
            lambdas = np.repeat(Lambda, M)

        else:
            if lambda_distr[0] == 'power_law':
                lambdas = sample_power_law(gamma = lambda_distr[1], kappa_0 = lambda_distr[2], \
                                          sample_size = M)
            elif lambda_distr[0] == 'poisson':
                lambdas = np.repeat(lambda_distr[1], M)
            else:
                raise ValueError('Lambda distribution not supported')

        #creating the graph pbject in networkx
        G = nx.Graph()
        #adding nodes
        G.add_nodes_from(range(N), bipartite=0)
        G.add_nodes_from(range(N, N+M), bipartite=1)

        for i in range(N):
            for j in range(M):
                #adding edges using the connection_power_law probability
                if np.random.uniform() <= \
                    connection_power_law(distance(thetas[i], phis[j], R) / \
                                         distance_scale(kappas[i], lambdas[j], mu),\
                                             connection_parameter):
                    G.add_edge(i, N+j)

        self.thetas, self.phis, self.kappas, self.lambdas = thetas, phis, kappas, lambdas
        self.graph = G
```