NUMERICAL APPROXIMATIONS FOR SPEEDING UP MCMC INFERENCE IN THE INFINITE RELATIONAL MODEL

Mikkel N. Schmidt and Kristoffer Jon Albers

Cognitive Systems, DTU Compute
Technical University of Denmark
Richard Petersens Plads, DTU Bldg. 321.
2800 Lyngby, Denmark.

ABSTRACT

The infinite relational model (IRM) is a powerful model for discovering clusters in complex networks; however, the computational speed of Markov chain Monte Carlo inference in the model can be a limiting factor when analyzing large networks. We investigate how using numerical approximations of the log-Gamma function in evaluating the likelihood of the IRM can improve the computational speed of MCMC inference, and how it affects the performance of the model. Using an ensemble of networks generated from the IRM, we compare three approximations in terms of their generalization performance measured on test data. We demonstrate that the computational time for MCMC inference can be reduced by a factor of two without affecting the performance, making it worthwhile in practical situations when on a computational budget.

Index Terms— Nonparametric Bayesian modeling, Infinite Relational Model, Numerical approximation.

1. INTRODUCTION

A common approach to modeling the structure in complex network data is to cluster the nodes of the network into groups which have similar structural properties. Discovering groups of nodes which connect to other nodes in a similar fashion is useful for unsupervised, explorative analysis of complex networks. Using non-parametric Bayesian models, one can find cluster structure which is statistically salient and learn the apropriate number of clusters at the same time.

Many different non-parametric Bayesian models of complex networks exist in the literature. The arguably simplest model is the so-called infinite relational model (IRM) [1–3], which is a non-parametric Bayesian extension of the stochastic blockmodel [4, 5]. Since exact inference in the IRM is intractable for networks with more than a few nodes, the clustering is typically learned using approximate inference techniques such as Markov chain Monte Carlo (MCMC) [6], or variational Bayes [7].

When analyzing very large complex networks, the computational speed of the inference procedure can become an issue [8]. There are, at least, four different ways in which one might consider speeding up the inference procedure when performing cluster analysis of complex networks. i) The model can be simplified in a way to permit analytic solutions to part of the problem or in other ways achieve faster inference. ii) Approximate inference algorithms with a better speed-accuracy trade-off can be employed. iii) Computational optimizations such as parallelization, vectorization, and lookup tables can be implemented. iv) Numerical approximations can be computed at a lower precision.

In this paper we investigate how low-precision numerical approximations can be used to speed up MCMC inference in the infinite relational model. We examine how changes in the numerical precision affects the inferred clustering structure. In particular we test different numeric approximations to the evaluation of the log-gamma function, which consitutes the majority of the work in a Gibbs sampler for the IRM.

2. THE INFINITE RELATIONAL MODEL

The IRM extends the stochastic blockmodel, by relying on a nonparametric prior over partitions to flexibly allow the number of clusters in the model to be learned from the data. This allows the model to adapt to the size and complexity of the data.

Restrict the discussion to the modeling of simple unipartite networks, the IRM can be formulated as the following generative process,

$$z|\alpha \sim \mathrm{CRP}(\alpha), \tag{1}$$
$$\theta_{k,\ell}|a,b \sim \mathrm{Beta}(a,b), \tag{2}$$
$$A_{i,j}|\theta,z \sim \mathrm{Bernoulli}(\theta_{z_i,z_j}), \tag{3}$$

where $A_{i,j}$ is a binary variable indicating whether or not there exists a link between node i and j. The prior for the cluster assignment, z is a Chinese restaurant process (CRP) governed by the concentration parameter α,

$$p(z|\alpha) = \frac{\Gamma(\alpha)\alpha^K}{\Gamma(\alpha + N)} \prod_{k=1}^{K} \Gamma(m_k), \tag{4}$$

where N is the number of nodes, K is the number of clusters, and m_k is the number of nodes in cluster k. The probability of observing a link between two nodes i and j, follows a Bernoulli distribution, depending only on z and the parameters $\theta_{k,\ell}$ which specifies the link probability between nodes in the two clusters k and ℓ, that i and j are assigned to. A Beta distribution with parameters a and b is used as a prior for these link probabilities,

$$p(\theta_{k,\ell}|a, b) = \frac{\theta_{k,\ell}^{a-1}(1 - \theta_{k,\ell})^{b-1}}{B(a, b)}. \tag{5}$$

Due to the conjugacy of the Bernoulli likelihood and Beta prior, the parameters θ can be analytically marginalized yielding the following likelihood,

$$p(A|z, a, b) = \prod_{k=1}^{K} \prod_{\ell=k+1}^{K} \frac{\mathrm{B}(m_{k,\ell} + a, \bar{m}_{k,\ell} + b)}{\mathrm{B}(a, b)}, \tag{6}$$

where $m_{k,\ell}$ and $\bar{m}_{k,\ell}$ denote the number of links and non-links between nodes in cluster k and ℓ, and B denotes the Beta function,

$$\mathrm{B}(x, y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x + y)}. \tag{7}$$

In the following we will keep α constant and assume improper flat priors over a and b.

2.1. Computations for a Gibbs sampler

The computations required to implement a Gibbs sampler for the cluster assignments z can be found by considering the change in the likelihood (and prior) when moving a node to each of the existing clusters or to a new cluster. Reassigning node n to the cluster k will change the likelihood by the factor

$$\prod_{\ell} \frac{\mathrm{B}(m_{k,\ell}^{\setminus n} + r_{n,\ell} + a,\ \bar{m}_{k,\ell}^{\setminus n} + n_\ell - r_{n,\ell} + b)}{\mathrm{B}(m_{k,\ell}^{\setminus n} + a,\ \bar{m}_{k,\ell}^{\setminus n} + b)}, \tag{8}$$

where the count statistics $m_{k,\ell}^{\setminus n}$ and $\bar{m}_{k,\ell}^{\setminus n}$ are the number of links and non-links between cluster k and ℓ, ignoring node n, and $r_{n,\ell}$ is the number of links between n and all nodes in cluster ℓ (see [3] for details). The count statistics for links and non-links between clusters can be kept and updated, instead of recomputed for each Gibbs iteration, and thus the required computations for implementing the Gibbs sampler is dominated by evaluating the Beta function.

In a practical implementation, computations will be performed in the log domain in order to ensure numeric stability. Calculating the logarithm of the Beta function is therefore the central operation for evaluating the likelihood within the Gibbs sampler. Thus, it is important to have efficient means for computing the logarithm of the Gamma function, in order to efficiently compute the logarithm of the Beta function,

$$\log \mathrm{B}(x, y) = \log \Gamma(x) + \log \Gamma(y) - \log \Gamma(x + y). \tag{9}$$

2.2. Computational optimization

In some situations, computing the log-Gamma function can be completely avoided by computational optimization. From Eq. (8) it is evident, that if a and b are constant, the log-Gamma function need only be evaluated at integer steps ($I + a$, $I + b$, and $I + a + b$, for integer I). Thus, it might be practical to simply precompute a large lookup table of log-Gamma values. However, depending on the data, the required lookup table might be impractically large, and if a and b are allowed to vary, computing a large lookup table before each Gibbs sweep is not practical.

2.3. Approximation by maximization

The reason that the Beta function arises is the analytical marginalization of θ. The joint distribution of θ and the data for a single block (all links and non-links between nodes in cluster k and ℓ) is given by

$$p(A_{k,\ell}, \theta|z, a, b) = \frac{\theta^{m+a-1}(1 - \theta)^{\bar{m}+b-1}}{\mathrm{B}(a, b)}, \tag{10}$$

where, to simplify the exposition, we have omitted the k, ℓ subscripts in the parameter and count statistics. Marginalizing θ yields the term $\mathrm{B}(m + a, \bar{m} + b)$. A crude approximation is to replace the marginalization by plugging in the maximum a-posteriori (MAP) estimate of θ, which yields

$$\frac{(m + a - 1)^{m+a-1}(\bar{m} + b - 1)^{\bar{m}+b-1}}{(n + a + b - 2)^{n+a+b-2}}. \tag{11}$$

Taking the logarithm and comparing with Eq. (9) we see that the MAP-plugin estimate corresponds exactly to approximating the log-Gamma function using Stirling's approximation (the variant for the factorial function), $\log n! = \log \Gamma(n) \approx n \log(n) - n$. This gives some understanding as to what happens algorithmically when approximating the log-Gamma function in this manner.

2.4. Directly approximating the log-Gamma function

Many well-known approximations to the log-Gamma function exist, having different trade-off between computational complexity and precision (see Fig. 1). One of the simplest is Stirling's approximation, given by

$$\log \Gamma(x) \approx \tfrac{1}{2} \log(2\pi) + (x - \tfrac{1}{2}) \log(x) - x. \tag{12}$$

Stirling's approximation is relatively fast to compute, as it only involves a single logarithm and a multiplication (disregarding additions and computing the constant in advance). Stirling's approximation yields an asymptotically accurate approximation, but is not very precise for small values of its argument.

We have discovered a very similar approximation with the same computational complexity,

$$\log \Gamma(x) \approx 1 + (x - 2 + \tfrac{1}{\log(2)}) \log(x) - x. \tag{13}$$

This approximation is not an asymptotic formula, but it has better precision for small values, $x < 4$, and thus a better worst case error. As we have not been able to find this approximation in the literature, we refer to it in the following as Schmidt's approximation.

Gosper's approximation,

$$\log \Gamma(x) \approx 1 + \tfrac{1}{2} \log([2x - \tfrac{5}{3}]\pi) + (x-1)\log(x-1) - x, \tag{14}$$

involving two logarithms and two multiplications, yields better asymptotic behavior but is still very imprecise for low values.

Lanczos' family of approximations [9] include the particularly interesting $\gamma = 1.5$ approximation with only two terms, which we refer to as Lanczos 1.5,

$$\log \Gamma(x) \approx \tfrac{1}{2}\log(2\pi) + (x + \tfrac{1}{2})\log(x+1) - (x+1) + \log(c_0 + \tfrac{c_1}{x}), \tag{15}$$

where c_0 and c_1 are constants. It involves two logarithms, one multiplication, and one division but yields a very good precision also for small values, having a relative error of no more than $2.4 \cdot 10^{-4}$ [9]. By increasing the number of terms in the Lanczos' approximation to N (requiring two logarithms, one multiplication and N divisions) the log-Gamma function can be computed to arbitrary precision.

2.5. Approximating the logarithm

Since the discussed approximations of the log-Gamma functions still require the computation of one or more logarithms, which itself requires evaluating some series expansion, nothing much appears to be gained. However, the logarithm can be approximated very cheaply. In most computer systems, floating point numbers are represented by a sign (S), a mantissa (M), and an exponent (E),

$$x = (-1)^S \cdot M \cdot 2^E, \tag{16}$$

where $0.5 \leq M \leq 1$. For positive x, the logarithm is given as

$$\log(x) = \log(M) + \log_2(e)^{-1}E. \tag{17}$$

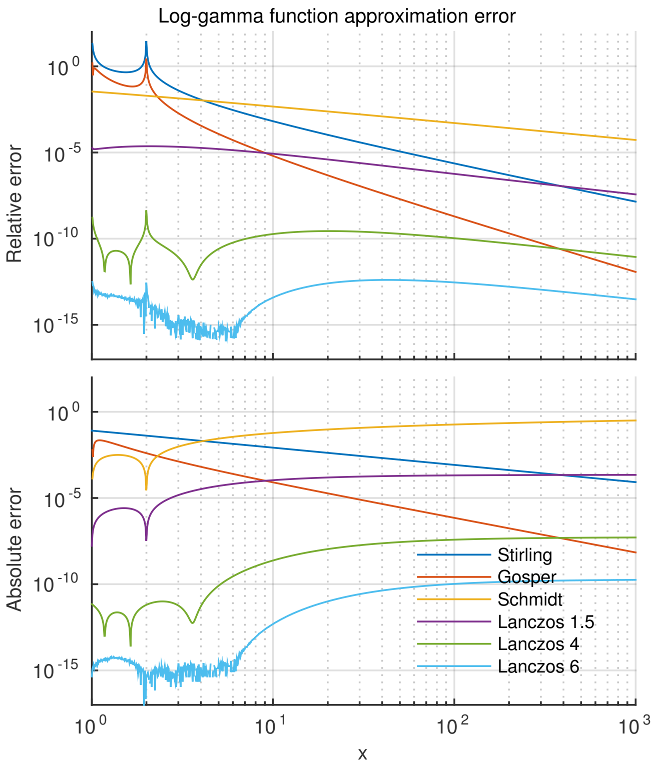

Fig. 1. Relative and absolute error in approximating $\log \Gamma(x)$ using various methods.

Thus, we need only one multiplication and the computation of the logarithm of the mantissa. Since the mantissa by definition is between one half and one, its logarithm can be approximated very efficiently.

Classical expansions, such as the rational expansion

$$\log(x) \approx 2\left(r + \frac{r^3}{3}\right), \quad r = \frac{x-1}{x+1}, \tag{18}$$

and the Taylor series (at a)

$$\log(x) \approx \log(a) + \frac{x-a}{a} - \frac{(x-a)^2}{2a^2} + -\frac{(x-a)^3}{3a^3} - \dots, \tag{19}$$

are not particularly precise when using a low order (see Fig 2). Approximating the logarithm direcly using lookup table is also not very precise, even for quite large tables. However, creating a lookup table of Taylor expansions yields very fast and precise approximations. For example, using a table with 1024 first order Taylor approximations yield a relative error below 10^{-5} and requires only one lookup and one division. Preferably, the division can be changed to a lookup and a multiplication by storing precomputed values of $\frac{1}{a}$.

The number of elements in the lookup table can be decided in advance, and by keeping the lookup table small enough, the entire is likely to fit the CPU cache at runtime, avoiding expensive access to the main memory.

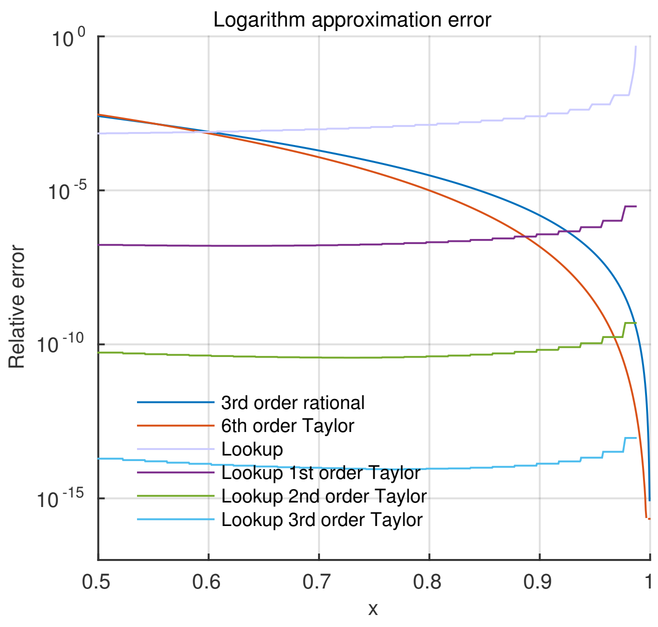

Fig. 2. Relative error in approximating $\log(x)$. The Taylor series shown is expanded around $a = 1$, and all lookup tables are of size 1024.

3. EXPERIMENTAL EVALUATION

To examine the influence of different approximation strategies in the IRM, we conducted an empirical evaluation of the performance of the model on a set of generated networks. We created 1000 random networks sampled from the IRM with 50 nodes, and parameters $a = b = \alpha = 1$. For each of the 1000 networks we generated z and θ from the prior and then sampled two network realizations, using one set for training and one set for testing.

To evaluate and compare the influence of the numeric approximations on the performance of IRM, the model was run on the same data with three different ways of approximating the log-Gamma function as well as with the log-Gamma computed to machine precision. In all approximations we computed logarithms using a length 1024 lookup table of first order Taylor expansions.

For each network, the model was fitted using 10,000 Gibbs sweeps over the clustering z interleaved with 100,000 Metropolis-Hastings (MH) updates of a and b. We repeatedly conducted 10 Gibbs sweeps followed by 100 MH updates, thinning the MCMC sample by a factor of 10. The proposal distribution for the MH update was a Normal distribution with standard deviation 0.1.

To compare the generalization performance, we computed the posterior mean log-likelihood averaged over the 1000 test networks. For each iteration number, we averaged over all previous MCMC samples in order to evaluate the test log-likelihood as a function of the number of iterations (see Figure 3.)

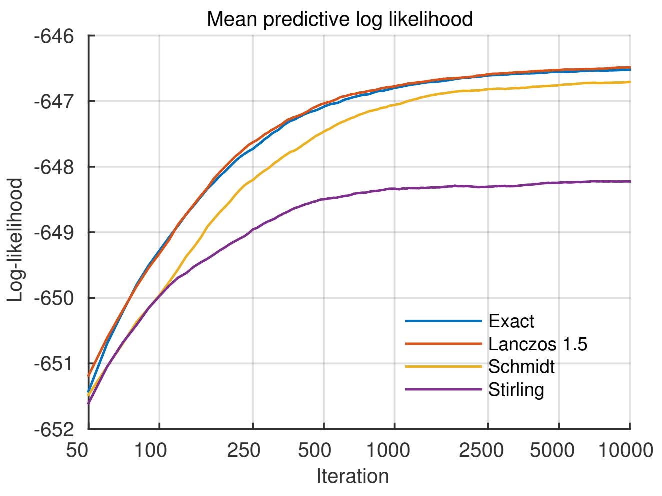

Fig. 3. Average test log-likelihood of the inferred model as a function of the number of iterations (Gibbs sweeps) of the MCMC sampler.

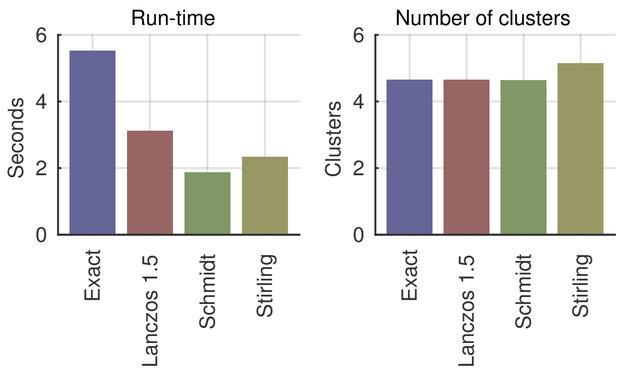

Fig. 4. Performance of the model using the different approximations, in terms of the average computation time and estimated number of clusters.

We recorded only the time it took to conduct the Gibbs sweeps, disregarding time for loading data, conducting MH updates, and storing intermediate results etc.

3.1. Results

Figure 3 shows that the performance of the Lanczos 1.5 approximation is indistinguishable from the exact computation. The performance of Schmidt's approximation is close but significantly worse, and Stirling's approximation performs much worse.

Figure 4 shows the run-time as well as the number of clusters discovered. For comparison, we note that the average number of clusters in the ensemble of networks used for training and test is $\alpha\psi(\alpha + n) - \psi(\alpha) \approx 4.49$. As expected, the three approximations are significantly faster than the machine precision computations: Lanczos 1.5 is around 2 times faster, and Schmidt is around 3 times faster. Although Stirling's approximation has the same complexity as Schmidt's, it runs slower: This can be explained by examining the number of clusters discovered by the different algorithms. While Lanczos 1.5 and Schmidt find almost exactly the same number of clusters as the exact computations, Stirlings approximation leads the inference procedure to erroneously discover more clusters which in turn impacts the computation time, making Stirlings approximation both less accurate and slower.

4. CONCLUSIONS

Introducing numerical approximation in evaluating the likelihood of a Bayesian model will influence the inference. In non parametric models, the inferred model complexity depends on the complexity of the data. Hence, introducing numerical approximations will likely affect the inferred complexity of the model making it difficult to assess the repercussions of the approximation compared to parametric models.

In our experiments we observed that the inferred number of components in the IRM depends on the chosen approximation of the log-Gamma function. Stirling's approximation is very imprecise for small values of its arguments, and using this approximation introduces bias in the model, which turns out to have a significant influence on the performance of the model: The MCMC procedure converges, but to solutions with a lower test likelihood and with notably more clusters. The proposed Schmidt's approximation is more precise in the low range, and though it does not appear to overestimate the number of components it converges to a lower test likelihood than the exact computation. The Lanczos 1.5 approximation on the other hand appears to yield results indistinguishable from the exact computation at around half the computational cost.

There appear to be no reason not to use the Lanczos 1.5 approximation in practical applications of IRM analyses, when one is on a tight computational budget. If one is willing to accept some error in approximation, Schmidt's approximation is to be preferred over Stirling's approximation.

In a practical implementation it is likely to be beneficial to use a hybrid approach, e.g. using a Lanczos approximation for small values, while relying on Gosper's approximation (for better precision) and/or Stirling's approximation (for lower computation complexity) for larger values. As mentioned, alternative to the approximations discussed here, one should also consider if it is more efficient to simply precompute a lookup table of the needed log-Gamma value: This, however, also depends on the data—both on the size of the network, the number of links, and the number of inferred clusters.

Acknowledgement: This project was supported by the Lundbeck Foundation, grant nr. R105-9813.

REFERENCES

[1] Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffiths, Takeshi Yamada, and Naonori Ueda, "Learning systems of concepts with an infinite relational model," *Cognitive Science*, vol. 21, no. 1, pp. 381, 2006.

[2] Zhao Xu, Volker Tresp, Kai Yu, and Hans-Peter Kriegel, "Infinite Hidden Relational Models," in *Proceedings of the 22nd International Conference on Uncertainity in Artificial Intelligence*, 2006.

[3] Mikkel N. Schmidt and Morten Mørup, "Nonparametric Bayesian modeling of complex networks: An introduction," 2013.

[4] Tom A. B. Snijders and Krzysztof Nowicki, "Estimation and prediction for stochastic blockmodels for graphs with latent block structure," *Journal of Classification*, vol. 14, no. 1, pp. 75–100, 1997.

[5] Paul W. Holland, Kathryn B. Laskey, and Samuel Leinhardt, "Stochastic blockmodels: First steps," *Social Networks*, vol. 5, no. 2, pp. 109–137, 1983.

[6] Kristoffer Jon Albers, Andreas Leon Aagard Moth, Morten Mørup, and Mikkel N. Schmidt, "Large scale inference in the Infinite Relational Model: Gibbs sampling is not enough," in *IEEE International Workshop on Machine Learning for Signal Processing, MLSP*, 2013.

[7] Katsuhiko Ishiguro, Issei Sato, and Naonori Ueda, "Collapsed Variational Bayes Inference of Infinite Relational Model," arXiv:1409.4757v1, 2014.

[8] Karen S. Ambrosen, Tue Herlau, Tim Dyrby, Mikkel N. Schmidt, and Morten Mørup, "Comparing Structural Brain Connectivity by the Infinite Relational Model," in *Pattern Recognition in NeuroImaging (PRNI)*, 2014.

[9] Cornelius Lanczos, "A Precision Approximation of the Gamma Function," *Journal of the Society for Industrial and Applied Mathematics*, vol. 1, pp. 86–96, 1964.