The Infinite Contextual Graph Markov Model

Daniele Castellana [*1] **Federico Errica** [*1 2 3] **Davide Bacciu** [1] **Alessio Micheli** [1]

Abstract

The Contextual Graph Markov Model (CGMM) is a deep, unsupervised, and probabilistic model for graphs that is trained incrementally on a layer-by-layer basis. As with most Deep Graph Networks, an inherent limitation is the need to perform an extensive model selection to choose the proper size of each layer's latent representation. In this paper, we address this problem by introducing the Infinite Contextual Graph Markov Model (ICGMM), the first deep Bayesian nonparametric model for graph learning. During training, ICGMM can adapt the complexity of each layer to better fit the underlying data distribution. On 8 graph classification tasks, we show that ICGMM: i) successfully recovers or improves CGMM's performances while reducing the hyper-parameters' search space; ii) performs comparably to most end-to-end *supervised* methods. The results include studies on the importance of depth, hyper-parameters, and compression of the graph embeddings. We also introduce a novel approximated inference procedure that better deals with larger graph topologies.

1. Introduction

It can be argued that one of the most daunting processes in machine learning is the selection of appropriate hyper-parameters for the task at hand. Indeed, due to the data-dependent nature of the learning problem, there usually exists no single model configuration that works well in all contexts. The most straightforward approach to mitigate this issue has typically been to rely on standard model selection techniques such as grid and random searches (Bergstra & Bengio, 2012), where the range of values to try are fixed a priori by the user. Nonetheless, there has always been an interest in alternative methods that can *estimate* the "right" values for some hyper-parameters (Gershman & Blei, 2012; He et al., 2021), thus intrinsically reducing the hyper-parameters' search space. In the Bayesian nonparametric (BNP) literature, which is of particular interest for this work, the complexity of Bayesian models can grow *with the data* (Teh et al., 2006), e.g., a BNP mixture model can adjust the number of its mixtures to better fit the empirical data distribution, thus freeing the user from the burden of choosing the most important (if not all) hyper-parameters.

In recent years, much research effort has been devoted to the theoretical and practical development of Deep Graph Networks (DGNs), which originated from Micheli (2009); Scarselli et al. (2009) and were later re-discovered from a spectral graph theory perspective (Defferrard et al., 2016; Kipf & Welling, 2017). DGNs can deal with graphs of varying topology without the need for human intervention, and they rely on local and iterative processing of information commonly known as *message passing*; for a thorough description of some of the most popular DGNs in the literature (and of the more general graph representation learning field), we refer the reader to recent surveys on the topic (Bronstein et al., 2017; Battaglia et al., 2018; Bacciu et al., 2020b; Wu et al., 2020). Despite most of these methods belonging to the neural world, the Contextual Graph Markov Model (CGMM) stands out as a deep, unsupervised, constructive and fully probabilistic model that has shown competitive performances on downstream graph classification tasks (Bacciu et al., 2018; 2020a). CGMM trains a stack of Bayesian networks, where each layer is conditioned on the *frozen* posteriors of the nodes of the graph computed at previous layers. Each layer optimizes the likelihood of the data using the Expectation Maximization (EM) algorithm (Moon, 1996) with closed-form solutions.

Like its neural counterparts, for which the number of hidden units in each layer has typically been selected as a hyper-parameter, CGMM relies on model selection to choose the "reasonable" number of hidden states associated with the categorical latent variables. Differently from the neural methods though, CGMM is amenable to a BNP extension, as each layer is essentially a conditional mixture model.

[*]Equal contribution [1]Department of Computer Science, University of Pisa, Italy [2]NEC Laboratories Europe, Heidelberg, Germany [3]Work primarily done as a PhD student at the University of Pisa. Correspondence to: Daniele Castellana <daniele.castellana@di.unipi.it>, Federico Errica <federico.errica@neclab.eu>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

For these reasons, our goal is the design of a deep, Bayesian nonparametric model for graph learning that can estimate the value of most hyper-parameters, e.g., the number of states, from the data itself. We achieve this by providing a Bayesian nonparametric treatment of CGMM. The principal difficulty lies in how to handle the variable-size number of neighbors of each node, which in CGMM is solved by (possibly weighted) convex combinations of the neighbors' posterior distributions. The resulting model, called Infinite Contextual Graph Markov Model (ICGMM), can generate as many latent states as needed to solve the unsupervised density estimation task at each layer. To the extent of our knowledge, this is the first Bayesian nonparametric model for adaptive graph processing. We compare ICGMM against CGMM as well as end-to-end supervised methods on eight different graph classification tasks, following a fair, robust and reproducible experimental procedure (Errica et al., 2020). The results demonstrate that ICGMM can infer most of its hyper-parameters without sacrificing its predictive performances. In particular, we will show that ICGMM's scores are on par or better than CGMM and that these are comparable to state of the art *supervised* methods. We complement the analysis with studies on how depth affects the performances and how the remaining hyper-parameters influence the number of chosen latent states. In turn, the latter has a significant effect on the final unsupervised graph embeddings' size. As a further contribution, we provide a faster implementation of our method that scales to the social datasets considered in this work without any performance variations. Our hope is to show how the cross-fertilization of ideas from different research fields can help us advance the state of the art, both in the methodological and empirical sense.

2. Related Works

The fundamental Bayesian nonparametric literature that is relevant to our work relates to the families of Dirichlet Processes (DPs) (Gershman & Blei, 2012) and Hierarchical Dirichlet Processes (HDPs) (Teh et al., 2006). In its most essential definition, a DP is a stochastic process that defines a probability distribution over other probability distributions. A DP is parametrized by a base distribution $\boldsymbol{G_0}$, i.e., the expected value of the process, and a scaling parameter α_0 that controls the concentration of DP realizations around $\boldsymbol{G_0}$ (Teh, 2010). In particular, the Chinese Restaurant Process (Aldous, 1985), the Stick-breaking Construction (Sethuraman, 1994) and the Pòlya urn scheme (Hoppe, 1984) are all alternative ways to formalize a DP. Moving to HDPs is conceptually straightforward, in that it considers the base distribution $\boldsymbol{G_0}$ as a draw from another DP parametrized by a base distribution $\boldsymbol{H}$ and a scaling parameter γ. For a detailed treatment of learning with DP and HDPs, the reader can check a number of tutorials and surveys (Teh et al., 2006; Orbanz & Teh, 2010; Gershman & Blei, 2012). Our work shares similarities with the Infinite Hidden Markov Model for temporal series (Beal et al., 2002), with the fundamental differences that causality assumptions have to be relaxed to deal with graphs and that the hidden variables' distributions are conditioned on a varying number of observations.

Most of the recent advances of the graph representation learning field are based on the so-called feedforward DGNs (Bacciu et al., 2020b). These models rely on "spatial" graph convolutional layers, i.e., the state of each node in the graph is determined by applying a permutation invariant function to its neighboring states computed at the previous layers. Combined with the depth of the architecture, these models propagate contextual information across the graph, a process also known as "message passing" (Gilmer et al., 2017). However, to the best of our knowledge, the only neural method for graphs that constructs part of its architecture in a principled way is the pioneering work of Micheli (2009). In fact, the Neural Network for Graphs (NN4G), known to be the first spatial DGN (see Wu et al. 2020), relies on the Cascade Correlation learning algorithm (Fahlman & Lebiere, 1990) to determine a suitable number of layers for the task under investigation. Recently, in this area of research, other works apply regularization terms (Zhou et al., 2021) and partial differential equations (Eliasof et al., 2021) to dynamically control the behavior of the network and counteract the oversmoothing effect.

Orthogonally to our research direction, the AutoML field seeks to automatically choose the architectural details and hyper-parameter values without the need for human intervention (He et al., 2021). We fundamentally differ from AutoML methods as ICGMM adapts its complexity during training, thus reducing the search space of hyperparameters. In contrast, AutoML strategies automatically explore the given hyperparameters' space in smarter ways than classical grid/random search algorithms.

3. Method

This Section introduces the details of our method. Since we borrow ideas from two relatively distant fields, we define a unified mathematical notation and jargon as well as a high-level overview of the CGMM and HDP models to ease the subsequent exposition.

We define a graph as a tuple $g = (\mathcal{V}_g, \mathcal{E}_g, \mathcal{X}_g)$ where $\mathcal{V}_g$ is the set of entities (also referred to as nodes or vertices), $\mathcal{E}_g$ is the set of oriented edges (u, v) connecting node u to v, and the symbol $\mathcal{X}_g$ stands for the set of node attributes associated with the graph g. Also, the neighborhood of a node u is the set of nodes connected to u, i.e., $\mathcal{N}_u = \{v \in \mathcal{V}_g | (v, u) \in \mathcal{E}_g\}$. For the purpose of this work, we will define the (categorical or continuous) node feature of a node

u with the term $x_u \in \mathcal{X}_g$.

3.1. Basics of CGMM

To best understand how and why this work extends CGMM, we now give a brief but essential description of its main characteristics. CGMM is, first and foremost, a deep architecture for the adaptive processing of graphs. Like other DGNs, it maps the entities of a graph, if not the graph itself, into latent representations. More specifically, we can get one of such representations for each layer of the architecture and then concatenate all of them to obtain richer node and graph embeddings. The latter is usually obtained as a global aggregation of the former.

The second peculiarity of CGMM is that it is constructive, i.e., trained in an incremental fashion: after one layer is trained, another one can be stacked atop of it and trained using the frozen outputs of the previous layer. This idea is borrowed from NN4G (Micheli, 2009), and it allows CGMM to relax the mutual dependencies between latent variables in a cyclic graph. However, because the local and iterative message passing mechanism used by spatial methods (Micheli, 2009; Kipf & Welling, 2017) is responsible for information propagation across the graph, this relaxation is not restrictive.

Thirdly, the node/graph embedding construction is fully probabilistic and unsupervised, since layer ℓ is represented as the Bayesian network on the left-hand-side of Figure 1. A latent variable q_u^ℓ is attached to each node u, and it is responsible for the generation of the node feature x_u. To take into account structural information, the hidden state q_u^ℓ is conditioned on the neighboring hidden states computed at the previous layer, i.e., the set $\{q_v^{\ell-1} \mid v \in \mathcal{N}_u\}$. Importantly, the constructive approach allows us to treat the hidden (frozen) states of the previous layer as observable variables. Each layer is trained to fit the data distribution of node features using the EM algorithm, thus guaranteeing the convergence to a local minimum. Once inference is performed, the state of each node is frozen and we can move to the subsequent layer. Lastly, the embedding of each node at layer ℓ is encoded as the posterior of its hidden state.

3.2. Basics of HDP

The HDP is a Bayesian nonparametric prior for the generation of grouped data using different infinite mixture models with shared mixture components. Let $\{x_1, x_2, \dots\}$ be a set of observations that are grouped into J groups, i.e., each observation x_u belongs to the group $j_u \in \{1, \dots, J\}$. Using the stick-breaking representation (Sethuraman, 1994), the HDP mixture model that generates the observations can be defined as (Teh et al., 2006):

$$\begin{aligned}
&\boldsymbol{\beta} \mid \gamma \sim \text{Stick}(\gamma) && q_u \mid j_u, (\boldsymbol{\pi}_j)_{j=1}^{J} \sim \boldsymbol{\pi}_{j_u} \\
&\boldsymbol{\pi}_j \mid \boldsymbol{\beta}, \alpha_0 \sim \text{DP}(\alpha_0, \boldsymbol{\beta}) && x_u \mid q_u, (\boldsymbol{\theta}_{\boldsymbol{c}})_{c=1}^{\infty} \sim F(\boldsymbol{\theta}_{q_u}) \\
&\boldsymbol{\theta} \mid \boldsymbol{H} \sim \boldsymbol{H},
\end{aligned} \tag{1}$$

where $F(\boldsymbol{\theta}_{q_u})$ denotes the emission distribution, parametrized by $\boldsymbol{\theta}_{q_u}$, that generates the observation x_u. The latent state q_u indicates which mixture component should be used to generate x_u. The value of q_u is sampled from the distribution $\boldsymbol{\pi}_{j_u}$, which stands for the mixture weights of group j_u. All $(\boldsymbol{\pi}_j)_{j=1}^{J}$ are obtained from a DP with concentration parameter α_0 and base distribution β. Notably, all groups' mixture weights are defined on the same set of mixture components, meaning there is a form of parameter sharing across different groups. Finally, we sample the distribution $\boldsymbol{\beta}$ via the stick-breaking process Stick(γ) of Sethuraman (1994).

To generate a possibly infinite number of emission distributions, we exploit a prior distribution $\boldsymbol{H}$ that allows us to create new mixture components on demand. Thanks to the stick-breaking construction, even though an infinite number of mixture components can be used, only a finite number of them is instantiated during the inference phase. Hereinafter, we indicate with the symbol C the number of mixture components that are chosen by the HDP at inference time.

3.3. Model Definition

Architecturally speaking, ICGMM shares the same characteristics of CGMM described in Section 3.1, whereas the differences of each layer's graphical model are highlighted in Figure 1. In particular, ICGMM assumes that the generation of the node features x_u at each layer is governed by an HDP mixture model. Thus, following the stick-breaking construction detailed in Section 3.2, the generative process of a single ICGMM layer ℓ can be formalized as follows:

$$\begin{aligned}
&\boldsymbol{\beta}^\ell \mid \gamma^\ell \sim \text{Stick}(\gamma^\ell) && j_u^\ell \mid \mathbf{q}_{\mathcal{N}_u}^{\ell-1} = \psi(\mathbf{q}_{\mathcal{N}_u}^{\ell-1}) \\
&\boldsymbol{\pi}_j^\ell \mid \boldsymbol{\beta}^\ell, \alpha_0^\ell \sim \text{DP}(\alpha_0^\ell, \boldsymbol{\beta}^\ell) && q_u^\ell \mid j_u^\ell, (\boldsymbol{\pi}_j)_{j=1}^{C^{\ell-1}} \sim \boldsymbol{\pi}_{j_u}^\ell \\
&\boldsymbol{\theta}^\ell \mid \boldsymbol{H} \sim \boldsymbol{H} && x_u \mid q_u^\ell, (\boldsymbol{\theta}_c^\ell)_{c=1}^{\infty} \sim F(\boldsymbol{\theta}_{q_u^\ell}^\ell),
\end{aligned} \tag{2}$$

where we add the superscript ℓ to the HDP mixture model quantities to highlight that they are different at each ICGMM layer. Similarly to the HDP case, we use C^ℓ to denote the number of states chosen by the model at the current layer. When clear from the context, we will omit such a superscript to ease the notation.

As mentioned before, in any HDP mixture model each observation must belong to a group, and such group is typically known *in advance*. In this work, instead, each ICGMM layer can assign a group to each observation by exploiting the structural information in the graph. This way, we effec-

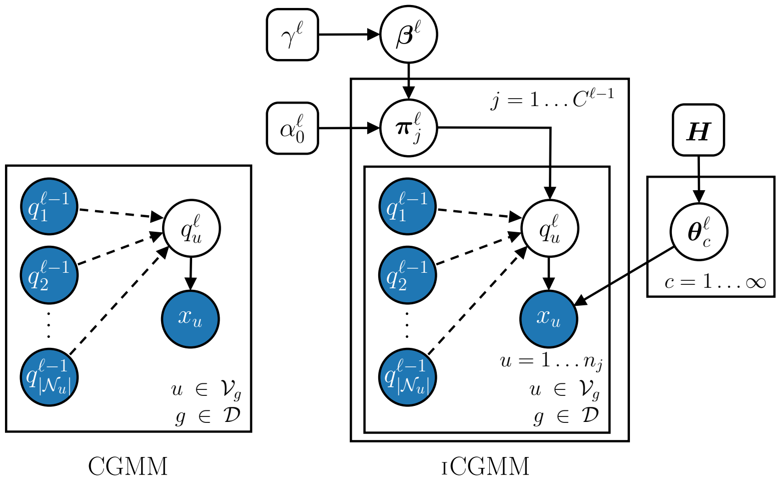


Figure 1. Differences between layer ℓ's graphical model of the original CGMM and the proposed iCGMM. Observable variables are blue circles, latent ones are empty circles, and white boxes denote prior knowledge. Each iCGMM is an HDP mixture model where the group j for each node observation x_u is pre-determined by the set of states of neighboring nodes $\mathbf{q}_{\mathcal{N}_u}^{\ell-1}$ computed at layer $\ell - 1$. Contrarily to CGMM, the number of values that the latent indicator variable q_u can assume is adjusted to fit the underlying data distribution. Dashed arrows denote the flow of contextual information from previous layers through the neighbors of each node u.

tively let nodes propagate contextual information between each other. In particular, we select the group j_u^ℓ of the feature node x_u based on the neighbors' observable posteriors $\mathbf{q}_{\mathcal{N}_u}^{\ell-1} = \{\boldsymbol{q}_v^{\ell-1} \in [0,1]^{C^{l-1}} \mid v \in \mathcal{N}_u\}$ approximated by the inference phase of the previous layer (see Section 3.4):

$$j_u^\ell = \psi(\mathbf{q}_{\mathcal{N}_u}^{\ell-1}) = \underset{j \in \{1,\dots,C^{\ell-1}\}}{\arg\max} \left(\frac{1}{|\mathcal{N}_u|} \sum_{v \in \mathcal{N}_u} \boldsymbol{q}_v^{\ell-1} \right)_j. \quad (3)$$

As we can see, j_u has been chosen as the most likely position in the $C^{\ell-1}$-sized macrostate obtained by averaging the neighbors' probabilities in $\mathbf{q}_{\mathcal{N}_u}^{\ell-1}$.

It follows that nodes with the same feature may have a different latent state c, due to the fact that they are assigned to different groups, i.e., different $\boldsymbol{\pi}_j$, on the basis of their neighborhood; this mimics the role of CGMM neighborhood aggregation but in an HDP mixture model. In the first layer, where no previous layer exists, we shall just assume that all observables belong to the same group.

Summing up, we depart from the basic CGMM layer of Bacciu et al. (2020a) in more than one way. First and foremost, we do not parametrize nor learn the CGMM transition distribution, which was responsible for the convex combination of neighboring states when computing the E-step of the EM algorithm. Instead, we rely on the most probable choice of the group j_u that is encoded by the neighbors' macrostate. Secondly, due to the sheer complexity of the Bayesian non-parametric treatment, we do not train the model via EM as done with CGMM; instead, we will exploit Gibbs sampling (Geman & Geman, 1984) to compute the quantities of interest. Finally, iCGMM retains one important architectural characteristic of CGMM, i.e., it prevents vanishing gradient effects and over-smoothing by default (Bacciu et al., 2020a), thus allowing us to construct deeper architectures that propagate contextual information.

3.4. Inference

The inference phase of every BNP method is meant to estimate the posterior of the model parameters. For each iCGMM layer ℓ, we wish to compute the quantities $\boldsymbol{q}_u^\ell, \boldsymbol{\beta}^\ell, \boldsymbol{\pi}_j^\ell, \boldsymbol{\theta}^\ell$. Thanks to the incremental construction of the iCGMM architecture, we can do so one layer at a time. Thus, since each iCGMM layer is an HDP mixture model, we can infer its parameters, following the Gibbs sampling schema of Teh et al. (2006). Crucially, the BNP approach also allows us to estimate the value of α_0^ℓ and γ^ℓ, with a further reduction of the hyper-parameters' search space. For the interested reader, we report the iCGMM complete Gibbs sampling equations and pseudo-code in Appendix A and B, respectively. Finally, it is worth mentioning that the constructive approach of CGMM and iCGMM is not an approximation of a more complex graphical model, rather it is a design choice that applies the basic principle of iterative computation underpinning all DGNs. At the same time, the BNP extension is mostly suitable for graphical models like CGMM, so, in general, it is not obvious how to extend neural networks to obtain the same benefits provided by iCGMM.

Graph Embedding Generation. In a similar vein with (Bacciu et al., 2020a), we prefer to use the sample distribution of q_u (Eq. (4)) at the last iteration, rather than the last sampled state, as an approximation of node u's posterior distribution. This way, we encode more information about state occupancy into node/graph embeddings.

As in Bacciu et al. (2020a), node embeddings of each layer are represented as *unibigrams*. A unibigram concatenates the posterior of a node, i.e., a vector called *unigram*, with its *bigram*. A bigram counts, for each possible state q_u, how many of u's neighbors are in another state, and it is represented as a vector of size C^2. The final graph representation is obtained by concatenation of node unibigrams across all layers followed by global aggregation. To tackle supervised tasks, we apply a "standard" predictor, e.g., a Multi-Layer Perceptron (MLP), on top of the unsupervised node/graph embeddings.

Faster Inference with Node Batches (ICGMM$_f$). Due to the sequential nature of the Gibbs sampling procedure, a naive implementation is slow when applied to the larger social graphs considered in this work. In the literature, there exist several exact distributed inference methods for HDP (Lovell et al., 2012; Williamson et al., 2013; Chang & Fisher III, 2014; Ge et al., 2015), but their effectiveness might be limited due to the unbalanced workload among workers or the elevated rejection rate (Gal & Ghahramani, 2014).

In this work, we prefer to speed up the inference procedure by introducing an approximation rather than relying on an exact distributed computation. As suggested in Gal & Ghahramani (2014), an approximated inference procedure may indeed suffice for many problems. What we propose is to perform sampling for a batch of node observations in parallel. This way, the necessary statistics are updated in batches rather than individually, and matrix operations can be used to gain efficiency. To maintain a good trade-off between the quality and speedup, we stick to 1 graph as the size of our batch. Such a trade-off provides a CPU speedup of up to $60\times$ at training time, and we empirically observed that performances remain unchanged with respect to the original version on the smaller chemical tasks considered. While this faster version of ICGMM, which we call ICGMM$_f$, does not strictly adhere to the technical specifications of the previous section, we believe that the pros largely outperform the cons. The interested reader can refer to Appendix D for an analysis of the speedup gains on the different datasets.

3.5. Limitations

Due to the complexity of the BNP treatment, one limitation of this work is that naive Gibbs sampling does not scale to very large datasets. The node independence assumption made by CGMM enables a faster batch computation, which can also be run on GPU. Despite having provided a simple, but approximated, sampling process that guarantees a substantial speedup and allows us to process graphs of non-negligible size, it would be interesting in the future to explore other inference methods to increase ICGMM's speedup, e.g., variational inference (Bryant & Sudderth, 2012; Wang & Blei, 2012; Hoffman et al., 2013; Hughes et al., 2015). The second limitation of ICGMM is that edge features are not taken into account. While there exist many neural models that do the same, we know that CGMM and its variant E-CGMM (Atzeni et al., 2021) can deal with discrete and arbitrary features, respectively. Our research directions for the future will investigate these aspects, providing an exact and efficient version of ICGMM that can process edge features as well.

4. Experiments

We evaluated the performances of ICGMM using the fair, robust, and reproducible evaluation setup for graph classification defined in Errica et al. (2020). It consists of an external 10-fold cross validation for model assessment, followed by an internal hold-out model selection for each of the external folds. Stratified data splits were already provided; in this respect, we had to re-assess CGMM and E-CGMM (Atzeni et al., 2021), a recently proposed variant, by trying all the hyper-parameters specified in the original papers (in particular, the values of C tried were 5, 10 and 20). We first experiment on the three chemical datasets D&D (Dobson & Doig, 2003), NCI1 (Wale et al., 2008) and PROTEINS (Borgwardt et al., 2005), where node features represent atom types. Then, we consider social datasets, including IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, REDDIT-MULTI-5K, and COLLAB (Yanardag & Vishwanathan, 2015), where the degree of each node is the sole continuous feature available. All datasets are publicly available (Kersting et al., 2016) and their statistics are summarized in Appendix C. Finally, we relied on Pytorch Geometric (Fey & Lenssen, 2019) for the implementation of our method.[1]

Apart from CGMM's variants, we will compare ICGMM against the following end-to-end supervised neural architectures for graphs: DGCNN (Zhang et al., 2018), DIFFPOOL (Ying et al., 2018), ECC (Simonovsky & Komodakis, 2017), GIN (Xu et al., 2019), GRAPHSAGE (Hamilton et al., 2017), and a structure-agnostic baseline method BASELINE, described in Errica et al. (2020), which was competitive on a number of benchmarks. We recall that these supervised methods construct the graph embeddings leveraging the supervision information coming from the target label; on the contrary, ICGMM embeddings are built in an unsupervised and constructive way. This represents a challenging

[1]The code to rigorously reproduce our results is provided here: `https://github.com/diningphil/iCGMM`.

comparison for our approach, as partially supervised methods generally struggle against fully supervised ones when trained on the same amount of supervised labels. In general, however, unsupervised methods can also: i) exploit large amounts of unlabeled data when there is a scarcity of ground-truth labels; ii) tackle different tasks on the same input data without retraining everything from scratch, by separating the representation learning phase from the prediction phase.. Results for the supervised models are taken from (Errica et al., 2020).

The formalism introduced so far provides a principled way to estimate the value of all hyper-parameters by introducing suitable hyper-priors. Moreover, it is known that information gain decreases as one considers higher levels of hyper-parameters (Bernardo & Smith, 2009; Goel & Degroot, 1981), so that the choice of the hyper-prior becomes less critical. That said, being this the first work to study HDP methods in the context of graph classification, we also explored the hyper-parameter space to best assess and characterize the behavior of the model. From now on, we refer to ICGMM$_{\alpha\gamma}$ as the model that relies on pre-defined values for α_0 and γ, whereas we place uninformative $Gamma(1, rate = 0.01)$ hyper-priors on both $\alpha_0^\ell, \gamma^\ell$ hyper-parameters for ICGMM.

For the chemical tasks, the prior $\boldsymbol{H}$ over the emission parameters $\boldsymbol{\theta}_c$ was the uniform Dirichlet distribution. The range of the remaining ICGMM hyper-parameters tried in this case were:

- Number of layers $\in \{5, 10, 15, 20\}$,
- Unibigram aggregation $\in \{\text{sum}, \text{mean}\}$,
- Gibbs sampling iterations $\in \{100\}$ for ICGMM and $\in \{10, 20, 50, 100\}$ for ICGMM$_{\alpha\gamma}$,
- $\alpha_0 \in \{1, 5\}$, $\gamma \in \{1, 2, 3\}$ (Only for ICGMM$_{\alpha\gamma}$)

Instead, for the social tasks we implemented a Normal-Gamma prior $\boldsymbol{H}$ over a Gaussian distribution. Here the prior is parametrized by the following hyper-priors: μ_0, the mean node degree extracted from the data; λ_0, which is inversely proportional to the prior variance of the mean; and (a_0, b_0), whose ratio $t = \frac{b_0}{a_0}$ represents the expected variance of the data. The ICGMM hyper-parameters here were:

- Number of layers $\in \{5, 10, 15, 20\}$,
- Unibigram aggregation $\{\text{sum}, \text{mean}\}$,
- $\lambda_0 \in \{\text{1e-6}\}$ ($\{\text{1e-4}, \text{1e-5}\}$ for COLLAB),
- Gibbs Sampling iterations $\in \{100\}$,
- $a_0 \in \{1.\}$, $b_0 \in \{0.09, 1.\}$,
- $\alpha_0 \in \{1, 5, 10\}$, $\gamma \in \{2, 5, 10\}$ (Only for ICGMM$_{\alpha\gamma}$)

To conclude, we list the hyper-parameters tried for the one-layer MLP classifier trained on the unsupervised graph embeddings:

Table 1. Results on chemical datasets (mean accuracy and standard deviation) are shown. Best performances are highlighted in bold.

	D&D	**NCI1**	**PROTEINS**
BASELINE	$\mathbf{78.4} \pm 4.5$	69.8 ± 2.2	$\mathbf{75.8} \pm 3.7$
DGCNN	76.6 ± 4.3	76.4 ± 1.7	72.9 ± 3.5
DIFFPOOL	75.0 ± 3.5	76.9 ± 1.9	73.7 ± 3.5
ECC	72.6 ± 4.1	76.2 ± 1.4	72.3 ± 3.4
GIN	75.3 ± 2.9	$\mathbf{80.0} \pm 1.4$	73.3 ± 4.0
GRAPHSAGE	72.9 ± 2.0	76.0 ± 1.8	73.0 ± 4.5
CGMM	74.9 ± 3.4	76.2 ± 2.0	74.0 ± 3.9
E-CGMM	73.9 ± 4.1	78.5 ± 1.7	73.3 ± 4.1
ICGMM$_{\alpha\gamma}$	75.6 ± 4.3	76.5 ± 1.8	72.7 ± 3.4
ICGMM$_{f_{\alpha\gamma}}$	75.0 ± 5.6	76.7 ± 1.7	73.3 ± 2.9
ICGMM	76.3 ± 5.6	77.6 ± 1.5	73.1 ± 3.9
ICGMM$_f$	75.1 ± 3.8	76.4 ± 1.4	73.2 ± 3.9

- Adam optimizer with batch size 32 and learning rate 1e-3,
- Hidden units $\in \{32, 128\}$,
- L2 regularization $\in \{0., \text{5e-4}\}$,
- epochs $\in \{2000\}$,
- early stopping on validation accuracy, with patience 300 on chemical tasks and 100 on social ones.

Notably, we also experimented with 2 and 3 layer MLPs, but found no advantage whatsoever in terms of validation performance.

5. Results

The empirical results on chemical and social benchmarks are reported in Tables 1 and 2, respectively. There are several observations to be made, starting with the chemical tasks. First of all, ICGMM performs similarly to CGMM, E-CGMM, and most of the *supervised* neural models; this suggests that the selection of j_u based on the neighboring recommendations is a subtle but effective form of information propagation between the nodes of the graph. In addition, results indicate that we have succeeded in effectively choosing the number of latent states without compromising the overall accuracy, which was the main goal of this work. Finally, ICGMM$_f$ performs as well as the exact version, and for this reason we safely applied the faster variant to the larger social datasets (including IMDB-B and IMDB-M to ease the exposition).

Moving to the social datasets, we observe that ICGMM achieves better average performances than other methods on IMDB-B, REDDIT-B and COLLAB. One possible reason for such an improvement with respect to CGMM variants may be how the emission distributions are initialized. On the one hand, and differently from the chemical tasks, CGMM and E-CGMM use the k-means algorithm (with fixed k=C), to initialize the mean values of the C Gaussian distributions,

Table 2. Results on social datasets (mean accuracy and standard deviation) are shown, where the node degree is used as the only node feature. The best performances are highlighted in bold.

	IMDB-B	IMDB-M	REDDIT-B	REDDIT-5K	COLLAB
BASELINE	70.8 ± 5.0	$\mathbf{49.1} \pm 3.5$	82.2 ± 3.0	52.2 ± 1.5	70.2 ± 1.5
DGCNN	69.2 ± 3.0	45.6 ± 3.4	87.8 ± 2.5	49.2 ± 1.2	71.2 ± 1.9
DIFFPOOL	68.4 ± 3.3	45.6 ± 3.4	89.1 ± 1.6	53.8 ± 1.4	68.9 ± 2.0
ECC	67.7 ± 2.8	43.5 ± 3.1	-	-	-
GIN	71.2 ± 3.9	48.5 ± 3.3	89.9 ± 1.9	$\mathbf{56.1} \pm 1.7$	75.6 ± 2.3
GRAPHSAGE	68.8 ± 4.5	47.6 ± 3.5	84.3 ± 1.9	50.0 ± 1.3	73.9 ± 1.7
CGMM	72.7 ± 3.6	47.5 ± 3.9	88.1 ± 1.9	52.4 ± 2.2	77.32 ± 2.2
E-CGMM	70.7 ± 3.8	48.3 ± 4.1	89.5 ± 1.3	53.7 ± 1.0	77.45 ± 2.3
ICGMM$_{f_{\alpha\gamma}}$	$\mathbf{73.0} \pm 4.3$	48.6 ± 3.4	91.3 ± 1.8	55.5 ± 1.9	78.6 ± 2.8
ICGMM$_f$	71.8 ± 4.4	49.0 ± 3.8	$\mathbf{91.6} \pm 2.1$	55.6 ± 1.7	$\mathbf{78.9} \pm 1.7$

which can be stuck in a local minimum around the most frequent degree values. On the other hand, ICGMM adopts a fully Bayesian treatment, which combined with the selection of the latent states allows for better modeling of outliers by adding a new state when the posterior probability of a data point is too low.

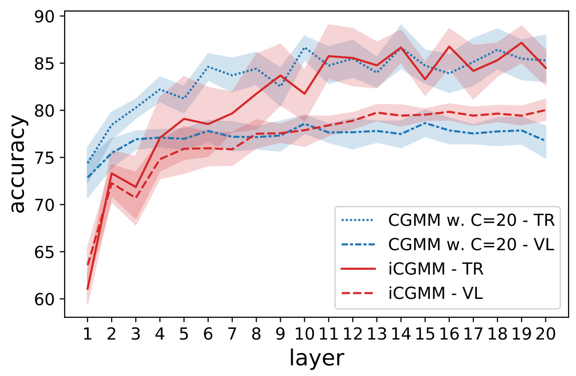
By estimating all hyper-parameters of our models using uninformative priors, we almost always (but for COLLAB) managed to *avoid the model selection* for the unsupervised graph embeddings creation. In turn, this amounts to a $6\times$ reduction in the number of configurations tried, but most importantly it frees the user from making hard choices about which values of hyper-parameters to use.

In what follows, we will try to shed more light on the improved generalization performances of ICGMM, by analyzing the exact model from a layer-wise perspective.

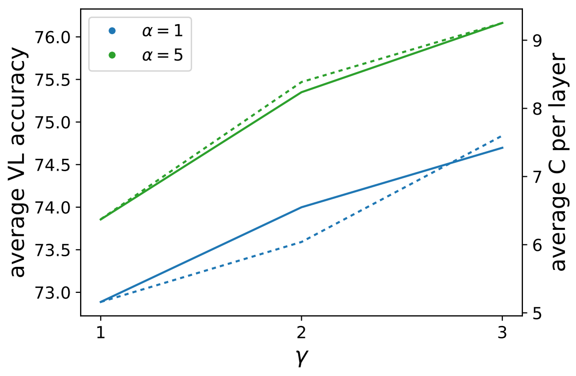
On the effectiveness of depth and hyper-parameters. To confirm our intuition about the benefits of the proposed information propagation mechanism, Figure 2(a) shows the NCI1 training and validation performances of both CGMM and ICGMM as we add more layers. For simplicity, we picked the best ICGMM configuration on the first external fold, and we compared it against the CGMM configuration with the most similar performances. Note that $C = 20$ was the most frequent choice of CGMM states by the best model configurations across the 10 outer folds: this is because having more emission distributions to choose from allows the CGMM model to find better local minima, whereas ICGMM can add new states whenever the data point's sampling probabilities are too low. We trained the same classifier at different depths, and we averaged scores across the 10 outer folds. We observe that the validation performance of both models is similar but in favor of ICGMM, with an asymptotic behavior as we reach 20 layers; it follows that depth remains fundamental to improve the generalization performances (Bacciu et al., 2020a).

We now study how ICGMM$_{\alpha\gamma}$ behaves as we vary the main hyper-parameters α_0 and γ. We continue our experimentation on NCI1; Figure 2(b) depicts the average validation performance and number of states C over all configurations and folds, subject to changes of α_0 and γ values. The trend indicates how greater values for both hyper-parameters achieve, on average, better validation performance. Also, smaller values of the two hyper-parameters tend to strongly regularize the model by creating fewer states, with a consequent reduction in validation accuracy. The relation between the number of states and these hyper-parameters remains consistent with the mathematical details of Section 3.

On the quality of graph embeddings. So far, we have argued that ICGMM selects the appropriate number of states for its unsupervised task at each layer (for more details, see Appendix F). As a matter of fact, Figure 3(a) reports such a statistic on the same NCI1 configuration as before: ICGMM preferred a lower number of latent states than CGMM, i.e., around 5 per layer. In turn, the resulting graph embeddings become much smaller, with important savings in terms of memory footprint and computational costs to train the subsequent classifier. Figure 3(b) displays the cumulative graph embedding size across layers, using the unibigram representation without loss of generality. We see that, when compared with CGMM (C=20), the size of graph embeddings produced by ICGMM is approximately 7% of those of the original model, while still preserving the same performance as CGMM. Additionally, we observe that the number of chosen states and the consequent graph embedding size is very similar to that of ICGMM$_{\alpha\gamma}$ with $\alpha_0 = 5, \gamma = 3$, despite the fact that these two hyper-parameters have been adjusted by ICGMM on the basis of the data. Last but not least, we asked how many of the states used by the model are actually populated. Figure 4 answers this question: when compared with the quasi-linear trend of CGMM, where training leads to solutions with a consistent number of unused states, ICGMM creates near-zero

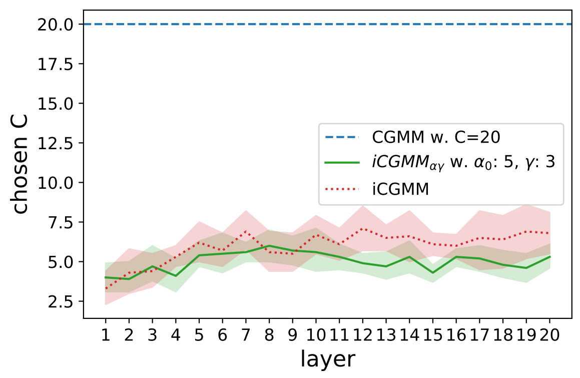


(a) Effect of depth on training/validation accuracy

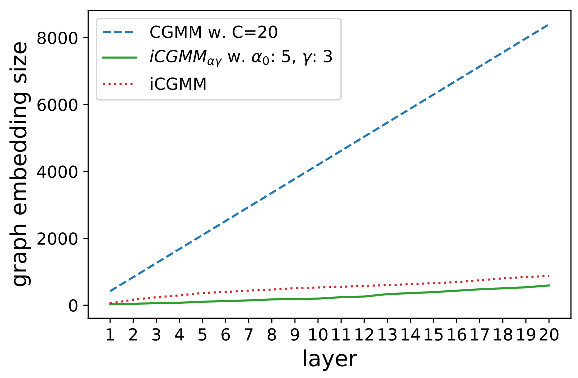


(b) Average VL accuracy (solid line) and number of chosen states (dashed line) w.r.t α_0 and γ values

Figure 2. Figures 2(a) and 2(b) analyze the relation between depth, performances, and the number of chosen states on NCI1.



(a) # states chosen at each layer



(b) Cumulative graph embedding size on NCI1

Figure 3. We show comparative results on the size and quality of graph embeddings between CGMM and ICGMM. Overall, ICGMM generates fewer latent states, with consequent savings in terms of memory and compute time of the classifier with respect to CGMM. See the text for more details.

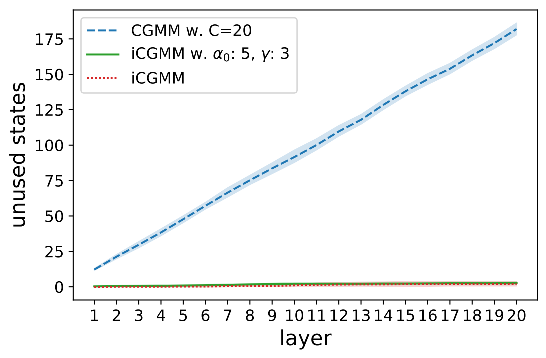


Figure 4. Cumulative unused states on NCI1. Overall, ICGMM generates ≈ 0 unused latent states; in contrast, CGMM's unused states grow linearly with respect to the number of layers.

unpopulated states[2].

To sum up, we have shown that: i) the information propagation mechanism introduced in the HDP is effective; ii) the model can successfully infer its number of latent states; iii) we can estimate the choice of ICGMM hyper-parameters using uninformative hyper-priors, which further reduces the cost of the model selection phase; iv) we can get a much lower memory and computational footprints due to the previous points without sacrificing the predictive performance (see Appendix E for a time-to-result comparison); v) our model has very competitive performances with respect to the state of the art.

[2]For simplicity, our code does not remove the possibly unpopulated state in the last Gibbs sampling iteration.

6. Conclusions

With the Infinite Contextual Graph Markov Model, we have bridged the gap between Bayesian nonparametric techniques and machine learning for graphs. We have described how our approach can infer the number of states and most hyperparameters at each unsupervised layer, thus reducing the number of configurations to try during model selection. As the empirical analyses show, not only can the model exploit depth to increase its generalization performances, but it also produces smaller embeddings than CGMM, with consequent savings in terms of memory footprint and training time of the subsequent classifier. For these reasons, we believe that ICGMM represents the first relevant step toward the theoretically grounded construction of fully probabilistic deep learning models for graphs.

References

Aldous, D. J. Exchangeability and related topics. In *École d'Été de Probabilités de Saint-Flour XIII—1983*, pp. 1–198. Springer, 1985.

Atzeni, D., Bacciu, D., Errica, F., and Micheli, A. Modeling edge features with deep bayesian graph networks. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

Bacciu, D., Errica, F., and Micheli, A. Contextual Graph Markov Model: A deep and generative approach to graph processing. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pp. 294–303. PMLR, 2018.

Bacciu, D., Errica, F., and Micheli, A. Probabilistic learning on graphs via contextual architectures. *Journal of Machine Learning Research*, 21(134):1–39, 2020a.

Bacciu, D., Errica, F., Micheli, A., and Podda, M. A gentle introduction to deep learning for graphs. *Neural Networks*, 129:203–221, 9 2020b.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., and others. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Beal, M. J., Ghahramani, Z., and Rasmussen, C. E. The infinite hidden markov model. *Proceedings of the 16th Conference on Neural Information Processing Systems (NIPS)*, 1:577–584, 2002.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2):281–305, 2012.

Bernardo, J. M. and Smith, A. F. *Bayesian theory*, volume 405. John Wiley & Sons, 2009.

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., and Kriegel, H.-P. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34 (4):25. 18–42, 2017.

Bryant, M. and Sudderth, E. Truly nonparametric online variational inference for hierarchical dirichlet processes. In *Proceedings of the 26th Conference on Neural Information Processing Systems (NIPS)*, 2012.

Chang, J. and Fisher III, J. W. Parallel sampling of hdps using sub-cluster splits. In *Proceedings of the 28th Conference on Neural Information Processing Systems (NIPS)*, 2014.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS)*, pp. 3844–3852, 2016.

Dobson, P. D. and Doig, A. J. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

Eliasof, M., Haber, E., and Treister, E. PDE-GCN: Novel architectures for graph neural networks motivated by partial differential equations. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*, pp. 3836–3849, 2021.

Errica, F., Podda, M., Bacciu, D., and Micheli, A. A fair comparison of graph neural networks for graph classification. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020.

Fahlman, S. E. and Lebiere, C. The Cascade-Correlation learning architecture. In *Proceedings of the 3rd Conference on Neural Information Processing Systems (NIPS)*, pp. 524–532, 1990.

Fey, M. and Lenssen, J. E. Fast graph representation learning with pytorch geometric. In *Representation Learning on Graphs and Manifolds Workshop, International Conference on Learning Representations (ICLR)*, 2019.

Fox, E. B., Sudderth, E. B., Jordan, M. I., and Willsky, A. S. The sticky hdp-hmm: Bayesian nonparametric hidden markov models with persistent states. *Preprint*, 2007.

Fox, E. B., Sudderth, E. B., Jordan, M. I., and Willsky, A. S. An hdp-hmm for systems with state persistence. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pp. 312–319, 2008.

Gal, Y. and Ghahramani, Z. Pitfalls in the use of parallel inference for the dirichlet process. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, volume 32, pp. 208–216. PMLR, 22–24 Jun 2014.

Ge, H., Chen, Y., Wan, M., and Ghahramani, Z. Distributed inference for dirichlet process mixture models. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 2276–2284. PMLR, 2015.

Geman, S. and Geman, D. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, PAMI-6(6):721–741, 1984.

Gershman, S. J. and Blei, D. M. A tutorial on bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1–12, 2012.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272. PMLR, 2017.

Goel, P. K. and Degroot, M. H. Information about hyperparamters in hierarchical models. *Journal of the American Statistical Association*, 76(373):140–147, 1981. ISSN 01621459. URL `http://www.jstor.org/stable/2287059`.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, pp. 1024–1034. Curran Associates, Inc., 2017.

He, X., Zhao, K., and Chu, X. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *Journal of Machine Learning Research*, 14(5), 2013.

Hoppe, F. M. Pólya-like urns and the ewens' sampling formula. *Journal of Mathematical Biology*, 20(1):91–94, 1984.

Hughes, M., Kim, D. I., and Sudderth, E. Reliable and scalable variational inference for the hierarchical dirichlet process. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.

Kersting, K., Kriege, N. M., Morris, C., Mutzel, P., and Neumann, M. Benchmark data sets for graph kernels, 2016. URL `http://graphkernels.cs.tu-dortmund.de`.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations (ICLR)*, 2017.

Lovell, D., Adams, R. P., and Mansinghka, V. K. Parallel markov chain monte carlo for dirichlet process mixtures. In *Workshop on Big Learning, Advances in Neural Information Processing Systems (NIPS)*, 2012.

Micheli, A. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009. Publisher: IEEE.

Moon, T. K. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, 1996. Publisher: IEEE.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL `www.graphlearning.io`.

Neal, R. M. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.

Orbanz, P. and Teh, Y. W. Bayesian nonparametric models. *Encyclopedia of machine learning*, 1, 2010.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. Publisher: IEEE.

Sethuraman, J. A constructive definition of dirichlet priors. *Statistica sinica*, pp. 639–650, 1994.

Simonovsky, M. and Komodakis, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29–38, 2017.

Teh, Y. W. *Dirichlet Process*, pp. 280–287. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. Hierarchical dirichlet processes. *Journal of the american statistical association*, 101(476):1566–1581, 2006.

Wale, N., Watson, I. A., and Karypis, G. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14 (3):347–375, 2008.

Wang, C. and Blei, D. Truncation-free stochastic variational inference for bayesian nonparametric models. In *Proceedings of the 26th Conference on Neural Information Processing Systems (NIPS)*, 2012.

Williamson, S., Dubey, A., and Xing, E. Parallel Markov chain Monte Carlo for nonparametric mixture models. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, volume 28, pp. 98–106. PMLR, 2013.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Yanardag, P. and Vishwanathan, S. V. N. Deep graph kernels. In *Proceedings of the 21th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 1365–1374, 2015.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS)*, pp. 4800–4810. Curran Associates, Inc., 2018.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 4438–4445, 2018.

Zhou, K., Huang, X., Zha, D., Chen, R., Li, L., Choi, S.-H., and Hu, X. Dirichlet energy constrained learning for deep graph neural networks. In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*, pp. 21834–21846, 2021.

A. ICGMM Gibbs Sampling Procedure

HDP Gibbs sampling is an iterative procedure (Neal, 2000; Teh et al., 2006; Fox et al., 2007) that we use to estimate all node latent states and ICGMM's parameters at each layer. Hereinafter, to keep the notation less cluttered, we shall omit the superscript ℓ of the current layer and define $\bar{C}$=$C^{\ell-1}$.

Sampling q_u. The conditional distribution $\boldsymbol{q}_u$ of q_u given all the other variables is given by:

$$P(q_u = c \mid j_u = j, \boldsymbol{q}^{-u}, \boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{x}) \propto (\alpha_0\beta_c + n_{jc}^{-u}) f(x_u \mid \theta_c), \quad c \in \{1, \dots, C+1\}; \tag{4}$$

where we recall that C denotes the number of current states in the mixture model, f is the *p.d.f.* associated with emission distribution $F(\theta)$ and the distribution $\boldsymbol{\pi}_j$ has been integrated out (Teh et al., 2006). Here, n_{jc}^{-u} indicates the number of observations assigned so far to the mixture component c of group j. Whenever we have that $q_u = C+1$, we create a new state and sample a new emission distribution $\boldsymbol{\theta}_{C+1}$ from $\boldsymbol{H}$. On the contrary, if at the end of an iteration there are no observation of any group associated with a certain mixture component, we can remove that mixture component and decrement the current number of states C. This is how ICGMM varies in complexity to fit the data distribution. Also, note that $\boldsymbol{q}_u^\ell$ will be used in Eq. 3 at the next layer $\ell+1$. When inferring the latent states of a new data point, no statistics of the model are updated.

Sampling $\boldsymbol{\beta}$. In the HDP stick-breaking representation that we use to define the ICGMM in Section 3.3, we require an auxiliary variable method to sample the base distribution $\boldsymbol{\beta}$ (Teh et al., 2006). We therefore introduce the auxiliary variables $\boldsymbol{m} = \{m_{jc} \mid \forall j \in \{1, \dots, \bar{C}\}, \forall c \in \{1, \dots, C\}\}$ that need to be sampled in order to compute $\boldsymbol{\beta}$. However, being m_{jc} dependent on n_{jc}, the sampling step of these variables is very inefficient for large values of n_{jc}, as the probability values are proportional the Stirling number of the first-kind $s(n_{jc}, \cdot)$ (Fox et al., 2008). Luckily, we can avoid this step by observing that the value m_{jc} corresponds to the number of tables where dish $q_u = c$ is served at restaurant j in the Chines Restaurant Franchise (CRF) representation (Teh et al., 2006; Fox et al., 2007). Thus, we can compute each m_{jc} by simply simulating the table assignments process. We recall that, in the CRF representation, each customer (i.e., observation) of each restaurant (i.e., group) is assigned to a table where just a single dish (i.e., mixture component) is served. Thus, while all customers sitting at the same table must be eating the same dish, there can be multiple tables serving the same dish as well.

Knowing that customer u is eating the dish $q_u = c$, its table assignment t_u can be sampled according to:

$$P(t_u = t \mid q_u = c, j_u = j, \boldsymbol{c}, \boldsymbol{t}^{-u}, \boldsymbol{\beta}, \alpha_0) \propto \begin{cases} \tilde{n}_{jt}^{-u}, & \forall t \text{ s.t. } c_{jt} = c; \\ \alpha_0\beta_c, & t = t_{new}, \end{cases} \tag{5}$$

where $\boldsymbol{t}^{-u}$ represents the tables assigned to all the other nodes except u, $c_{jt} \in \boldsymbol{c}$ specifies the dish assigned to table t at restaurant j and $\tilde{n}_{jt}^{-u}$ denotes the number of customers (except u) sitting at table t of restaurant j. Since we know the dish q_u selected by the customer u, there is zero probability that the customer sits to a table where that dish is not served. The creation and deletion of tables is very similar to that of Eq. 4, so we skip it in the interest of the exposition and refer to the pseudocode in Appendix B for a complete treatment.

At last, after computing m_{jc} as described above (i.e., $\sum_{t'} \mathbb{I}[c_{jt'} = c]$), the base distribution $\boldsymbol{\beta}$ is sampled from:

$$\boldsymbol{\beta} \mid \boldsymbol{q}, \boldsymbol{m} \sim \text{Dir}(\sum_{j=1}^{\bar{C}} m_{j1}, \dots, \sum_{j=1}^{\bar{C}} m_{jC}, \gamma), \tag{6}$$

where Dir stands for the Dirichlet distribution.

Sampling $\boldsymbol{\theta}$. To update the emission parameters $\boldsymbol{\theta}$, we rely on its posterior given $\boldsymbol{q}$ and $\boldsymbol{x}$:

$$P(\boldsymbol{\theta}_c \mid \boldsymbol{q}, \boldsymbol{x}) \propto h(\boldsymbol{\theta}_c) \prod_{\forall u \mid q_u = c} f(x_u \mid \boldsymbol{\theta}_c). \tag{7}$$

By choosing the family of the base distribution $\boldsymbol{H}$ to be a conjugate prior for F, e.g., a Dirichlet distribution for Categorical emissions or a Normal-Gamma distribution for Normal emissions, we can compute the posterior in closed form.

Let the emission distribution be a categorical distribution with M possible states. When creating a new state, we can sample the emission parameter according to a Dirichlet distribution, which is a conjugate prior for the categorical distribution:

$$\theta_c \sim \text{Dir}(\eta, \dots, \eta), \tag{8}$$

where the subscript c indicates the mixture component. Thanks to the conjugate prior, the emission parameters can be updated by sampling its Dirichlet posterior distribution:

$$\theta'_c \sim \text{Dir}(\eta + N_c^1, \dots, \eta + N_c^M), \tag{9}$$

where N_c^m indicates the number of times the visible label m has been associated with the latent state c, i.e., $N_m^c = \sum_u \mathbb{I}[q_u = c \wedge x_u = m]$.

Similarly to the categorical case, let the emission distribution be an univariate Gaussian. In this case, for each state, we can sample the emission parameter according to a Normal-Gamma distribution:

$$\mu_c \sim \mathcal{N}(\mu_0, 1/(\lambda_0 \tau_c)) \tag{10}$$

$$\tau_c \sim \text{Gamma}(a_0, b_0), \tag{11}$$

where the subscript c indicates a mixture component ant τ_c is the inverse of the variance. Then, the emission parameters of the Gaussian can be updated as follows:

$$\mu'_c \sim \mathcal{N}\left(\frac{\lambda_0\mu_0 + N_c\bar{x}_c}{\lambda_0 + N_c}, \frac{1}{(\lambda_0 + N_c)\tau'_c}\right) \tag{12}$$

$$\tau'_c \sim \text{Gamma}\left(a_0 + \frac{N_c}{2}, b_0 + \frac{1}{2}\left(N_c s_c + \frac{\lambda_0 N_c(\bar{x}_c - \mu_0)^2}{\lambda_0 + N_c}\right)\right), \tag{13}$$

where N_c indicates the number of visible labels associated with the latent state c (i.e., $N_c = \sum_u \mathbb{I}[q_u = c]$), $\bar{x}_c$ is the mean of the data associated with the class c (i.e., $\bar{x}_c = \frac{1}{N_c}\sum_{\forall u | q_u = c} x_u$), and s_c is the variance of the data associated with the class c (i.e., $s_c = \frac{1}{N_c}\sum_{\forall u|q_u=c}(x_u - \bar{x}_u)^2$).

Sampling α_0. Following (Teh et al., 2006), the concentration parameter α_0 can be updated between Gibbs sampling iterations by exploiting an auxiliary variable schema. Assume that α_0 has a Gamma prior distribution Gamma(a, b) (i.e., $\alpha_0 \sim$ Gamma(a, b)). Then, we define the auxiliary variables $w_1, \dots, w_{\bar{C}}$ and $s_1, \dots, s_{\bar{C}}$, where each w_j variable takes a value between 0 and 1, and each s_j is a binary variable. Then, the value of α_0 can be sampled according to the following schema:

$$w_j \sim \text{Beta}(\alpha_0 + 1, n_{j.}), \tag{14}$$

$$s_j \sim \text{Bernoulli}\left(\frac{n_{j.}}{n_{j.} + \alpha_0}\right), \tag{15}$$

$$\alpha_0 \sim \text{Gamma}\left(a + m_{..} - \sum_{j=1}^{\bar{C}} s_j, b - \sum_{j=1}^{\bar{C}} \log w_j\right), \tag{16}$$

where $n_{j.}$ is the number of costumer eating in the j-th restaurant, and $m_{..}$ is the total number of tables in all the restaurants.

Sampling γ. Similarly, assuming that the hyperparameter γ has a gamma prior distribution Gamma(a', b') (i.e., $\gamma \sim$ Gamma(a', b')), its value can be updated by following the auxiliary variable schema below (Teh et al., 2006; Fox et al., 2008):

$$r \sim \text{Beta}(\gamma + 1, m_{..}), \tag{17}$$

$$p \sim \text{Bernoulli}\left(\frac{m_{..}}{m_{..} + \gamma}\right), \tag{18}$$

$$\gamma \sim \text{Gamma}(a' + C - p, b' - \log r). \tag{19}$$

B. ICGMM Pseudocode

To ease the understanding of our model, we provide the pseudocode of the Gibbs sampling method employed in this work.

Algorithm 1 Gibbs sampling method for exact ICGMM

```
Require: A dataset of graphs $\mathcal{D} = \{g_1, \dots, g_N\}$. Initialize $C = 1$, $\boldsymbol{\theta} = \{\theta_1\}$ (where $\theta_1 \sim H$), $\mathcal{T}_j = \emptyset$ (for all restaurant $j$),
    $\boldsymbol{q} = \boldsymbol{t} = \boldsymbol{c} = \perp$, and $\boldsymbol{n} = \tilde{\boldsymbol{n}} = \mathbf{0}$.
    repeat
        for $g \in \mathcal{D}$ do                                        ▷ For each graph
            for $u \in \mathcal{V}_g$ do                                   ▷ For each node
                // assign the restaurant
                $j_u \leftarrow \psi(\mathbf{q}^{\ell-1}_{\mathcal{N}_u})$                ▷ Can be done once $\forall u$
                // assign the dish
                $n_{j_u q_u} \leftarrow n_{j_u q_u} - 1$                   ▷ If $q_u \neq \perp$, remove $q_u$ from the counting
                $q_u \leftarrow$ SAMPLING$(j_u, \boldsymbol{n}, \boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{\beta}, \alpha_0)$     ▷ Sample the dish according to Eq. (4)
                if $q_u$ is new then                                       ▷ Create a new state
                    $\theta_{\text{new}} \sim H$
                    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} \cup \{\theta_{\text{new}}\}$
                    $C \leftarrow C + 1$
                    $n_{j q_u} \leftarrow 0 \quad \forall j \in \{1, \dots, \bar{C}\}$          ▷ Initialize the counters
                end if
                $n_{j_u q_u} \leftarrow n_{j_u q_u} + 1$                   ▷ Update the counter
                // assign the table
                $\tilde{n}_{j_u t_u} \leftarrow \tilde{n}_{j_u t_u} - 1$   ▷ If $t_u \neq \perp$, remove $t_u$ from the counting
                $t_u \leftarrow$ SAMPLING$(j_u, q_u, \boldsymbol{c}, \tilde{\boldsymbol{n}}, \boldsymbol{\beta}, \alpha_0)$    ▷ Sample the table according to Eq. (5)
                if $t_u$ is new then                                       ▷ Create a new table
                    $\mathcal{T}_j \leftarrow \mathcal{T}_j \cup \{t_u\}$
                    $c_{j_u t_u} \leftarrow q_u$                           ▷ Save the dish-table assignment
                    $m_{j_u q_u} \leftarrow m_{j_u q_u} + 1$               ▷ Update the table count
                    $\tilde{n}_{j_u t_u} \leftarrow 0$                     ▷ Initialize customer counter
                end if
                $\tilde{n}_{j_u t_u} \leftarrow \tilde{n}_{j_u t_u} + 1$
            end for
        end for
        // remove unused dishes
        for $c \in \{1, \dots, C\}$ do
            if $\sum_{j=1}^{\bar{C}} n_{jc} = 0$ then                      ▷ No customers eats the dish $c$
                $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} \setminus \{\theta_c\}$
                $C \leftarrow C - 1$
            end if
        end for
        // remove empty tables
        for $j \in \{1, \dots, \bar{C}\}$ do
            for $t \in \mathcal{T}_j$ do
                if $\tilde{n}_{jt} = 0$ then                               ▷ No customers eat at the table $t$ in the restaurant $j$
                    $\mathcal{T}_j \leftarrow \mathcal{T}_j \setminus \{t\}$
                    $m_{jc_{jt}} \leftarrow m_{jc_{jt}} - 1$
                end if
            end for
        end for
        // update model parameters
        $\boldsymbol{\beta} \leftarrow$ SAMPLING$(\boldsymbol{q}, \boldsymbol{m})$     ▷ Sample according to Eq. (6)
        $\boldsymbol{\theta} \leftarrow$ SAMPLING$(\boldsymbol{q}, \boldsymbol{x})$    ▷ Sample according to Eq. (7)
        if not ICGMM$_{\alpha\gamma}$ then
            $\alpha_0 \leftarrow$ SAMPLING$(a, b, \boldsymbol{n})$         ▷ Sample according to Eq. (14), (15), (16)
            $\gamma \leftarrow$ SAMPLING$(a', b', \boldsymbol{m})$         ▷ Sample according to Eq. (17), (18), (19)
        end if
    until stopping criteria
```

C. Dataset Statistics

Below we report some statistics for the chosen benchmarks.

Table 3. Dataset statistics.

		# Graphs	# Classes	# Nodes	# Edges	# Node labels
CHEM.	DD	1178	2	284.32	715.66	89
	NCI1	4110	2	29.87	32.30	37
	PROTEINS	1113	2	39.06	72.82	3
SOCIAL	IMDB-BINARY	1000	2	19.77	96.53	-
	IMDB-MULTI	1500	3	13.00	65.94	-
	REDDIT-BINARY	2000	2	429.63	497.75	-
	REDDIT-5K	4999	5	508.82	594.87	-
	COLLAB	5000	3	74.49	2457.78	-

D. Speedup gains with faster inference

We compare the performances of the exact version of ICGMM against the faster implementation. As we can see, the speedup increases for the datasets with a larger average number of nodes (see Table 3).

Table 4. Approximate speedup between the exact ICGMM and the faster version on all datasets.

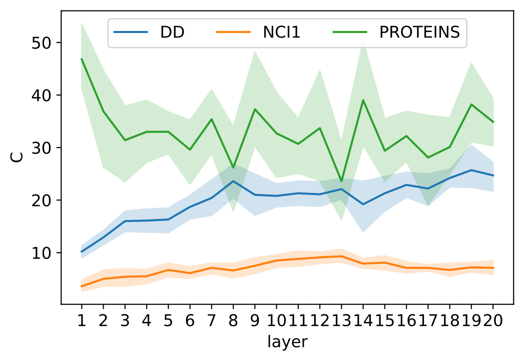
		ICGMM	ICGMM$_f$
		ref.	**min/max**
CHEM.	DD	1×	17.8×/30.8×
	NCI1	1×	3.1×/5.1×
	PROTEINS	1×	4.2×/5.7×
SOCIAL	IMDB-B	1×	2.4×/5.1×
	IMDB-M	1×	1.6×/3.6×
	REDDIT-B	1×	11.1×/45.6×
	REDDIT-5K	1×	36.7×/60.6×
	COLLAB	1×	3.1×/8.6×

E. Time-to-result Comparison

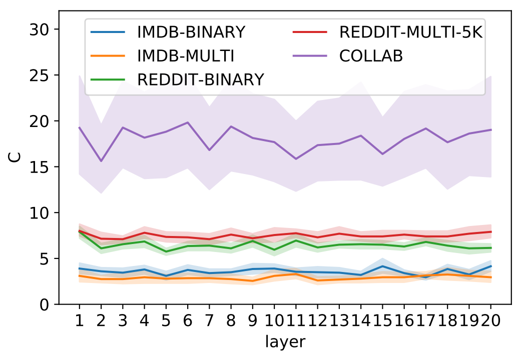
A fair time comparison between all models requires to look at the "time to result" using the same resources - in our case CPUs - which would imply a complete re-evaluation of all models in Errica et al. (2020). Due to the sheer amount of experiments needed to do so, we did our best to compare the sequential time to result of the fastest network GraphSAGE (72 configurations), against ICGMM on NCI1 and COLLAB. The results are shown below: while ICGMM is slower than GraphSAGE when comparing single runs, we are able to save a considerable amount of time and energy with ICGMM due to the ability to estimate most hyper-parameter's values.

Dataset	iCGMM Total (Embed + Classify)	GraphSAGE	Relative CPU Speedup (time to completion)
NCI1	58h+44h = 102h	2010h	19.7x
COLLAB	334h + 621h = 955h	35280h	36.9x

Table 5. Time-to-result to train ICGMM and GraphSAGE on NCI1 and COLLAB.

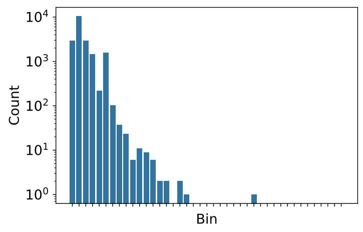


(a) Chemical datasets.

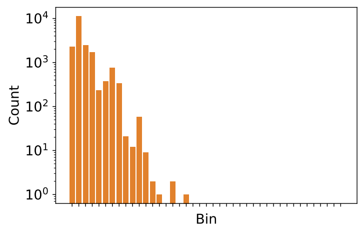


(b) Social datasets.

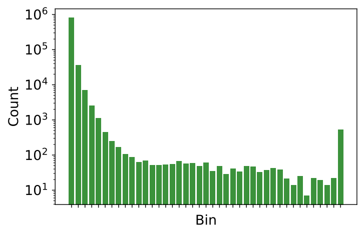
Figure 5. Per-layer analysis of the average number of states generated by ICGMM.



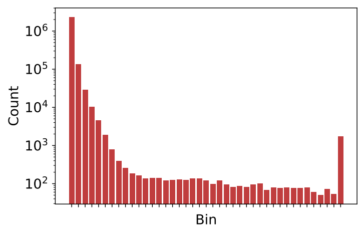
(a) IMDB-BINARY.



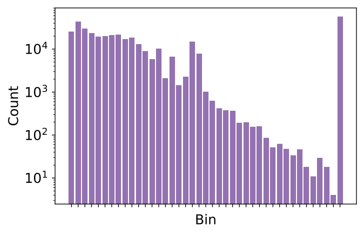
(b) IMDB-MULTI.



(c) REDDIT-BINARY



(d) REDDIT-MULTI-5K.



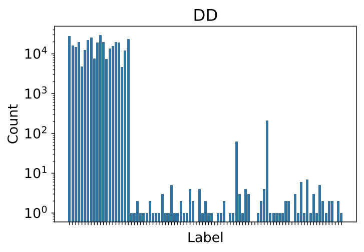
(e) COLLAB.

Figure 6. The node labels distribution of each social dataset considered. For the sake of clarity, we cluster the labels into 41 bins; in the plots, each bar indicates the number of node labels in a bin. The first 40 bins (from left to right) have a width of 5; thus, they cover all the labels between 0 and 200. The last bin is used to count all the nodes with a label that is greater than 200.

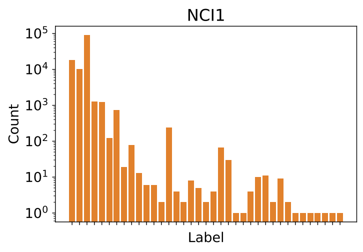
F. State Analysis

We additionally performed an analysis of the average number of states generated by each layer of ICGMM on the chemical and social datasets. Results are shown in Figure 5. At each layer, ICGMM is trained to solve a node density estimation problem. Thus, we argue that the number of states selected by the model in each dataset is at least related to the complexity of the node label distributions.

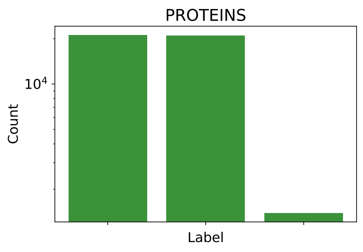
In Figure 6, we report the node label distributions of the social datasets considered (we recall that in this datasets the node label indicates the node out-degree). The node label distributions in the IMDB datasets are almost uni-modal. Thus, each ICGMM layer can effectively represent them using a small number of states (around 5). In the REDDIT datasets, most of the labels appear uniformly; thus, ICGMM exploits more states (around 10) to obtain a good representation of them. Finally, in the COLLAB dataset, the missing peak among the first bins force ICGMM to use the highest number of states (around 20) among the social datasets.



(a) DD.



(b) NCI1.



(c) PROTEINS.

Figure 7. The node labels distribution of each chemical dataset considered. In the plots, each bar is related to a different node label. The bar-label link follows the label id generated in the data pre-processing step (see Morris et al. 2020 for more details): the leftmost bar is related to the label with id 0, while the rightmost one is related to the label with the highest id. The height of a bar indicates the number of nodes in the dataset with the corresponding label.

On the chemical datasets, we expect a similar behavior. In Figure 7, we report the node label distributions of the chemical datasets considered (we recall that in these datasets the node label is a categorical value). The DD dataset contains more node labels than the NCI1 datasets. Accordingly, ɪCGMM exploits a higher number of states on the DD dataset than on the NC11 dataset. The PROTEINS dataset does not follow this trend: even if it has the smallest number of node labels (only 3), ɪCGMM uses the highest number of states (around 30). We believe that our model creates too many states due to a poor initialization. Further investigations are required to shade the light on this behavior.