ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

# Hidden Markov Models for Temporal Graph Representation Learning

Federico Errica[1*], Alessio Gravina[2*], Davide Bacciu[2], Alessio Micheli[2]

1 - NEC Laboratories Europe
Kurfürsten-Anlage, 36, 69115 Heidelberg - Germany

2 - University of Pisa - Department of Computer Science
Largo Bruno Pontecorvo, 3, 56127, Pisa - Italy

**Abstract**. We propose the Hidden Markov Model for temporal Graphs, a deep and fully probabilistic model for learning in the domain of dynamic time-varying graphs. We extend hidden Markov models for sequences to the graph domain by stacking probabilistic layers that perform efficient message passing and learn representations for the individual nodes. We evaluate the goodness of the learned representations on temporal node prediction tasks, and we observe promising results compared to neural approaches.

## 1 Introduction

In various scientific disciplines, ranging from molecular dynamics to behavior modeling, we often understand complex phenomena as systems of interacting entities and model them accordingly as graph structures. Some of these systems evolve over time, but modeling the dependencies of an evolving graph can be a tricky process. Indeed, entities might leave or join the network and the interactions between them vary over time. Inspired by the data-driven approach of graph representation learning, researchers have recently started to extend machine learning models for static graphs to the temporal domain to automatically capture the patterns of such evolving systems [1]. Most of these approaches are confined to the class of neural networks, and we argue that probabilistic methods for temporal graphs have been widely understudied despite their potential. For instance, a probabilistic model can easily capture the multimodality of the data distribution, which is useful in the context of stochastic processes [2]. Additionally, they can deal with missing data and exploit large amounts of unlabeled data to build rich unsupervised embeddings [3]. In this work, we propose the Hidden Markov Model for temporal Graphs (HMM4G), a deep and purely probabilistic model for sequences of graphs. We evaluate HMM4G on temporal node prediction tasks and show competitive performance with neural network counterparts.

## 2 Related Work

Our work is inspired by two different lines of research. The first is the one of deep probabilistic models for static graphs [3] that rely on message passing mechanisms

[*]The authors contributed equally to this work.

[4]. These methods are trained incrementally, one layer after another, and the depth of the architecture is functional to the spreading of messages between nodes of the graph. The second is the one of temporal graph representation learning [1, 5], which develops ad-hoc approaches to deal with the different technical and methodological challenges that the temporal extension, such as the varying topology of graphs across time, the sudden (dis)appearance of nodes, and the memory capacity of temporal models. Our work lies at the intersection between these two fields, by proposing a purely probabilistic method for graph representation learning. It also profoundly differs from [6], where a multi-agent filtering algorithm is proposed to determine an underlying state of the graph, but it is orthogonal to the topic of graph representation learning.

## 3 The HMM4G Model

We first introduce the mathematical notation and then we describe how to deal with temporal graph representation learning in a purely probabilistic fashion.

*Notation.* In this paper, a graph $g = (\mathcal{V}_g, \mathcal{E}_g, \mathcal{X}_g)$ is a tuple where $\mathcal{V}_g$ is the set of nodes and $\mathcal{E}_g$ is the set of directed edges $(u, v)$ connecting node $u$ to node $v$. Each node $u$ is associated with a vector $\boldsymbol{x}_u \in \mathcal{X}_g$. We also define the neighborhood of node $u$ as $\mathcal{N}_u = \{v \in \mathcal{V}_g \mid (v, u) \in \mathcal{E}_g\}$. (W.l.o.g.) A temporal graph, or discrete-time dynamic graph, is a sequence of $T$ graphs $(g^1, \dots, g^T)$ such that $\mathcal{V}_{g^i}{=}\mathcal{V}_{g^j} = \mathcal{V}_g \; \forall i, j \in [1, T]$. We assume we can discretize the time steps of the sequence, and we will use the superscript $t$ to refer to the time dimension. We introduce random variable (r.v.) $X^t_u$ with realization $\boldsymbol{x}_u$ to model the distribution of node $u$ attributes at time-step $t$. Similarly, we model the latent state of a node $u$ at time-step $t$ with a categorical r.v. $Q^t_u$ with $C$ possible states and discrete realization $q^t_u$. The posterior distribution of $Q^t_u$ conditioned on the evidence is another categorical distribution, and we refer to its parametrization with a vector $\boldsymbol{h}^t_u \in \mathbb{R}^C$ belonging to the $C$-1-simplex. Generally speaking, such a parametrization can be seen as the realization of a Dirichlet distribution of order $C$ that we denote with the letter $H^t_u$.

In HMM4G we mirror the same message passing mechanism of temporal deep graph networks by stacking layers of temporal graph convolutions on top of each other. The key difference is that we implement each layer as a special case of an Input-Output HMM (IO-HMM) for sequences [7], later extended to trees [8]. We present HMM4G's graphical model for a generic node $u$ and layer $\ell$ in Figure 1, abstracting from the layer to ease the exposition. Compared to an HMM, in a classical IO-HMM the prior distribution of the latent variable $Q^t$ is replaced by the conditional distribution of $Q^t$ given the input evidence at time-step $t$. Similarly, in a generic layer $\ell$ of HMM4G, we define a similar conditional distribution that takes into account the "messages" of the neighbors of $u$ computed at a previous layer $\ell - 1$.
Formally, the input evidence for node $u$ at time $t$ is modeled by a Dirichlet r.v.

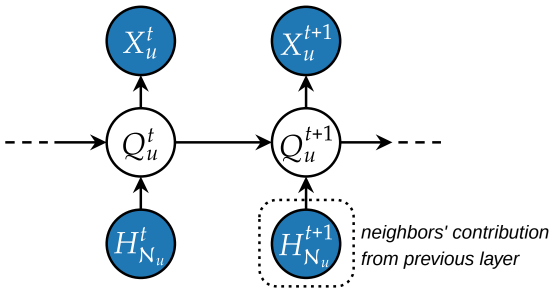
$X_u^t$
$X_u^{t+1}$
$Q_u^t$
$Q_u^{t+1}$
$H_{\mathcal{N}_u}^t$
$H_{\mathcal{N}_u}^{t+1}$
neighbors' contribution from previous layer

Fig. 1: Graphical model of HMM4G at layer $\ell$ for node $u$ in the graph. Observed r.v.s (values estimated at layer $\ell - 1$) are in blue and latent ones in white.

$H^t_{\mathcal{N}_u}$ of order $C$, whose realization $\boldsymbol{h}^t_{\mathcal{N}_u} \in \mathbb{R}^C$ is computed as follows:

$$\boldsymbol{h}^t_{\mathcal{N}_u} = \frac{1}{|\mathcal{N}_u|} \sum_{v \in \mathcal{N}_u} \boldsymbol{h}^t_v, \tag{1}$$

where we have aggregated the parameters of the posterior distributions of the neighbors computed at the previous layer. This is akin to what happens in deep graph neural networks, where the latent representations of neighboring nodes are combined by a permutation invariant function. We use $\boldsymbol{h}^t_{\mathcal{N}_u}$ to explicitly parametrize the (categorical) transition distribution for node $u$ at layer $\ell$:

$$P_{\boldsymbol{\theta}}(Q^t_u = i \mid Q^{t-1}_u = q^{t-1}_u, H^t_{\mathcal{N}_u} = \boldsymbol{h}^t_{\mathcal{N}_u}) \tag{2}$$

$$= \sum_{j=1}^{C} P_{\boldsymbol{\theta}_j}(Q^t_u = i \mid q^{t-1}_u)\boldsymbol{h}^t_{\mathcal{N}_u}(j), \tag{3}$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_C)$ are the transition parameters to be learned and $\boldsymbol{h}^t_{\mathcal{N}_u}(j)$ denotes the $j$-th component of a vector. The parameters $\boldsymbol{\theta}$ are *shared* across all nodes to generalize to unseen graphs of arbitrary topology. The only other distribution of the model, that is $P(X^t_u|Q^t_u)$, is learned as in standard IO-HMMs and also shared across all nodes, e.g., a Gaussian for continuous attributes.

At each layer, due to the presence of cycles in the graphs, we break the mutual dependencies between the node variables as a product of conditional probabilities, and we maximize the following pseudo-log-likelihood w.r.t. the parameters $\boldsymbol{\Theta}$:

$$\log \prod_{u \in \mathcal{V}_g} P_{\boldsymbol{\Theta}}(X^1_u, \ldots, X^T_u | \boldsymbol{h}^1_{\mathcal{N}_u}, \ldots, \boldsymbol{h}^T_{\mathcal{N}_u}). \tag{4}$$

Therefore, sequences of node attributes can be processed in parallel as it happens for temporal deep graph networks, meaning that the inference phase has the same linear complexity in the number of edges when processing the graph. We train HMM4G incrementally: we apply Expectation Maximization to layer $\ell$, and we infer the parameters of the posterior distribution of the variable $Q^t_u$ for all nodes and time-steps in the graph sequence. Training complexity is also similar to other models, except for [9] where the embedding layer is untrained. We use this information to compute $\boldsymbol{h}^t_{\mathcal{N}_u}$ in the subsequent layer $\ell+1$. When $\ell = 0$, the layer reduces to an HMM. We can compute closed-form update equations for the M-step by extending the classical HMM derivation; we do not show them here in the interest of space. The final latent representation of each node $u$ at time $t$, which is eventually used to make predictions about the nodes, is the concatenation of the realizations $\boldsymbol{h}^t_u$ across the layers of the architecture. In particular, we apply the same *unibigram* technique of [3] to the learned representations, which modifies each $\boldsymbol{h}^t_u$ of layer $\ell$ to take into account some neighboring statistics as well. Indeed, it was shown that unibigrams improve the quality of the learned representation for static graphs.

We could extend the model to account for discrete edge types and multiple prior layers using the techniques in [3], but we leave these extensions to future works. In addition, we do not assume a static graph structure over time [5] but rather we can let both the nodes' attributes and their interactions freely change.

## 4 Experiments

We consider four known graph datasets for node regression in the temporal graph domain, i.e., Twitter Tennis [10], Chickenpox [11], Pedalme, and Wikimath [12]. Twitter Tennis is a mention graph in which the underlying topology is dynamic, i.e., changes over time. Nodes are Twitter accounts and their labels encode the number of mentions between them. In contrast, the remaining three datasets have a static underlying topology in which only node features change over time. In these tasks, node attribute/labels represent reported cases of chickenpox in Hungary, delivery demands by Pedal Me in London, and daily user visits to Wikipedia pages, respectively. The task consists of predicting future node labels given the previous evolution of the graph (also known as graph snapshots). We compare our method against 11 state-of-the-art temporal deep graph networks for discrete-time dynamic graphs from [9], i.e., DCRNN, GCRN-GRU, GCRN-LSTM, GC-LSTM, DyGrAE, EGCN-H, EGCN-O, A3T-GCN, T-GCN, MPNN LSTM, and DynGESN. Such baselines differ in the learning strategy, attention mechanisms, and employed temporal and graph neural network layers [1, 5].

Each model is designed as a combination of two main components. The first is the recurrent graph encoder which maps each node's input features into a latent representation. The second is the readout, which maps the output of the first component into the output space. The readout is a Multi-Layer Perceptron for almost all models in the experiments (DynGESN uses ridge regression). For each timestamp of the sequence, we first obtain the latent node representations of the corresponding graph snapshot using the recurrent encoder, and then we feed them into the readout to obtain a prediction for each node.

We leverage the same experimental setting and data splits reported in [9]. Specifically, we performed hyper-parameter tuning via grid search, optimizing the Mean Square Error (MSE). We train using the Adam optimizer for a maximum

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

Table 1: Hyper-parameters tried during model selection.

| | **n. layers** | **epochs** | **C** | **lr** | **weight decay** | **hidden dim** |
|---|---|---|---|---|---|---|
| HMM4G | [1-5] | 10, 20, 40 | 5, 10 | – | – | – |
| Readout | [1-3] | 1000 | – | $10^{-3}$, $10^{-2}$ | 0, 0.0005, 0.005 | $2^2, 2^3, 2^5, 2^6, 2^7$ |

Table 2: Test MSE with standard deviation averaged over 10 final runs; best and second best results are shown as bold and underlined. Baselines taken from [9].

| | **Chickenpox** | **Tennis** | **Pedalme** | **Wikimath** |
|---|---|---|---|---|
| Mean baseline | 1.117 | 0.482 | 1.484 | 0.843 |
| Linear baseline | 0.952 | 0.356 | 1.499 | 0.663 |
| DCRNN | $1.097_{\pm 0.006}$ | $0.478_{\pm 0.004}$ | $1.454_{\pm 0.050}$ | $0.679_{\pm 0.007}$ |
| GCRN-GRU | $1.103_{\pm 0.004}$ | $0.477_{\pm 0.007}$ | $\mathbf{1.420}_{\pm 0.054}$ | $0.680_{\pm 0.021}$ |
| GCRN-LSTM | $1.097_{\pm 0.006}$ | $0.477_{\pm 0.006}$ | $1.453_{\pm 0.085}$ | $0.678_{\pm 0.008}$ |
| GC-LSTM | $1.095_{\pm 0.005}$ | $0.475_{\pm 0.010}$ | $1.490_{\pm 0.088}$ | $0.677_{\pm 0.009}$ |
| DyGrAE | $1.102_{\pm 0.013}$ | $0.480_{\pm 0.005}$ | $\underline{1.426}_{\pm 0.089}$ | $0.621_{\pm 0.012}$ |
| EGCN-H | $1.137_{\pm 0.026}$ | $0.481_{\pm 0.003}$ | $1.446_{\pm 0.168}$ | $0.779_{\pm 0.031}$ |
| EGCN-O | $1.135_{\pm 0.011}$ | $0.484_{\pm 0.002}$ | $1.469_{\pm 0.137}$ | $0.807_{\pm 0.047}$ |
| A3T-GCN | $1.078_{\pm 0.009}$ | $0.477_{\pm 0.005}$ | $1.494_{\pm 0.049}$ | $0.618_{\pm 0.008}$ |
| T-GCN | $1.083_{\pm 0.011}$ | $0.478_{\pm 0.004}$ | $1.515_{\pm 0.059}$ | $0.616_{\pm 0.011}$ |
| MPNN LSTM | $1.125_{\pm 0.005}$ | $0.482_{\pm 0.001}$ | $1.580_{\pm 0.102}$ | $0.856_{\pm 0.021}$ |
| DynGESN | $\mathbf{0.907}_{\pm 0.007}$ | $\mathbf{0.300}_{\pm 0.003}$ | $1.528_{\pm 0.063}$ | $\underline{0.610}_{\pm 0.003}$ |
| HMM4G | $\underline{0.939}_{\pm 0.013}$ | $\underline{0.333}_{\pm 0.004}$ | $1.769_{\pm 0.370}$ | $\mathbf{0.542}_{\pm 0.008}$ |

of 1000 epochs. We employ an early stopping criterion that stops the training if the validation error does not decrease for 100 epochs. We report in Table 1 the grid of hyper-parameters explored in our experiments.[1]

## 5 Results

We present the MSE test results of our experiments in Table 2. The first observation is that HMM4G has promising performances compared to the baselines employed in the experiments, ranking first or second in three out of four tasks. Indeed, HMM4G achieves an error score that is on average 16% better than the other baselines. The larger gain is achieved on Tennis and Wikimath datasets, where HMM4G is on average 39% and 28% better than the baselines, respectively. It is worth noting that Tennis and Wikimath datasets are more challenging than the other tasks in our experiments, as they contain two orders of magnitude more nodes than the others. In general, our method performs comparably with DynGESN. The worst performance is achieved on the Pedalme dataset, which consists of only 36 timestamps and 15 nodes, making it the smallest temporal graph in our experiments. The amount of nodes is so

[1]Code available at github.com/nec-research/hidden_markov_model_temporal_graphs

small that the initialization of the Gaussian emission distributions is problematic (probably connected to the high standard deviation), meaning there is not enough incentive for the model to differentiate the embeddings based on the structural information when maximizing the likelihood. On the other hand, HMM4G achieves the best performance on Wikimath, which is also the largest of the datasets considered, and it improves over DynGESN and the other baselines by a large margin. This suggests that, with enough data, our probabilistic model can learn good representations of the temporal graph dynamics.

## 6 Conclusions

We introduced a new probabilistic framework for learning in temporal graphs. Our method combines the sequential processing of HMMs with message passing to deal with topologically varying structures over time. We showed how the learned representations are useful for temporal node prediction tasks, especially on larger datasets. We believe that our contribution is one of the first attempts at bridging the gap between probabilistic models and temporal graph learning.

## References

[1] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(1):2648–2720, 2020.

[2] Federico Errica, Davide Bacciu, and Alessio Micheli. Graph mixture density networks. In *Proceedings of the 38th ICML*, pages 3025–3035, 2021.

[3] Davide Bacciu, Federico Errica, and Alessio Micheli. Probabilistic learning on graphs via contextual architectures. *Journal of Machine Learning Research*, 21(134):1–39, 2020.

[4] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1263–1272, 2017.

[5] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207:117921, 2022.

[6] Mert Kayaalp, Virginia Bordignon, Stefan Vlaski, and Ali H Sayed. Hidden markov modeling over graphs. In *In Proceedings of the IEEE Data Science and Learning Workshop (DSLW)*. IEEE, 2022.

[7] Yoshua Bengio and Paolo Frasconi. Input-output hmms for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249, 1996.

[8] Davide Bacciu, Alessio Micheli, and Alessandro Sperduti. An input-output hidden markov model for tree transductions. *Neurocomputing*, 112:34–46, 2013.

[9] Alessio Micheli and Domenico Tortorella. Discrete-time dynamic graph echo state networks. *Neurocomputing*, 496:85–95, 2022.

[10] Ferenc Béres, Róbert Pálovics, Anna Oláh, and András A. Benczúr. Temporal walk based centrality metric for graph streams. *Applied Network Science*, 3(1):32, 2018.

[11] Benedek Rozemberczki, Paul Scherer, Oliver Kiss, Rik Sarkar, and Tamas Ferenci. Chickenpox cases in hungary: A benchmark dataset for spatiotemporal signal processing with graph neural networks. In *The Web Conference, GLB workshop*, 2021.

[12] Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, and Oliver Kiss et al. PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (ICKM)*, 2021.