Metric Learning for comparison of HMMs using Graph Neural Networks

Rajan Kumar Soni RAJANSONI424@GMAIL.COM
Robert Bosch Centre for Data Science and AI, Indian Institute of Technology, Madras

Karthick Seshadri KARTHICK.SESHADRI@NITANDHRA.AC.IN
Department of Computer Science and Engineering, National Institute of Technology, Andhra Pradesh

Balaraman Ravindran RAVI@CSE.IITM.AC.IN
Robert Bosch Centre for Data Science and AI, Indian Institute of Technology, Madras

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

Hidden Markov models (HMMs) belong to the class of double embedded stochastic models which were originally leveraged for speech recognition and synthesis. HMMs subsequently became a generic sequence model across multiple domains like NLP, bio-informatics and thermodynamics to name a few. Literature has several heuristic metrics to compare two HMMs by factoring in their structure and emission probability distributions in HMM nodes. However, typical structure-based metrics overlook the similarity between HMMs having different structures yet similar behavior and typical behavior-based metrics rely on the representativeness of the reference sequence used for assessing the similarity in behavior. Further, little exploration has taken place in leveraging the recent advancements in deep graph neural networks for learning effective representations for HMMs. In this paper, we propose two novel deep neural network based approaches to learn embeddings for HMMs and evaluate the validity of the embeddings based on subsequent clustering and classification tasks. Our proposed approaches use a Graph variational Autoencoder and diffpooling based Graph neural network (GNN) to learn embeddings for HMMs. The graph autoencoder infers latent low-dimensional flat embeddings for HMMs in a task-agnostic manner; whereas the diffpooling based graph neural network learns class-label aware embeddings by inferring and aggregating a hierarchical set of clusters and sub-clusters of graph nodes. Empirical results reveal that the HMM embeddings learnt through the Graph variational autoencoders and diffpooling based GNN outperform the popular heuristics as measured by the cluster quality metrics and the classification accuracy in downstream tasks.

Keywords: Deep metric learning; Graph Neural Networks; Hidden Markov Models; Task agnostic embeddings; Graph variational autoencoders; Diff-pooling based graph convolutional networks.

1. Introduction

Hidden markov models are well known for their role as an enabler in different real word applications, such as in customer relationship, molecular biology, body posture identification, fraud detection and speech technology. Such applications benefit from the ability of HMMs in modeling long sequences of observations accurately. Typically, the following three

standard problems (Rabiner and Juang, 1986) are considered to be of interest in HMMs: (i) Computing the likelihood of generating a sequence of observations, (ii) Inferring the most likely sequence of states that might have generated an observation sequence and, (iii) Computing the parameters of the HMM given an observation sequence. However an accurate estimation of the distance between HMMs is important for the performance of several descriptive, predictive and prescriptive HMM-based models and hence the problem of learning embeddings for HMMs in a metric space has become a research problem of interest.
Some of the earlier attempts to find a good metric are based on the following: (i) co-emission probabilities (Lyngsø et al., 1999), (ii) Monte Carlo approximation for entropy divergence (Falkhausen et al., 1995) and Kullback–Leibler (KL) divergence (Juang and Rabiner, 1985), (iii) graph-matching (Sahraeian and Yoon, 2011), (iv) Bayes probability of error (Bahlmann and Burkhardt, 2001), (v) BP Metric (Panuccio et al., 2002) based on stationary cumulative probability distribution function (Zeng et al., 2010) and (vi) system statistics (Lu et al., 2013).
Typically, approaches cited above perform well only if models of similar structure exhibit similar behaviour. The structure-based metrics have an inherent lapse of not accounting for cases where the HMMs have different structures yet similar behavior. Metrics based on graph-matching will become intractable and impractical as the number of nodes and edges in the HMM graph increases (Lubiw, 1981). Further, the accuracy of the distance estimated by these metrics are heavily impacted by the inherent noise present in the observations used for training HMMs. Behavior based metrics are influenced by the representativeness of the reference sequence used for gauging the similarity between HMMs. Various graph network models in the deep learning literature have been shown to effectively infer feature representations to encode the key properties of graphs. However, to the best of our knowledge little exploration has been done in studying the applicability of these recently developed deep graph neural network models in the context of learning representations for HMMs. As a first-of-its-kind attempt, we propose a graph variational autoencoder (GVAE) based task agnostic model and a diffpoolng based graph convolutional neural network (GCN) model to learn embeddings for HMMs. The following are the key contributions of this paper:

(i) We propose two task-agnostic learnt embeddings for HMMs based on Autoencoders and Graph variational autoencoders in an attempt to effectively encode both the structure and behavior of HMMs in the embeddings learnt.

(ii) We apply the learnt embeddings in the context of a representative complete-linkage and a single-linkage clustering algorithm to showcase their validity. We have also analyzed the efficacy of the embeddings learnt in the context of a classification task.

(iii) We also propose and evaluate a supervised class-label aware embedding for HMMs by leveraging a Diffpooling based graph neural network in the event of a small amount of labelled data being available.

The rest of the paper is organized as follows: Section 2 reviews some of the state of art prior work and discusses the preliminaries. Section 3 describes the models and their architecture we experimented with to propose the learned metric. Section 4 describes the experimental setup, dataset, packages used and also discusses the evaluation methods and results with

respect to performance metrics. Section 5 has our concluding remarks along with some pointers for further research.

2. Prior work and Preliminaries

Juang and Rabiner (1985) proposed a probabilistic distance measure based on K-L divergence for HMMs. It was a consistent probabilistic modelling technique which employs a Monte Carlo procedure for evaluation. However its performance is affected by length of observations and by the initial values of the parameters. Later another metric based on co-emission probabilities was proposed by Rabiner and Juang (1986) for different types of models. Zhong and Ghosh (2003) proposed variants of KL such as minKL and maxKL which can be used to measure the distance between two generative models H_1 and H_2 using a dataset generated by one of the models as the reference set R. minKL and maxKL are respectively the minimum and the maximum over all $x \in R$ of the differences between the log likelihoods of x given the parameters of H_1 and H_2 respectively. minKL and maxKL are similar to single and complete-linkage metrics typically used in the context of an agglomerative clustering model. Zeng et al. (2010) proposed a metric based on stationary cumulative probability distribution function which uses the characteristics of a stationary HMM and exhibited a lesser time complexity than KL. Earlier works on graph based metrics mostly include metrics based on maximum common sub graphs (Bunke and Shearer, 1998) and based on graph isomorphism (Dijkman et al., 2009) to cite a few. As these metrics are NP-hard to compute, these metrics can be applied only on a restricted class of smaller sized problem instances. With the advancements in deep learning techniques many deep graph similarity-based learning models have been proposed in the recent past. Deep graph learning models are popular as these models learn a better representation of the graph features that helps in accurately performing the target task (Wu et al., 2021).

Hidden Markov model

A hidden markov model (Rabiner and Juang, 1986) is represented as $H = (\pi, A, B, n, C)$, where π represents the prior probability distribution over states; n is the number of states; A represents the transition matrix; B represents the matrix of emission probabilities in case of discrete observations and linearized parameters of the emission probability distribution in case of continuous observations. C represents either an alphabet of symbols to be emitted or a continuous space of observations depending upon whether the support of the observation sequence is continuous or discrete. As we have experimented with audio datasets, we have assumed that the emission distribution in a state i of H follows a multivariate normal distribution with the probability density for the observation x in state i given by $B_i(x) = \frac{1}{(2\pi)^{d/2}}|\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)^T\right\}$ of dimensionality d, where Σ is the diagonal co-variance matrix and μ is the mean vector. Hence, for continuous distributions row i of the B matrix is assumed to contain the mean and linearized diagonal co-variance of the emission distribution in state i. The HMMs considered in this paper are ergodic in which transitions are permitted from any state to any other state.

3. Methodology

This section contains two subsections; the first of which discusses baselines that approximate the distance between HMMs by comparing their structure, behavior or both structure cum behavior and based on embeddings derived through factorization of HMM's adjacency cum feature matrix. The next subsection discusses two task agnostic embeddings learnt through autoencoders and graph autoencoders; it also contains a discussion on class-aware representation learning for HMMs by training a diffpooling based network.

3.1. Baselines

3.1.1. Behavior based Metric using Cross log-likelihood estimates

The first metric is based on computing the original and the cross log-likelihoods (Rabiner and Juang, 1986) of a pair of observation sequences each generated by a different HMM. This metric is entirely based on the behavior of the HMMs being compared. The distance returned by this algorithm is the mean of the cross-likelihood distances taken across q such pairs of observation sequences. However, the validity of the distance computed is entirely dependent on the representativeness and discriminatory ability of the observation sequences considered. Further, this is a direct metric which does not produce any embedding that can be used in downstream tasks. This approach is outlined in Algorithm 1:

Algorithm 1 CrossLikelihoodBehaviorMetric

1: **Input:** Two Hidden Markov Models H_1, H_2.
2: Generate q pairs of observation sequences $(O_1^1, O_2^1), (O_1^2, O_2^2), ...(O_1^q, O_2^q)$ such that O_1^i and O_2^i are of length k each and are generated by H_1 and H_2 respectively.
3: $\forall i \in [1, q]$ $distance(i) = \frac{1}{k} \times ([\log(P(O_1 \mid H_1)) + \log(P(O_2 \mid H_2))] - [\log(P(O_2 \mid H_1)) + \log(P(O_1 \mid H_2))])$
4: $d = \frac{1}{q} \times (\sum_{i=1}^{q} distance(i))$
5: **Return** d

3.1.2. Structural Metric based on state mapping

Given two HMMs $H_1 = (\pi_1, A_1, B_1, n_1, C)$ and $H_2 = (\pi_2, A_2, B_2, n_2, C)$ as inputs, we normalize the HMM having a relatively more number of states. This normalization is done by treating the transition probability matrix of the larger HMM as the adjacency matrix of the graph to be clustered. We have used a weighted min cut based formulation for directed graphs proposed by Meila and Pentney (2007) for grouping the states. Post clustering, the clusters containing more than one node are treated as super-states in the normalized HMM. Multiple edges incident on super-states are coalesced with a weight equal to the sum of the weights on the edges being coalesced. The emission probability distribution of a super-state is estimated as the weighted mean of the parameters of the distributions of the nodes in the state. The weight attributed to a state in a super-state is taken to be the sum of the transition probabilities on the edges coming into the state from other states.

Subsequently, a random walk is performed on the two HMMs to determine the probability of being in a state after a certain number of steps. States in H_1 are mapped to

Algorithm 2 StateMapping Metric

Input: Two Hidden Markov Models $H_1 = (\pi_1, A_1, B_1, n_1, \mathrm{C})$ and $H_2 = (\pi_1, A_2, B_2, n_2, \mathrm{C})$
$n = \min(n_1, n_2)$
$H(\pi', A', B', n', C) \longleftarrow$ Model having greater than n states.
$\Psi = \text{spectralClustering}(A', n', n)$; //reduces the number of states from n' to n.
For each node $c_i \in \Psi$**:**
    If $(|c_i| > 1)$**:**
        $c_i \cdot \mu = 1/|c_i| \sum_{c_{ij} \in c_i} \left(\left(\sum_{(a,c_{ij}) \in E} A'[a, c_{ij}] \right) \times c_{ij} \cdot \mu \right)$
        $c_i \cdot \Sigma = 1/|c_i| \sum_{c_{ij} \in c_i} \left(\left(\sum_{(a,c_{ij}) \in E} A'[a, c_{ij}] \right) \times c_{ij} \cdot \Sigma \right)$
Perform a Random walk of t steps on H_1 and H_2.
For each $i \in [1, n]$**:**
    $P_1[i]$ = probability of being in state i in H_1 after t steps.
    $P_2[i]$ = probability of being in state i in H_2 after t steps.
J = Map node i in H_1 to a node j in H_2 such that $P_1[i]$ and $P_2[j]$ are the k^{th} order statistic in P_1 and P_2 respectively for some $k \in [1, n]$.
$distance = 0$
For each mapping $(i, j) \in J$**:**
    distance $+= (KL(B_1[i] \| B_2[j]) + KL(A_1[i] \| A_2[j]))$
Return *distance*

those in H_2 depending upon the agreement between the two states with respect to these probability estimates. Subsequently the distance between the HMMs is approximated as the sum of pairwise distances between the emission probability and transition probability distributions of the corresponding mapped states as outlined in the Algorithm 2 entitled *StateMappingMetric*, which is similar to the metric by Sahraeian and Yoon (2011). Though the structures of HMMs are normalized, based on our empirical observations, this metric fails to account for cases where two structurally different HMMs exhibit similar behavior.

3.1.3. Behavior based Metric using likelihood of a common sequence

Let D be a sequence of n observations. The metric entitled UniSequenceLikelihoodMetric (Falkhausen et al., 1995), approximates the distance between two HMMs H_1 and H_2 as the distance between the log-likelihood estimates of each of the n observations in D given H_1 and H_2. This metric shares the same demerits as enlisted for the Crosslikelihood based behavior metric and is computed as $distance(\log(P(D \mid H_1)), \log(P(D \mid H_2)))$.

3.1.4. Hybrid Metric based on both structure and behavior

A hybrid metric based on both structure and behavior is arrived at by approximating the distance d between the HMMs H_1 and H_2 as the weighted average of the distances d_b and d_s inferred by the behavior metric described by Algorithm 1 and the structure metric illustrated in Algorithm 2 respectively. The weighting factor α is in the range $[0, 1]$. The distance between the HMMs is computed as follows: $d(H_1, H_2) = \alpha \times d_b + (1 - \alpha) \times d_s$. From our experiments we could observe that depending upon the value of α, the performance of

the hybrid embeddings lies between that of the structure based metric $M2$ and the behavior based metric $M1$. When $\alpha = 1$ or $\alpha = 0$ the performance is same as that of $M1$ or $M2$ respectively.

3.1.5. Helper Routines

The following approaches are used by the matrix factorization-based models and deep neural net models to convert HMMs into matrices or a vector. The first approach converts an HMM with n nodes into emission and transition matrices. For each node we compute the emission vector by concatenating the linearized diagonal co-variance and mean vectors. We form an emission matrix where each row represents the emission vector of a state; Row i of the transition matrix is the out-transition probability distribution of the state i. Dimensionality of the emission matrix obtained is $n \times 2d$. The second approach concatenates the emission and transition matrices and linearizes the concatenated outputs to emit a vector V of size $(n^2 + 2nd) \times 1$.

3.1.6. Matrix factorization based linear embeddings

Given a pair of HMMs, H_1 and H_2, we construct the feature matrix of the HMMs by invoking $Convert\_HMM\_to\_FeatureMatrices$. The transition and emission matrices are concatenated to obtain matrix representations H_1', H_2' of the HMMs each of dimensionality $n \times (n + 2d)$. We subject these representations to Singular Value Decomposition (SVD) to factorize them as follows: $H_1' = L_1 S_1 R_1^T$ and $H_2' = L_2 S_2 R_2^T$, where L_1 and L_2 are the matrices of left singular values of dimensionality $n \times r_1$ and $n \times r_2$ respectively, S_1 and S_2 are the singular diagonal matrices containing the eigen values of dimensionality $r_1 \times r_1$ and $r_2 \times r_2$ respectively and R_1^T and R_2^T are the matrices of right eigen vectors of dimensionality $r_1 \times (n + 2d)$ and $r_2 \times (n + 2d)$ respectively.

To obtain the projections of the HMM nodes, we fix a $k << (n + 2d)$ and compute $E(H_1) = R_1^{(.,k)} \times S_1^{(k,k)}$ and $E(H_2) = R_2^{(.,k)} \times S_2^{(k,k)}$, where $R_1^{(.,k)}$ and $R_2^{(.,k)}$ are respectively the restrictions of R_1 and R_2 to the first k columns and $S_1^{(k,k)}$, $S_2^{(k,k)}$ are respectively the restrictions of S_1 and S_2 to the first k rows and columns. Each row in $E(H_1)$ and $E(H_2)$ represents the k-dimensional embedding of the respective nodes in H_1 and H_2. Each node in H_1 is mapped to its closest node in H_2, the closeness is measured in the k-dimensional Euclidean space. The distance between the HMMs is computed as the sum of the KL-divergences between the corresponding transition probability and emission probability distributions over all the pairs of mapped nodes in H_1 and H_2. However, our empirical observations reveal the lack of modeling power of such linear approaches in encoding HMMs; Such linear embeddings of HMMs are observed to perform poorly in the downstream clustering and classification tasks.

3.2. Embeddings based on Deep Neural Nets

This section outlines our proposed approach to learn task agnostic embeddings for HMMs through autoencoders and graph variational autoencoders, followed by our discussion on class-aware representation learning for HMMs by training a diffpooling based graph convolutional network.

3.2.1. Autoencoder based task agnostic HMM embeddings

We convert an HMM into a vector as mentioned in the section 3.1.5. The vector has both the structural aspects and the emission probabilities which represent the first-order behavior of the HMM. Our first approach uses the standard version of autoencoder architecture to obtain low dimensional embeddings for the vector representations of the HMMs to be compared and returns the distance between the embeddings. However, the embeddings learnt through autoencoders are optimized just with respect to the reconstruction loss and do not have any regularization in the latent space to encode the generative ability of the HMMs in the embeddings learnt. As this demerit is evident in our empirical observations of the performance of Autoencoder based embeddings in the downstream tasks, we propose a Variational Graph Autoencoder based embedding in the next section to address this concern.

3.2.2. Graph variational Autoencoder based task agnostic HMM embeddings

We have adopted the graph encoder model proposed by Kipf and Welling (2016) to build a variational autoencoder for an HMM $H = (\pi, A, B, n, C)$. The proposed GVAE converts H into a latent embedding ℓ in a lower dimension space. Let the emission probability distribution be characterized by a d-dimensional Gaussian. The feature of a state i in H (denoted by B_i) has a dimensionality of $2d$ to accommodate the parameters of the emission distribution of the state namely μ_i and the linearized form of diagonal σ_i^2. The GVAE infers a latent vector ℓ_i of dimensionality d_r for each state i in H, such that $d_r << 2d$.
The GVAE uses a GCN for inferring $\ell = [\ell_1, \ell_2, \ell_3, ..., \ell_n]$, which is a matrix of size $n \times d_r$. Assuming that, the likelihood of ℓ_i is independent of that of the other states given A and B, we get,

$$\hat{p}(\ell \mid A, B) = \prod_{i=1}^{n} \hat{p}\left(\ell_i \mid A, B\right) \tag{1}$$

There are two key problems to be overcome while learning the embedding. Firstly, as it is intractable to compute $\hat{p}(A, B)$ by marginalizing over all possible distributions for ℓ, we assume that $\hat{p}\left(\ell_i \mid A, B\right) \sim N(\hat{\mu}_i, \hat{\sigma}_i^2)$ for some $\hat{\mu}_i \in \mathbb{R}^{d_r}$ and $\hat{\sigma}_i^2 \in \mathbb{R}^{d_r}$. The second problem is to encode the generative ability of the model in the embedding to incorporate behavioral aspects of the model in the embedding. This is done by bifurcating the objective function θ into a typical term to capture the decoder's reconstruction error and a regularizer to optimize the KL-divergence between the inferred distribution $\hat{p}(.)$ and the ground truth $p(.)$. Parameters of the GCN are learnt using stochastic gradient descent. The optimization objective is given as θ in equation 2.

$$\theta = -E_{\hat{p}(\ell|A_iB)}[\log p(A \mid \ell) + KL[\hat{p}(\ell \mid A, B) \| p(\ell))]] \tag{2}$$

$\hat{\mu}_i$ and $\hat{\sigma}_i^2$ are initialized using the Xavier uniform initializer for all i. $p(\ell)$ denotes the prior distribution in which each ℓ_i is independently sampled from $N(0, 1)$. Reparameterization of the latent variables are done, to use back propagation for adjusting the parameters of GCN based on the error observed in the output layer. To obtain an embedding for H, we input the A and B matrices to the GCN and in the inferred output matrix ℓ, we treat each row as the embedding of the corresponding state in H. The GCN used as the inference

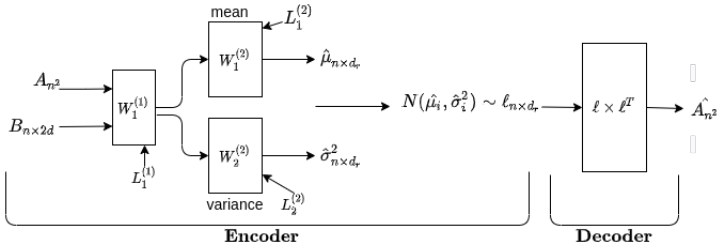

Figure 1: Latent variable inference using a graph AutoEncoder

model is two layered as shown in Figure 1. The transformation performed by the GCN is given by $A \times \delta(ABW^{(1)}) \times W^{(2)}$, where $W^{(1)}$ are the parameters or weights of the GCN in the first layer. $W^{(1)}$ is the shared weight matrix of $L^{(1)}$ which is shared by the two sub layers $L_1^{(2)}$ and $L_2^{(2)}$. $L_1^{(2)}$ and $L_2^{(2)}$ separately infers $\hat{\mu}_i$ and $\hat{\sigma}_i^2$. δ is the *Relu* activation given by $max(ABW^{(1)}, 0)$. The decoder that generates the state transition matrix A given the latent parameters, is modeled using the dot product between the corresponding latent vectors as given in Equation 3.

$$p(A \mid \ell) = \prod_{i=1}^{n} p\left(A_i \mid \ell_i, \ell\right) \tag{3}$$

The conditional likelihood of $A_i(\forall i \in [1, n])$ is assumed to follow a Dirichlet distribution as in equation 4. The distance between two HMMs can be computed using the Graph variational autoencoder as outlined in the Algorithm 3.

$$p\left(A_i \mid \ell_i, \ell\right) = \mathrm{Dir}\left(A_i \mid \left(\mathrm{softmax}\left(\left(\ell_i, \ell^{\top}\right)\right)\right)\right) \tag{4}$$

Algorithm 3 Embedding HMMs using Graph variational autoencoder

1: **Input:** $H_1 = (\pi_1, A_1, B_1, n_1, \mathrm{C})$ and $H_2 = (\pi_1, A_2, B_2, n_2, \mathrm{C})$
2: $X_1 = Graph\_VAE(A_1, B_1)$, $X_2 = Graph\_VAE(A_2, B_2)$
3: $distance = 0$
4: **for each** $v_i \in H_1$ **do**
5: $\quad v_j = \arg\min_{v_k \in H_2} euclidean\_distance(X_1(v_i), X_2(v_k))$
6: $\quad$ distance $+= euclidean\_distance(X_1(v_i), X_2(v_j))$
7: **end for**
8: **return** distance

3.2.3. Inference of hierarchical embeddings for HMMs using a supervised diffpooling GCN

One of the demerits of the previous two approaches based on autoencoders is that the HMM representations learnt are flat. The approaches have no innate provision to encode the hierarchical structures which are typically prevalent in HMMs. The overall behavior exhibited by an HMM may be viewed as the aggregation of local behaviors exhibited by a

set of closely knit clusters of states with high transition probabilities within a cluster than that across clusters. Similarly, the behavior exhibited by each of these clusters of states can be interpreted as the aggregate behavior exhibited by a set of sub-clusters within the clusters. The idea proposed by Ying et al. (2018) for learning structural graph representations has been adapted by us to encode both the hierarchical spectral-structures and behavior exhibited by the spectral-sub-structures in HMMs as illustrated in Figure 2.
The hierarchical view captures detailed prominent localized features responsible for the overall behavior of an HMM, as opposed to Autoencoder based approaches that at one stroke pool the individual node embeddings of an HMM to emit a global embedding, thereby losing important hierarchical local patterns.

Each layer $L(i)$ in the diffpooling network corresponds to a pair of GNNs $GNN_M^{(i)}$ and

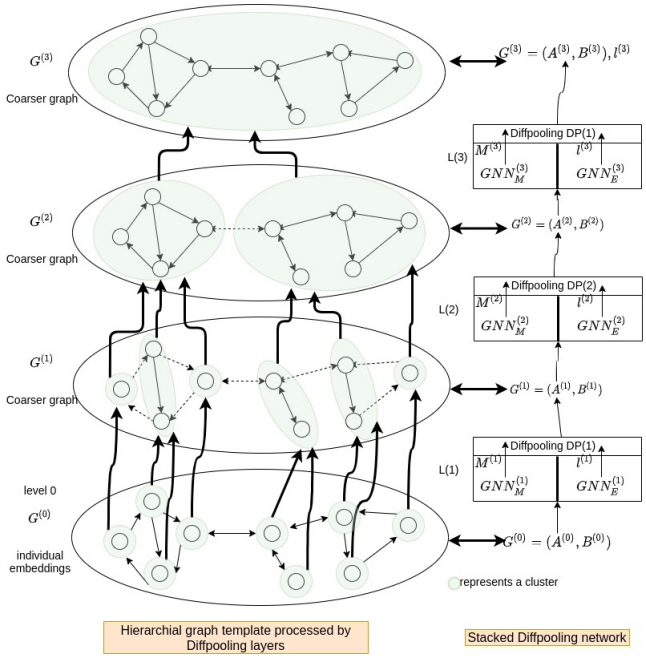

Figure 2: Learning hierarchical HMM embedding -an illustration

$GNN_E^{(i)}$, having n_{i-1} input nodes and n_i output nodes followed by a diffpooling sub layer named $DP(i)$. In the layer L(i), n_{i-1} corresponds to the number of nodes in the input graph $G^{(i-1)}$ and n_i indicates the number of clusters in the output coarsened graph $G^{(i)}$ at level i.
Each $GNN_M^{(i)}$ maps nodes in $G^{(i-1)}$ to their degree of association with respect to each output cluster node in $G^{(i)}$, this mapping is done using the node embeddings in $G^{(i-1)}$. $GNN_M^{(i)}$ outputs the mapping of the node in $G^{(i-1)}$ to a set of nodes in $G^{(i)}$ in the form of a $n_{i-1} \times n_i$ matrix $M^{(i)}$, where each row j in $M^{(i)}$ corresponds to the strength of the mapping of j to each of the nodes in $G^{(i)}$ as specified in equation 5.

$$M^{(i)} = \text{softmax}\left(\text{GNN}_M^{(i)}\left(A^{(i-1)}, B^{(i-1)}\right)\right). \tag{5}$$

Similarly each of the $GNN_E^{(i)}$ accepts the adjacency and emission matrices of $G^{(i-1)}$ and generates the embedding $\ell^{(i)}$ of each cluster in $G^{(i)}$ as outlined in equation 6.

$$\ell^{(i)} = \text{GNN}_E^{(i)}\left(A^{(i-1)}, B^{(i-1)}\right) \tag{6}$$

For each layer $L(i)$, $A^{(i-1)}$ and $B^{(i-1)}$ correspond to the transition and feature probability matrices of the HMM respectively. The number of output nodes in $GNN_M^{(i)}$ and $GNN_E^{(i)}$ is a hyper parameter that corresponds to the maximum number of clusters to be inferred in $L(i)$. The diffpooling sub-layer in $L(i)$ takes as inputs the embeddings of the nodes in $\ell^{(i)}$, the mapping matrix $M^{(i)}$ and $A^{(i-1)}$ to generate the feature matrix of the coarsened graph $G^{(i)}$ i.e. $(B^{(i)})$ and the adjacency matrix of $G^{(i)}$ i.e. $(A^{(i)})$ of sizes $n_i \times d$ and $n_i \times n_i$ respectively. The feature matrix $B^{(i)}$ is computed as the weighted aggregation of embeddings in $\ell^{(i)}$, where the weight of an embedding is the strength of association of the node to the output clusters as mentioned in $B^{(i)} = {M^{(i)}}^{\top}\ell^{(i)}$. The strength of the association between pairs of clusters/nodes in $G^{(i)}$ is computed as $A^{(i)} = {M^{(i)}}^{\top}A^{(i-1)}M^{(i)}$.
If there are m diffpooling layers then the embedding $l^{(m)}$ is considered as the final embedding of the HMM. The end-to-end training of the diffpooling network has been done using stochastic gradient descent by using the cross-entropy loss function. The distance between two HMMs can be computed as the distance between the hierarchical embeddings of the HMMs obtained by a diffpooling network. The embeddings compared are the ones emitted from the last diffpooling layer. The distance between two HMMs can be computed using a Diffpooling network as outlined in the Algorithm 4.

Algorithm 4 Inferring an HMM Embedding using Diffpooling

1: **Input:** M_1, M_2.
2: $(T_1, E_1) = Convert\_HMM\_to\_FeatureMatrices(M_1)$ //as mentioned in section 3.1.5
3: $(T_2, E_2) = Convert\_HMM\_to\_FeatureMatrices(M_2)$
   //T_1 and T_2 are transition matrices; E_1 and E_2 are emission matrices.
4: $V_1 = Diffpooling(T_1, E_1)$ and $V_2 = Diffpooling(T_2, E_2)$
5: **Return** distance(V_1, V_2)

4. Experimental setup and Performance evaluation

The dataset used for our experiments is the open source Free Spoken Digi Dataset (FSDD). FSDD dataset has $2K$ audio files each containing the utterance of a digit by one of the four speakers. We have used Google Collab for performing our experiments. The key libraries used are pytorch, numpy and scipy.
To validate the embeddings generated, we have performed two extrinsic tasks namely, clustering and classification using the embeddings and measured the validity of the embeddings through the performance metrics computed on the outputs of the tasks. For the clustering task we have experimented using a complete-linkage based agglomerative clustering and a single-linkage based Minimum Spanning Tree (MST) clustering algorithm. In the agglomerative clustering, initially we treat each HMM in the test-set as a singleton cluster and repeatedly combine two closest clusters; closeness is measured by computing a complete-linkage based distance between pairs of clusters. The merging of pairs of clusters is repeated

until we obtain the desired number of clusters. In the MST based clustering procedure (Jana and Naik, 2009), we treat each HMM in the test-set as a vertex of a complete graph in which the distance between two vertices is taken to be the distance between the HMM embeddings. An MST is formed on this graph, and we disconnect $m-1$ largest edges in the MST to obtain m connected components. A DFS is performed to retrieve the m components as m clusters. The validity of the clusters is assessed through the following metrics: Cluster Purity (CP), Normalized Mutual Information (NMI) and Rand Index (RI). Classification is leveraged as the second extrinsic task to validate the embeddings and classification accuracy is used as the metric. $M1$ to $M8$ denote the following metrics: (i) $M1$: Cross Likelihood based metric, (ii) $M2$: State mapping based metric, (iii) $M3$: Unisequence Likelihood metric, (iv) $M4$: Matrix Factorization based metric (v) $M5$: Hybrid metric based on both structure and behavior, (vi) $M6$: Autoencoder based metric (vii) $M7$: Graph Autoencoder based metric, and (viii) $M8$: Diffpooling based metric.

4.1. Clustering task

The following sets of experiments are designed to validate the proposed embedding for HMMs:

Training Set Size vs. Cluster quality: The objective of this experiment is to determine the training set size on the clustering performance. We trained a set of HMMs each with ten audio files of the same digit, sampled uniformly at random without replacement from the FSDD database. Let S be the set of HMMs trained which has an equal representation of all the digits in $[0, 9]$. We have split S into S_{train} and S_{test} such that $|S_{train}|$: $|S_{test}| = 4:1$ through a stratified sampling. We trained the Autoencoder, Graph Autoencoder and Diffpooling based models using S_{train} and tested the models using the tuples in S_{test}. Once an embedding is obtained for the HMMs in S_{test}, we have clustered the embeddings using both the agglomerative and MST based clustering algorithms and measured the quality of the clusters generated using CP, RI and NMI metrics. We repeated this process for different sizes of the set S such as 1000, 1300, 1600, 1900 and 2200, such that each of the smaller sized sets is a subset of any larger sized set, to infuse fairness in our comparisons. The results are plotted in the Figure 3. From the plots, it can be observed that the graph autoencoder based embeddings performed better than the autoencoder based embeddings and matrix factorization based linear embeddings. The behavioral baselines perform better than the structural baseline. Further the autoencoder and graph autoencoder based embeddings required lesser number of samples in the training set to achieve a better accuracy in the complete-linkage clustering as compared to the MST-based single-linkage clustering. As the number of test samples increases, the accuracy of the MST based clustering output dips sharply for the graph autoencoder based embedding.

Number of Audio files vs. Cluster quality: We followed a similar experimental design as that of the previous experiment; however, in this experiment for creating the set S, we have used different numbers of audio files of a digit to train the HMMs. We repeated an iteration of the previous experiment five times by setting $|S| = 2000$ with 5, 10, 15, 20 and 25 audio files each to train the HMMs in S. The cluster quality metrics obtained are plotted in the Figure 4. The relative trends in the performance of the baselines and the

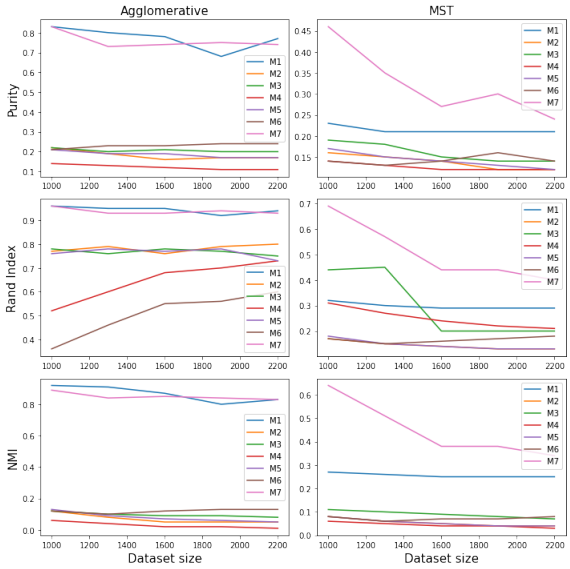

Figure 3: Training set size vs. Cluster quality

learnt embeddings are similar to that of the previous experiment. The Graph variational autoencoder based embedding seems to capture the generative ability of the HMMs and preserves the behavior as opposed to the autoencoder based approach, which has resulted in the former exhibiting a consistently better performance than the latter and also compared to the other metrics. As the number of sequences used for training HMMs increases the performance of the $M7$ embedding also exhibits an upward trend.

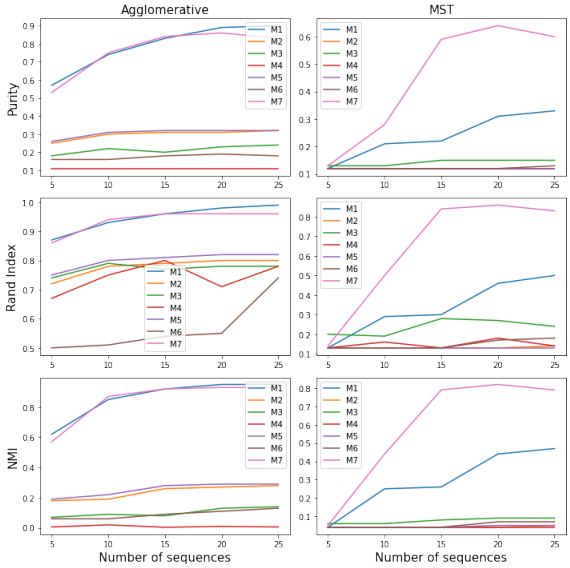

Figure 4: No. of audio files vs. Cluster quality

Variance in the number of HMM states vs. Cluster quality: This experiment is designed to assess the ability of the metric to recognize the similarity between two HMMs having different number of states but similar behavior. This experimental set-up is similar to the previous two experiments. We have repeated an iteration of the experiment five times by setting $|S| = 1000$, number of audio files to train an HMM $= 25$ and the number of states in an HMM is $n = \lceil r \sim N(13, i) \rceil$, where $i \in [1, 11]$ for each of the five iterations. The mean number of states is inferred to be 13, by plotting the mean likelihoods of the dataset given the HMMs for different number of states. The results of this experiment are plotted in the Figure 5. The graph variational autoencoder based metric exhibited a robust performance as the variance in n increases. This asserts that the embeddings learnt by graph variational autoencoders have the ability to recognize structurally different yet behaviorally similar HMMs.

We remark that the metrics that are not based on the learnt HMM embeddings, are not

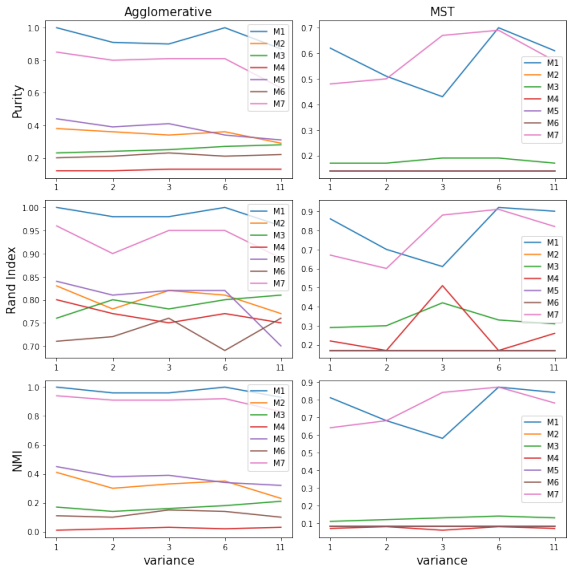

Figure 5: Variance in the No. of HMM states vs. Cluster quality

impacted by the variations in the training set; however, for comparison we have used the same test-set S_{test} as that of the learned metric, to assess the performance of the first four metrics discussed in this paper.

Table 1: T-test stats for Agglomerative and MST clustering $[df = 11, p = 0.05, 1 - tailed]$

Metric	M7 vs. M1	M7 vs. M2	M7 vs. M3	M7 vs. M4	M7 vs. M5	M7 vs. M6
CP	-3.44, 2.18	17.59, 7.04	17.81, 6.94	21.23, 7.04	15.82, 7.04	18.61, 6.94
RI	-3.52, 1.93	26.44, 7.38	13.11, 6.53	18.16, 6.64	14.54, 7.37	09.04, 7.21
NMI	-3.37, 2.19	20.46, 7.72	25.55, 7.53	26.86,26.36	17.90, 7.74	27.03, 7.58

Statistical tests $M7$ vs. *Rest*: We conducted 36 paired samples one-tailed t-tests for comparing the performance of the graph variational autoencoder based metric $M7$ with each of the other six metrics $M1$ to $M6$ with respect to the three performance indicators (CP, RI and NMI) for each of the two clustering algorithms. The metrics used for the t-test are spooled from the 15 experiments conducted and reported across the figures 3, 4 and 5. The t-test statistics computed with 11 degrees of freedom (df) and 95% confidence level for the agglomerative clustering algorithm and the MST-based algorithm are reported in the Table 1 as the first and the second values respectively in each cell. The *df* is set to 11 as the fifth experiment reported in 3, subsumes the other four. An entry $M7$ vs. Mi in the table 1, with respect to an algorithm and a performance indicator ($PI \in \{CP, RI, NMI\}$) is computed using the mean of the differences $\hat{\mu} = \mu(PI(M7) - PI(Mi))$ and the standard deviation of the differences $\hat{\sigma} = \sigma(PI(M7) - PI(Mi))$, as test-stat(M7 vs. Mi) $= \frac{\hat{\mu} \times \sqrt{df}}{\hat{\sigma}}$. As the stats except $M1$ (agglomerative) in Table 1 are much higher than the rejection threshold in the t-table, we remark that there is enough statistical evidence to accept the alternate hypothesis that $M7$ performs better than the metrics $M2$, $M3$, $M4$, $M5$, $M6$ with respect to both the clustering tasks across the 11 independent datasets sampled from FSDD. $M1$ beats $M7$ in the MST clustering task (highighted in red in Table 1). We further remark that as the performance of $M1$ is heavily reliant on the reference sequence generated for measuring the cross-likelihood, to make the competition tough, we have generated 10 reference sequences and averaged the cross-likelihood to give a fair advantage to $M1$. We refer the reader to the supplementary material with this submission for a visualization of the embeddings learnt by GVAE and diffpooling based GCN. The supplementary material also contains the mean and confidence intervals of the performance indicators across the experiments reported in this section.

4.2. Remarks on the superior performance of Diffpooling based GCN

While the other metrics performed poorly when the single-linkage MST clustering algorithm is employed, the diffpooling based metric exhibited robustness with NMI, CP and RI values of 1 for both the clustering algorithms across all the 15 experiments performed. We remark that this is not surprising as the HMM representation trained by diffpooling is class-label aware. The Diffpooling based metric achieved a perfect clustering accuracy even with lesser number of audio files being used to train the HMMs in the dataset. The Diffpooling based metric is resilient to the increase in the variance of the number of HMM states in the test-set, thereby asserting that it is able to recognize the similarity between two structurally different yet behaviorally similar HMM pairs like the graph autoencoder based embedding.

4.3. Classification task

The second extrinsic task of classification is performed using a multi-class classifier to recognize each of the ten digits in the audio files. For this task we have used 1600 training data points and 400 test points. Each HMM is trained using 10 uniformly sampled audio files of the utterance of the same digit in the FSDD dataset. A multi-class classifier neural network is employed using a standard classification network architecture. The mean classification accuracy across different test sets for the metrics $M6$, $M7$ and $M8$ are 0.45, 0.96 and 1.00 respectively. The diffpooling based model achieves a perfect classification accuracy for the

HMMs constructed from the FSDD dataset due to its class-label aware training of HMM embeddings.

5. Conclusion

Over the past few years, Graph Neural Networks have emerged as powerful and practical tools for machine learning tasks over graph structures. In this paper, we have proposed two novel methods for learning embeddings for Hidden Markov models that use graph variational autoencoders and diffpooling based graph convolutional networks to effectively learn the structure and behavior of HMMs. The following are the significant inferences based on our experimental results:

(i) Baseline metrics based on behavior tend to learn better quality embeddings than those that are based only on the structure.

(ii) Distance metrics based on learnt embeddings, typically yield better accuracy for the clustering and classification tasks than those generated by structure-based baselines.

(iii) Graph variational autoencoder based embeddings are effective even in tasks where the dataset contains structurally dissimilar yet behaviorally similar HMMs due to the regularized, behavior-preserving and generative latent space learnt by the model.

(iv) Distance metrics based on learnt embeddings, that exploit the hierarchical graphical structure of HMMs perform better with respect to downstream tasks than the structure agnostic flat embeddings.

(v) While other metrics falter when used with a single-linkage MST clustering algorithm, diffpooling exhibits a robust performance for both single and complete linkage based clustering algorithms.

A potential direction for future research will be to extend the proposed models to infer embeddings for other temporal-behavioral graphical models.

References

C. Bahlmann and H. Burkhardt. Measuring hmm similarity with the bayes probability of error and its application to online handwriting recognition. In *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pages 406–411, Sep. 2001.

H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3):255–259, 1998.

R. Dijkman, M. Dumas, and L. García-bañuelos. Graph matching algorithms for business process model similarity search. In *Proc. International Conference on Business Process Management*, pages 48–63, 2009.

M. Falkhausen, H. Reininger, and D. Wolf. Calculation of distance measures between hidden markov models. In *EUROSPEECH*, pages 1487–1490, 1995.

P. K. Jana and A. Naik. An efficient minimum spanning tree based clustering algorithm. In *2009 Proceeding of International Conference on Methods and Models in Computer Science (ICM2CS)*, pages 1–5, 2009.

B. H. Juang and L. R. Rabiner. A probabilistic distance measure for hidden markov models. *AT & T Technical Journal*, 64(2):391–408, 1985.

T. N. Kipf and M. Welling. Variational graph auto-encoders. In *NeurIPS Workshop on Bayesian Deep Learning*, 2016.

C. Lu, J. M Schwier, R. M. Craven, L. Yu, R. R. Brooks, and C. Griffin. A normalized statistical metric space for hidden markov models. *IEEE transactions on cybernetics*, 43 (3):806–819, 2013.

Anna Lubiw. Some np-complete problems similar to graph isomorphism. *SIAM Journal on Computing*, 10(1):11–21, 1981.

R. B. Lyngsø, C. N. Pedersen, and H. Nielsen. Metrics and similarity measures for hidden markov models. *Proceedings. International Conference on Intelligent Systems for Molecular Biology*, page 178—186, 1999.

M. Meila and W. R. Pentney. Clustering by weighted cuts in directed graphs. In *Proceedings of the 2007 SIAM International Conference on Data Mining (SDM)*, pages 135–144, 2007.

A. Panuccio, M. Bicego, and V. Murino. A hidden markov model-based approach to sequential data clustering. In *Proceedings of IAPR International workshop on statistical technique in pattern recognition*, pages 734–743, Windsor, Canada, 2002.

L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

S. M. E. Sahraeian and B-J. Yoon. A novel low-complexity hmm similarity measure. *IEEE Signal Processing Letters*, 18(2):87–90, 2011.

Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, Jan 2021. ISSN 2162-2388.

R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the NIPS'18*, page 4805–4815, Red Hook, NY, USA, 2018. Curran Associates Inc.

J. Zeng, J. Duan, and C. Wu. A new distance measure for hidden markov models. *Expert systems with applications*, 37(2):1550–1555, 2010.

S. Zhong and J. Ghosh. A unified framework for model-based clustering. *Journal of Machine Learning Research*, 4(6):1001–1037, 12 2003.