Neuralizing Efficient Higher-order Belief Propagation

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Graph neural network models have been extensively used to learn node representations for graph structured data in an end-to-end setting. These models often rely on localized first order approximations of spectral graph convolutions and hence are unable to capture higher-order relational information between nodes. Probabilistic Graphical Models form another class of models that provide rich flexibility in incorporating such relational information but are limited by inefficient approximate inference algorithms at higher order. In this paper, we propose to combine these approaches to learn better node and graph representations. First, we derive an efficient approximate sum-product loopy belief propagation inference algorithm for higher-order PGMs. We then embed the message passing updates into a neural network to provide the inductive bias of the inference algorithm in end-to-end learning. This gives us a model that is flexible enough to accommodate domain knowledge while maintaining the computational advantage. We further propose methods for constructing higher-order factors that are conditioned on node and edge features and share parameters wherever necessary. Our experimental evaluation shows that our model indeed captures higher-order information, substantially outperforming state-of-the-art k-order graph neural networks in molecular datasets.

1 Introduction

Graph representation learning [11, 10] has gained more attention of researchers in recent years since much of the data available is structured and can be easily represented with graphs. The aim of representation learning on graphs is to encode structural information of graphs in order to learn better and richer node embeddings. These embeddings can then be used for downstream tasks like node classification, link prediction, graph classification and regression.

In recent years, Graph Neural Networks (GNNs) [17, 6, 9, 37] have shown remarkable success in learning better representations for graph structured data. GNNs learn node representations by iteratively passing messages between nodes within their neighbourhood and updating the node embeddings based on the messages received. Though successful, these models are limited by the first order approximations they make in aggregating information from the neighbouring nodes; graphs often exhibit complex substructures which cannot be captured by only pairwise modeling. Many graphs exhibit repeating substructures like motifs, and nodes satisfy higher-order constraints e.g., valence constraints in molecular data [38, 1]. We can learn better node representations if we can design better message passing schemes that have sufficient representational power to capture such higher-order structures and constraints.

Node interactions have also been modeled with Probabilistic Graphical Models (PGMs) [34, 19] wherein nodes are seen as random variables with a factorized joint distribution defined over them. Research focus in such models has been to find approximate inference algorithms to compute node marginals. There is rich literature of theoretically grounded PGM inference algorithms to find the state of a node given its statistical relations with other variable nodes in the graph. Knowledge of

such inference algorithms can provide good inductive bias if it can be encoded in the neural network. Inductive bias lets the network favour certain solutions over others and if the bias is indeed consistent with the target task, it helps in better generalization [2]. So, while we use deep learning methods to learn representations in an end-to-end setting, better generalization may be achievable for some tasks with the inductive bias of classical inference algorithms. Unfortunately, these approximate inference algorithms become inefficient at higher-order.

One such well-known algorithm to find approximate node marginals is Loopy Belief Propagation (LBP) [26, 27]. In this paper, we propose to leverage sum-product LBP to formulate the message passing updates and build a better graph representation learning model. To this end, we derive an efficient message passing algorithm based on LBP with arbitrary higher-order factors, with the assumption that the higher-order factors are of low rank and can be represented as a mixture of small number of rank-1 tensors. The derived message passing updates only need two operations, matrix multiplication and Hadamard product and their complexity grows linearly with the number of variables in the factor.

Further, we show how the message passing updates can be represented in a neural network module and unroll the inference as a computational graph. We allow the messages to be arbitrary real valued vectors (instead of being constrained to be positive as in LBP) and treat the messages as latent vectors in a network; the latent vectors produced by the network can then be used for the learning the target task through end-to-end training. We refer to the process of unrolling the algorithm, relaxing some of the constraints, and using the resulting network as a component for end-to-end learning as *neuralizing* the algorithm. With LBP, this gives us a neural network message passing scheme, where higher-order factors can be added based on domain knowledge.

We focus on molecular data as a case-study. High-order information is likely useful in inference problems on molecular data but modeling the high-order interactions is challenging; nodes and edges are typed and the variable number of neighbours of each node makes fixed sized factors inappropriate. We show how to construct higher-order variable sized factors that share parameters through the use of edge and node features, allowing our model to be applied not only to molecular data, but also to other similar datasets where the graphs may have typed nodes and edges, with different nodes having different degrees. Experiments on two challenging large molecular datasets (QM9 and Alchemy) show that our model outperforms other recently proposed k-order GNNs substantially.

2   Related work

There have been significant number of works which have tried to incorporate inference algorithms in deep networks [44, 4, 21, 20, 31, 13, 39]. A large number of these works focus on learning pairwise potentials of graphical models on top of CNN to capture relational structures among their outputs for structured prediction. These methods are largely focused on modeling for a specific task like semantic segmentation [44, 21], scene graph prediction [39], and image denoising [36]. Further, [31, 13] represent planning algorithms as neural network layers in sequential decision making tasks.

The inspiration for our low-rank formulation of factors comes from [35] where higher-order potentials were decomposed to derive an efficient sampling based junction-tree algorithm. Further, significant number of low-rank bilinear models have seen successful application in visual question-answering task [16, 15, 41]. Similar low-rank methods were used in restricted Boltzmann machines as well for 3-way energy functions in [23, 24]. Our formulation of higher-order factors as tensor decompositions can be seen as a generalization of low-rank bilinear pooling in [16].

There has been some recent work on extending the graph convolutional neural networks to hyper-graphs in order to capture higher-order information [7, 40, 12, 42]. Feng et al. [7], Jiang et al. [12] used clique-expansion of hyper-edges to extend convolutional operation to hyper-graphs. Such modeling is equivalent to decomposing an hyper-edge to a set of pairwise edges. Similar approximation is applied in [40] where the number of pairwise edges added are reduced and are linearly dependent on the size of the hyperedge. Although, these methods operate on hyper-graphs, effectively the hyper-graphs are reduced to graphs with pairwise edges.

Recently, Zhang et al. [43] proposed a graph network expressive enough to represent max-product belief propagation. We work with the sum-product algorithm instead and further develop higher-order factors that can be flexibly applied to graphs containing nodes of different degrees. Further, Morris

et al. [25] and Maron et al. [22] used Weisfeiler Lehman (WL) graph isomorphism tests to construct increasingly powerful GNNs. They proposed models of message passing to capture higher order structures and compared their expressiveness with higher-order WL tests. As these models are theoretically shown to be more expressive in capturing higher-order information, we set these models as benchmarks and experimentally compare our model with the k-order GNNs.

3   Preliminaries

In this section, we briefly review two concepts central to our approach, low rank tensor decomposition and the sum product form of the loopy belief propagation algorithm.

3.1   Tensor decompositions

Tensors are generalizations of matrices to higher dimensions. A order-m tensor $\mathbf{T}$ is an element in $\mathbb{R}^{N_1 \times N_2 \cdots \times N_m}$ with N_k possible values in k^{th} dimension for $k \in \{1, 2, \ldots, m\}$. Tensor rank decompositions provide succinct representation of tensors. In CANDECOMP / PARAFAC (CP) decomposition, a tensor $\mathbf{T}$ can be represented as a linear combination of outer products of vectors as

$$\mathbf{T} = \sum_{r=1}^{R} \lambda_r w_{r,1} \otimes w_{r,2} \otimes \cdots \otimes w_{r,m} \tag{1}$$

where $\lambda_r \in \mathbb{R}$, $w_{r,k} \in \mathbb{R}^{N_k}$, $\otimes$ is the outer product operator i.e. $\mathbf{T}(i_1, i_2 \ldots, i_m) = \sum_{r=1}^{R} \lambda_r w_{r,1}^{i_1} w_{r,2}^{i_2} \cdots w_{r,m}^{i_m}$, and the term $w_{r,1} \otimes w_{r,2} \otimes \cdots \otimes w_{r,m}$ is a rank-1 tensor. The scalar coefficients λ_r can optionally be absorbed into $\{w_{r,k}\}$. The smallest R for which an exact R-term decomposition exists is the rank of tensor $\mathbf{T}$ and the decomposition (1) is its R-rank approximation. With this compact representation, an exponentially large tensor $\mathbf{T}$ with $N_1 \times N_2 \cdots \times N_m$ entries can be represented with R vectors for each variable in $\mathbf{T}$, i.e. with a total of $R(N_1 + N_2 + \cdots + N_m)$ parameters. More information about tensor decompositions can be found in [18, 28].

3.2   Graphical models and Loopy belief propagation

Consider a hyper-graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V} = \{1, \ldots, n\}$ and hyperedges $\mathcal{E} \in 2^{\mathcal{V}}$. An undirected graphical model $(\mathcal{G}, \mathcal{F})$ associates every vertex $i \in \mathcal{V}$ with a discrete random variable $x_i \in X$ and every hyperedge $a \in \mathcal{E}$ with a non-negative potential function $f_a \in \mathcal{F}$ such that the joint distribution of variables factorizes over $\mathcal{F}$ as

$$P(x_1, x_2 \ldots, x_n) = \frac{1}{Z} \prod_{f_a \in \mathcal{F}} f_a(X_a) \qquad Z = \sum_X \prod_{f_a \in \mathcal{F}} f_a(X_a) \tag{2}$$

where $X_a = [x_i : i \in a]$ and Z is the normalizing constant. Without loss of generality, we assume all variables can take d values and consequently $f_a(X_a) \in \mathbb{R}^{d^{|X_a|}}$. A factor graph is a convenient data structure used to represent such distributions. It is a bipartite graph whose edges connect variable nodes to factor nodes such that every x_i is connected to f_a if $x_i \in X_a$. Throughout the paper, we index variables with $\{i, j, \ldots\}$ and factors with $\{a, b, \ldots\}$.

Loopy belief propagation (LBP) [27, 26] is the procedure of computing approximate marginals $p(x_i)$ at each node x_i on a factor graph by sending messages between factor and variables nodes. Essentially, LBP starts by initializing two kinds of messages, factor-to-node $m_{a \to i}(x_i)$ and node-to-factor $m_{i \to a}(x_i)$. Messages are a function of the variable in the receiving node updated with the following recursive equations,

$$m_{i \to a}(x_i) = \prod_{c \in N(i) \setminus \{a\}} m_{c \to i}(x_i) \tag{3}$$

$$m_{a \to i}(x_i) = \sum_{X_a \setminus \{x_i\}} f_a(X_a) \prod_{j \in N(a) \setminus \{i\}} m_{j \to a}(x_j) \tag{4}$$

where $N(i)$ is the set of neighbours of i and $m_{a \to i}(x_i) \in \mathbb{R}^d$. After sufficient number of iterations, the belief of variables is computed by

$$b_i(x_i) = f_i(x_i) \prod_{a \in N(i)} m_{a \to i}(x_i). \tag{5}$$

3

4 Proposed Method

4.1 Low Rank Loopy Belief Propagation

Consider a factor $f_a(X_a)$ over $|X_a| = n_a$ variables with $X_a = [x_1, x_2 \ldots, x_{n_a}]$. Since, $f_a(X_a)$ is a tensor in $\mathbb{R}^{d^{n_a}}$, it can be represented as a sum of R fully factored terms in CP decomposition form (1).

$$f_a(X_a) = \sum_{r=1}^{R} \lambda_r w_{a_{r,1}} \otimes w_{a_{r,2}} \otimes \cdots \otimes w_{a_{r,n_a}} \tag{6}$$

where $w_{a_{r,j}} \in \mathbb{R}^d$. This representation is efficient if f_a is of low-rank i.e., $R \ll d^{n_a}$. We posit that such a low-rank representation is often a good approximation of f_a in practice; the method is likely to be useful when this assumption holds.

Absorbing λ_r in (6) into weights $\{w_{a_{r,j}}\}$ and substituting in (4), we have

$$m_{a \to i}(x_i) = \sum_{X_a \setminus \{x_i\}} \Big(\sum_{r=1}^{R} w_{a_{r,1}} \otimes w_{a_{r,2}} \otimes \cdots \otimes w_{a_{r,n_a}} \Big) \prod_{j \in N(a) \setminus \{i\}} m_{j \to a}(x_j) \tag{7}$$

$$m_{a \to i}(x_i) = \sum_{X_a \setminus \{x_i\}} \sum_{r=1}^{R} \big(w_{a_{r,1}} m_{1 \to a}(x_1) \big) \otimes \cdots w_{a_{r,i}} \cdots \otimes \big(w_{a_{r,n_a}} m_{n_a \to a}(x_{n_a}) \big) \tag{8}$$

Now we can marginalize out a variable in the same manner as if f_a was fully factorized. We simply push the outer summation inside, distribute and separately evaluate it over each of the univariate products $\big(w_{a_{r,j}} m_{j \to a}(x_j) \big)$. This gives us

$$m_{a \to i}(x_i) = \sum_{r=1}^{R} w_{a_{r,i}} \Big(\sum_{x_1} w_{a_{r,1}} m_{1 \to a}(x_1) \Big) \cdots \Big(\sum_{x_{n_a}} w_{a_{r,n_a}} m_{n_a \to a}(x_{n_a}) \Big) \tag{9}$$

$$m_{a \to i}(x_i) = \sum_{r=1}^{R} w_{a_{r,i}} \gamma_{a_{r,1}} \cdots \gamma_{a_{r,n_a}} \tag{10}$$

where $\sum_{x_j} w_{a_{r,j}} m_{j \to a}(x_j) = w_{a_r}^T m_{j \to a} = \gamma_{a_{r,j}} \in \mathbb{R}$. If we stack R weight vectors for each variable as $W_{a_i} = [w_{a_{1,i}}, w_{a_{2,i}}, \ldots w_{a_{R,i}}] \in \mathbb{R}^{d \times R}$ and $\boldsymbol{\gamma}_{a_j} = [\gamma_{a_{1,j}}, \gamma_{a_{2,j}}, \ldots \gamma_{a_{R,j}}]^T \in \mathbb{R}^{R \times 1}$, we can rewrite the (10) in matrix form as

$$m_{a \to i} = W_{a_i} \big[\boldsymbol{\gamma}_{a_1} \odot \boldsymbol{\gamma}_{a_2} \cdots \odot \boldsymbol{\gamma}_{a_{n_a}} \big] \tag{11}$$

where $\boldsymbol{\gamma}_{a_j} = W_{a_j}^T m_{j \to a}$ and $\odot$ is the Hadamard product or element-wise multiplication.

Now we have new message passing updates for the loopy belief propagation algorithm,

$$m_{a \to i} = W_{a_i} \Big[\big(W_{a_1}^T m_{1 \to a} \big) \odot \big(W_{a_2}^T m_{2 \to a} \big) \cdots \odot \big(W_{a_{n_a}}^T m_{n_a \to a} \big) \Big] \tag{12}$$

$$m_{i \to a} = \bigodot_{c \in N(i) \setminus \{a\}} m_{c \to i} \tag{13}$$

Belief update is simple. For the variable x_i, we project the messages from the other variables sharing a factor with x_i to $\mathbb{R}^R$, perform elementwise multiplication and then project the product back to $\mathbb{R}^d$. Thereafter, multiply such messages from all the factors x_i is connected to get the updated belief of x_i.

Clearly, the computational complexity of the message updates grows linearly with the addition of variables to factors and thereby the algorithm is efficent enough to run with higher-order factors.

4.2 Neuralizing Low Rank LBP

To learn better node and consequently, graph representations in an end-to-end setting, we seek to neuralize the Low Rank LBP algorithm by writing the message passing updates as a functionally equivalent neural network module and *unroll* the inference algorithm as a computational graph. We further replace the positive message vectors in LBP with unconstrained real valued hidden latent vector representations initialized from a feature extractor network. Depending on the problem, the feature extractor can be modeled to either capture zeroth or first order information. Thereafter, the message updates can follow the higher-order factor graph structure. In our experiments on molecular datasets, we use MPNN [9] as the feature extractor network and on an optical character recognition dataset, we use CNN as a base feature extractor.

The LBP algorithm contains multiplication of several terms leading to numerical instability due to overflow and underflow errors. It has been shown that normalization does not alter the final beliefs in LBP [27] and hence standard practice of handling these numerical errors is normalization of messages after every iteration of the algorithm. In our experiments, we tried methods like normalization after multiplication in equation (13) and addition of log-normalized messages from higher-order factors. Unfortunately, we found that such methods quickly lead to sub-optimal local minima, and the network fails to learn discriminative representations. Empirically, we found that replacing the multiplication operation in the message update function of (13) with summation followed by an MLP layer with relu activation works well in enabling stable learning of useful representations. We note that the network no longer represents (13) exactly, although it retains the remaining structure of the algorithm, capturing most of its inductive bias. We leave it to future work to handle the numerical issues related to exactly representing (13) during training.

Figure 1: (a) Factor graph with x_1 and its neighbours. (b) Message passing update for node x_1

We now describe the Low Rank LBP network. Consider a Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node and edge features. Let $\mathcal{F}$ be a factor graph defined on $\mathcal{G}$. Let $h_{x_i}^t$ be the hidden state at node x_i at iteration t. Then, we define the Low Rank LBP network with the following message passing update:

$$h_{x_i}^{t+1} = h_{x_i}^t + MLP\left(\sum_{a \in N(i)} W_{a_i} \Big[\bigodot_{j \in N(a)\backslash\{i\}} W_{a_j}^T h_{x_j}^t \Big] \right) \tag{14}$$

where $N(i)$ are the set of factors adjacent to x_i, $N(a)$ the set of variables adjacent to factor a and $\{W_{a_j}\}$ are learnable parameters. We add the previous hidden state with the new message as residual connection to improve learning. After T iterations, we can use a readout function on the learnt node representations for prediction tasks on nodes or the whole graph.

A factor is described by the set of matrices $\{W_{a_j}\}$ in (14) i.e. it has as many W_{a_j}'s as the number of variables adjacent to it in the factor graph. We further relax this constraint and maintain $2|\{W_{a_j}\}|$ matrices at each factor, one set is used for transformation before Hadamard product and the other set after the product. This increases the representative power of the network while still being able to represent (14) as the extra set of matrices can be learnt to be same as the first set.

5 Models

5.1 Modeling handwritten character sequences with higher-order factors

A sequences is one of the simplest graph structures. In a handwritten character sequence, the previous few characters are likely to contain useful information for predicting the next character. We consider one of the simplest higher order model: a higher order factor is placed at each node in the sequence. We use this model to experimentally study the scaling behaviour of our algorithm.
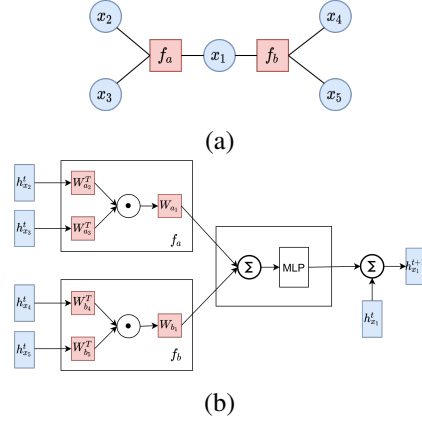
5

5.2 Modeling molecular data with higher-order factors

Molecular data has several properties needed for an effective study of higher-order representation learning models of graphs. A molecule is a graph with atoms as nodes and bonds that exist between atoms as edges. Higher-order information is present in the form of valence constraints of atoms which determine number of bonds they can form. Molecules often contain subgraphs, also called functional groups (e.g., OH, CHO etc.), which are identifiable and significant in determining their functionality and geometry. Furthermore, molecules can have highly variable connectivity, arbitrary size and variety of 3D molecular shapes. They come with broad range of chemical properties to predict with large heterogeneity in their structures [38]. Any graph learning model should be powerful enough to capture such higher-order structures and dependencies in order to learn highly discriminative representations. Thereby, we focus on molecular data to study the effectiveness of the proposed model and show its modeling flexibility in incorporating domain knowledge in constructing the factors.

In (14), the set of weights $\{W_{a_j}\}$ are separate for all factors and within a factor, separate for all variables connected to the factor. This gives much freedom in modeling to leverage domain knowledge in order to share some of these parameters and be able to work with large graphs as well. Given a molecular graph, we discuss possible ways of constructing higher-order factors, conditioning and sharing of parameters.

One way to capture higher-order information is to add a factor for every node in the molecule connecting that node (we will call this node *central atom* in that factor) and its neighbours to the factor. Then factor weights can be shared by conditioning on the following

- **Central atom type (CAT):** Weights within the factor are shared but different factors share parameters only if they have the same central atom type. In this case, we maintain only as many weight matrices as the number of atom types.

- **Bond type (BT):** Weights are shared if the bond type between the central atom and its neighbour is same. In this case, we maintain as many weight matrices as the number of bond types.

- **Central atom and bond type (CABT):** Weights are shared if both the central atom type and bond type are same. Number of weight matrices needed is the product of number of atom and bond types.

- **Central atom, bond and neighbour atom type (CABTA):** Weights are shared if the bond type and the atom types of atoms sharing the bond are same.

Do note that most molecular datasets have small number of atom types and bond types. Further, if we have continuous edge features or other features types such as strings, then each weight matrix W_{a_j} can be learnt by end-to-end training using an MLP with node and edge features as inputs and the weight matrix as output. This shows the proposed model of message passing is flexible and provides sufficient freedom in modeling.

6 Experiments

We first evaluate Low Rank Belief Propagation (LRBP) network on recognizing handwriting characters in a sequence, a task where higher order dependencies is likely to be useful. We use this task to better understand the scaling behaviour of the model. Further, to validate the effectiveness of LRBP-net in capturing higher-order information from more general graph structured data, we report results on two molecular datasets and compare with other higher-order GNNs that capture such information as well – the methods we are comparing with also give the state-of-the-art performances.

6.1 Optical character recognition data

To study the properties of LRBP network, we used the handwriting recognition data set [32], originally collected by [14]. The dataset consists of a subset of ~ 6100 handwritten words with the average length of ~ 8 characters. The words are segmented into characters and each character is rasterized into an image of 16 by 8 binary pixels. The dataset is available in 10 folds with each fold containing ~ 6000 characters. The task is to recognise characters by leveraging the neighbourhood of characters within the context of word. Since, the words come from a small vocabulary, there is a strong higher-order correlation present in the dataset. We follow the 10-fold cross-validation setup in [32] with 1

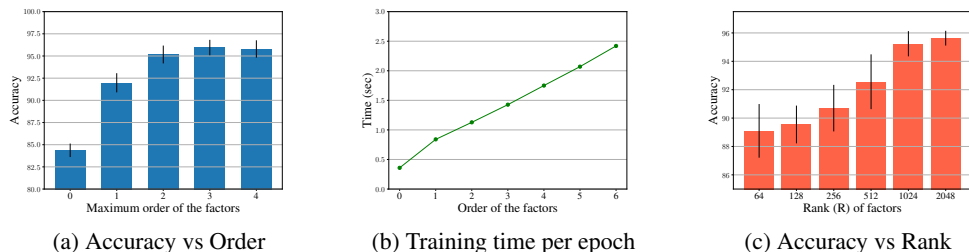(a) Accuracy vs Order　　　　(b) Training time per epoch　　　　(c) Accuracy vs Rank

Figure 2: Handwriting character Recognition

fold for training and 9 folds for testing and report averaged results. In our framework, each individual character image is a node x_i with the label $y_i \in \{a, b, \dots z\}$. Depending on the order, x_i can share a factor with other character nodes x_j within the same word. We evaluate the performance of LRBP-net by varying order and rank of the factors.

Implementation: We use 3 standard convolution and a fully connected layer as the base the feature extractor to get the zero-th order feature of 512 dimension. We then run 3 iterations of higher-order message passing followed by a classifier. We fix the rank of all factors at 1024 and share parameters between factors with same order. We train for 50 epochs with a learning rate of 0.001.

Results in Figure (2a) show strong improvements as maximum order of factors used is successively increased before saturating at 4th-order and above. To evaluate the efficiency of (14), we analysed the computation time as we vary the order of factors used. Figure (2b) shows that the training time per epoch grows almost linearly with the order of factors.

To evaluate the effect of rank of the factors, we fixed the zero-th order feature dimension to 64 and used factors up to order-3. We then ran three message passing iterations by varying the rank of the factors from 64 to 2048. Results in Figure (2c) show consistent improvement in performance with the increasing rank. We suggest the rank should be at least twice the message hidden vector dimension.

6.2 Molecular data

Datasets: We evaluate our model on QM9 [30, 29] and Alchemy [3] datasets on the task of regression on 12 quantum-mechanical properties. QM9 is composed of 134K drug-like organic molecules with sizes varying from 4 to 29 atoms per molecule. Atoms belong to one of the 5 types, Hydrogen (H), Carbon (C), Oxygen (O), Nitrogen(N), and Fluorine (F) and each molecule may contain up to 9 heavy (non-Hydrogen) atoms. Nodes come with discrete features along with their 3D positions. We follow the standard 80:10:10 random split for training, validation and testing.

Alchemy [3] is a recently released more challenging dataset of 119K organic molecules comprising of 7 atom-types H, C, N, O, F, S (Sulphur) and Cl (Chlorine) with up to 14 heavy atoms. These molecules are screened for being more likely to be useful for medicinal chemistry based on functional group and complexity. Furthermore, we follow the split based on molecule size where almost all training set contains molecules with 9 or 10 heavy atoms while the validation and test set contain molecules with 11 or 12 heavy atoms. As quantum-mechanical properties depend on the molecule size, this split tests how well the model can generalize to heavier molecules. The regression targets are same as in QM9 dataset.

Implementation: We plug our message passing scheme within Message Passing Neural Network (MPNN) framework [9] with the standard implementation provided by Pytorch-geometric [8]. MPNN has Edge-conv [9] as message function, GRU [5] as update function followed by a set2set [33] model as readout function for the graph. We run 3 iterations of MPNN message passing scheme followed by 3 iterations of higher-order LRBP message passing described in (14). We then combine the MPNN output from the third iteration and the LRBP output with concatenation followed by the set2set readout function. For LRBP module, we set the hidden vector (h_{x_i}) dimension to 64 and the rank R of factors to 512. We use Adam optimizer initialized with learning rate of $1e^{-3}$. All other hyperparameters are maintained as is provided by Pytorch-geometric implementation. All targets are normalized and are trained with the absolute error loss for 200 epochs with batch size of 64. Refer supplementary material for more details.

Table 1: Graph Regression results on QM9 dataset

Target	MPNN	Joint training of targets				Separate training of targets			
		123-GNN	PPGNN	LRBP	Gain(%)	123-GNN	PPGNN	LRBP	Gain(%)
μ	0.358	0.407	0.231	**0.092**	60.19	0.476	0.0934	**0.0688**	26.33
α	0.89	0.334	0.382	**0.183**	45.20	0.27	0.318	**0.14028**	48.04
ϵ_{homo}	0.00541	0.07807	0.00276	**0.00199**	27.89	0.0037	0.00174	**0.00168**	2.98
ϵ_{lumo}	0.00623	0.08492	0.00287	**0.00205**	28.57	0.00351	0.0021	**0.0016**	22.95
Δ_ϵ	0.0066	0.10939	0.00406	**0.00279**	31.28	0.0048	0.0029	**0.0024**	15.51
$\langle R^2 \rangle$	28.5	22.83	16.07	**2.81**	82.53	22.9	3.78	**1.41**	62.69
$ZPVE$	0.00216	0.01129	0.00064	**0.00018**	71.87	0.00019	0.000399	**0.00010**	47.37
U_0	2.05	8.910	0.234	**0.0907**	61.25	0.0427	0.022	**0.0163**	25.90
U	2	8.910	0.234	**0.0907**	61.23	0.111	0.0504	**0.0139**	72.42
H	2.02	8.910	0.229	**0.0906**	60.42	0.0419	0.0294	**0.0175**	40.47
G	2.02	8.910	0.238	**0.0907**	61.91	0.0469	0.024	**0.0134**	44.16
C_v	0.42	0.1184	0.184	**0.084**	29.05	0.0944	0.144	**0.0552**	41.52

Results:

For QM9 dataset, following [22] we report Mean Absolute Error (MAE) in two settings, one where all targets are jointly trained and the other where each target is separately trained. Factors are constructed as described in Section 5.2 with factor weights conditioned on central atom and bond type (CABT). 123-GNN [25] and PPGNN [22] are the k-order methods which capture higher-order information with PPGNN having the state-of-the-art

Table 2: Graph Regression results on Alchemy dataset

Target	MPNN*	LRBP	Gain(%)	LRBP Network Ablation Models			
				CAT	BT	CABT	CABTA
μ	**0.1026**	0.1041	-1.41	0.1091	0.1041	0.1092	0.1233
α	0.0557	**0.0451**	19.05	0.0473	0.0451	**0.0446**	0.0449
ϵ_{homo}	0.1151	**0.1004**	12.74	0.1065	0.1004	**0.1001**	0.1007
ϵ_{lumo}	0.0817	**0.0664**	18.74	0.0712	**0.0664**	0.0685	0.0696
Δ_ϵ	0.0832	**0.0691**	16.88	0.0739	**0.0691**	0.0703	0.0720
$\langle R^2 \rangle$	0.0271	**0.0099**	63.47	0.0099	0.0099	**0.0094**	0.0120
$ZPVE$	0.0259	**0.0115**	55.42	0.0116	0.0115	**0.0108**	0.0140
U_0	0.0131	**0.0044**	65.80	**0.0042**	0.0044	0.0046	0.0054
U	0.0131	**0.0044**	65.90	**0.0041**	0.0044	0.0046	0.0054
H	0.0130	**0.0044**	65.77	**0.0042**	0.0044	0.0046	0.0054
G	0.0130	**0.0044**	65.77	**0.0042**	0.0044	0.0046	0.0054
C_v	0.0559	**0.0481**	13.93	**0.0472**	0.0481	0.0488	0.0502
MAE	0.0499	**0.0394**	21.18	0.04115	**0.0394**	0.0400	0.0424

score in QM9 dataset. Table 1 shows that LRBP-net outperforms PPGNN by significant margin in all the targets under both the settings. Furthermore, the margin of improvement indicates that much of the higher-order information was not sufficiently captured by the k-order GNNs.

For Alchemy dataset, following [3] we report MAE on jointly trained normalized targets and compare with MPNN which was the best performing model in the benchmark results [3]. We use the validation set to select among the models in Section 5.2. As LRBP-net is built on MPNN as the backend network, the margin of improvement in Table 2 is mainly because of higher-order message passing module. We also did an ablation study using models described in Section 5.2. Results indicate that conditioning of parameters on either central atom or edge type helps most. Conditioning in these ways help capture most of the higher-order information centered around an atom (node). For the CABTA method, there is slight decrease in performance which is likely caused by the large parameter size in the model. Collectively, the ablation results suggest that major improvements are coming from the message passing scheme itself since conditioning on only bond types (BT) seems to be sufficient for better performance.

7 Conclusion

In this paper, we present an efficient Low-rank sum-product belief propagation procedure for inference in factor graphs where the complexity of message updates grows linearly with the number of variables in the factor. Further, we neuralize the message passing updates to learn better node representations with end-to-end training. This gives us a fairly simple, flexible, powerful and efficient message passing scheme for representation learning of graph data where higher-order information is present. We validate the proposed model with experiments on molecular data where it outperforms other state-of-the-art k-order GNNs substantially.

Broader Impact

We propose a new efficient representation learning technique for graph structured data. It is a general algorithm to learn better graph representations and would impact all the areas where data can be represented in the form of sparse graphs. Particularly, we show better performance on molecular data and hence our method will likely be helpful in applications which require predictions on molecular data. We foresee our method to be helpful in applications like drug discovery where new drugs can be discovered.

These graph based learning techniques can also be used to learn behaviour of communities. Such technologies may be misused by persons with mal-intent. Such outcomes can be curbed by increased research into privacy technologies and appropriate regulations by governmental organizations.

References

[1] Agarwal, S., Branson, K., and Belongie, S. (2006). Higher order learning with graphs. In *Proceedings of the 23rd international conference on Machine learning*, pages 17–24.

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

[3] Chen, G., Chen, P., Hsieh, C.-Y., Lee, C.-K., Liao, B., Liao, R., Liu, W., Qiu, J., Sun, Q., Tang, J., et al. (2019). Alchemy: A quantum chemistry dataset for benchmarking ai models. *arXiv preprint arXiv:1906.09427*.

[4] Chen, L.-C., Schwing, A., Yuille, A., and Urtasun, R. (2015). Learning deep structured models. In *International Conference on Machine Learning*, pages 1785–1794.

[5] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[6] Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

[7] Feng, Y., You, H., Zhang, Z., Ji, R., and Gao, Y. (2019). Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3558–3565.

[8] Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*.

[9] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org.

[10] Hamilton, W., Ying, Z., and Leskovec, J. (2017a). Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034.

[11] Hamilton, W. L., Ying, R., and Leskovec, J. (2017b). Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*.

[12] Jiang, J., Wei, Y., Feng, Y., Cao, J., and Gao, Y. (2019). Dynamic hypergraph neural networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2635–2641.

[13] Karkus, P., Hsu, D., and Lee, W. S. (2017). Qmdp-net: Deep learning for planning under partial observability. In *Advances in Neural Information Processing Systems*, pages 4694–4704.

[14] Kassel, R. H. (1995). *A comparison of approaches to on-line handwritten character recognition*. PhD thesis, Massachusetts Institute of Technology.

[15] Kim, J.-H., Jun, J., and Zhang, B.-T. (2018). Bilinear attention networks. In *Advances in Neural Information Processing Systems*, pages 1564–1574.

[16] Kim, J.-H., On, K.-W., Lim, W., Kim, J., Ha, J.-W., and Zhang, B.-T. (2016). Hadamard product for low-rank bilinear pooling. *arXiv preprint arXiv:1610.04325*.

[17] Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[18] Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3):455–500.

[19] Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

[20] Lin, G., Shen, C., Reid, I., and van den Hengel, A. (2015). Deeply learning the messages in message passing inference. In *Advances in Neural Information Processing Systems*, pages 361–369.

[21] Lin, G., Shen, C., Van Den Hengel, A., and Reid, I. (2016). Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3194–3203.

[22] Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. (2019). Provably powerful graph networks. In *Advances in Neural Information Processing Systems*, pages 2153–2164.

[23] Memisevic, R. and Hinton, G. (2007). Unsupervised learning of image transformations. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

[24] Memisevic, R. and Hinton, G. E. (2010). Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural computation*, 22(6):1473–1492.

[25] Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609.

[26] Murphy, K., Weiss, Y., and Jordan, M. I. (2013). Loopy belief propagation for approximate inference: An empirical study. *arXiv preprint arXiv:1301.6725*.

[27] Pearl, J. (2014). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier.

[28] Rabanser, S., Shchur, O., and Günnemann, S. (2017). Introduction to tensor decompositions and their applications in machine learning. *arXiv preprint arXiv:1711.10781*.

[29] Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A. (2014). Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1:140022.

[30] Ruddigkeit, L., Van Deursen, R., Blum, L. C., and Reymond, J.-L. (2012). Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875.

[31] Tamar, A., Wu, Y., Thomas, G., Levine, S., and Abbeel, P. (2016). Value iteration networks. In *Advances in Neural Information Processing Systems*, pages 2154–2162.

[32] Taskar, B., Guestrin, C., and Koller, D. (2004). Max-margin markov networks. In *Advances in neural information processing systems*, pages 25–32.

[33] Vinyals, O., Bengio, S., and Kudlur, M. (2015). Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*.

[34] Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305.

[35] Wrigley, A., Lee, W. S., and Ye, N. (2017). Tensor belief propagation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3771–3779. JMLR. org.

[36] Wu, Z., Lin, D., and Tang, X. (2016). Deep markov random field for image modeling. In *European Conference on Computer Vision*, pages 295–312. Springer.

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

[38] Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. (2018). Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530.

[39] Xu, D., Zhu, Y., Choy, C. B., and Fei-Fei, L. (2017). Scene graph generation by iterative message passing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5419.

[40] Yadati, N., Nimishakavi, M., Yadav, P., Nitin, V., Louis, A., and Talukdar, P. (2019). Hypergcn: A new method for training graph convolutional networks on hypergraphs. In *Advances in Neural Information Processing Systems*, pages 1509–1520.

[41] Yu, Z., Yu, J., Fan, J., and Tao, D. (2017). Multi-modal factorized bilinear pooling with co-attention learning for visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 1821–1830.

[42] Zhang, R., Zou, Y., and Ma, J. (2019a). Hyper-sagnn: a self-attention based graph neural network for hypergraphs. *arXiv preprint arXiv:1911.02613*.

[43] Zhang, Z., Wu, F., and Lee, W. S. (2019b). Factor graph neural network. *arXiv preprint arXiv:1906.00554*.

[44] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1529–1537.