Graph Neural Processes

Towards Bayesian Graph Neural Networks

Andrew Carr

Perception, Control, and Cognition Lab

Table of contents

Introduction

Problem

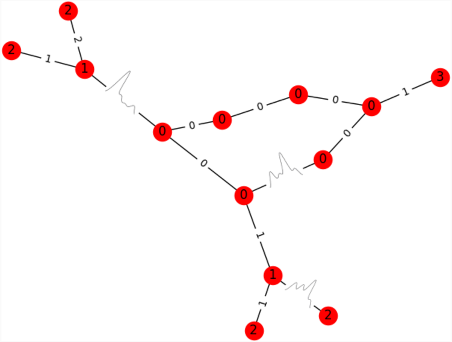
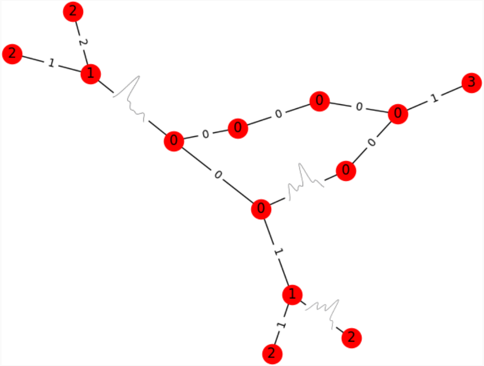


Figure 1: Graph with imputed edge distributions

Problem



$$G = (a, V, E)$$
$$V = \{v_i\}_{i=1:N^v}$$
$$E = \{e_k, v_k, u_k\}_{k=1:N^e}$$

Motivation

- Visual scene understanding
- Few-shot learning
- Learning dynamics of physical systems
- Traffic prediction
- Multi-agent systems
- Natural language processing
- semi-supervised text classification

Background

Graph Neural Networks

- Operate on Graph Structured Input

Graph Neural Networks

- Operate on Graph Structured Data
- Neural Operations on Graphs (e.g., spectral convolution)

Use Eigeninformation from traditional Graph Laplacian to perform convolution

$$L = D - A$$

Graph Neural Networks

- Operate on Graph Structured Input
- Neural Operations on Graphs (e.g., spectral convolution)
- Fixed sized input

Graph Neural Networks

- Operate on Graph Structured Input
- Neural Operations on Graphs (e.g., spectral convolution)
- Fixed sized input
- Difficult to scale

Graph Neural Networks

- Operate on Graph Structured Input
- Neural Operations on Graphs (e.g., spectral convolution)
- Fixed sized input
- Difficult to scale
- No measure of uncertainty

Conditional Neural Processes

$$P(Y_T|X_T, X_C, Y_C) \iff P(Y_T|X_T, r_C)$$

Conditional Neural Processes

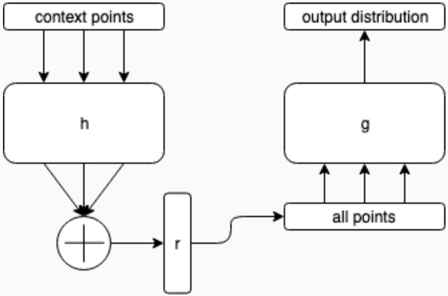
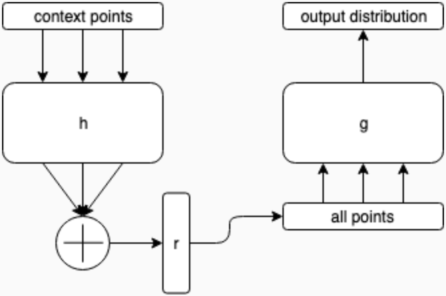


Figure 2: Conditional Neural Process Architecture

Conditional Neural Processes



$$
\begin{aligned}
r_i &= h_\theta(\vec{x}_i) && \forall \vec{x}_i \in X_C && (1)\\
r_C &= r_1 \oplus r_2 \oplus r_3 \oplus \cdots \oplus r_n && && (2)\\
z_i &= g_\phi(\vec{y}_i | r_C) && \forall \vec{y}_i \in X_T && (3)
\end{aligned}
$$

Conditional Neural Processes

- Flexible

Conditional Neural Processes

- Flexible
- Scalable

Conditional Neural Processes

- Flexible
- Scalable
- Measures uncertainty

Our Method

Graph Neural Processes

- Conditional Neural Process on Graphs

Graph Neural Processes

- Conditional Neural Process on Graphs
- Edges are context points

Graph Neural Processes

- Conditional Neural Process on Graphs
- Edges are context points
- Spectral Features

Features

Normalized Symmetric Graph Laplacian

$$L = I_n - D^{-1/2} A D^{-1/2}$$

$$\Lambda = \sigma(L)$$

Features

Take the eigenvectors of the graph laplacian and select the first k to obtain the local spectral eigen features

$$\Lambda|_k = (\Lambda_{kj})_{\substack{k \in r \\ 1 \leq j \leq m}}$$

Features

Or, more clearly

$$\Lambda|_k = (\Lambda_{kj})_{\substack{k \in r \\ 1 \le j \le m}}$$

```
def get_local_spectral_eigen_features(G, ind, k):
    """
    G - graph (networkx graph)
    ind - location of the context point edge in the adjacency matrix
    k - hyperparameter tuned based on the size of the input graphs in X_C
    """
    A = nx.adjacency_matrix(G).toarray()
    N = A.shape[0]

    diags = A.sum(axis=1)**(-1/2)
    D = scipy.sparse.spdiags(diags.flatten(), [0], N, N, format='csr').toarray()

    L = np.eye(N) - D.dot(A).dot(D) # calculate normalized graph laplacian
    val, vec = np.linalg.eig(L)

    return vec[ind][:k]
```

Features

Additionally the value and degree of each node on the edge are used as features

Algorithm

How does it work?

Algorithm Walk Through

Let X input graphs
for $t = 0, \cdots, n_{\text{epochs}}$ **do**
  for x_i in X **do**
    Sample $p \leftarrow \text{unif}(p_0, p_1)$
    Assign $n_{\text{context points}} \leftarrow p \cdot |\text{Edges}(x_i)|$
    Sparsely Sample $x_i^{cp} \leftarrow x_i|_{n_{\text{context points}}}$
    Compute degree and adj matrix D, A for graph x_i^{cp}
    Compute $L \leftarrow I_{n_{\text{context points}}} - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$
    Define F^{cp} as empty feature matrix for x_i^{cp}
    Define F as empty feature matrix for full graph x_i
    for edge k in x_i **do**
      Extract eigenfeatures $\Lambda|_k$ from L
      Concatenate $[\Lambda|_k; v_k; u_k; d(v_k); d(u_k)]$ where v_k, u_k are the attribute values at the node, and $d(v_k), d(u_k)$ the degree at the node.
      if edge $k \in x_i^{cp}$ **then**
        Append features for context point to F^{cp}
      end if
      Append features for all edges to F
    end for
    Encode and aggregate $r_C \leftarrow h_\theta(F^{cp})$
    Decode $\tilde{x}_i \leftarrow g_\phi(F|r_C)$
    Calculate Loss $l \leftarrow \mathcal{L}(\tilde{x}_i, x_i)$
    Step Optimizer
  end for
end for

Experiments

16 Datasets

Dataset	$\|X\|$	$\|\bar{N}\|$	$\|\bar{E}\|$	$\| \cup \{e_k\}\|$
AIDS	2000	15.69	16.20	3
BZR_MD	306	21.30	225.06	5
COX2_MD	303	26.28	335.12	5
DHFR_MD	393	23.87	283.01	5
ER_MD	446	21.33	234.85	5
Mutagenicity	4337	30.32	30.77	3
MUTAG	188	17.93	19.79	4
PTC_FM	349	14.11	14.48	4
PTC_FR	351	14.56	15.00	4
PTC_MM	336	13.97	14.32	4
Tox21_AHR	8169	18.09	18.50	4
Tox21_ARE	7167	16.28	16.52	4
Tox21_aromatase	7226	17.50	17.79	4
Tox21_ARLBD	8753	18.06	18.47	4
Tox21_ATAD5	9091	17.89	18.30	4
Tox21_ER	7697	17.58	17.94	4

Table 1: Features of the explored data sets

5 Baselines

- Random Value

5 Baselines

- Random Value
- Common Value

5 Baselines

- Random Value
- Common Value
- Common Local Value

5 Baselines

- Random Value
- Common Value
- Common Local Value
- Random Forest

5 Baselines

- Random Value
- Common Value
- Common Local Value
- Random Forest
- Neural Network

Evaluation of Success

- Acceptable Success: Gain insight into Bayesian Neural Networks on Graph Structured Data

Evaluation of Success

- Acceptable Success: Gain insight into Bayesian Neural Networks on Graph Structured Data
- Stretch Success: Beat all baselines in Precision, Recall, and F1-Score

Timeline

- Proposal May 2019
- Experiments May - September 2019
- Submit Thesis to Committee October 1 2019
- Defense around October 15 2019

References i

See https://arxiv.org/abs/1902.10042 for complete list of references