Work Log

Liu Yanpei
elliott_liu@outlook.com

November 14, 2024

Hands-on GCN with the Karate Club Dataset

Dataset

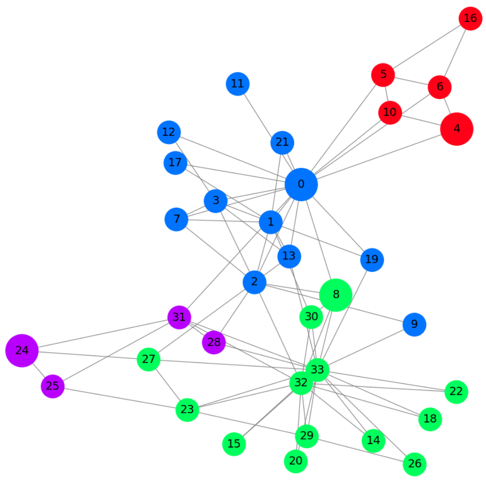


Figure 1: Karate Club Network

GCN formulation

Here, GCN tries to learn the following mapping

$$f : \mathcal{G}(V, E) \times \mathcal{X}^{|V| \times d} \to \mathcal{Y},$$

where the graph structure is given by edge index (adjacency list, or less efficiently, adjacency matrix), and the node features are given by a matrix as always.

When it comes to train-test split for semi-supervised learning, the graph structure is kept as a whole and the technique of `train_mask` comes as a handy tool for the feature matrix.

I find the following equations comes as fundamental to GCNs:

$$h_i = \sum_{j \in \tilde{\mathcal{N}}_i} \frac{1}{\sqrt{\deg(i)}\sqrt{\deg(j)}} W^T x_j.$$

Using matrix notations, we have

$$H^{(l+1)} = \sigma(\hat{A} H^{(l)} W^{(l)}),$$

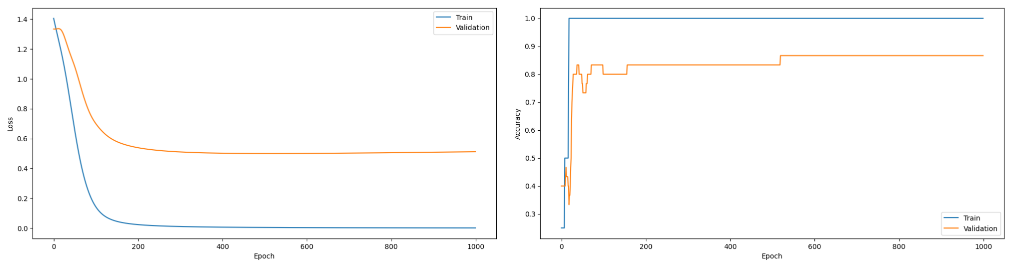
where $\hat{A} = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}$ is the normalized adjacency matrix, and $W^{(l)}$ is of dimension $C^{(l)} \times C^{(l+1)}$. Note that how diffrent nodes are sharing the same weight matrix. (*Question: I guess this is different from the lecture, where the weight matrix is different for the node itself and its neighbors? Are there any advantages?*)

Caveat: default feature matrix

In this problem setting, only the graph structure is used for inference and no predefined node features are given or used. Therefore, in that no information is used from the feature space, why not use a simple feature matrix, as simple as an all-ones vector in the length of $|V|$? (Above anything else, it turned out that an all-zeros vector would not work as it leads to zero gradients.)

The resulting performance was outrageously poor and unfortunately it was after a while when I found out the reason to be the poorly designed feature matrix. Looking back at the equations of GCN, it becomes kind of obvious that with this all-ones feature vector, the model is not able to distinguish between different nodes and the information contained in the graph is reduced to only the degrees of nodes. Instead of this choice, an identity matrix which one-hot encodes the nodes would solve the problem.

Performance



(a) Loss (b) Accuracy

Figure 2: Training Loss and Accuracy

Neural Enhanced Belief Propagation

Introduction and background

- **Motivation:** The author suggests two cons of traditional BP. First, in real world scenarios, we may only have access to a poor approximation of the true distribution of the graphical model, leading to suboptimal estimates. Second, BP is not guaranteed to converge in loopy graphs.

- **Error correction decoding algorithm LDPC:**

$$P(\mathbf{x}) \propto \prod_s \left[\sum_{n \in N(s)} x_n = 0 \mod 2 \right] = \prod_s f_s(\mathbf{x}_s),$$

$$P(\mathbf{r}|\mathbf{x}) = \prod_n P(r_n|x_n),$$

$$P(\mathbf{x}|\mathbf{r}) \propto P(\mathbf{x})P(\mathbf{r}|\mathbf{x}).$$

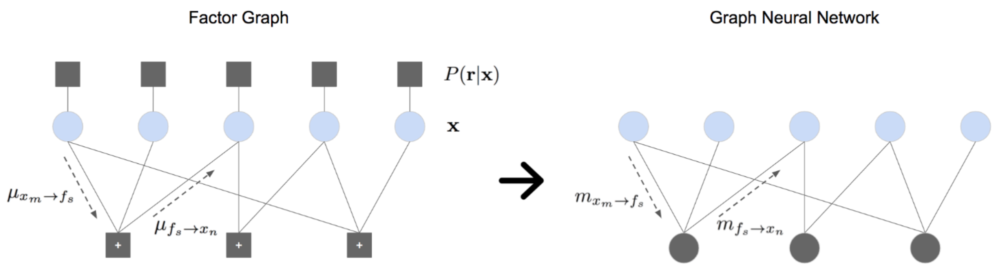


Figure 3: Illustration of the FG-GNN and LDPC

To associate belief propagation with LDPC, it is helpful to think of any factor graphs as a bipartite graph.

Question: When we talk about i.i.d. noises (the second equation above), what are we actually referring to?

- **Belief Propagation formulation:**
Variable to factor message:

$$\mu_{x_m \to f_s}(x_m) = \prod_{l \in N(x_m) \backslash \{f_s\}} \mu_{f_l \to x_m}(x_m);$$

Factor to variable message:

$$\mu_{f_s \to x_n}(x_n) = \sum_{x_s \backslash x_n} f_s(x_s) \prod_{m \in N(f_s) \backslash \{x_n\}} \mu_{x_m \to f_s}(x_m);$$

Marginal estimate:

$$p(x_n) \propto \prod_{s \in N(x_n)} \mu_{f_s \to x_n}(x_n).$$

NEBP formulation

$$\begin{aligned}
\tilde{\mu}_{f\to x}, \tilde{\mu}_{x\to f} &= \mathrm{BP}(\mu_{f\to x}) \\
M^t_{f\to x} &= \text{FG-GNN}(h^t, \tilde{\mu}_{f\to x}, \tilde{\mu}_{x\to f}) \\
\mu^{t+1}_{f\to x} &= \tilde{\mu}_{f\to x} \cdot f_s(M^t_{f\to x}) + f_u(M^t_{f\to x})
\end{aligned}$$

Many technical details to figure out...

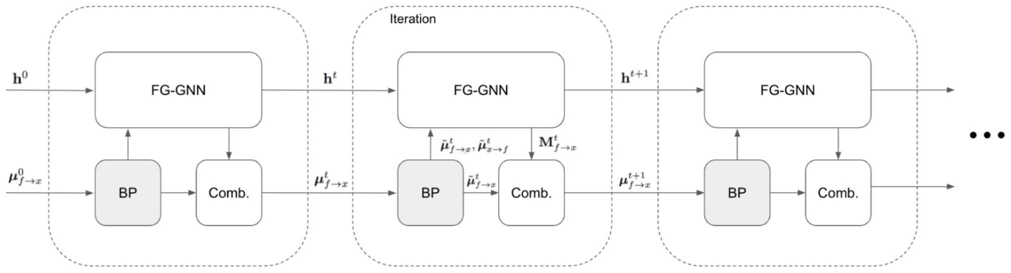

Iteration
$\mathbf{h}^0$
FG-GNN
$\mathbf{h}^t$
FG-GNN
$\mathbf{h}^{t+1}$
FG-GNN
$\boldsymbol{\mu}^0_{f\to x}$
BP
Comb.
$\boldsymbol{\mu}^t_{f\to x}$
$\tilde{\boldsymbol{\mu}}^t_{f\to x}, \tilde{\boldsymbol{\mu}}^t_{x\to f}$
$\mathbf{M}^t_{f\to x}$
BP
$\tilde{\boldsymbol{\mu}}^t_{f\to x}$
Comb.
$\boldsymbol{\mu}^{t+1}_{f\to x}$
BP
Comb.
•••

Figure 4: Illustration of the NEBP