TP3: Graph Neural Networks

omar (dot) darwiche-domingues (at) inria.fr
pierre (dot) perrault (at) inria.fr

December 6, 2019

Abstract

The report and the code are due in 2 weeks (deadline 23:59, 20/12/2019). You will find instructions on how to submit the report on piazza, as well as the policies for scoring and late submissions.

1 Neural Relational Inference

This practical session is based on the paper *Neural Relational Inference for Interacting Systems* by Kipf et al., 2018.

We will use the following material provided by Marc Lelarge and Timothée Lacroix: `https://github.com/timlacroix/nri_practical_session`.

1.1 Motivation and problem formulation

A wide range of dynamical systems can be seen as a group of interacting components. For example, we can think of a set of 2-dimensional particles coupled by springs. Assume that we are given only a set of trajectories of such interacting dynamical system. How can we learn its dynamical model in an unsupervised way?

Formally, we are given as input a set of trajectories of N objects, and each trajectory has length T. Each object i, for $i = 1, \dots, N$, is represented by a vertex v_i. Let $\mathbf{x}_i^t$ be the feature vector of object i at time t (e.g., position and velocity) with dimension D. Let $\mathbf{x}^t = \{\mathbf{x}_1^t, \dots, \mathbf{x}_N^t\}$ be the set of features of all N objects at time t and let $\mathbf{x}_i = (\mathbf{x}_i^1, \dots, \mathbf{x}_i^T)$ be the trajectory of object i. The input data can be stored in a 3-dimensional array $\mathbf{x}$ of shape $N \times T \times D$, denoted by $\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^T)$, such that $\mathbf{x}_{i,t,d}$ is the d-th component of the feature vector of object i at time t.

In addition, we assume that the dynamics can be modeled by a graph neural network (GNN) given an unknown graph $\mathbf{z}$ where $\mathbf{z}_{i,j}$ represents the discrete edge type between objects v_i and v_j.

In this context, we want to learn, simultaneously:

- The edge types $\mathbf{z}_{i,j}$ (**edge type estimation**);
- A model that, for any time t, takes $\mathbf{x}^t$ as input and predicts $\mathbf{x}^{t+1}$ as output (**future state prediction**).

1.2 Model

The Neural Relational Inference (NRI) model consists of:

- An **encoder** that uses trajectories $\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^T)$ to infer pairwise interaction vectors $\mathbf{z}_{i,j} \in \mathbb{R}^K$ for i, j in $\{1, \dots, N\}$, where K is the number of *edge types*.
- A **decoder** that takes $\mathbf{x}^t$ and $\mathbf{z} = \{\mathbf{z}_{i,j}\}_{i,j}$ as input to infer $\mathbf{x}^{t+1}$.

Both the encoder and the decoder are implemented using graph neural networks. For more details, read Section 3 of the paper here.

2 Questions

Complete the code in the following notebook

`https://github.com/timlacroix/nri_practical_session/blob/master/NRI_student.ipynb`

and answer the questions below in your report. **For the report, no code submission is required**. Note that this Github repository contains a `solutions` folder, which you are allowed to use to complete the notebook.

2.1. Explain what are the edge types $\mathbf{z}_{i,j}$.

2.2. In the NRI model, explain how the encoder and the decoder work.

2.3. Explain the LSTM baseline used for joint trajectory prediction. Why is it important to have a "burn-in" phase?

2.4. Consider the training of the LSTM baseline. Notice that the negative log-likelihood is lower after the burn-in than before. Why is this surprising? Why is this happening?

2.5. Consider the problem of trajectory prediction. What are the advantages of the NRI model with respect to the LSTM baseline?

2.6. Consider the training the of NRI model. What do you notice about the edge accuracy during training? Why is this surprising?

2.7. What do you expect to happen with the NRI model when there is no interaction between the objects?