Hybrid Machine Learning Approaches & Applications

Seminar - WS 20/21

Coding Exercise on Graph Neural Network

Michael Feld
Guillermo Reyes
Amr Gomaa
Matthias Klusch

Akshay Joshi
Matrikelnummer: 2581346
s8akjosh@stud.uni-saarland.de

I. Questions

1. **What are GNN and what are its possible applications?**
   Answer: Graph neural networks capture the dependencies in a graph through message passing between the nodes of a graph. In contrast to standard neural networks, GNNs retain a state that can represent information from its neighborhood with arbitrary depth. Major applications are: Modeling Physics System, Predicting Protein Structures, and Drug discovery etc.

2. **What is the difference between GNN, GCN, CNN and FCNN?**
   Answer: General graphs presents certain challenges: they are mostly irregular, there is no ordering to the neighboring nodes and even if they're ordered, they don't carry any semantic meaning. So, traditional connectionist networks like CNN/FCNN would not work. Thus, we need GNN/GCNN to learn features without relying on canonical spatial representations.

3. **How can we insert prior knowledge into the learning process?**
   Answer: Graphs let us impose a relational inductive bias or prior knowledge into the neural model. For an instance, an adjacency matrix representing information of skeletal joints can be inserted as prior in a pose estimation model. This is based on our intuition/knowledge that the pixels for limbs should always be below the pixels for head in general scenarios.

4. **What are the possible methods to implement a GCN?**
   Answer: The major methods are Spectral GCN (introduce filters from the perspective based on graph spectral theory) & Spatial GCN (formulate graph convolutions as aggregation of feature information from neighbors). Other recent methods complementary to GCN are: Graph Attention Networks, Graph Spatio-Temporal Networks, and Graph Auto-Encoders etc.

5. **How does your adj matrix work and why did you choose it?**
   Answer: Standard training methods for GNN usually tend to overfit to the scarce label info in the dataset and using gaussian adj. matrix does not help the model in useful ways (as it instead blurs the overall image). Thus, I have implemented an extremely sparse adj. matrix where each node's features can be randomly dropped either partially or fully. Later, the perturbed adj. matrix is propagated over the input images. As a result, each node is enabled to be insensitive to specific neighborhoods, thus increasing the overall robustness.

II. Results

Method	Accuracy [10 Epochs]
Fully Connected	91%
Convolutional Neural Network	95%
GNN + Predict Edges	89%
GNN + Gaussian Adj. Matrix	91%
GNN + My Adj. Matrix	**92%**