Assignment-2

GMLFA (AI60007) - Autumn,2024 - IIT Kharagpur

Release Date: [23/08/2024]
Submission Date: [13/09/2024]
Total Marks: 21

Instructions:

- All graded questions are compulsory to solve, non-graded questions are optional.
- Each group has to ***submit only one file*** named 'group_number_assignment2.ipynb'.
- ***Negative marking*** will be there as per our ***plagiarism policy*** given in the course webpage.
- You can use any language for coding questions, but ***'python'*** is preferred.
- Frameworks like Pytorch, Tensorflow are encouraged to construct deeper neural network architectures.
- You will be provided with one supporting code notebook (.ipynb) file with pseudocode if required.
- Any required help will be provided to you in the code notebook regarding data or any specific library.

Dataset:

For all the questions asked in this assignment you have to use the **QM9 dataset.**

QM9 Dataset:

The QM9 dataset is a widely used benchmark dataset in the field of graph neural networks (GNNs) and molecular property prediction. It contains about 134,000 small organic molecules with up to 9 heavy atoms (C, O, N, F). Each molecule is represented as a graph, where atoms are nodes and bonds are edges.

Key features of QM9:
- Number of graphs: ~134,000
- Node features: Atom properties (e.g., atomic number, charge)
- Edge features: Bond properties (e.g., bond type)
- Graph labels: Various molecular properties (e.g., energy, dipole moment) as '**regression targets**'. For detailed information please visit the provided data link.

Link:
https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.datasets.QM9.html
The dataset is used for regression tasks, predicting molecular properties from graph structures.

Use Case:

- **We are going to use the first 1000 graphs ([:1000]) for training, 100 graphs ([1000:1100]) for validation and 100 graphs ([1100:1200]) for the test.**
- **We will use the first property 'μ (dipole moment)' which is a continuous value as the target label for the graph, stored at index 0 of targets.**
- You will get the Data-Loaded in the code notebook.
- This is the Regression task so you have to take one label for every graph.

Part (A): [5 marks]

Use the library implementation of following shallow embedding methods to generate the node embeddings and then compute the graph features by averaging all the node features.

- DeepWalk (embedding_dimensions= 64, walk_length=10, num_walks=50)
- Node2Vec (embedding_dimensions= 64, walk_length=10, num_walks=50, p=1, q=0.5)

Now, implement a custom Deep Neural Network for the regression task. [Every graph has one embedding and corresponding label to be predicted]

Report the following:

- Root Mean Square Error (RMSE) Metric for each of the methods in the test set.

Part (B): [8 marks]

Graph Convolutional Network (GCN) with Node Features:

- GCN Layer you have to implement:
  $$H^{(l+1)} = \sigma\left(\widehat{D}^{-1\backslash 2}\widehat{A}\,\widehat{D}^{-1\backslash 2}H^{(l)}W^{(l)}\right)$$
  where $\widehat{A} = A + I$ is the adjacency matrix with added self-loops, $\widehat{D}$ is the degree matrix, $H^{(l)}$ is the node feature matrix at layer lll, and $W^{(l)}$ is the weight matrix.

- **Task:**

- Implement a Graph Convolutional Network (GCN) using the original node features.
- You can try out various aggregators like ‘sum’, ‘mean’ etc to get graph features at the end.
- Show the effect of GCN layers into the learning [use upto 4 GCN layers].
- Perform Regression on the test set and report RMSE.

Part (C): [8 marks]

Attention Mechanism in GNN (EGATConv):

- You have to implement an attention based GNN as given by the following equations.
  - Attention Mechanism: $e_{ij} = LeakyReLU\left(a^T\left[Wh_i \,||\, Wh_j \,||\, W_e e_{ij}\right]\right)$, where W and W_e are the learnable weight metrics , a is a learnable attention function (e.g., an MLP) and || is the concatenation operator.
  - Normalised attention coefficient: $\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k \in N(i)} exp(e_{ik})}$
  - Node Update: $h_i^{(l+1)} = \sigma\left(\sum_{j \in N(i)} \alpha_{ij} W^{(l)} h_j^{(l)}\right)$
- Task:
  - Implement an attention-based GNN, incorporating the concepts of the Edge-Weighted Graph Attention Network (EGATConv)
  - You can try out various aggregators like ‘sum’, ‘mean’ etc to get graph features at the end.
  - Show the effect of EGATConv layers into the learning [use upto 4 layers].
  - Perform Regression on the test set and report RMSE.