Homework 3: Learning on Sequence and Graph Data

Deep Learning (84100343-0)
Spring 2022
Tsinghua University

1 Part One: RNN and Transformer

1.1 Background: Language Modeling

Language modeling is a central task in NLP and language models can be found at the heart of speech recognition, machine translation, and many other systems. In this homework, you are required to solve a language modeling problem by designing and implementing recurrent neural networks (RNNs), as well as Transformers. A language model is a probability distribution over sequences of words. Given such a sequence $(\mathbf{x}_1, ..., \mathbf{x}_m)$ with length m, it assigns a probability $P(\mathbf{x}_1, ..., \mathbf{x}_m)$ to the whole sequence. In detail, given a vocabulary dictionary of words $(\mathbf{v}_1, ..., \mathbf{v}_m)$ and a sequence of words $(\mathbf{x}_1, ..., \mathbf{x}_t)$, a language model predicts the following word $\mathbf{x}_{t+1}$ by modeling: $P(\mathbf{x}_{t+1} = \mathbf{v}_j | \mathbf{x}_1, ..., \mathbf{x}_t)$ where $\mathbf{v}_j$ is a word in the vocabulary dictionary. Conventionally, we evaluate our language model in terms of perplexity (PP) https://en.wikipedia.org/wiki/Perplexity. Note that, $\mathrm{PP} = \exp(\text{Average Cross Entropy Loss})$.

1.2 Dataset

- We use the Penn Treebank [6] dataset. This dataset have two parts: training set and validation set. The directory structure consists of two parts:
  - "`./src/`" contains the start code.
  - "`./data/`" contains the train set and the valid set.

1.3 Tasks and Scoring

You need to finish the following tasks on the given dataset:

- **Task A**: Construct a kind of standard RNN (*e.g.* LSTM[3], GRU[1], etc.) and train your model from scratch using the recommended deep learning framework. Note that fine-tuning from a large-scale pre-trained model is forbidden. Validate your model on the *valid* set, and report training and validation curves. [**15pts**]
- **Task B**: Construct the standard Transformer (Attention is All You Need[7]) and train your model from scratch using the recommended deep learning framework. Still, fine-tuning from a large-scale pre-trained model is forbidden. Validate your model on the *valid* set, and report training and validation curves. [**15pts**]

- **Data Preparation**:
  - We have provided the data preparation code "`data.py`". By running "`python main.py`", you will see how the data is preprocessed. Please summarize the differences between preprocessing text data for language modeling with preprocessing image data for object recognition. [**10pts**]
- **Technical Details**:
  - As we have learned in class, mask is used in Transformer to keep it autoregressive. Please explain why mask is necessary in Transformer and how it is implemented in your code. [**10pts**]

- **Attention Visualization**:
  - Please visualize the attention values of some typical words and check whether the Transformer can learn meaningful attention values. [**10pts**]
- **Extra Techniques**:
  - Please adopt an extra technique you find in other materials to further improve your model. Please explain why you choose it, check whether it works on our given dataset and give a convincing reason. [**5pts**]
  - The transformer consists of the multi-head attention, feed-forward network, and Layernorm. Please do the ablation study to analyze which part of the transformer is more important to the final performance of the transformer. [**5pts**]

2 Part Two: GNN and Node2Vec

2.1 Background: Dataset Split in Node Classification Task

In Part One, each data point is an independent sentence. Hence, it is easy to split the dataset (e.g. 80% as training set and the remaining 20% as validation set). While in a graph, different nodes are **not independent** because of message passing. In this case, there are two options.

- Transductive setting: the input graph can be observed in all the dataset splits (training, validation and test set). In other words, we only split the labels.
- Inductive setting: we break the edges between splits to get multiple graphs. These graphs are thus independent.

2.2 Dataset and Library

- We adopt the citation network dataset *Cora* from [10]. In a citation network, nodes represent documents and edges represent citation links.
- We will use Pytorch Geometric (PyG) in this assignment. You can install it according to the tutorial. Besides, it's **strongly** recommended to read through PyG's documentation and code.

2.3 Tasks and Scoring

You need to finish the following tasks on the given dataset:

- **Task A**: We provide implementations of **GCN**[4], **GAT**[8] and **Node2Vec**[2] for you. Read through and run `gcn.py`, `gat.py`, `node2vec.py`, report the performance of these methods. [**10pts**]
- **Task B**: Implement **DeepGCN**[5] **or GIN**[9] (You **only** need to implement **one** of them to get full grades) based on PyG, report the performance. (**Hint**: PyG has implemented basic layers for you) [**20pts**]

3 What you should submit

Please submit your *code and report* as an Archive (zip or tar). The document is supposed to cover your **insights** of the proposed model, the **technical details**, the **experimental results**, and the necessary **references**.

References

[1] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NeurIPS*, 2015.

[2] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.

[3] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[4] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[5] G. Li, M. Muller, A. Thabet, and B. Ghanem. Deepgcns: Can gcns go as deep as cnns? In *ICCV*, 2019.
[6] M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Using Large Corpora*, 1994.
[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.
[8] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. In *ICLR*, 2018.
[9] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.
[10] Z. Yang, W. Cohen, and R. Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.