Worksheet #1
April 16th, 2020

Paper: Relational inductive biases, deep learning, and graph networks

1. What is the meaning of Inductive Bias? Be formal.

   In machine learning, the inductive bias is associated with assumptions that algorithms use to learn the target function that will predict new inputs. Ideally, we would like to find an unbiased generalization procedure that does not give preference to a model over the other, however, due to large search spaces and, often, due to the nature of a problem (for example, preference to reduce number of false negatives), bias need to be introduced. Moreover, according to Mitchell [2],

   > *Unbiased generalization programs that use consistency with the training instances as their only source of information, cannot outperform programs that use rote learning. Additional information or biases are therefore critical to the ability to classify instances that are not identical to the training instances.*

   Although necessary for some cases, biases need to be used criteriously.

2. Given that you understood the meaning of inductive bias, what is the meaning of **Relational** Inductive Bias?

   Relational Inductive Bias are **relational** features added to the background knowledge that help an algorithm to learn a target function. For example, knowledge bases allow for the introduction of first order rules that describe a particular domain in a relational way. This knowledge may help to learn other new knowledge.

3. Visit the Google Knowledge Graph API web page. Do you think that the API can be used to **learn** new knowledge? In other words, only

using that API could you **predict new links** in the graph? If so, give an example.

Knowledge Graph is based on Freebase [1], a well known knowledge base bought by Google [1]. Therefore, we can only query the knowledge graph for information already stored. The system itself does not perform any learning, although we could implement algorithms to learn relations from the data stored in the knowledge graph.

4. Visit the DeepMind Graph Nets web page. Go through the example to find paths in the graph. How do you think the algorithm works? This library is built on top of Tensorflow. What kind of representation is used for the input graphs? How do you think the shortest paths are represented in order that the network can use them as ground truth for training?

   If you digged the python code (`https://github.com/deepmind/graph_nets/blob/master/graph_nets/graphs.py`) you may have noticed that there is a complete description of the structures used to represent the graph, which is transcribed here:

```
"""A class that defines graph-structured data.
The main purpose of the `GraphsTuple` is to represent multiple graphs with
different shapes and sizes in a way that supports batched processing.
This module first defines the string constants which are used to represent
graph(s) as tuples or dictionaries: `N_NODE, N_EDGE, NODES, EDGES, RECEIVERS,
SENDERS, GLOBALS`.
This representation could typically take the following form, for a batch of
`n_graphs` graphs stored in a `GraphsTuple` called graph:
  - N_NODE: The number of nodes per graph. It is a vector of integers with shape
    `[n_graphs]`, such that `graph.N_NODE[i]` is the number of nodes in the i-th
    graph.
  - N_EDGE: The number of edges per graph. It is a vector of integers with shape
    `[n_graphs]`, such that `graph.N_EDGE[i]` is the number of edges in the i-th
    graph.
  - NODES: The nodes features. It is either `None` (the graph has no node
    features), or a vector of shape `[n_nodes] + node_shape`, where
    `n_nodes = sum(graph.N_NODE)` is the total number of nodes in the batch of
    graphs, and `node_shape` represents the shape of the features of each node.
```

[1] in the news: https://www.cloudave.com/140/google-buys-freebase-this-is-huge/

The relative index of a node from the batched version can be recovered from the ‘graph.N_NODE‘ property. For instance, the second node of the third graph will have its features in the ‘1 + graph.N_NODE[0] + graph.N_NODE[1]‘-th slot of graph.NODES.
Observe that having a ‘None‘ value for this field does not mean that the graphs have no nodes, only that they do not have node features.

- EDGES: The edges features. It is either ‘None‘ (the graph has no edge features), or a vector of shape ‘[n_edges] + edge_shape‘, where ‘n_edges = sum(graph.N_EDGE)‘ is the total number of edges in the batch of graphs, and ‘edge_shape‘ represents the shape of the features of each edge.
  The relative index of an edge from the batched version can be recovered from the ‘graph.N_EDGE‘ property. For instance, the third edge of the third graph will have its features in the ‘2 + graph.N_EDGE[0] + graph.N_EDGE[1]‘-th slot of graph.EDGES.
  Observe that having a ‘None‘ value for this field does not necessarily mean that the graph has no edges, only that they do not have edge features.
- RECEIVERS: The indices of the receiver nodes, for each edge. It is either ‘None‘ (if the graph has no edges), or a vector of integers of shape ‘[n_edges]‘, such that ‘graph.RECEIVERS[i]‘ is the index of the node receiving from the i-th edge.
  Observe that the index is absolute (in other words, cumulative), i.e. ‘graphs.RECEIVERS‘ take value in ‘[0, n_nodes]‘. For instance, an edge connecting the vertices with relative indices 2 and 3 in the second graph of the batch would have a ‘RECEIVERS‘ value of ‘3 + graph.N_NODE[0]‘.
  If ‘graphs.RECEIVERS‘ is ‘None‘, then ‘graphs.EDGES‘ and ‘graphs.SENDERS‘ should also be ‘None‘.
- SENDERS: The indices of the sender nodes, for each edge. It is either ‘None‘ (if the graph has no edges), or a vector of integers of shape ‘[n_edges]‘, such that ‘graph.SENDERS[i]‘ is the index of the node sending from the i-th edge.
  Observe that the index is absolute, i.e. ‘graphs.RECEIVERS‘ take value in ‘[0, n_nodes]‘. For instance, an edge connecting the vertices with relative indices 1 and 3 in the third graph of the batch would have a ‘SENDERS‘ value of ‘1 + graph.N_NODE[0] + graph.N_NODE[1]‘.
  If ‘graphs.SENDERS‘ is ‘None‘, then ‘graphs.EDGES‘ and ‘graphs.RECEIVERS‘ should also be ‘None‘.
- GLOBALS: The global features of the graph. It is either ‘None‘ (the graph has no global features), or a vector of shape ‘[n_graphs] + global_shape‘ representing graph level features.

Regarding the algorithm, we have discussed about it during class. But, the best way of understanding it is to dig the code.

References

[1] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD 08, page 12471250, New York, NY, USA, 2008. Association for Computing Machinery.

[2] Tom M. Mitchell. The need for biases in learning generalizations. Technical report, Rutgers University, New Brunswick, NJ, 1980.