EECS 182 Deep Neural Networks
Spring 2023 Anant Sahai

Discussion 7

1. Beam Search

When making predictions with an autoregressive sequence model, it can be intractable to decode the true most likely sequence of the model, as doing so would require exhaustively searching the tree of all $O(M^T)$ possible sequences, where M is the size of our vocabulary, and T is the max length of a sequence. We could decode our sequence by greedily decoding the most likely token each timestep, and this can work to some extent, but there are no guarantees that this sequence is the actual most likely sequence of our model.

Instead, we can use beam search to limit our search to only candidate sequences that are the most likely so far. In beam search, we keep track of the k most likely predictions of our model so far. At each timestep, we expand our predictions to all of the possible expansions of these sequences after one token, and then we keep only the top k of the most likely sequences out of these. In the end, we return the most likely sequence out of our final candidate sequences. This is also not guaranteed to be the true most likely sequence, but it is usually better than the result of just greedy decoding.

The beam search procedure can be written as the following pseudocode:

Algorithm 1 Beam Search

```
for each time step $t$ do
    for each hypothesis $y_{1:t-1,i}$ that we are tracking do
        find the top $k$ tokens $y_{t,i,1}$,...,$y_{t,i,k}$
    end for
    sort the resulting $k^2$ length $t$ sequences by their total log-probability
    store the top $k$
    advance each hypothesis to time $t+1$
end for
```

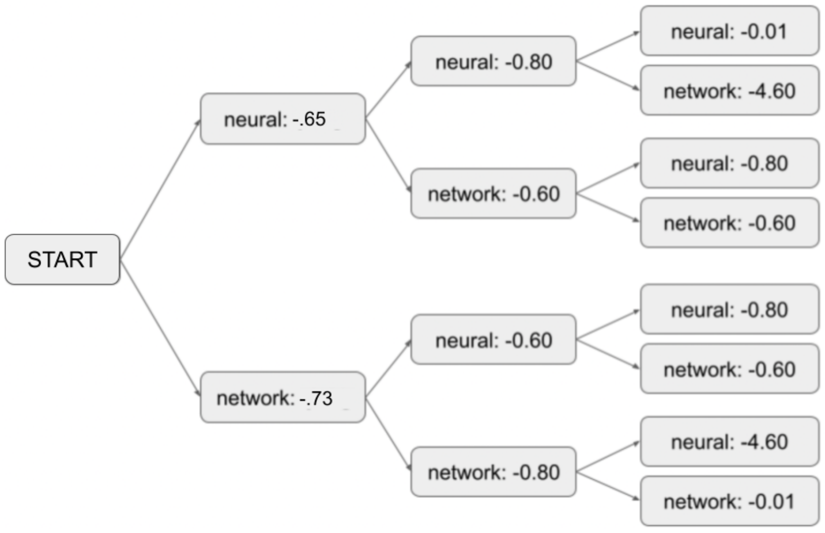

Figure 1: The numbers shown are the decoder's log probability prediction of the current token given previous tokens.

We are running the beam search to decode a sequence of length 3 using a beam search with $k = 2$. Consider predictions of a decoder in Figure 1, where each node in the tree represents the next token **log probability** prediction of one step of the decoder conditioned on previous tokens. The vocabulary consists of two words: "neural" and "network".

(a) **At timestep 1, which sequences is beam search storing?**

(b) **At timestep 2, which sequences is beam search storing?**

(c) **At timestep 3, which sequences is beam search storing?**

(d) **Does beam search return the overall most-likely sequence in this example? Explain why or why not.**

(e) **What is the runtime complexity of generating a length-T sequence with beam size k with an RNN?** Answer in terms of T and k and M. (Note: an earlier version of this question said to write it in terms of just T and k. This answer is also acceptable.)

2. Graph Neural Network Conceptual Questions

Diagrams in this question were taken from `https://distill.pub/2021/gnn-intro`. This blog post is an excellent resource for understanding GNNs and contains interactive diagrams:

A. You are studying how organic molecules break down when heated. For each molecule, you know the element and weight of each atom, which other atoms its connected to, the length of the bond the atoms, and the type of molecule it is (carbohydrate, protein, etc.) You are trying to predict which bond, if any, will break first if the molecule is heated.

   (a) How would you represent this as a graph? (What are the nodes, edges, and global state representations? Is it directed or undirected?)

(b) How would you use the outputs of the last GNN layer to make the prediction?

(c) How would you encode the node representation?

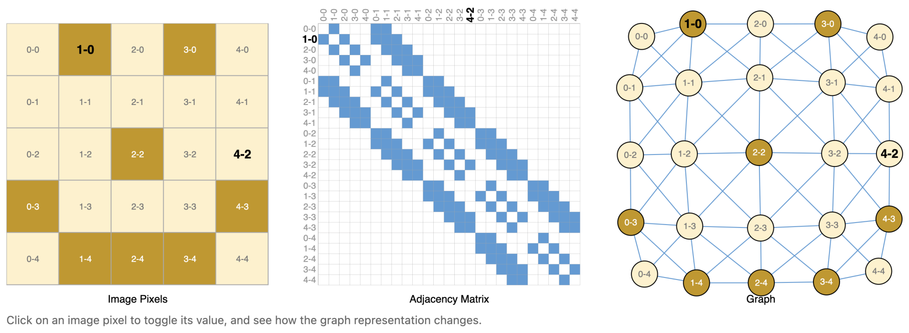
Image Pixels
Adjacency Matrix
Graph
Click on an image pixel to toggle its value, and see how the graph representation changes.

Figure 2: Images as Graphs

B. If you're doing a graph-level classification problem, but node values are missing for some of your graph nodes, how would you use this graph for prediction?

C. Consider the graph neural net architecture shown in Figure 3. It includes representations of nodes (V_n), edges (E_n), and global state (U_n). At each timestep, each node and edge is updated by aggregating neighboring nodes/edges, as well as global state. The global state is the updated by pooling all nodes and edges. For more details on the architecture setup, see https://distill.pub/2021/gnn-intro/#passing-messages-between-parts-of-the-graph.

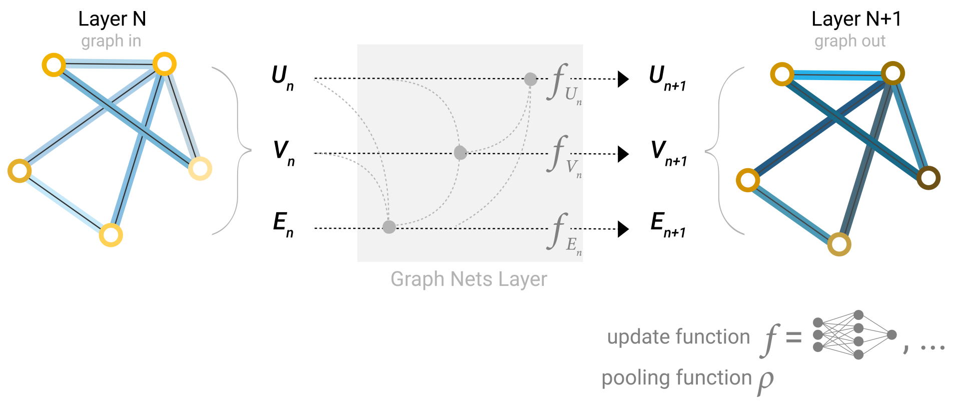

Figure 3: GNN architecture

(a) If we double the number of nodes in a graph which only has node representations, how does this change the number of learned weights in the graph? How does it change the amount of computation used for this graph if the average node degree remains the same? What if the graph if fully connected? (Assume you are not using a global state representation).

(b) Where in this network are learned weights incorporated?

(c) The diagram provided shows undirected edges. How would you incorporate directed edges?

(d) The differences between an MLP and a RNN include (a) weights are shared between timesteps and (b) data is fed in one timestep at a time. How would you make an MLP-like GNN? What about an RNN-like GNN?

D. Let's say you run a large social network and are trying to predict whether individuals in the network like a particular ad. Each individual is represented as a node in the graph, and we are doing a node-level prediction problem. You have labels for around half of the people in the network and are trying to predict the other half. Unlike a traditional ML problem, where there there are many independent training points, here we have a single social network. How would you do prediction on this graph?

E. (Optional) Play around with different GNN design choices in the GNN playground at `https://distill.pub/2021/gnn-intro/#gnn-playground`. Which design choices lead to the best AUC?

Contributors:

- CS 182 Staff from past semesters.
- Olivia Watkins.
- Kumar Krishna Agrawal.
- Dhruv Shah.