EECS 182 Deep Neural Networks
Fall 2022 Anant Sahai Discussion 8

1. Autoencoders (Conceptual)

State whether each of the statements below is True or False and explain why. If the type of autoencoder is not specified, you may consider all autoencoder types (vanilla, denoising, masked, etc.)

(a) There is no point in checking autoencoder reconstruction performance on a validation set because we will ultimately evaluate whether representations are useful by training on downstream tasks.

(b) If you train two different autoencoder variants on the same dataset, the one which produces lower validation loss will perform better on the downstream task.

(c) The autoencoder decoder is not used after pretraining.

(d) Using autoencoder representations can can sometimes produce worse performance on a downstream task than using raw inputs.

(e) Autoencoder representations can be useful even if the representation was trained on a very different dataset than the downstream task.

(f) Using an autoencoder representation (rather than using raw inputs) is most useful when you have few labels for your downstream task.

(g) With images, it is often more effective to mask patches than to mask individual pixels.

(h) We can think of masked and denoising autoencoders as vanilla autoencoders with data augmentation applied.

(i) If you trained an autoencoder with noise or masking, you should also apply noise/masking to inputs when using the representations for downstream tasks.

(j) Autoencoders always encode inputs into fixed-size lower-dimensional representations.

2. Beam Search

This problem will also be covered on the homework.

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.

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
```

The beam search procedure can be written as the following pseudocode ($y_{t,i,k}$ is the kth token at timestep t generated from hypothesis i. $y_{1:t-1}, i$ means hypothesis i is a string of tokens for timesteps 1 through $t-1$.)

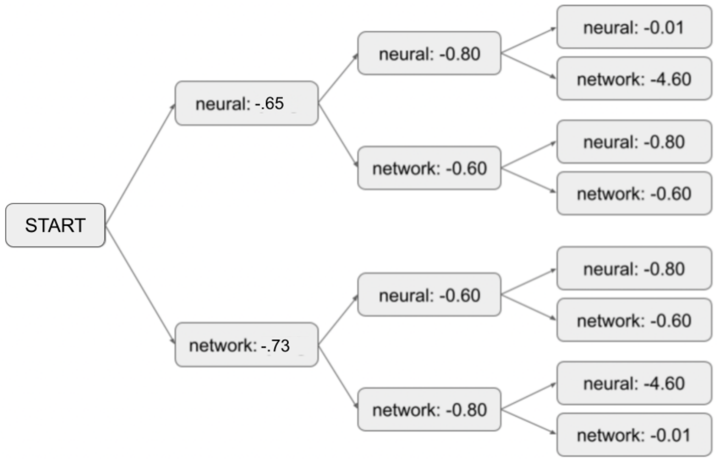

Figure 1: The numbers shown are the decoder's log probability prediction of the current token given previous tokens.

We are running beam search to decode a sequence of length 3 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 vocab 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? 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. You may assume that $M \geq K$, and that every time you need to compute the top K you do it by sorting the array and taking the last K.

(f) **What information needs to be stored for each of the k hypotheses in beam search with an RNN?**

Contributors:

- Olivia Watkins.
- CS 182 Staff from past semesters.