Lecture 8 Quiz

Quiz, 7 questions

7/7 points (100%)

✓ **Congratulations! You passed!**

Next Item

✓ 1 / 1 points

1.
Imagine that we have a fully trained RNN that uses multiplicative connections as explained in the lecture. It's been trained well, i.e. we found the model parameters with which the network performs well. Now we want to convert this well-trained model into an equivalent model with a different architecture. Which of the following statements are correct?

☐ We can use the additive input model proposed in the lecture (see the page about "An obvious recurrent neural net" in the video about multiplicative connections), with a modification inspired by what we saw in the older language models: instead of connecting the input character directly to the hidden state at the next time step, we use a different vector representation for each of the 86 input characters, and we connect that vector representation directly to the hidden state at the next time step. If the vector has as many elements as there are factors in the original model, this will be flexible enough that an equivalent model can always be built.

Un-selected is correct

☐ We can use the model where the input character chooses the whole hidden-to-hidden weight matrix, but only if there aren't too many factors in the multiplicative model. If there are too many factors, then the multiplicative model will have more parameters than this alternative model, which means that there will not always be an equivalent model of this alternative form.

Un-selected is correct

Lecture 8 Quiz

Quiz, 7 questions

7/7 points (100%)

☐ We can use the model where the input character chooses the whole hidden-to-hidden weight matrix. It can be more difficult to train, but it's sufficiently flexible.

Correct
The multiplicative factors effectively create a different hidden-to-hidden matrix for each input character.

☐ We can use the additive input model proposed in the lecture (see the page about "An obvious recurrent neural net" in the video about multiplicative connections). It can be more difficult to train, but it's sufficiently flexible.

Un-selected is correct

✓ 1 / 1 points

Lecture 8 Quiz

Quiz, 7 questions

7/7 points (100%)

2.

The multiplicative factors described in the lecture are an alternative to simply letting the input character choose the hidden-to-hidden weight matrix. Let's carefully compare these two methods of connecting the current hidden state and the input character to the next hidden state.

Suppose that all model parameters (weights, biases, factor connections if there are factors) are between -1 and 1, and that the hidden units are logistic, i.e. their output values are between 0 and 1. Normally, not all neural network model parameters are between -1 and 1 (although they typically end up being between -100 and 100), but for this question we simplify things and say that they are between -1 and 1.

For the simple model, this restriction on the parameter size and hidden unit output means that the largest possible contribution that hidden unit #56 at time t can make to the input (i.e. before the logistic) of hidden unit #201 at time $t+1$ is 1, no matter what the input character is. This happens when the hidden-to-hidden weight matrix chosen by the input unit has a value of 1 for the connection from #56 to #201, and hidden unit #56 at time t is maximally activated, i.e. its state (after the logistic) is 1. Those two get multiplied together, for a total contribution of 1.

Let's say that our factor model has 1000 factors and 1500 hidden units. What is the largest possible contribution that hidden unit #56 at time t can possibly make to the input (i.e. before the logistic) of hidden unit #201 at time $t+1$, in this factor model, subject to the same restriction on parameter size and hidden unit output?

1000

Correct Response

A total of 1 per factor. There are 1000 factors, so the answer is 1000.

✓ 1 / 1 points

Lecture 8 Quiz

Quiz, 7 questions

7/7 points (100%)

3.

The multiplicative factors described in the lecture are an alternative to simply letting the input character choose the hidden-to-hidden weight matrix. In the

lecture, it was explained that that simple model would have 86 x 1500 x 1500 = 193 500 000 parameters, to specify how the hidden units and the input

character at time t influence the hidden units at time $t+1$. How many parameters does the model with the factors have for that same purpose,

i.e. for specifying how the hidden units and the input character at time t influence the hidden units at time $t+1$? Let's say that there are

1500 hidden units, 86 different input characters, and 1500 factors.

4629000

Correct Response

Each factor has connections to all of the 86 input characters, to the hidden units at time t, and to the hidden units at time $t+1$, for a total of $86+1500+1500=3086$ parameters per factor. The total is $3086 \cdot 1500 = 4629000$.

✓ 1 / 1 points

4.

In the lecture, you saw some examples of text that Ilya Sutskever's model generated, after being trained on Wikipedia articles. If we ask the model to generate a couple of sentences of text, it quickly becomes clear that what it's saying is not something that was actually written in Wikipedia. Wikipedia articles typically make much more sense than what this model generates. Why doesn't the model generate significant portions of Wikipedia articles?

[✓ That would have been overfitting, which was carefully avoided. The model has to generalize well.]

Correct

○

It should learn to generate whole Wikipedia articles, eventually, with more training. However, that might require more compute power than is available nowadays (the model had to be trained for a month already, on a very fast computer).

○ Basic calculations about the size of the hidden state vector show that the model can never learn to reliably generate any fixed string of text that's more than 38 characters long (38 is the square root of 1500, the number of hidden units).

✓ 1 / 1 points

5.
Recall that Neural Networks for language modeling often learn word representation vectors, to avoid the need to have a weight from every possible input word to every hidden unit. Can we use the same learning method in an Echo State Network?

[✓ No]

Correct
That would require backpropagating through time, which is exactly what ESN's don't do.

○ Yes

✓ 1 / 1 points

6.
Visualizing what a Neural Network has learned is often difficult. Here are two ideas for how to generate text from Ilya Sutskever's language model. Which one will best show what the model has learned?

○ Give it the beginning of a sentence, e.g. "Once upon a time, ", and ask it which character it thinks will come next. That's a probability distribution over the 86 possible characters. As the next character in our sentence-under-construction, we use the character that the model considers most likely. Append that to the partial sentence, and repeat the procedure to get the next character. Repeat this until we have several pages of text.

Lecture 8 Quiz

Quiz, 7 questions

7/7 points (100%)

[✓ Give it the beginning of a sentence, e.g. "Once upon a time, ", and ask it which character it thinks will come next. That's a probability distribution over the 86 possible characters. Pick a character according to that distribution. Append that to the partial sentence, and repeat the procedure to get the next character. Repeat this until we have several pages of text.]

Correct

If we go for the most probable character, we'd probably see the model repeating the same sentence over and over again. A probability distribution is better visualized by samples from it than by showing only what has the highest probability.

✓ 1 / 1 points

7.

In Echo State Networks, does it matter whether the hidden units are linear or logistic (or some other nonlinearity)?

[✓ Yes. With linear hidden units, the output would become a linear function of the inputs, and we typically want to learn nonlinear functions of the input. Therefore, linear hidden units are a bad choice.]

Correct

No. The hidden units are not learned, so the usual Neural Network concerns don't apply here.