Scalable Machine Learning and Deep Learning - Review Questions 4

By

Akhil Yerrapragada (akhily@kth.se)

Gibson Chikafa (chikafa@kth.se)

Group Name – gibson_akhil

1. **1 point.** What's the vanishing problem in RNN?

Ans: During Backpropagation through time, when the computed gradient gets so small that it does not contribute to learning (weight) we can say it is a vanishing gradient problem in RNN.

---

2. **1 point.** Explain the impact of different gates in LSTM.

Ans:

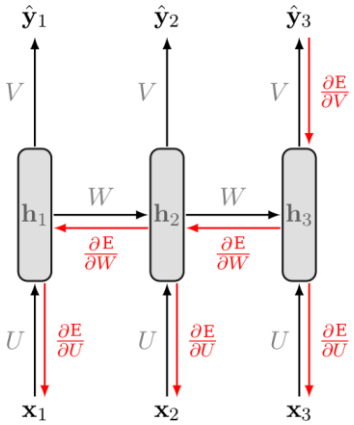
Forget Gate: The forget gate is responsible in deciding which information to keep or ignore. It contains a sigmoid function that takes a combined input of previous hidden state and current input. The output will be of range 0 to 1. Values near 0 can be forgotten and values near 1 can be kept. This output is then pointwise multiplied with the previous state.

Input Gate: It contains sigmoid and tanh functions where each takes a combined input of previous hidden state and current input. Here we use sigmoid to decide which values need to be updated and tanh fits the output of given input in the range of -1 to 1. Now we pointwise multiply tanh output with sigmoid output and finally, pointwise add the result to already modified (In Forget Gate) previous state (Form now, this is the new state).

Output Gate: This is used to produce next hidden state. It contains sigmoid and tanh functions. The sigmoid takes the combined input of previous hidden state and current input whereas tanh takes the input of new state. We pointwise multiply the outputs received from both the activation functions and use it as the hidden function in the next time step.

---

3. **1 point.** Assume the error of the following network is $E = E^{(1)} + E^{(2)}$, then compute the $\frac{\partial E}{\partial u}$.



Ans:

Given $E = E^{(1)} + E^{(2)}$

$\partial E/\partial u = X\ t\ \partial E^{(t)}/\partial u = \partial E^{(1)}/\partial u + \partial E^{(2)}/\partial u$

$\partial E^{(1)}/\partial u = \partial E^{(1)}/\partial y\hat{}^{(1)} * \partial y\hat{}^{(1)}/\partial z^{(1)} * \partial z^{(1)}/\partial h^{(1)} * \partial h^{(1)}/\partial s^{(1)} * \partial s^{(1)}/\partial u$

$\partial E^{(2)}/\partial u = \partial E^{(2)}/\partial y\hat{}^{(2)} * \partial y\hat{}^{(2)}/\partial z^{(2)} * \partial z^{(2)}/\partial h^{(2)} * \partial h^{(2)}/\partial s^{(2)} * \partial s^{(2)}/\partial u + \partial E^{(2)}/\partial y\hat{}^{(2)} * \partial y\hat{}^{(2)}/\partial z^{(2)} * \partial z^{(2)}/\partial h^{(2)} * \partial h^{(2)}/\partial s^{(2)} * \partial s^{(2)}/\partial h^{(1)} * \partial h^{(1)}/\partial s^{(1)} * \partial s^{(1)}/\partial u$

1 point. Explain how to use a seq-to-seq model for language translation.

Ans:

Let us consider English to French translation.

Using 2 multilayered (4 layers) LSTMs, one for the input sequence and the other for output sequence. The first LSTM maps the input sequence to a vector and the second LSTM obtains a translated sequence from the vector. One factor to consider here is that the length of output sequence may differ from the input sequence.

First, we train the model with sentence pairs. The training is done by maximizing the log probability of T and S. Here, T refers to correct translation and S refers to Source sentence.

$$1/|\mathcal{S}| \sum_{(T,S)\in\mathcal{S}} \log p(T|S)$$

Translations ($\hat{T}$) can be produced post training by finding most likely translation for a source sentence S which is represented in the below equation.

$$\hat{T} = \arg\max_{T} p(T|S)$$

Most likely translations can be obtained using left-to-right beam search decoder. It maintains small number (B) of partial hypothesis. Here, partial hypothesis is a prefix of some translation. For each time step we extend partial hypothesis with all possible words in the vocabulary. Among these we consider only most likely translations based on the log probability and add them to complete hypothesis set.

The analysis is carries out based on the research paper - Sequence to Sequence Learning with Neural Networks by Ilya Sutskever, Oriol Vinyals, Quoc V. Le

---

4. **1 point.** Briefly explain the attention and self-attention mechanisms.

Ans: In attention mechanism, weights are computed as a scaled dot-product between queries and keys. Here, query determines the values that need to be focused. Queries, keys and values are considered as vectors in attention. We apply softmax on the dot-product to obtain the probability distribution. The output from attention is weighted sum of values.

In self-attention, we use the embedding vectors to create queries, keys and values. Here, we use projection matrices that makes the model learnable. The models learn by projecting input to the embedding space. We now take the required information that makes necessary representations of the embedding vector from the projection matrix and run the attention mechanism over the queries, keys and values.