Deep Learning for NLP 2020
Exercise 08 Solution

July 12, 2020

1 Pingo

- What are components of GRUs?
  - ☐ Dropout Gate
  - ☑ Reset Gate
  - ☑ Update Gate
  - ☐ Gradient Clipping Gate
- How do GRUs differ from traditional RNNs?
  - ☑ They have better control over the impact of the input on the memory
  - ☑ They can overwrite the full memory if needed
  - ☑ They can, in theory, store dependencies to previous inputs which were fed arbitrarily far away in the past

2 RNN Extensions

1. What is the benefit of bidirectional RNNs over unidirectional RNNs? Explain in up to two sentences and give an example.

   Solution:

   In general: By adding an RNN which processes the input sequence in reverse, one improves the representation of inputs which are influenced by other inputs further down in the sequence. However, the full sequence needs to be known in advance. A case where bidirectionality is beneficial are, for example, adjective modifiers in French which can appear before or after a noun. In the sentence `"le vieux chien noir"` (the old black dog), the representation of `chien` would not include the adjective `noir`.

2. What is the benefit of adding "output connections" to RNNs? Explain in up to two sentences and give an example.

   Solution:

   Without output connections, the output y_t at each time step t is chosen independently of other outputs. When using output connections, one can choose outputs y_t depending on other outputs.

   A case where output connections are beneficial are generation tasks, for example machine translation. Compared to choosing the most likely output token individually at each time step, results improve significantly if one chooses the most likely output *sequence* based on the probabilities produced by the softmax activation at each step.

3 Theoretical Background of Vanishing Gradients

In the lecture on backpropagation, we derived the formula for the derivative of the loss function with respect to a neuron p_i in an MLP as:

$$\frac{\partial E}{\partial p_i} = \sum_j \frac{\partial E}{\partial y_j} \cdot \sigma'(z_j) \cdot w_{i,j}$$

If we refer to a neuron p_i in layer k as $p_i^{(k)}$, this becomes:

$$\frac{\partial E}{\partial p_i^{(k)}} = \sum_j \frac{\partial E}{\partial p_j^{(k+1)}} \cdot \sigma'\left(z_j^{(k+1)}\right) \cdot w_{i,j}^{(k+1)}$$

1. Unfold one iteration of backpropagation: Write $\frac{\partial E}{\partial p_r^{(k-1)}}$ as a function of $\frac{\partial E}{\partial p_j^{(k+1)}}$ by using the formula given above.

   Solution:

$$\begin{aligned}
\frac{\partial E}{\partial p_r^{(k-1)}} &= \sum_i \frac{\partial E}{\partial p_i^{(k)}} \cdot \sigma'\left(z_i^{(k)}\right) \cdot w_{r,i}^{(k)} \\
&= \sum_i \left(\sum_j \frac{\partial E}{\partial p_j^{(k+1)}} \cdot \sigma'\left(z_j^{(k+1)}\right) \cdot w_{i,j}^{(k+1)} \right) \cdot \sigma'\left(z_i^{(k)}\right) \cdot w_{r,i}^{(k)} \\
&= \sum_{i,j} \frac{\partial E}{\partial p_j^{(k+1)}} \cdot \sigma'\left(z_j^{(k+1)}\right) \cdot \sigma'\left(z_i^{(k)}\right) \cdot w_{i,j}^{(k+1)} \cdot w_{r,i}^{(k)}
\end{aligned}$$

2. Without applying the formula another time: How would $\frac{\partial E}{\partial p_s^{(k-2)}}$ depend on σ'?

   Solution It would be a function of $(\sigma')^3$.

3. Based on your findings in a) and b): Which activation function would you recommend for an MLP with more than 3 hidden layers? What would happen if you chose a less suited activation function?

   Solution In the hidden layers near the input layer of the network, we have values of $(\sigma')^M$, considering we have M hidden layers. When σ' is small, then this very quickly converges to zero. Specificially, the sigmoid has a derivative that is never higher than 0.3 and thus particularly suffers from this problem. Thus, the gradients of the loss w.r.t. the weights is very small or close to zero for all weights in almost all layers except, possibly, the hidden layers near the output layer. Thus, no learning happens in these layers and the net will get stuck at its initial weight choices. The recommendation is to use an activation function with gradients close to 1 (particularly for deep networks), i.e. ReLU or its variants.