Recurrent Neural Networks LSTM: Part-2

Short presentation on update of LSTM layer design and doubts.

Presentation Outline

- ***Forward Propagation*** - covers how forward pass in layer has been designed.
- ***Backward Propagation*** - covers how backward pass in layer has been designed.
- ***Discussion and doubts regarding layer design*** - issues which are preventing me from achieving my goal.

Forward Propagation

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t tanh(c_t)
\end{aligned}
$$

- The term highlighted in green box is referred to as *candidate value.*
- Each gate can be thought of as separate neural network.

Reference - LSTM: A Search Space Odyssey: https://arxiv.org/pdf/1503.04069.pdf

Example of working input gate of LSTM cell.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$$

```
//_____________________________________________________________________________
template <typename Architecture_t>
auto inline TBasicLSTMLayer<Architecture_t>::InputGate(const Matrix_t &input, Matrix_t &di)
-> void
{
   /*! Computes input gate values according to equation:
    *  input = act(W_input . input + W_state . state + bias)
    *  activation function: sigmoid. */
   const DNN::EActivationFunction fAF = this->GetSigmoidActivationFunction();
   Matrix_t tmpState(fInputValue.GetNrows(), fInputValue.GetNcols());
   Architecture_t::MultiplyTranspose(tmpState, fHiddenState, fWeightsInputGateState);
   Architecture_t::MultiplyTranspose(fInputValue, input, fWeightsInputGate);
   Architecture_t::ScaleAdd(fInputValue, tmpState);
   Architecture_t::AddRowWise(fInputValue, fInputGateBias);
   DNN::evaluateDerivative<Architecture_t>(di, fAF, fInputValue);
   DNN::evaluate<Architecture_t>(fInputValue, fAF);
}
```

Backward Propagation through time (BPTT)

$$
\begin{aligned}
\delta out_t &= \Delta_t + \Delta out_t \\
\delta state_t &= \delta out_t \odot o_t \odot (1 - \tanh^2(state_t)) + \delta state_{t+1} \odot f_{t+1} \\
\delta a_t &= \delta state_t \odot i_t \odot (1 - a_t^2) \\
\delta i_t &= \delta state_t \odot a_t \odot i_t \odot (1 - i_t) \\
\delta f_t &= \delta state_t \odot state_{t-1} \odot f_t \odot (1 - f_t) \\
\delta o_t &= \delta out_t \odot \tanh(state_t) \odot o_t \odot (1 - o_t) \\
\delta x_t &= W^T \cdot \delta gates_t \\
\Delta out_{t-1} &= U^T \cdot \delta gates_t
\end{aligned}
$$

Fig 1. *Derivative value of each internal gate during backpropagation in LSTM cell.*

$$
\begin{aligned}
\delta W &= \sum_{t=0}^{T} \delta gates_t \otimes x_t \\
\delta U &= \sum_{t=0}^{T-1} \delta gates_{t+1} \otimes out_t \\
\delta b &= \sum_{t=0}^{T} \delta gates_{t+1}
\end{aligned}
$$

Fig 2. *The final updates to internal parameters.*

- a(t) represents candidate gate values obtained at timestep ‘t’ during forward pass.
- state(t) represents **cell state** value at timestep ‘t’.

Reference - LSTM: A Search Space Odyssey: https://arxiv.org/pdf/1503.04069.pdf

Current work of LSTM Layer

- **Forward Propagation**:
  - Forward propagation feature is complete with tests are passing.
- **Backpropagation through time**:
  - Parameters update is done for full network after successful forward pass.
  - Most of the work is similar to RNN.
  - Final parameter updates: input-weights (W), state-weights (U) and biases (b) has been implemented in *RecurrentPropagation.cxx*
  - Implemented gradient flow logic but not working.
- **Issue preventing me from achieving goals**:
  - In current gradient flow design, **segmentation fault** occurs during backpropagation testing.
  - In forward pass feature, **output-gate** is resulting in a **zero matrix.** Not able to figure out the reason behind it.
  - Hence a discussion might be required for layer design.

Thank You :)