Multiplicative LSTM for sequence modelling

(Krause et al. 2016)

Consider the Elman RNN:

$$h^{(t)} = tanh(W_{hh}h^{(t-1)} + W_{ih}x^{(t)} + b)$$

and to be specific, we focus only on the pre-nonlinearity update to the previous hidden state, ignoring bias for brevity:

$$\hat{h}^{(t)} = W_{hh}h^{(t-1)} + W_{ih}x^{(t)}$$

The change to the previous hidden state is twofold – a linear transformation by a parameter matrix W_{hh}, which is *independent* of the current input x_t followed by the translation by adding the input vector x_t, which was beforehand projected into the hidden space by W_{ih}.

The hidden state of the RNN has a task to summarize information from the input sequence. We can imagine this is done by W_{hh} *allocating* space in the hidden vector through nulling irrelevant dimensions, or the information from the transformed input $W_{ih}x^{(t)}$ *forcefully* taking its space from the hidden vector by overwriting. Nevertheless, it is easy to see that the W_{hh} matrix has an incredibly difficult task – it is supposed to be able to free up space for the next sequence element *without knowing* what that element is, while keeping relevant information for any possible sequence.

A simple fix to this is to parametrize the transition matrix W_{hh} - instead of using a single matrix, we could use a W_{hhi}, where i is the vocabulary index of the input element $x^{(t)}$. The hidden state would then be updated dependent on the current input, simplifying the learning process. However, the space complexity easily explodes, as the input vocabulary sizes in most practical models are larger than tens of thousands, and it is not feasible to use such a large tensor.

References

Krause, Ben, Liang Lu, Iain Murray, and Steve Renals. 2016. "Multiplicative Lstm for Sequence Modelling." *arXiv Preprint arXiv:1609.07959*.