Project Proposal

Simas Glinskis

February 16, 2017

Abstract

Does batch normalization within the LSTM improve performance or training time?

Experiment

The experiment to test this proposal would be conducted in two parts.

First, the LSTM would be run to establish a baseline for performance and accuracy. Next, a normalization layer would be applied to the carry signal before computing the output of the LSTM cell.

The normalization layer would be applied after the carry update for the first run, and then before the Tanh computation for a second.

The intuition is that applying a normalization layer to the concatenated inputs would have no effect, as the data is normalized. Therefore, a normalization would have the most impact on C_{t+1} or H_{t+1}. For example,

$$C_{t+1} \rightarrow Norm(C_{t+1})$$

$$H_{t+1} \rightarrow H_{t+1} = \sigma(W_O[X_t, H_t]) * Norm(\tanh(W_H C_{t+1}))$$

Finally, accuracy and performance metrics would be compared based on the rate of decrease for the loss, time to completion, and rate of growth for the accuracy.

Outcomes

I do not think the normalization layers will improve the accuracy of the LSTM model. If anything it will only increase the computation time.

Normalization layers are used in CNN models because of their depth and constant weight multiplications. The normalization layers attempt to keep all activations within the active region of the nonlinearities, which prevents zero gradients.

In the LSTM model, the weights stay constant during computation through time, and the data continues to feed the model, which is assumed to be normalized. The carry is the only signal which may benefit from normalization, but I think the gates and diversions keep the signal active.

If the LSTM model is attempting to compute an extremely long time series, the normalization may increase the performance, but as the data is staying constant and the model is changing, it may be hard to see this feature.