



W



T



F



M



A



T



E





W



T



F



M



A



T



W





Long Short-Term Memory

- Long Short-Term Memory

Sepp Hochreiter and Jürgen Schmidhuber. *Neural Computation*, 1997

- Learning to Forget: Continual Prediction with LSTM

Felix A. Gers, Jürgen Schmidhuber, Fred Cummins. *Neural Computation*, 2000

David R. Morrison

The Goal

- Implementing memory in a neural network
 - Long term: weight changes
 - Short term: recent input events
 - Why?
-
-

Solutions?

- Recurrent Networks
 - Back-Propagation Through Time
 - Real-Time Recurrent Learning



Problems

- No clear advantages over feed-forward backprop
 - What to put in short-term memory?
 - Error Signals
 - Vanish (hard to bridge long time lags)
 - Explode (oscillating weights)
-
-

Long Short-Term Memory

- Enforces constant error flow
- Stores information in memory cells
 - What information to store
 - When to release information



Network Architecture



Experiments

- Embedded Reber Grammar
 - Noise-Free/Noisy Sequence
 - Noise and Signal on Same Channel
 - Adding Problem
 - Multiplication Problem
 - Temporal Order
-
-

Fixing LSTM

- Forget Gates
 - Learns when stored information is no longer necessary
- Experimental results



How it learns

Cell block c_j 's truncated derivatives are:

$$\frac{\partial y^{in_j}(t)}{\partial w_{lm}} = f'_{in_j}(net_{in_j}(t)) \frac{\partial net_{in_j}(t)}{\partial w_{lm}} \approx_{tr} \delta_{in_j l} f'_{in_j}(net_{in_j}(t)) y^m(t-1). \quad (12)$$

$$\frac{\partial y^{out_j}(t)}{\partial w_{lm}} = f'_{out_j}(net_{out_j}(t)) \frac{\partial net_{out_j}(t)}{\partial w_{lm}} \approx_{tr} \delta_{out_j l} f'_{out_j}(net_{out_j}(t)) y^m(t-1). \quad (13)$$

$$\frac{\partial s_{c_j^v}(t)}{\partial w_{lm}} = \frac{\partial s_{c_j^v}(t-1)}{\partial w_{lm}} + \frac{\partial y^{in_j}(t)}{\partial w_{lm}} g(net_{c_j^v}(t)) + y^{in_j}(t) g'(net_{c_j^v}(t)) \frac{\partial net_{c_j^v}(t)}{\partial w_{lm}} \approx_{tr} \quad (14)$$

$$\begin{aligned} & (\delta_{in_j l} + \delta_{c_j^v l}) \frac{\partial s_{c_j^v}(t-1)}{\partial w_{lm}} + \delta_{in_j l} \frac{\partial y^{in_j}(t)}{\partial w_{lm}} g(net_{c_j^v}(t)) + \\ & \delta_{c_j^v l} y^{in_j}(t) g'(net_{c_j^v}(t)) \frac{\partial net_{c_j^v}(t)}{\partial w_{lm}} = \\ & (\delta_{in_j l} + \delta_{c_j^v l}) \frac{\partial s_{c_j^v}(t-1)}{\partial w_{lm}} + \delta_{in_j l} f'_{in_j}(net_{in_j}(t)) g(net_{c_j^v}(t)) y^m(t-1) - \\ & \delta_{c_j^v l} y^{in_j}(t) g'(net_{c_j^v}(t)) y^m(t-1). \end{aligned}$$

$$\begin{aligned} \frac{\partial y^{c_j^v}(t)}{\partial w_{lm}} &= \frac{\partial y^{out_j}(t)}{\partial w_{lm}} h(s_{c_j^v}(t)) + h'(s_{c_j^v}(t)) \frac{\partial s_{c_j^v}(t)}{\partial w_{lm}} y^{out_j}(t) \approx_{tr} \\ & \delta_{out_j l} \frac{\partial y^{out_j}(t)}{\partial w_{lm}} h(s_{c_j^v}(t)) + (\delta_{in_j l} + \delta_{c_j^v l}) h'(s_{c_j^v}(t)) \frac{\partial s_{c_j^v}(t)}{\partial w_{lm}} y^{out_j}(t). \end{aligned}$$

To efficiently update the system at time t , the only (truncated) derivatives that need to be calculated at time $t-1$ are $\frac{\partial s_{c_j^v}(t-1)}{\partial w_{lm}}$, where $l = c_j^v$ or $l = in_j$.

$$\frac{\partial net_{in_j}(t)}{\partial y^m(t-1)} \approx_{tr} 0 \quad \forall m$$

$$\frac{\partial net_{out_j}(t)}{\partial y^m(t-1)} \approx_{tr} 0 \quad \forall m$$

$$\frac{\partial s_{c_j^v}(t)}{\partial y^m(t-1)} \approx_{tr} 0 \quad \forall m$$

and

Therefore we get

$$\frac{\partial y^{in_j}(t)}{\partial y^m(t-1)} = f'_{in_j}(net_{in_j}(t)) \frac{\partial net_{in_j}(t)}{\partial y^m(t-1)} \approx_{tr} 0 \quad \forall m$$

$$\frac{\partial y^{out_j}(t)}{\partial y^m(t-1)} = f'_{out_j}(net_{out_j}(t)) \frac{\partial net_{out_j}(t)}{\partial y^m(t-1)} \approx_{tr} 0 \quad \forall m$$

and

$$\frac{\partial y^{c_j^v}(t)}{\partial y^m(t-1)} = \frac{\partial y^{out_j}(t)}{\partial y^m(t-1)} h(s_{c_j^v}(t)) + \frac{\partial y^{out_j}(t)}{\partial y^m(t-1)} h'(s_{c_j^v}(t)) \frac{\partial s_{c_j^v}(t)}{\partial y^m(t-1)} \approx_{tr} 0 \quad \forall m$$

This implies for all w_{lm} not on connections to c_j , (in_j , out_j) that is, $l \notin \{c_j^v, in_j, out_j\}$:

$$\frac{\partial y^{c_j^v}(t)}{\partial w_{lm}} = \sum_{l' \in \{in_j, out_j\}} \frac{\partial y^{c_j^v}(t)}{\partial y^{l'}(t-1)} \frac{\partial y^{l'}(t-1)}{\partial w_{lm}} \approx_{tr} 0$$

The truncated derivatives of output unit k are:

$$\frac{\partial y^k(t)}{\partial w_{lm}} = f'(net_k(t)) \left(\sum_{l' \in \{in_k, out_k\}} w_{kl'} \frac{\partial y^{l'}(t-1)}{\partial w_{lm}} + h_k y^m(t-1) \right) \approx_{tr} \quad (15)$$

$$f'(net_k(t)) \left(\sum_{j \in \{1, \dots, k\}} \delta_{c_j^v} w_{kj} \frac{\partial y^{c_j^v}(t-1)}{\partial w_{lm}} + \sum_{j \in \{1, \dots, k\}} (h_{in_j} + h_{out_j}) \sum_{l' \in \{in_j, out_j\}} w_{jl'} \frac{\partial y^{l'}(t-1)}{\partial w_{lm}} + \right.$$

$$\left. \sum_{l' \in \{in_k, out_k\}} w_{kl'} \frac{\partial y^{l'}(t-1)}{\partial w_{lm}} + h_k y^m(t-1) \right) \approx_{tr}$$

$$f'(net_k(t)) \begin{cases} y^m(t-1) & l = k \\ w_{kj} \frac{\partial y^{c_j^v}(t-1)}{\partial w_{lm}} & l = c_j^v \\ \sum_{l' \in \{in_j, out_j\}} w_{jl'} \frac{\partial y^{l'}(t-1)}{\partial w_{lm}} & l = in_j \text{ OR } l = out_j \\ \sum_{l' \in \{in_k, out_k\}} w_{kl'} \frac{\partial y^{l'}(t-1)}{\partial w_{lm}} & l \text{ otherwise} \end{cases}$$

Learning Rules

- Forward Pass
 - Backward Pass and Weight Update
 - Output learning, neuron learning, output gate learning
 - Cell learning, forget gate learning, input gate learning
-
-

What does it all mean?

- LSTM is a unique network architecture that shows a lot of promise for learning how to remember things.
 - It is quickly growing in popularity, used successfully in areas of music composition, speech recognition, protein structure analysis, and others
-
-