Long Short-Term Memory

Managing Long-Term Dependencies within Sequences

August 25, 2013

Outline

Recurrent Neural Networks

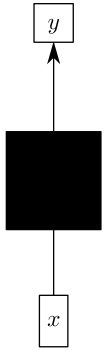
Long Short-Term Memory Cells

Example

Output layer for text recognition

Classification task

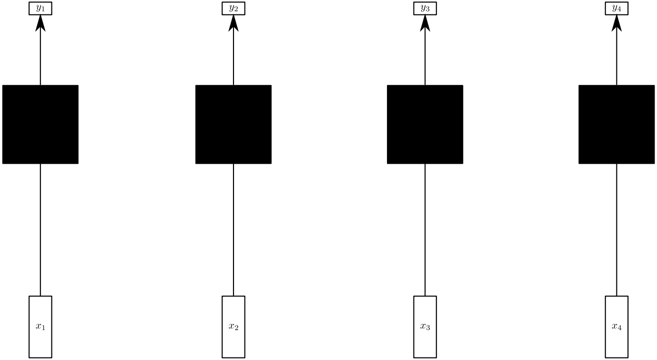
mapping x to y



$$\mathcal{F}: x \in \mathbb{X} \rightarrow y \in \mathbb{Y}$$

Sequence classification task

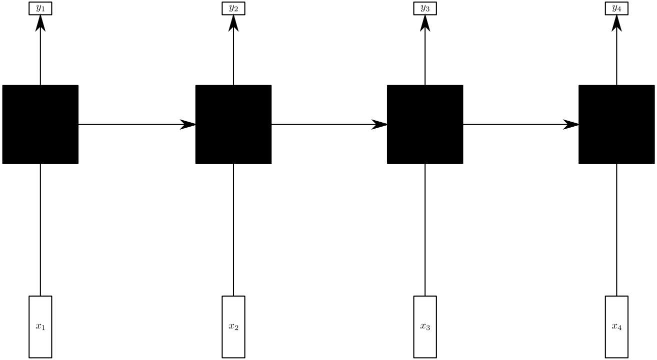
mapping $x_1 x_2 \ldots$ to $y_1 y_2 \ldots$



$$\mathcal{F} : \boldsymbol{x} \in \mathbb{X}^* \rightarrow \boldsymbol{y} \in \mathbb{Y}^*$$

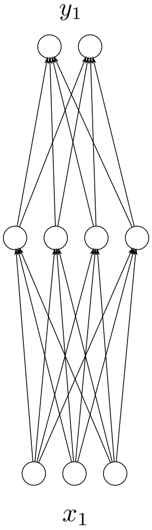
Sequence classification task with dependencies

mapping $x_1 x_2 \ldots$ to $y_1 y_2 \ldots$

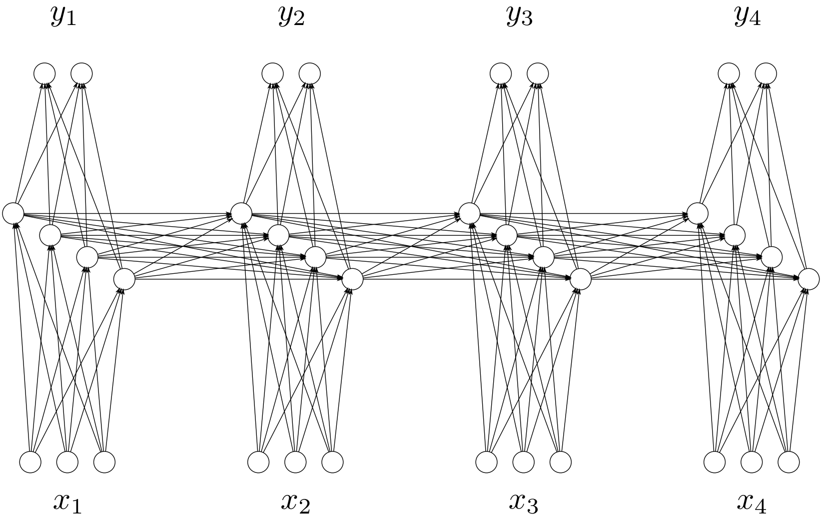


$$\mathcal{F} : \boldsymbol{x} \in \mathbb{X}^* \rightarrow \boldsymbol{y} \in \mathbb{Y}^*$$

Feed-forward Neural Networks



Recurrent Neural Networks

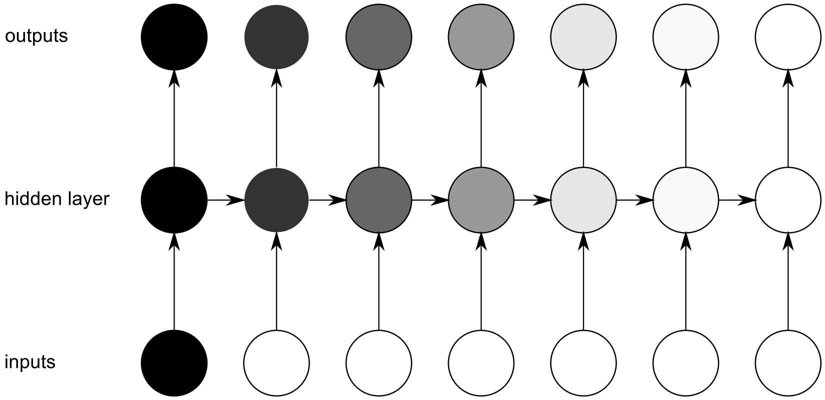


Problems with simple RNN architectures

Problems with simple RNN architectures

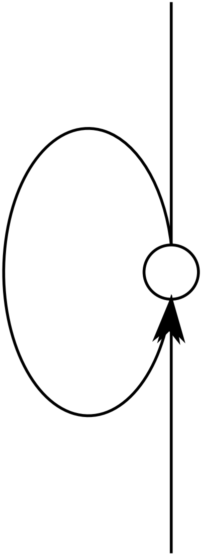
- Vanishing gradient in training
  - Sensitivity to an input at time t decreases rapidly
- Only limited context effectively usable

Vanishing gradient problem



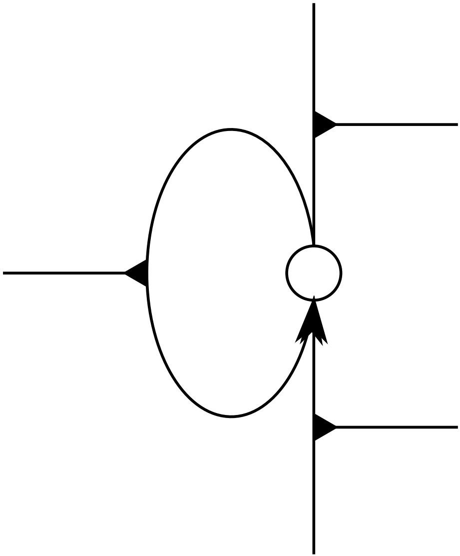
Constructing a differentiable memory cell

Start point



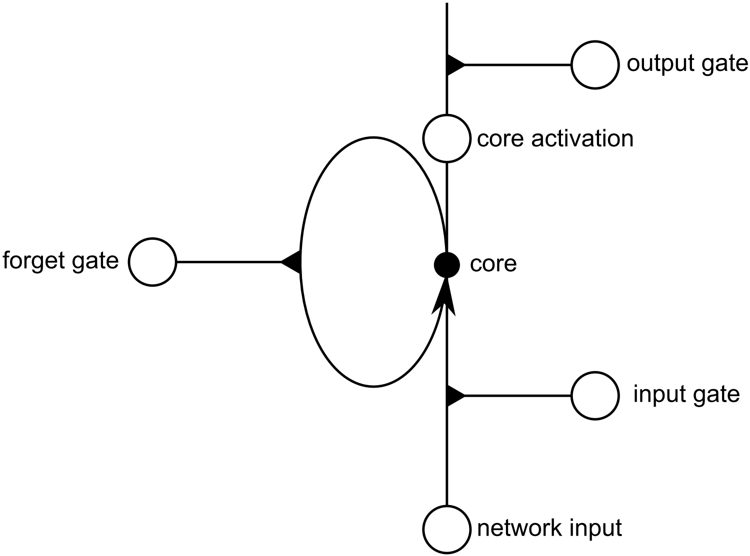
Constructing a differentiable memory cell

Adding controls



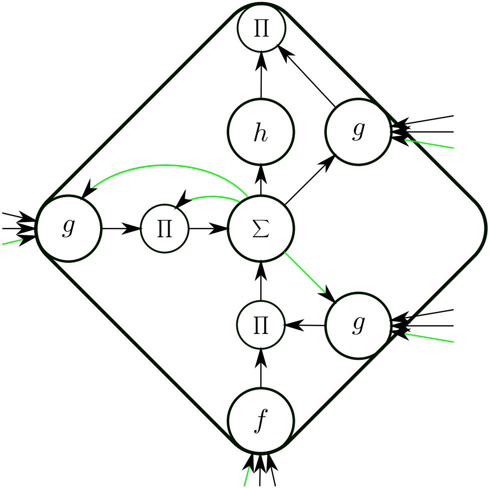
Constructing a differentiable memory cell

Let the controls be neurons



Constructing a differentiable memory cell

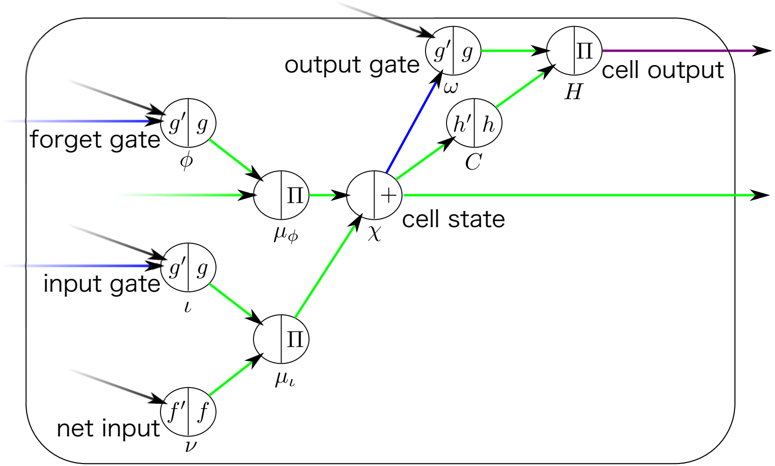
The final result



green lines are time-delayed connections

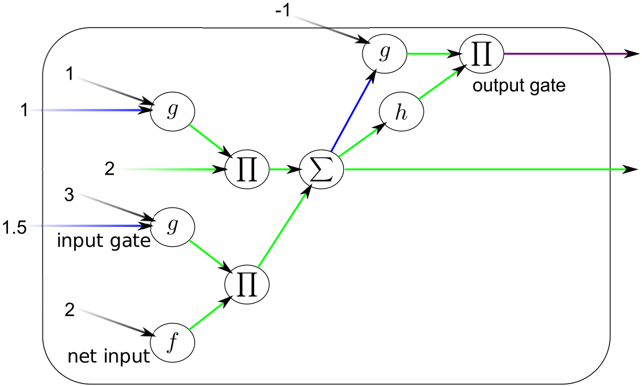
Constructing a differentiable memory cell

The final result



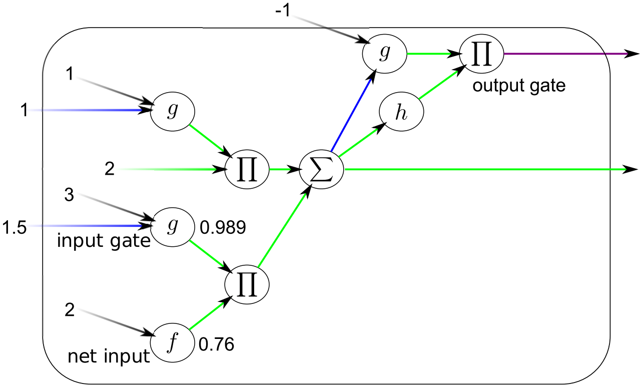
green lines are connection with a fixed weight = 1

Example



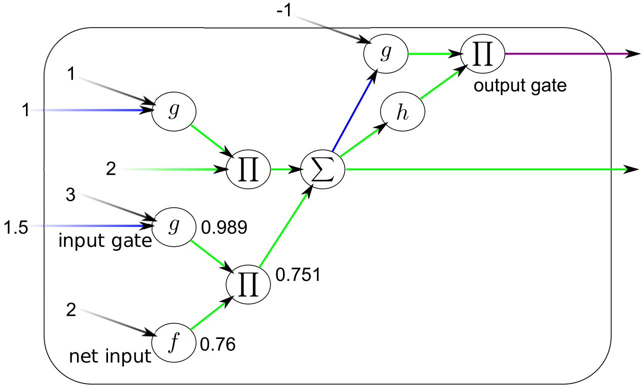
$$
\begin{aligned}
g(x) &= (1+\exp(-x))^{-1} \\
f(x) &= 2\,(1+\exp(-x))^{-1} - 1 \\
h(x) &= 2\,(1+\exp(-x))^{-1} - 1
\end{aligned}
$$

Example



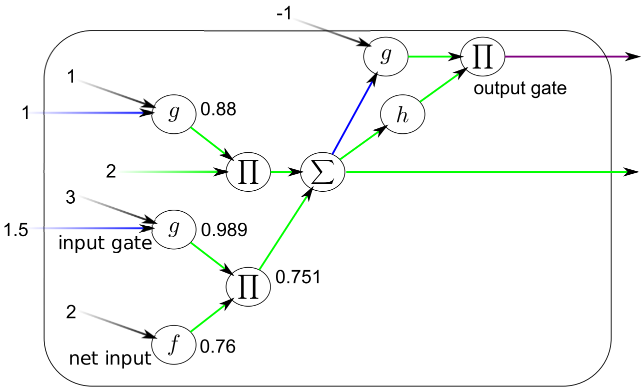
$$
\begin{aligned}
g(x) &= (1+\exp(-x))^{-1} \\
f(x) &= 2\,(1+\exp(-x))^{-1} - 1 \\
h(x) &= 2\,(1+\exp(-x))^{-1} - 1
\end{aligned}
$$

Example



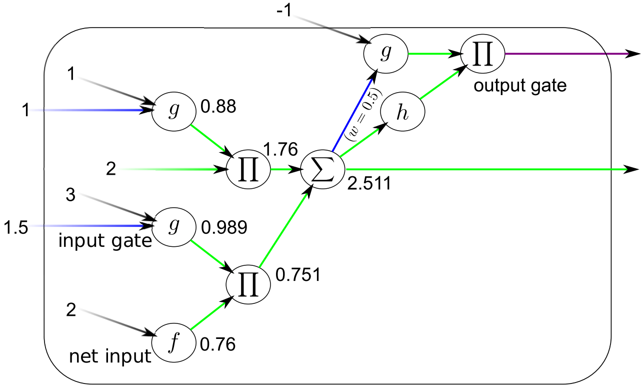
$$
\begin{aligned}
g(x) &= (1+\exp(-x))^{-1} \\
f(x) &= 2\,(1+\exp(-x))^{-1} - 1 \\
h(x) &= 2\,(1+\exp(-x))^{-1} - 1
\end{aligned}
$$

Example



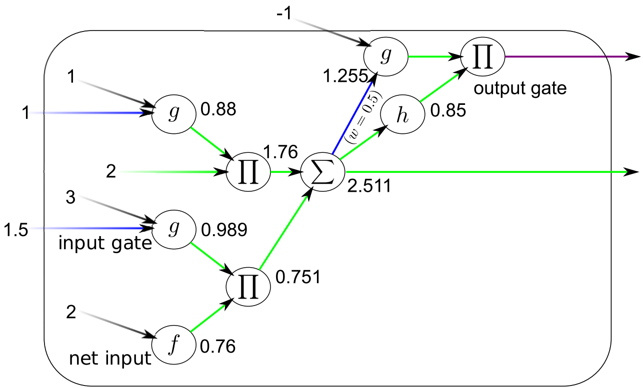
$$
\begin{aligned}
g(x) &= (1+\exp(-x))^{-1} \\
f(x) &= 2\,(1+\exp(-x))^{-1} - 1 \\
h(x) &= 2\,(1+\exp(-x))^{-1} - 1
\end{aligned}
$$

Example



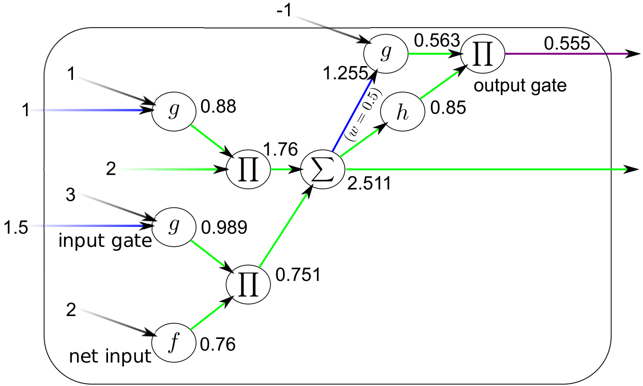
$$
\begin{aligned}
g(x) &= (1 + \exp(-x))^{-1} \\
f(x) &= 2\,(1 + \exp(-x))^{-1} - 1 \\
h(x) &= 2\,(1 + \exp(-x))^{-1} - 1
\end{aligned}
$$

Example



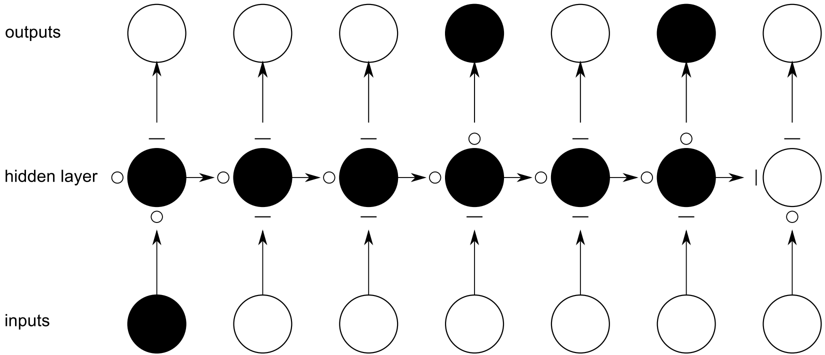
$$
\begin{aligned}
g(x) &= (1 + \exp(-x))^{-1} \\
f(x) &= 2\,(1 + \exp(-x))^{-1} - 1 \\
h(x) &= 2\,(1 + \exp(-x))^{-1} - 1
\end{aligned}
$$

Example

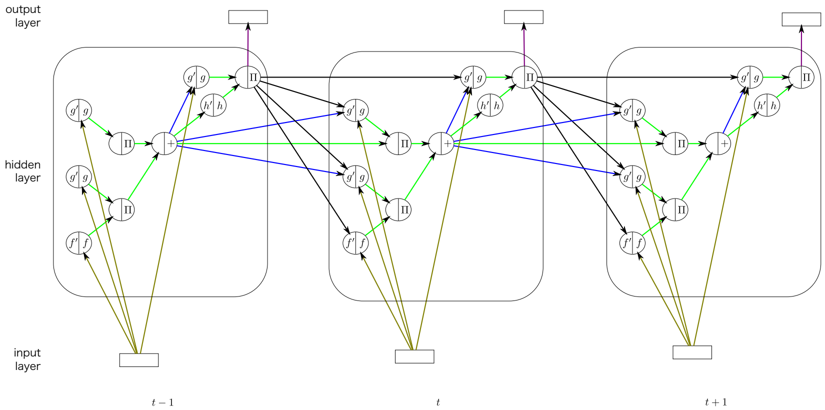


$$
\begin{aligned}
g(x) &= (1 + \exp(-x))^{-1} \\
f(x) &= 2\,(1 + \exp(-x))^{-1} - 1 \\
h(x) &= 2\,(1 + \exp(-x))^{-1} - 1
\end{aligned}
$$

LSTM preserving information



Complete Architecture

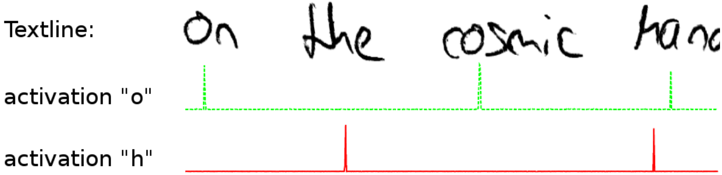


Output Layer

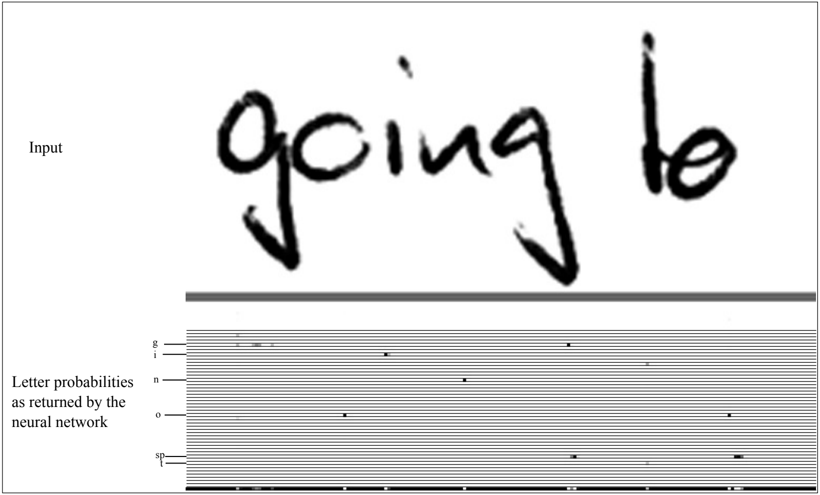
- The output layer can be any arbitrary output layer
- For text recognition, one output node is associated to each recognizable label
- The output layer is usually normalized via *softmax*

$$y_c = \frac{\exp(a_c)}{\sum_{c'} \exp(a_{c'})}$$

- An extra node, the blank or ε-node, in the output layer can be used as a default output

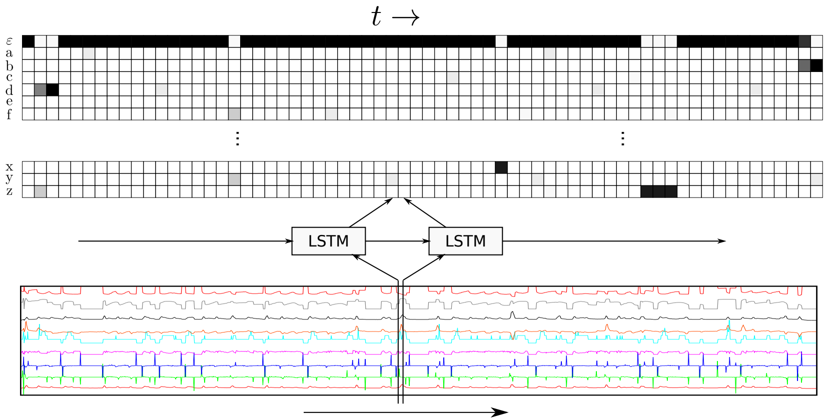


Real Word Example



Complete Architecture

The entire LSTM neural network for text recognition



Training a Neural Network

The process for training a (already initialized) neural network is

- take a new training example $(\boldsymbol{x}, \boldsymbol{z})$
- Let the network produce an output $\boldsymbol{y} = NN(\boldsymbol{x})$
- Compute the error $E(\boldsymbol{y}, \boldsymbol{z})$
- Compute the derivative of the error wrt. to every weight $\delta w = \frac{\partial E(\boldsymbol{x},\boldsymbol{y})}{\partial w}$
- Change each weight to reduce the error $w \leftarrow w - \eta \cdot \delta w$

Summary

LSTM

- Recurrent neural network
- Gates control the flow of information
- Differentiable version of a memory cell
- No vanishing gradient problem
- Trainable to consider long-term dependencies

Forward Pass

- Simple addition and multiplication operations
- Runs in $O\left(|\boldsymbol{x}| \cdot \left(|\mathrm{LSTM}|^2 + |x| \cdot |\mathrm{LSTM}| + |\mathrm{LSTM}| \cdot |y|\right)\right)$

Backward Pass

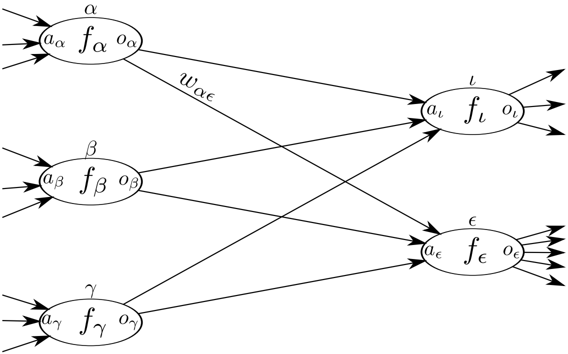
- Standard Back-propagation
- Complete unfolding in time feasible
- Different error functions possible, according to task

Time for Questions

Appendix: Back-propagation Training Formulas

Back-propagation

Error gradient wrt. weight



$$\frac{\partial E}{\partial a} = \delta_a$$

$$\frac{\partial E}{\partial w} = \delta_w$$

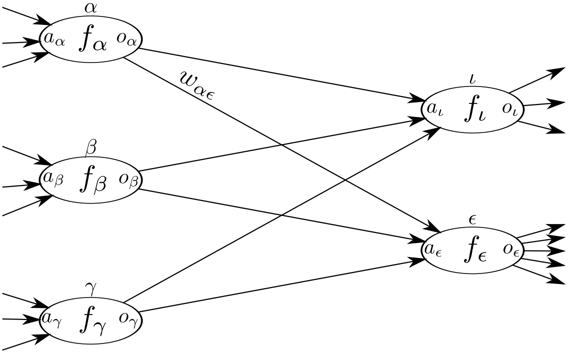
$$\frac{\partial E}{\partial o} = \delta_o$$

$$\frac{\partial E}{\partial w_{\alpha\epsilon}} = \frac{\partial E}{\partial a_\epsilon}\frac{\partial a_\epsilon}{\partial w_{\alpha\epsilon}}$$

$$= \delta_{a_\epsilon} \cdot o_\alpha$$

Back-propagation

Error Gradient wrt. node output



$$\frac{\partial E}{\partial a} = \delta_a$$

$$\frac{\partial E}{\partial w} = \delta_w$$

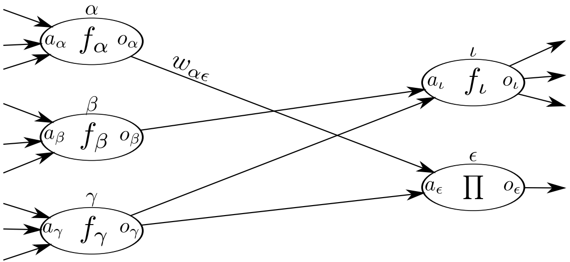
$$\frac{\partial E}{\partial o} = \delta_o$$

$$\delta_{o_\alpha} = \frac{\partial E}{\partial o_\alpha} = \sum_k \frac{\partial a_k}{\partial o_\alpha} \frac{\partial E}{\partial a_k} \quad k = \iota, \epsilon$$

$$= \sum_k w_{\alpha k} \delta_{a_k} \quad k = \iota, \epsilon$$

Back-propagation

Error Gradient wrt. node output (for multiplication nodes)



$$\frac{\partial E}{\partial a} = \delta_a$$

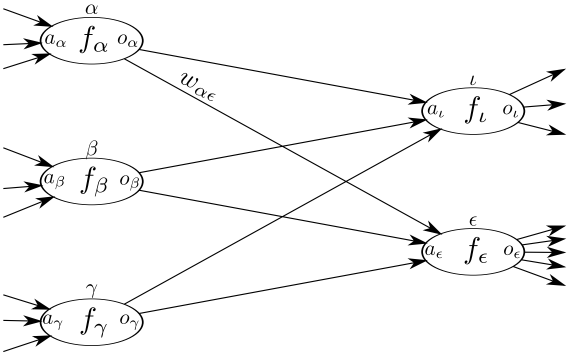
$$\frac{\partial E}{\partial w} = \delta_w$$

$$\frac{\partial E}{\partial o} = \delta_o$$

$$\begin{aligned}\delta_{o_\alpha} = \frac{\partial E}{\partial o_\alpha} &= \frac{\partial E}{\partial o_\epsilon}\frac{\partial o_\epsilon}{\partial o_\alpha} \\ &= \delta_{o_\epsilon} \cdot o_\gamma \cdot w_{\alpha\epsilon} \cdot w_{\gamma\epsilon}\end{aligned}$$

Back-propagation

Error Gradient wrt. node input



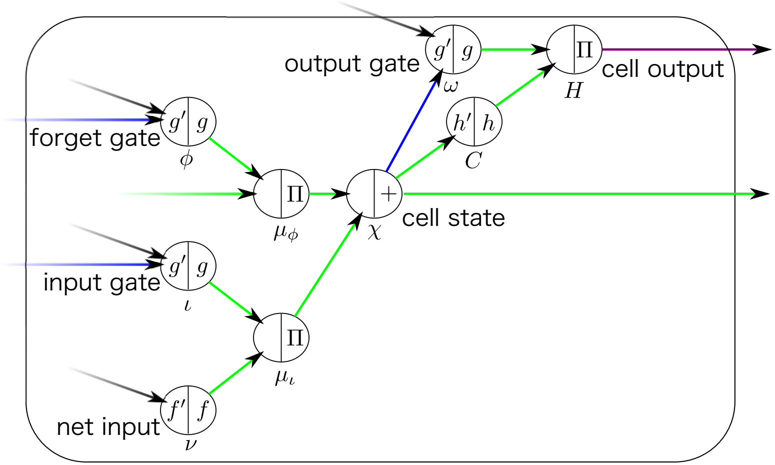
$$\frac{\partial E}{\partial a} = \delta_a$$

$$\frac{\partial E}{\partial w} = \delta_w$$

$$\frac{\partial E}{\partial o} = \delta_o$$

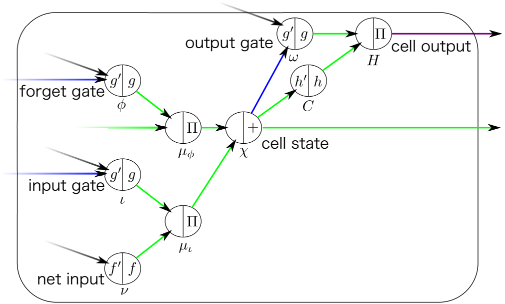
$$\begin{aligned}\frac{\partial E}{\partial a_\alpha} &= \frac{\partial o_\alpha}{\partial a_\alpha}\frac{\partial E}{\partial o_\alpha} \\ &= f'_\alpha(a_\alpha)\sum_k w_{\alpha k}\delta a_k \quad k = \iota, \epsilon\end{aligned}$$

Deriving the back-propagation formulas for the LSTM cell



Deriving the back-propagation formulas for the LSTM cell

Example: peephole weight output gate

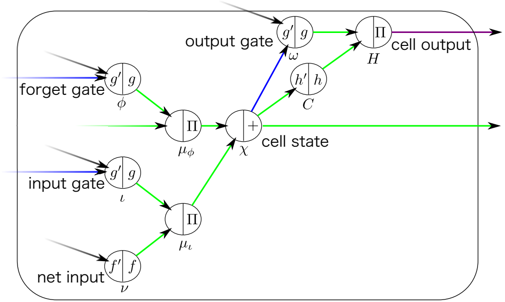


The error gradient at the output of cell output

$$\delta_H^t = \frac{\partial E}{\partial o_{\mathrm{H}_i}^t} = \sum_{k \in \mathrm{OutputLayer}} \delta_{a_k} w_{ik}^{(\mathrm{HO})} + \sum_{k \in \mathrm{hiddenLayer}} \delta_{a_k} w_{ih}^{(\mathrm{HH})}$$

Deriving the back-propagation formulas for the LSTM cell

Example: peephole weight output gate

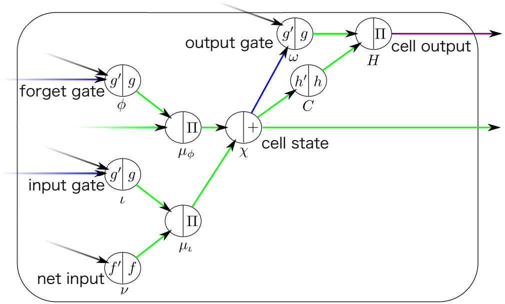


The error gradient at the input of output gate

$$\delta_\omega = g'(a_\omega^t) \cdot h(a_C^t) \cdot \delta_H^t$$

Deriving the back-propagation formulas for the LSTM cell

Example: peephole weight output gate

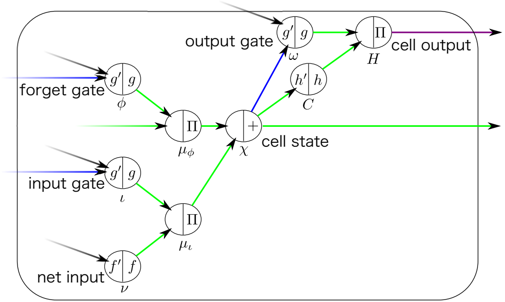


The error gradient at the input of core activation

$$\delta_C = h'(o_\chi^t) \cdot o_\omega^t \cdot \delta_H^t$$

Deriving the back-propagation formulas for the LSTM cell

Example: peephole weight output gate



The error gradient at the input of cell state

$$\delta_\chi^t = \frac{\partial O}{\partial o_\chi^t} = w_{\chi\omega}\delta_\omega + \delta_C$$
$$+ o_\phi^{t+1}\delta_\chi^{t+1}$$
$$+ w_{\chi\iota}\delta_\iota^{t+1} + w_{\chi\phi}\delta_{\phi^{t+1}}$$

Back-propagation

- All error gradients can be constructed in a similar way
- The result is a memory cell that can be trained via back-propagation