Graves 2013, "Generating Sequences with Recurrent Neural Networks"

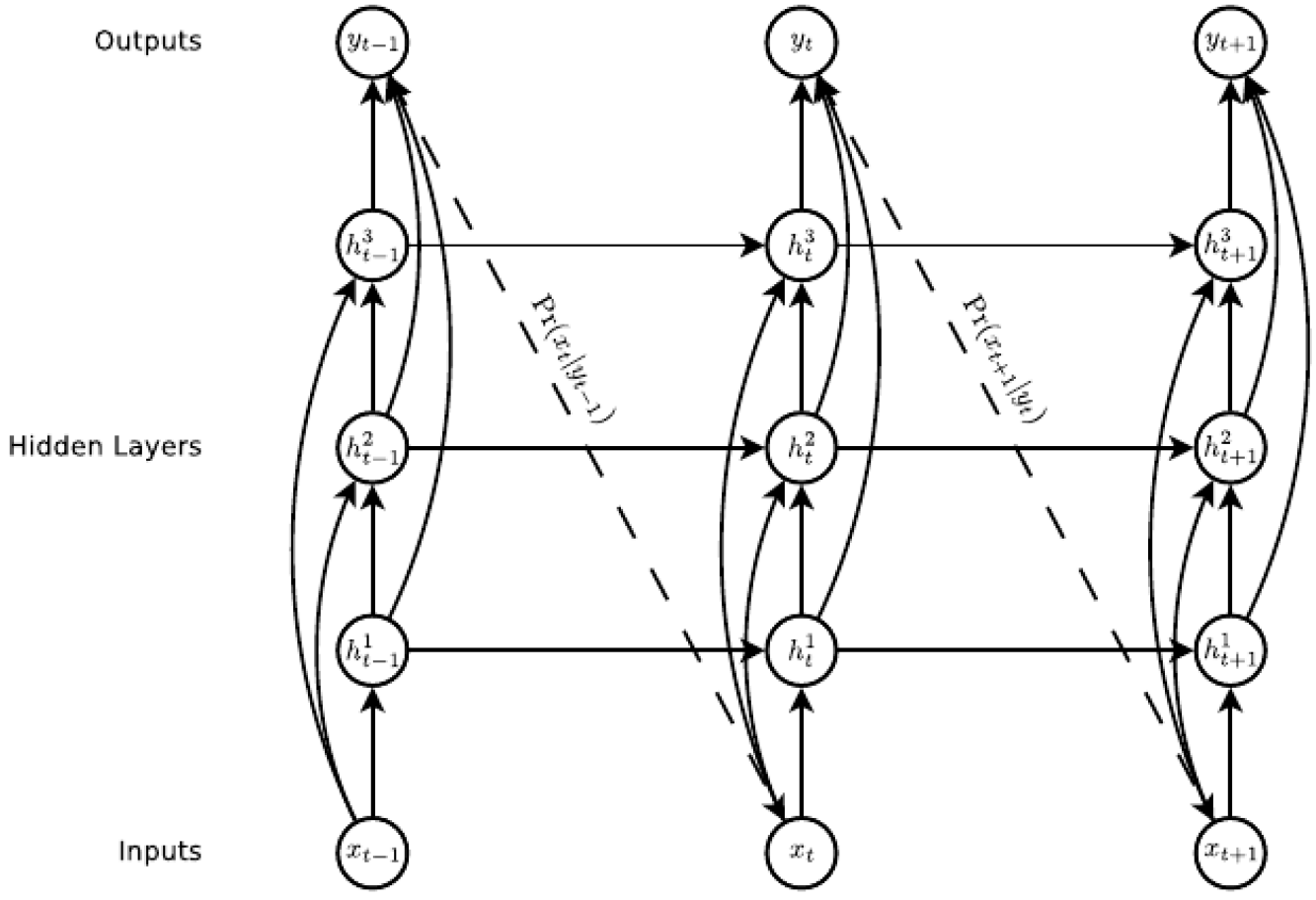
Johannes Bausch and Jack Kamm

14 November 2017

Outline

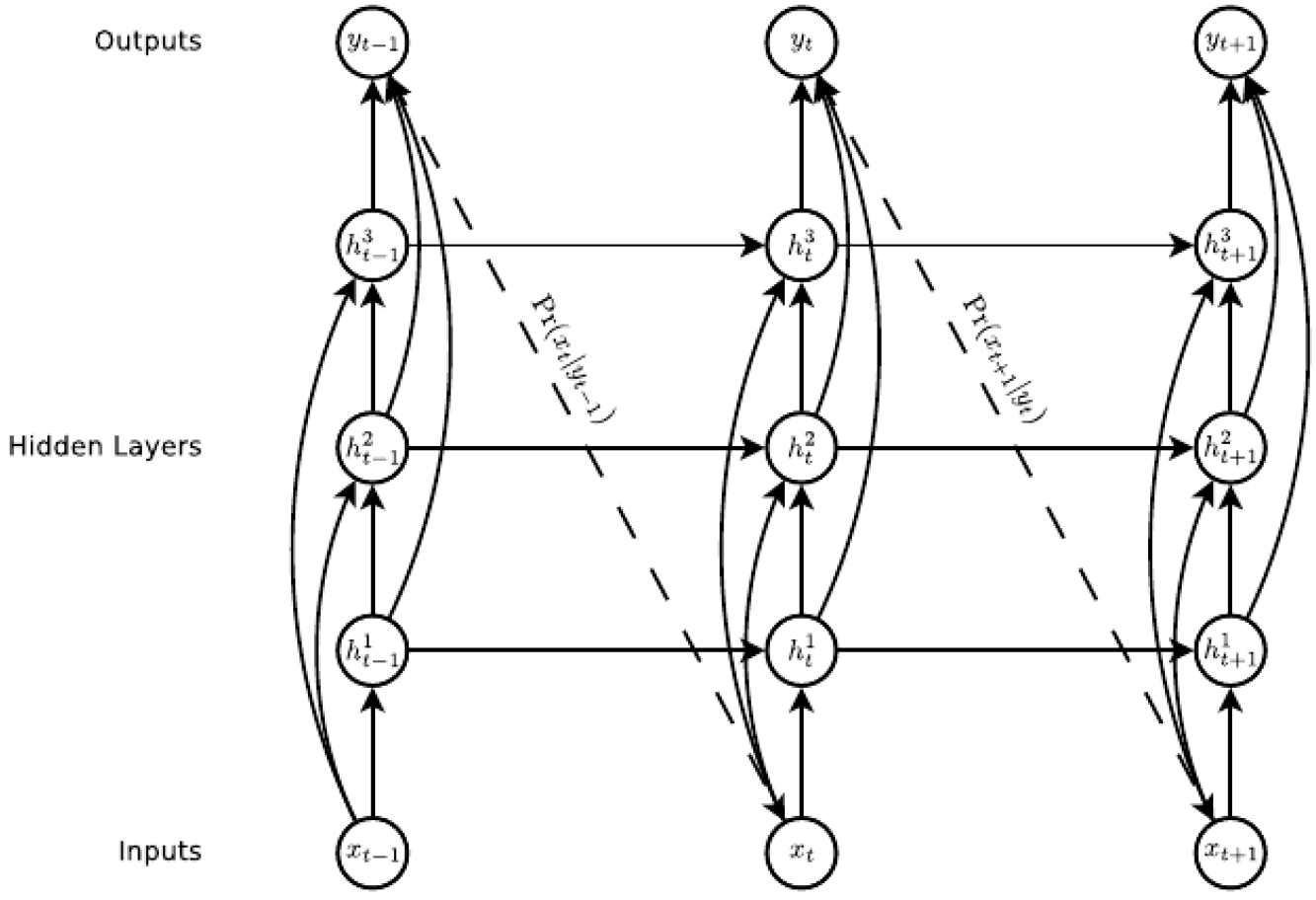
Table of Contents

Recurrent neural networks



Input x_t, hidden layers h^n_t, output y_t, generative model $\mathbb{P}(x_{t+1} \mid y_t)$

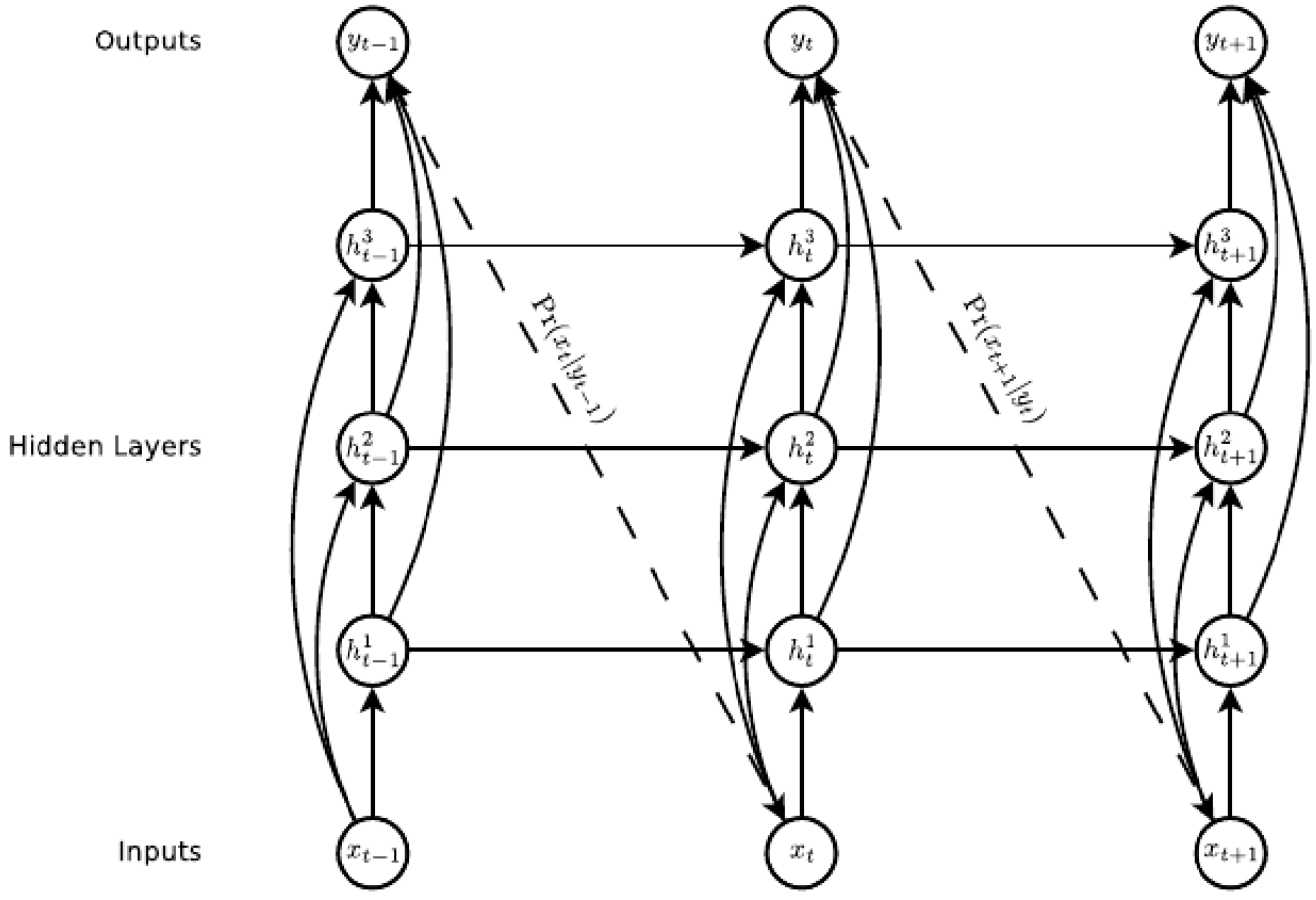
Recurrent neural networks



$h_t^n =$ nonlinear link $\circ$ affine combo of x_t, h_{t-1}^n, h_t^{n-1}

$$h_t^n = \mathcal{H}(W_{ih^n} x_t + W_{h^{n-1}h^n} h_t^{n-1} + W_{h^n h^n} h_{t-1}^n + b_h^n)$$

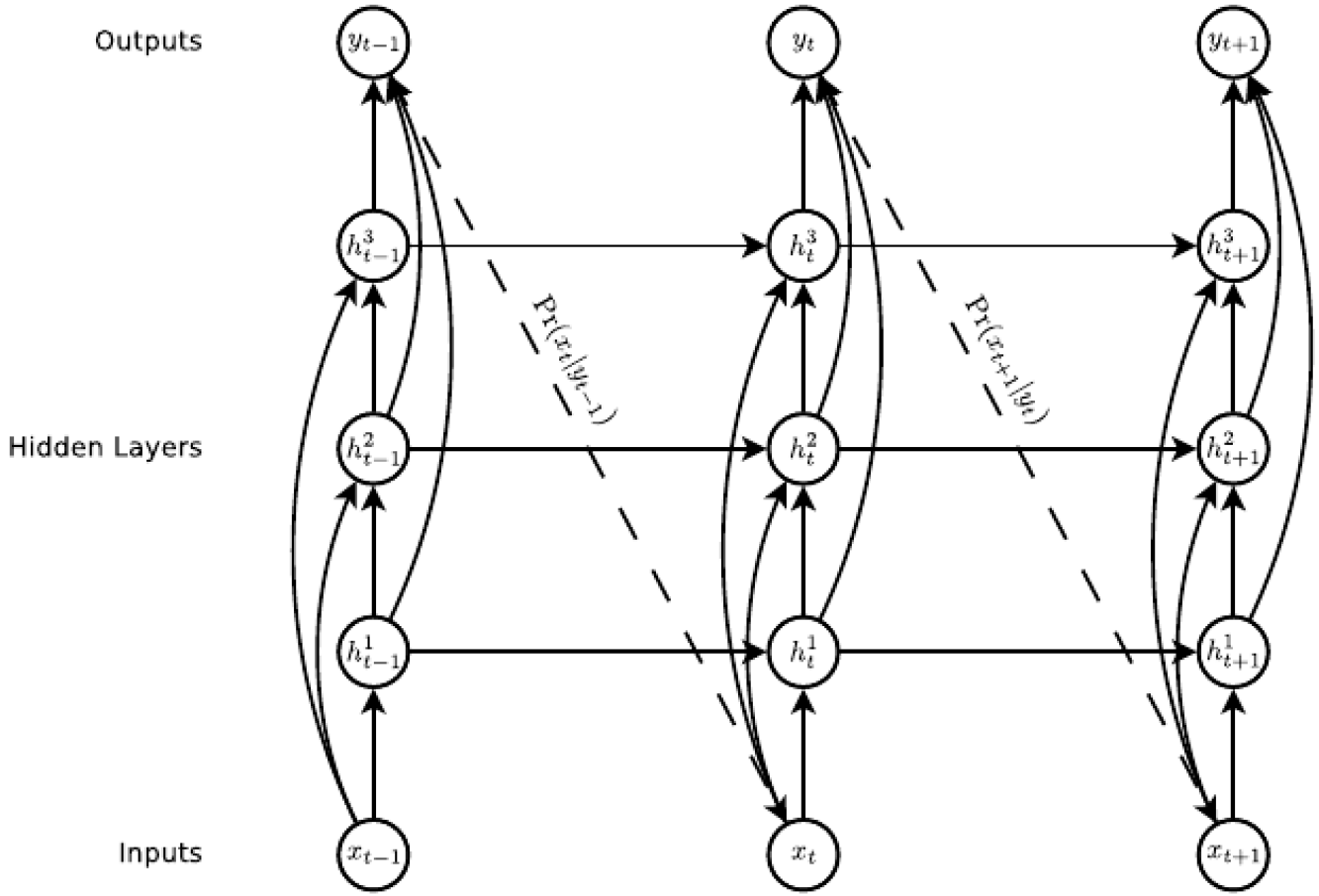
Recurrent neural networks



y_t = nonlinear link $\circ$ affine combo of h^n_t

$$y_t = \mathcal{Y}(b_y + \sum_{n=1}^{N} W_{h^n y} h^n_t)$$

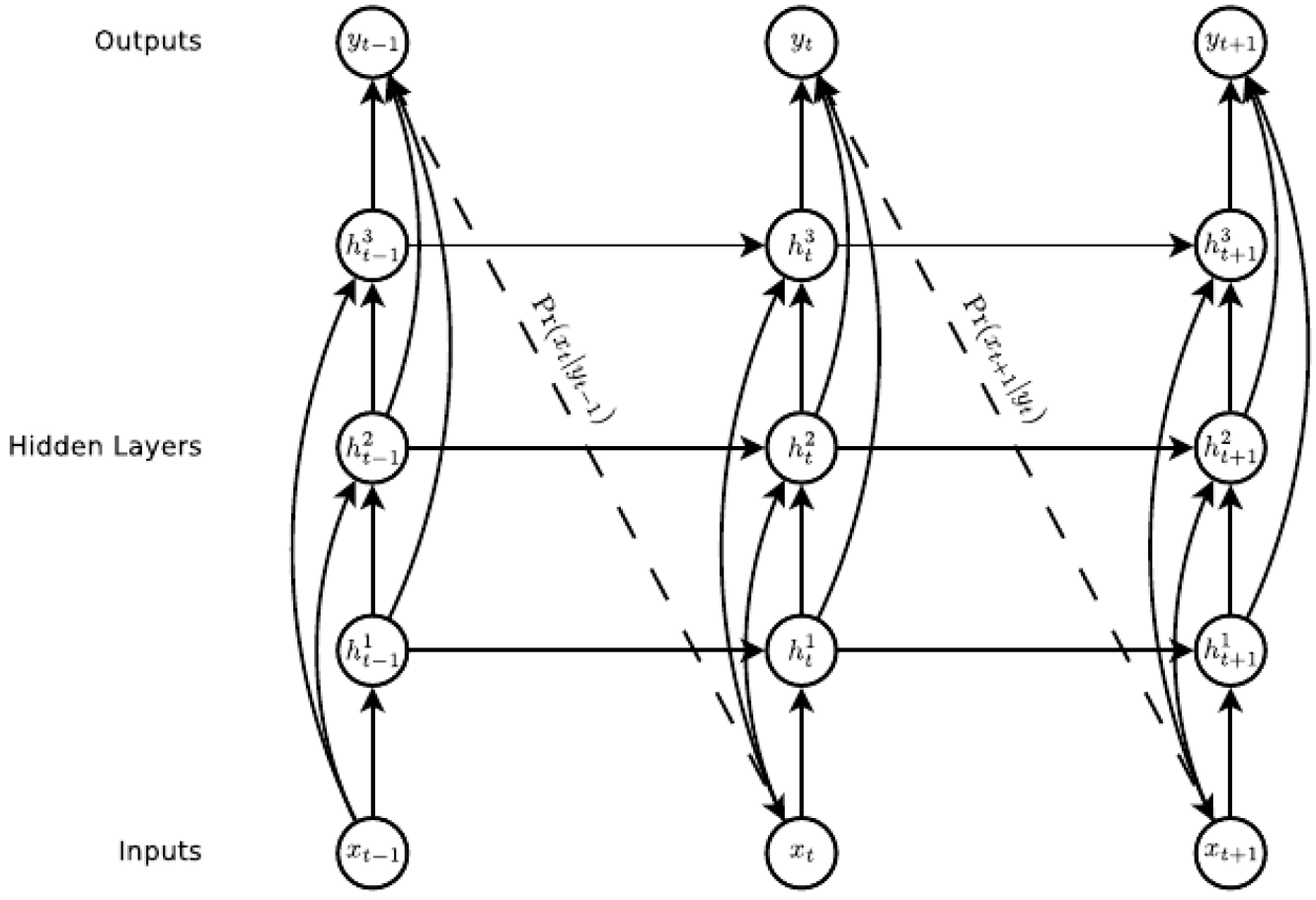
Recurrent neural networks



Train by maximizing likelihood of generative model:

$$\mathbb{P}(\mathbf{x}) = \prod_{t=1}^{T} \mathbb{P}(x_t \mid y_{t-1})$$

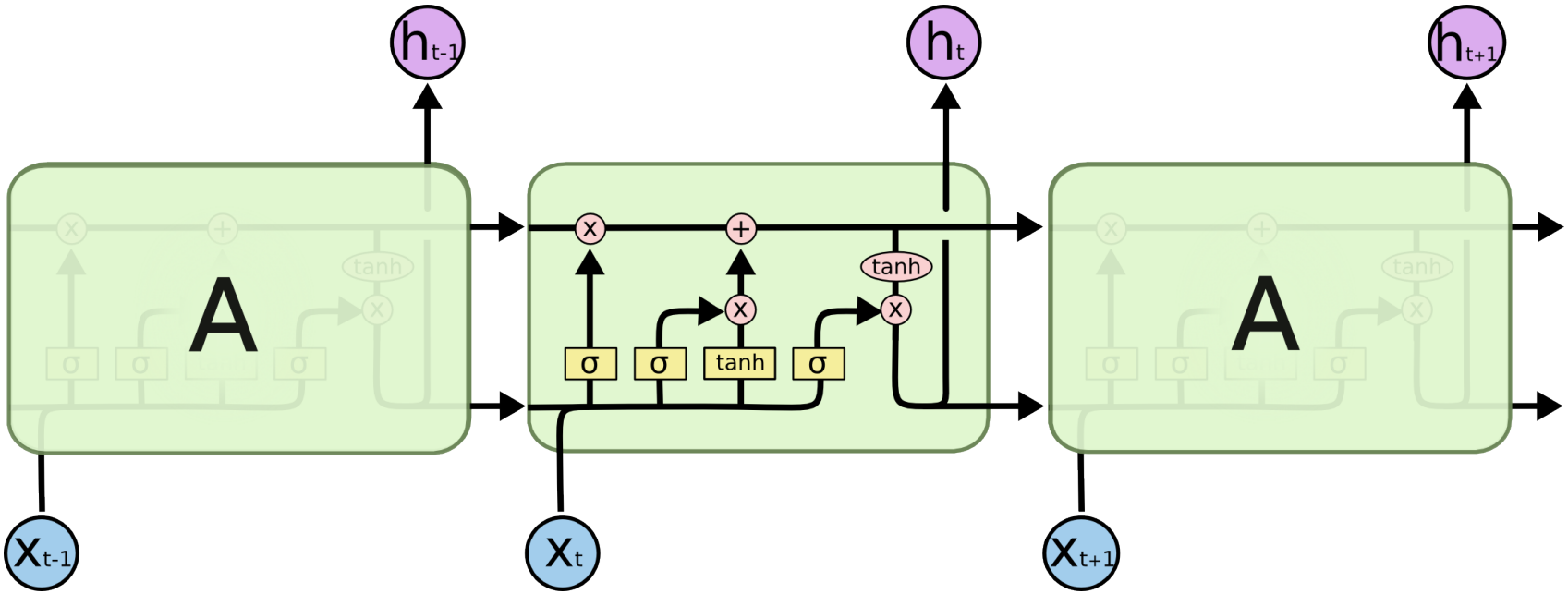
Recurrent neural networks



Compute $\nabla_\Theta \log \mathbb{P}_\Theta(\mathbf{x})$ by "truncated backpropagation through time"

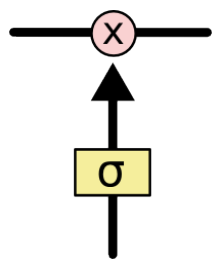
- i.e., reverse chain-rule + "clip" exploding derivatives

Long short term memory[1]

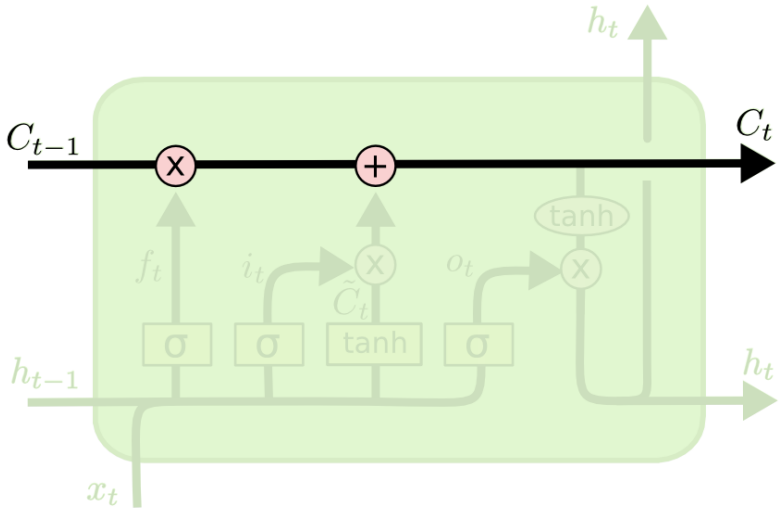


Information passes through a series of "gates"

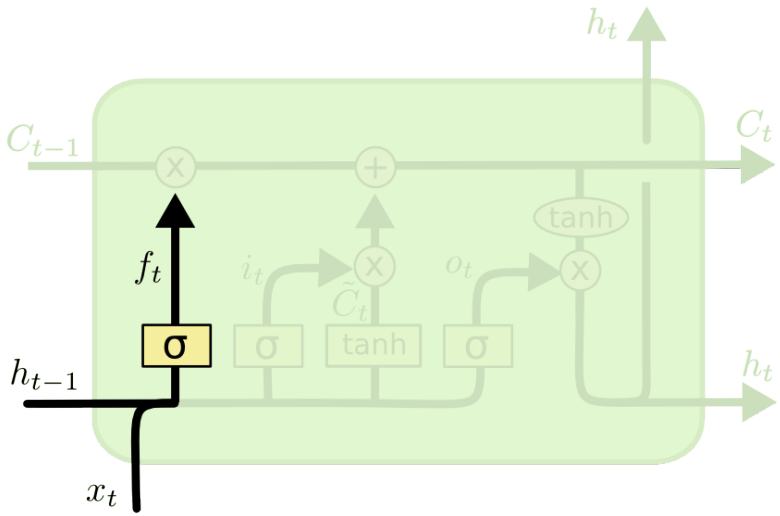
- "Gate" = multiplication with sigmoid
  - $\sigma = 0 \Rightarrow$ "let nothing thru"
  - $\sigma = 1 \Rightarrow$ "let all thru"



[1] graphics have slight differences with Graves 2013; they are taken from: https://colah.github.io/posts/2015-08-Understanding-LSTMs

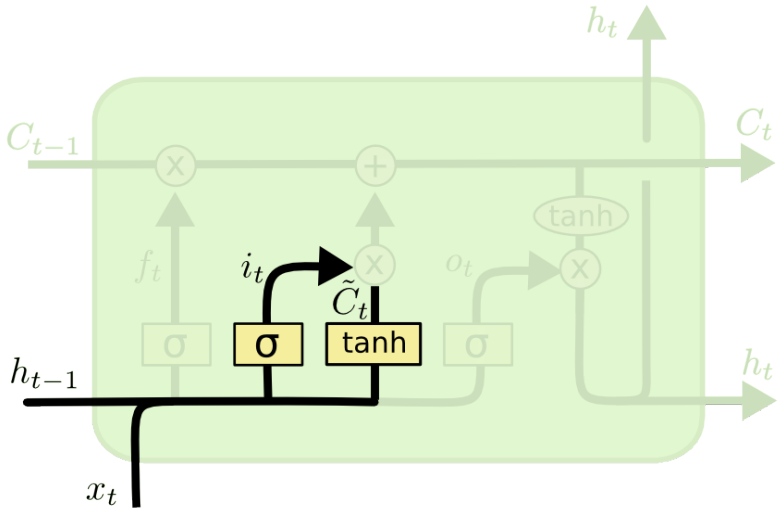


C_t = "cell state" = flows horizontally across LSTM units



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

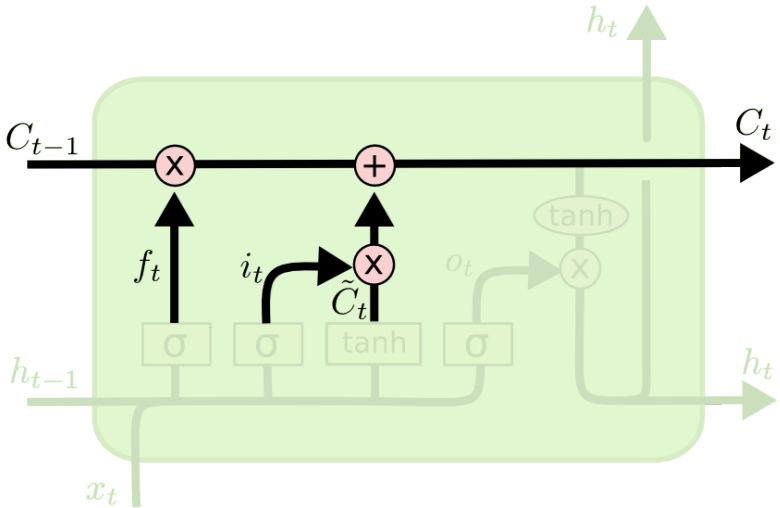
f_t = "forget gate" = gate to forget information from C_{t-1}



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

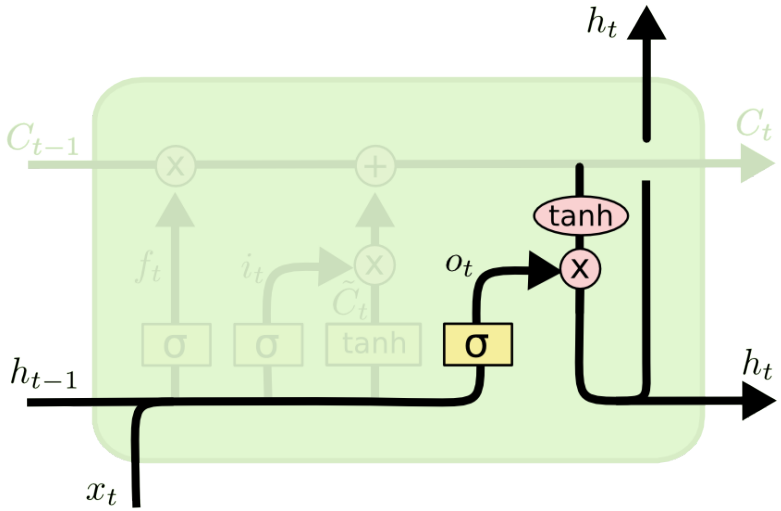
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

i_t = "input gate" = gate to add information from h_{t-1} and x_t to C_t



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Update C_t using f_t and i_t

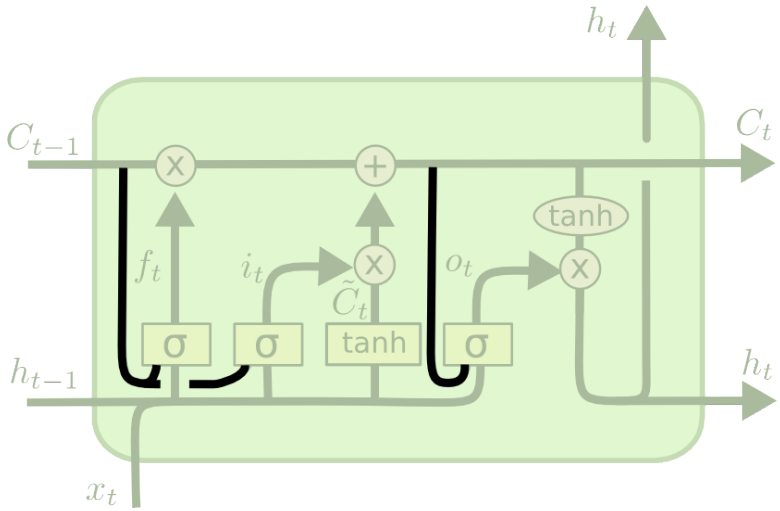


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

o_t = "output gate" = gate to output information to cell above/right

Many variations on LSTM exist. Here is the one used in Graves 2013:



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Notice the extra "peephole" connections from C to f, i, o

Table of Contents

Text Prediction

Basic framework for text prediction
$y_t = \mathbb{P}(x_{t+1})$
Per-character vs per-word prediction

Penn Treebank Test Set

Penn Treebank
Perplexity? BPC?
Regularization schemes

Wikipedia Experiments

- Wikipedia experiments
- `karpathy.github.io/2015/05/21/rnn-effectiveness` has some nice visualizations for understanding what's going on, e.g. what the inner neurons represent

Table of Contents

Handwriting experiments

- Handwriting experiments
- Mixture density outputs
  - Basic framework for real valued outputs
- Figure 10 is cool

Handwriting Synthesis

- Handwriting synthesis
- Input, output sequences have different lengths
- Biased vs unbiased vs primed sampling

Synthesis Network

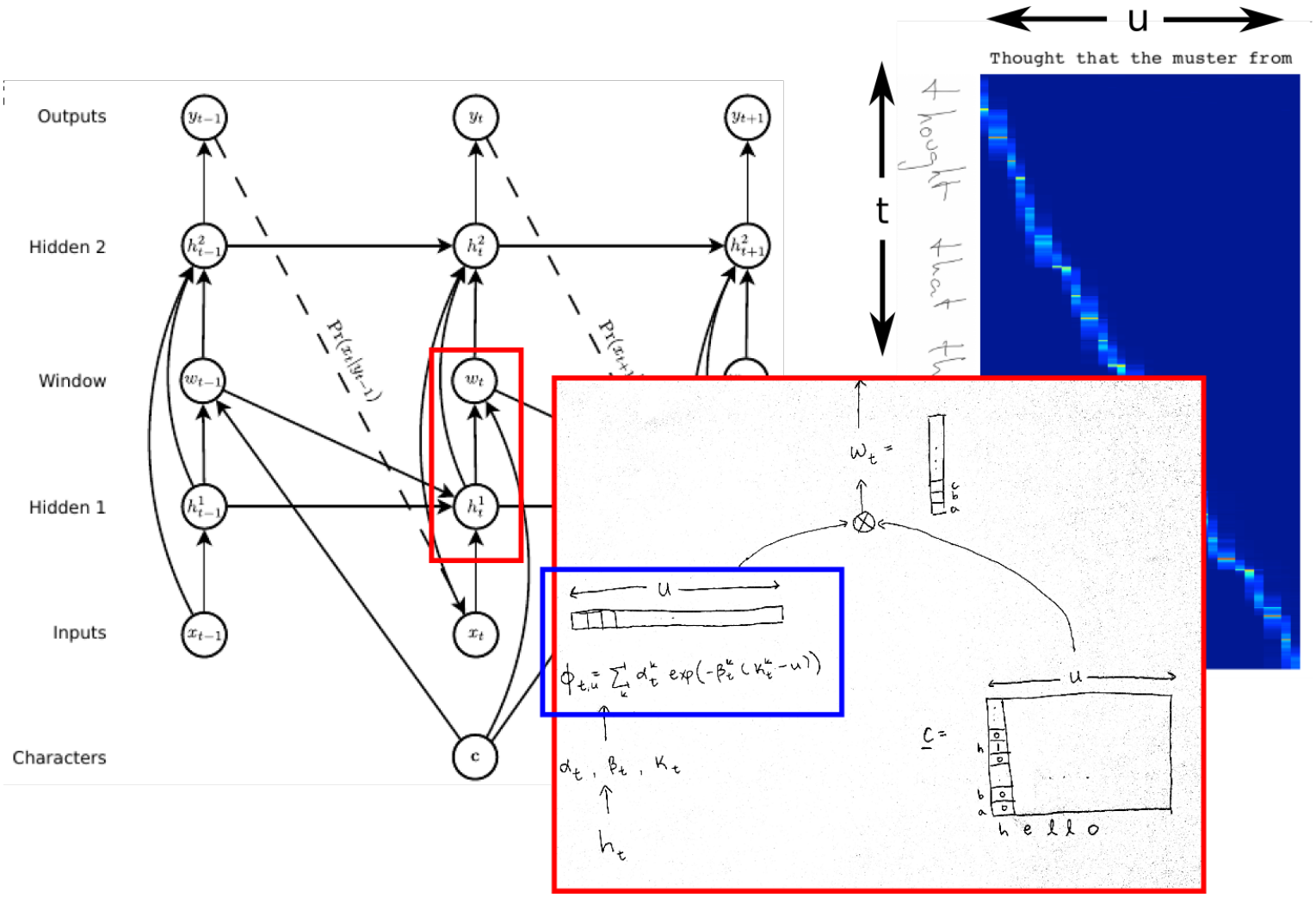


Figure: $\phi_t(h_t^1)$ = distn over positions; $w_t = \mathbf{c}\phi_t$ = distn over characters