Autoregressive Methods (RNNs/LSTMs/GRUs)

CS229: Machine Learning

Sanmi Koyejo
Stanford University, Winter 2024
(Adapted from slides by Matgus Telgarsky and Alexander Schwing)

Goals of this lecture

- Recurrent Neural Nets (RNNs)
- Long short term memory (LSTM)
- Gated recurrent unit (GRU)

Goals of this lecture

- Recurrent Neural Nets (RNNs)
- Long short term memory (LSTM)
- Gated recurrent unit (GRU)

Reading Material

- Course Notes, Section 14.3
- Goodfellow et al.; Deep Learning; Chapter 10

Lecture notation

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} \circ \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} = \begin{bmatrix} v_1 u_1 \\ \vdots \\ v_m u_m \end{bmatrix}$$

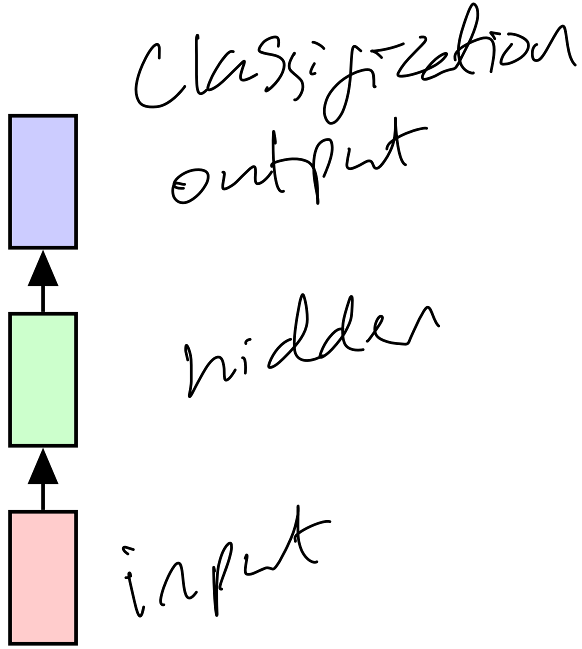
Notation	Usage
$h(\cdot)$	Feature function; $\phi(\cdot)$ in the notes
$f(\cdot), F(\cdot)$	Prediction function; $h(\cdot)$ in the notes
$l(\cdot,\cdot)$	Loss function; $J(\cdot)$ in the notes
$\boldsymbol{w}, \boldsymbol{W}$	Model Parameters, θ in the notes
$\boldsymbol{x}^{(i)}, \boldsymbol{x}$	Input(s)
$y^{(i)}, y$	Label(s)
$\hat{y}$	Prediction; o in the notes
α_k	step size in decent methods
λ	Regularization parameter(s); C in the notes
$\sigma(\cdot)$	Activation function, nonlinearity
$\circ$	Hadamard product, $\mathbf{a} \circ \boldsymbol{b}$ is the element-wise product of $\mathbf{a}$ and $\boldsymbol{b}$
$h^{(t)}$	Hidden variable(s) / Layer(s) at time/step t

Goal: Flexible models for sequences.

- Machine learning with variable length sequences of inputs and outputs.

Goal: Flexible models for sequences.

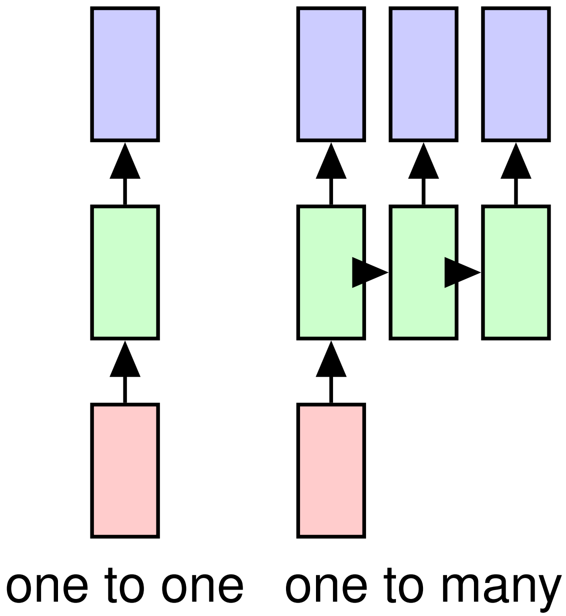
- Machine learning with variable length sequences of inputs and outputs.



one to one

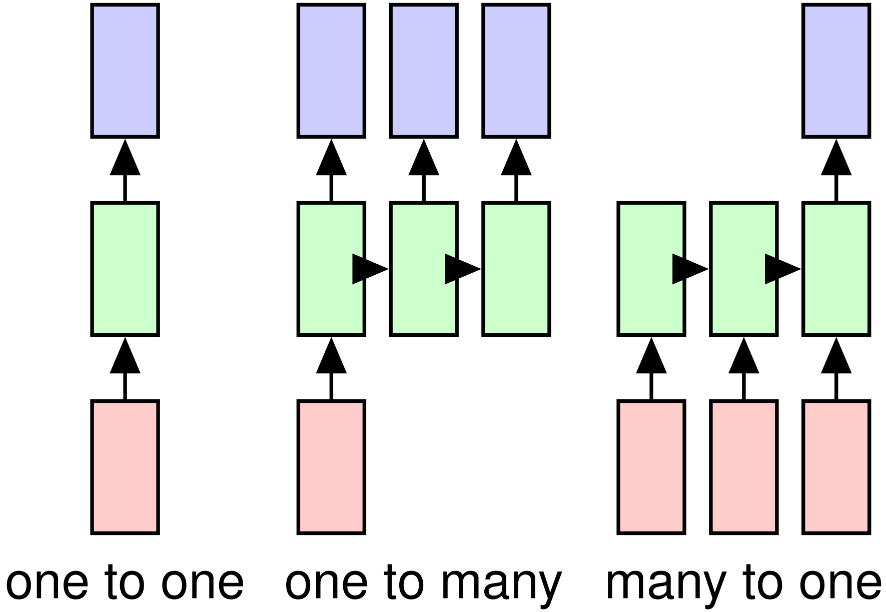
Goal: Flexible models for sequences.

- Machine learning with variable length sequences of inputs and outputs.



Goal: Flexible models for sequences.

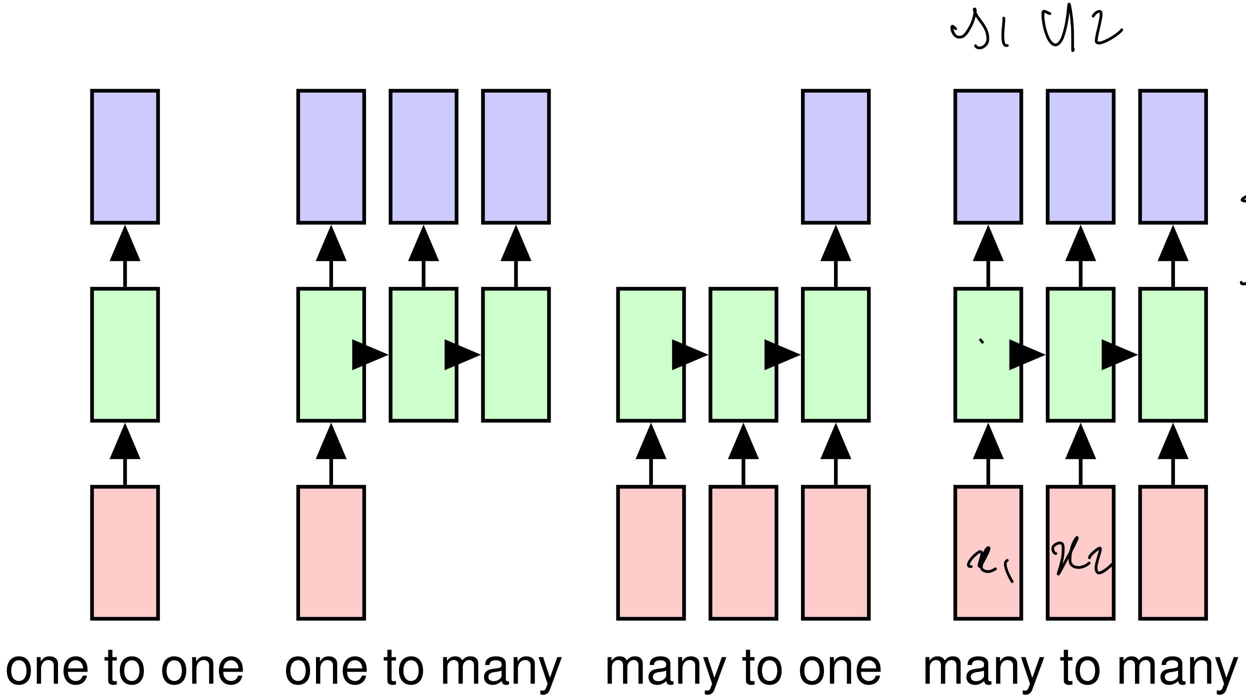
- Machine learning with variable length sequences of inputs and outputs.



Classification
- Image classification
- Sentiment "

Goal: Flexible models for sequences.

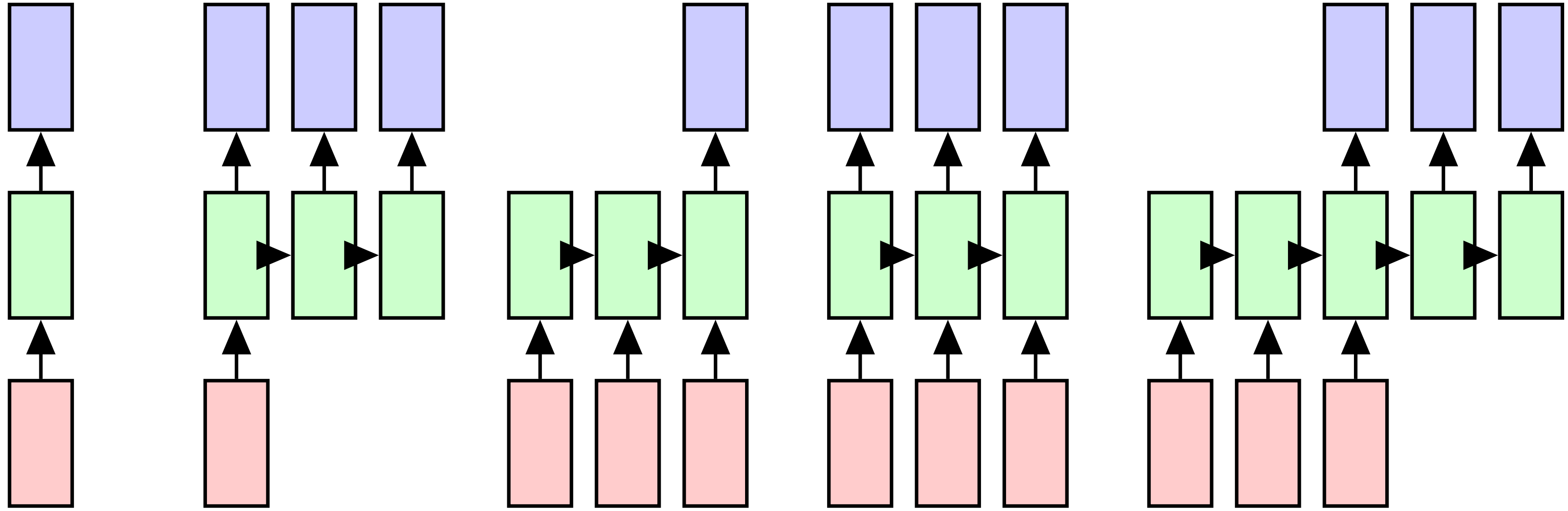
- Machine learning with variable length sequences of inputs and outputs.



- Translation (?)
- Time series forecasting (?)
- Recommendation Systems

Goal: Flexible models for sequences.

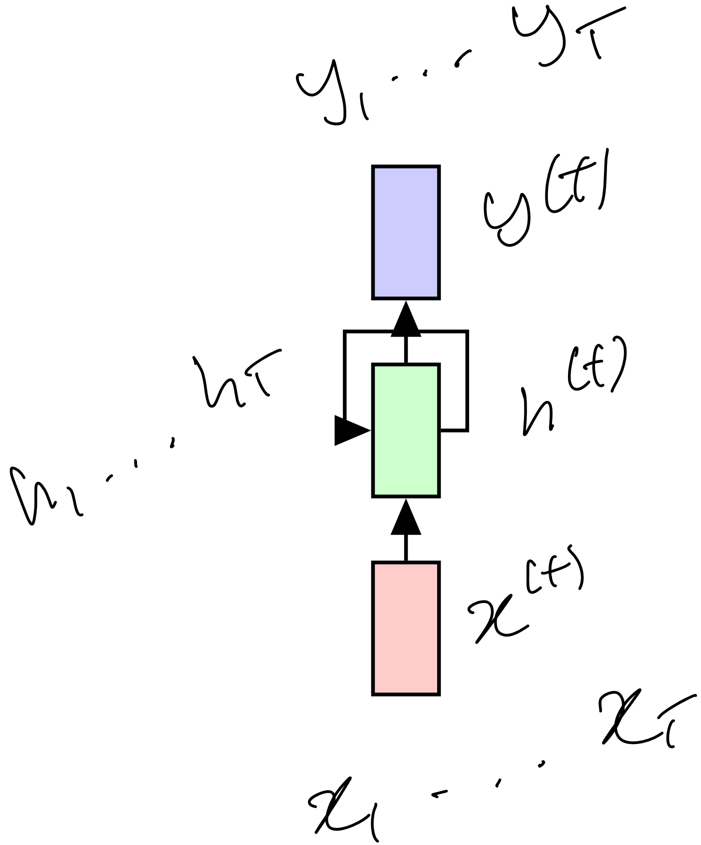
- Machine learning with variable length sequences of inputs and outputs.



Recurrent Neural Nets (RNNs)

- Input depends on previous output.

$$
\begin{aligned}
h^{(t)} &= f(h^{(t-1)}, x^{(t)}, \boldsymbol{w}) \\
y^{(t)} &= g(h^{(t)})
\end{aligned}
$$



Recurrent Neural Nets (RNNs)

$x = \{\text{"water"}, \text{"is"}, \text{"wet"}\}$

- Input depends on previous output.

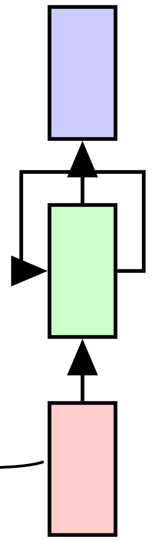
$x^{(1)} = \text{"water"}$

$x^{(2)} = \text{"is"}$

$x^{(3)} = \text{"wet"}$

$y^{(3)} = \{+, -\}$

$$
\begin{aligned}
h^{(t)} &= f(h^{(t-1)}, x^{(t)}, \boldsymbol{w}) \\
y^{(t)} &= g(h^{(t)})
\end{aligned}
$$

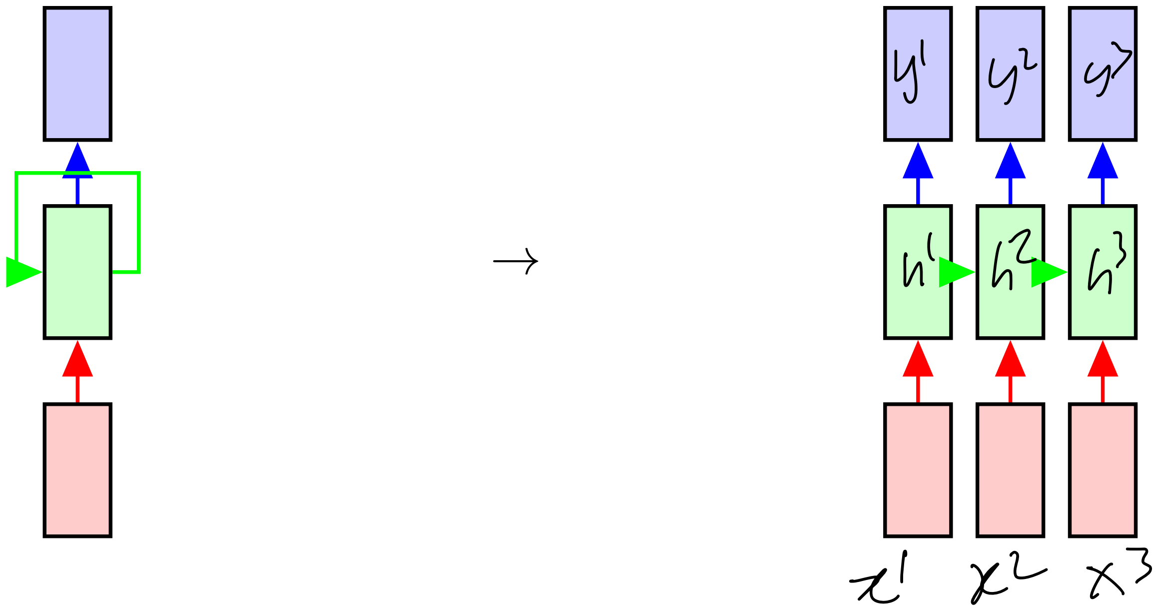


$T = 3$

Applications include:

- Natural language processing, speech recognition
- Image processing, video processing

Unrolling the RNN (parameter sharing).



$$
\begin{aligned}
h^{(t)} &= f(h^{(t-1)}, x^{(t)}, \boldsymbol{w}) \\
y^{(t)} &= g(h^{(t)})
\end{aligned}
$$

unfolded/unrolled network
performs identical operations

Note that f and g are independent of time

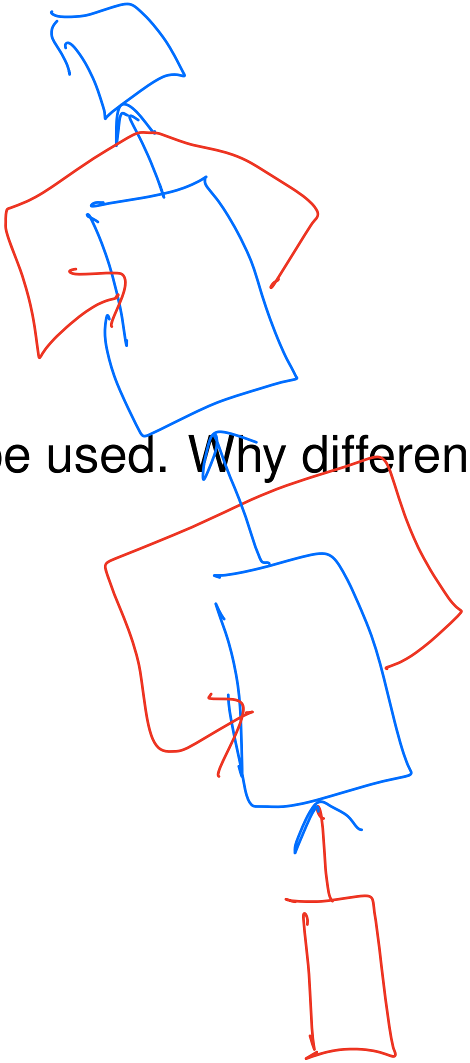
Model Specification.

What are f and g?

Any differentiable function can be used. Why differentiable functions?

Next we will cover examples of:

- Standard recurrent nets.
- LSTM nets.
- GRU nets.



Elman (1990), Finding Structure in Time

Standard recurrent nets.

$x^{(t)} \in \mathbb{R}^d$

$h^{(t)} \in \mathbb{R}^m$

$y^{(t)} \in \mathbb{R}^K$

Generally:

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}, \boldsymbol{w})$$

$$y^{(t)} = g(h^{(t)})$$

Specifically:

$\in \mathbb{R}^{m \times d}$ $\quad \in \mathbb{R}^{m \times m}$ $\quad \in \mathbb{R}$

$$h^{(t)} = \sigma_h(W_{hx}x^{(t)} + W_{hh}h^{(t-1)} + w_{hb})$$

$$y^{(t)} = \sigma_y(W_{yh}h^{(t)} + w_{yb})$$

$\in \mathbb{R}^{K \times m}$ $\quad \in \mathbb{R}$

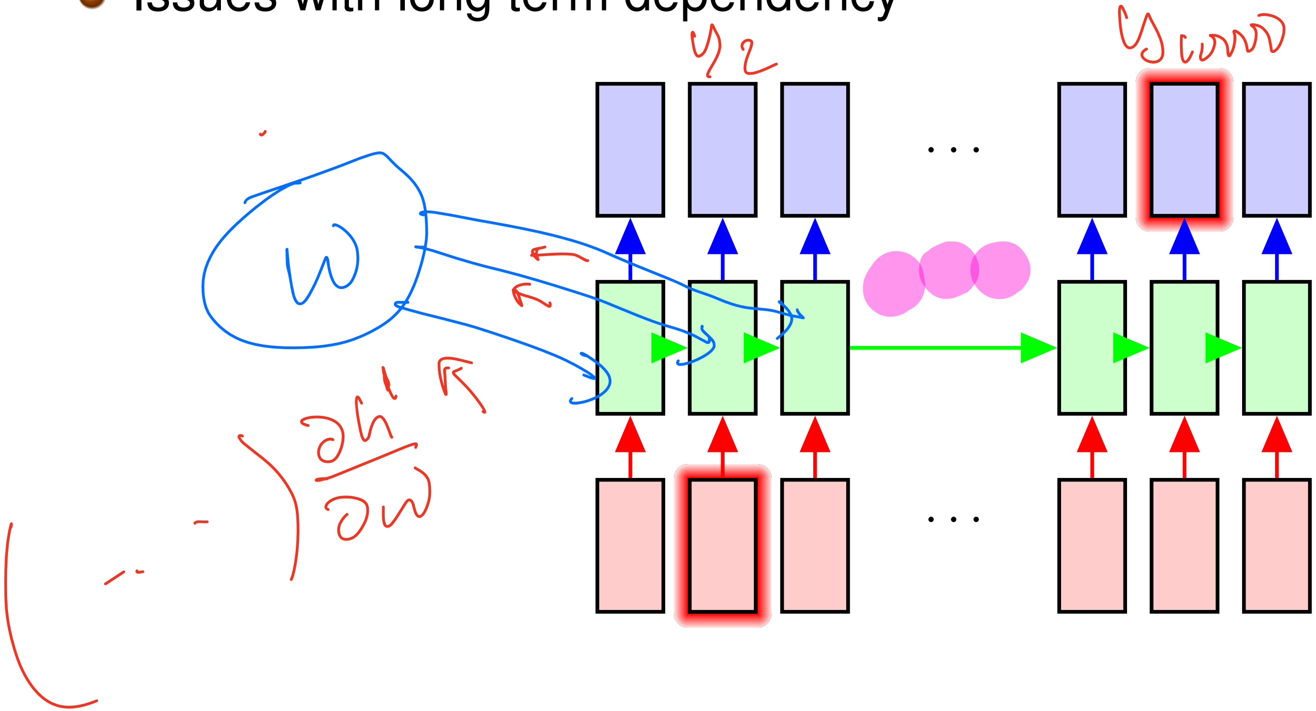
- σ_h and σ_y are activation functions (nonlinearities), e.g., $\tanh$, sigmoid, ReLU.
- Thus, RNN is constructed using affine transformations and point-wise non-linearities

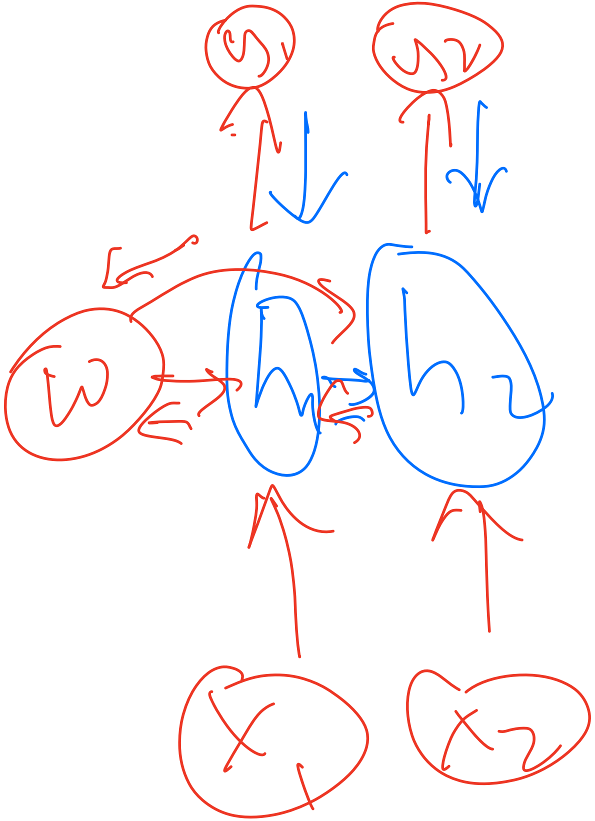
$$D = \left\{ \left\{ x_n^{(t)}, y_n^{(t)} \right\}_{t=1}^{T_n} \right\}_{n=}^{N}$$

Problems with classical recurrent neural nets

- Vanishing gradients
- Issues with long term dependency

$\cancel{\frac{\partial f}{\partial w}}$





Long short term memory (LSTM).

- Shown to better capture long-term dependencies
- Shown to address the vanishing gradient problem

Same general architecture:

$$
\begin{aligned}
h^{(t)} &= f(h^{(t-1)}, x^{(t)}, \boldsymbol{w}) \\
y^{(t)} &= g(h^{(t)})
\end{aligned}
$$

Long short term memory (LSTM).

- Shown to better capture long-term dependencies
- Shown to address the vanishing gradient problem

Same general architecture:

$$
\begin{aligned}
h^{(t)} &= f(h^{(t-1)}, x^{(t)}, \boldsymbol{w}) \\
y^{(t)} &= g(h^{(t)})
\end{aligned}
$$

However, hidden state $h^{(t)}$ is defined using additional components:
($\circ$ denotes Hadamard product; σ is activation function)

$$
\begin{aligned}
i^{(t)} &= \sigma_i(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + w_{bi}) && \text{Input gate} \\
f^{(t)} &= \sigma_f(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + w_{bf}) && \text{Forget gate}
\end{aligned}
$$

Long short term memory (LSTM).

- Shown to better capture long-term dependencies
- Shown to address the vanishing gradient problem

Same general architecture:

$$\begin{aligned} h^{(t)} &= f(h^{(t-1)}, x^{(t)}, \boldsymbol{w}) \\ y^{(t)} &= g(h^{(t)}) \end{aligned}$$

However, hidden state $h^{(t)}$ is defined using additional components:
($\circ$ denotes Hadamard product; σ is activation function)

$$\begin{aligned} i^{(t)} &= \sigma_i(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + w_{bi}) && \text{Input gate} \\ f^{(t)} &= \sigma_f(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + w_{bf}) && \text{Forget gate} \\ o^{(t)} &= \sigma_o(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + w_{bo}) && \text{Output/Exposure gate} \end{aligned}$$

Long short term memory (LSTM).

- Shown to better capture long-term dependencies
- Shown to address the vanishing gradient problem

Same general architecture:

$$\begin{aligned} h^{(t)} &= f(h^{(t-1)}, x^{(t)}, \boldsymbol{w}) \\ y^{(t)} &= g(h^{(t)}) \end{aligned}$$

However, hidden state $h^{(t)}$ is defined using additional components:
($\circ$ denotes Hadamard product; σ is activation function)

$i^{(t)} = \sigma_i(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + w_{bi})$	Input gate
$f^{(t)} = \sigma_f(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + w_{bf})$	Forget gate
$o^{(t)} = \sigma_o(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + w_{bo})$	Output/Exposure gate
$\tilde{c}^{(t)} = \sigma_c(W_{cx}x^{(t)} + W_{ch}h^{(t-1)} + w_{bc})$	New memory cell

Long short term memory (LSTM).

- Shown to better capture long-term dependencies
- Shown to address the vanishing gradient problem

Same general architecture:

$$
\begin{aligned}
h^{(t)} &= f(h^{(t-1)}, x^{(t)}, \boldsymbol{w}) \\
y^{(t)} &= g(h^{(t)})
\end{aligned}
$$

However, hidden state $h^{(t)}$ is defined using additional components:
($\circ$ denotes Hadamard product; σ is activation function)

$$
\begin{aligned}
i^{(t)} &= \sigma_i(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + w_{bi}) && \text{Input gate} \\
f^{(t)} &= \sigma_f(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + w_{bf}) && \text{Forget gate} \\
o^{(t)} &= \sigma_o(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + w_{bo}) && \text{Output/Exposure gate} \\
\tilde{c}^{(t)} &= \sigma_c(W_{cx}x^{(t)} + W_{ch}h^{(t-1)} + w_{bc}) && \text{New memory cell} \\
c^{(t)} &= f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{Final memory cell}
\end{aligned}
$$

Long short term memory (LSTM).

- Shown to better capture long-term dependencies
- Shown to address the vanishing gradient problem

Same general architecture:

$$\begin{aligned} h^{(t)} &= f(h^{(t-1)}, x^{(t)}, \boldsymbol{w}) \\ y^{(t)} &= g(h^{(t)}) \end{aligned}$$

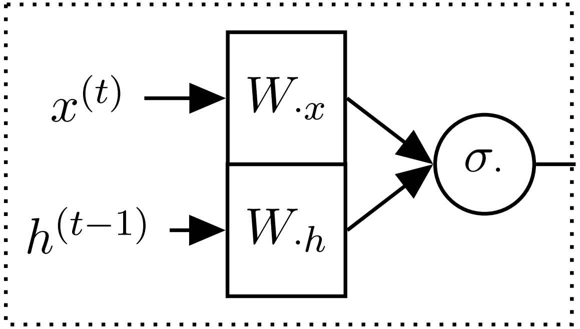
However, hidden state $h^{(t)}$ is defined using additional components:
($\circ$ denotes Hadamard product; σ is activation function)

$$\begin{aligned} i^{(t)} &= \sigma_i(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + w_{bi}) && \text{Input gate} \\ f^{(t)} &= \sigma_f(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + w_{bf}) && \text{Forget gate} \\ o^{(t)} &= \sigma_o(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + w_{bo}) && \text{Output/Exposure gate} \\ \tilde{c}^{(t)} &= \sigma_c(W_{cx}x^{(t)} + W_{ch}h^{(t-1)} + w_{bc}) && \text{New memory cell} \\ c^{(t)} &= f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{Final memory cell} \\ h^{(t)} &= o^{(t)} \circ \sigma_h(c^{(t)}) \end{aligned}$$

General structure of hidden state.

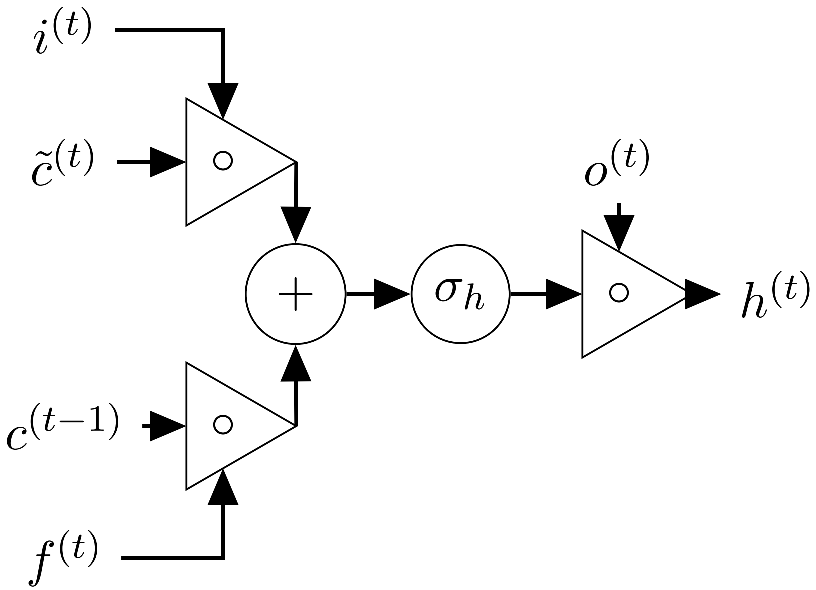
$$
\begin{aligned}
i^{(t)} &= \sigma_i(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + w_{bi}) && \text{Input gate}\\
f^{(t)} &= \sigma_f(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + w_{bf}) && \text{Forget gate}\\
o^{(t)} &= \sigma_o(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + w_{bo}) && \text{Output/Exposure gate}\\
\tilde{c}^{(t)} &= \sigma_c(W_{cx}x^{(t)} + W_{ch}h^{(t-1)} + w_{bc}) && \text{New memory cell}\\
c^{(t)} &= f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{Final memory cell}\\
h^{(t)} &= o^{(t)} \circ \sigma_h(c^{(t)})
\end{aligned}
$$

$i^{(t)}, f^{(t)}, o^{(t)}, \tilde{c}^{(t)}$ are standard feedforward blocks



Some Intuition.

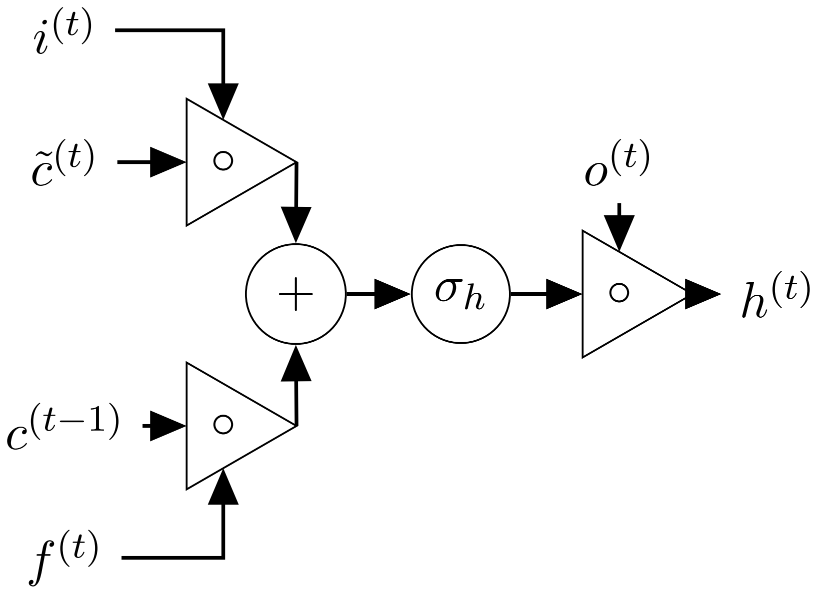
$$
\begin{aligned}
i^{(t)} &= \sigma_i(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + w_{bi}) && \text{Input gate} \\
f^{(t)} &= \sigma_f(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + w_{bf}) && \text{Forget gate} \\
o^{(t)} &= \sigma_o(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + w_{bo}) && \text{Output/Exposure gate} \\
\tilde{c}^{(t)} &= \sigma_c(W_{cx}x^{(t)} + W_{ch}h^{(t-1)} + w_{bc}) && \text{New memory cell} \\
c^{(t)} &= f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{Final memory cell} \\
h^{(t)} &= o^{(t)} \circ \sigma_h(c^{(t)})
\end{aligned}
$$



Some Intuition.

$$
\begin{aligned}
i^{(t)} &= \sigma_i(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + w_{bi}) && \text{Input gate} \\
f^{(t)} &= \sigma_f(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + w_{bf}) && \text{Forget gate} \\
o^{(t)} &= \sigma_o(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + w_{bo}) && \text{Output/Exposure gate} \\
\tilde{c}^{(t)} &= \sigma_c(W_{cx}x^{(t)} + W_{ch}h^{(t-1)} + w_{bc}) && \text{New memory cell} \\
c^{(t)} &= f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{Final memory cell} \\
h^{(t)} &= o^{(t)} \circ \sigma_h(c^{(t)})
\end{aligned}
$$

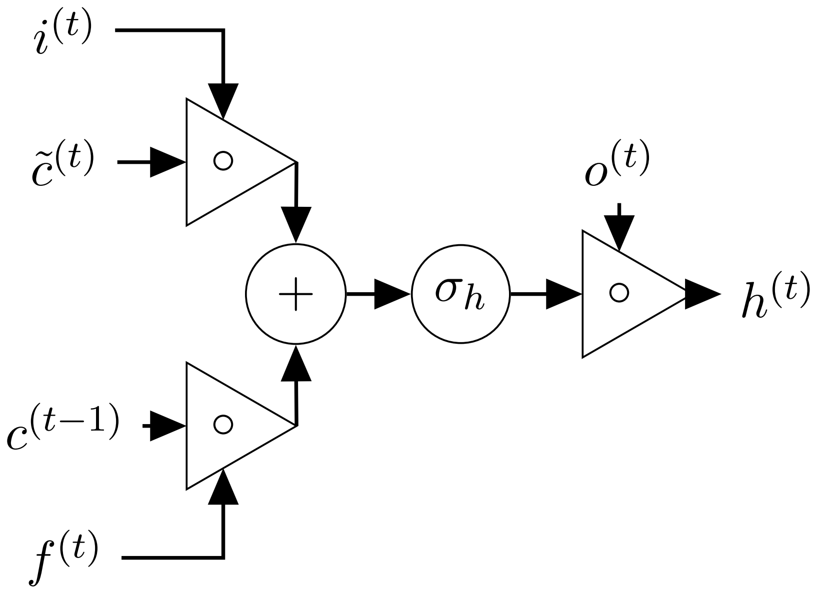
- $i^{(t)}$: Does $x^{(t)}$ matter?



Some Intuition.

$$
\begin{aligned}
i^{(t)} &= \sigma_i(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + w_{bi}) && \text{Input gate} \\
f^{(t)} &= \sigma_f(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + w_{bf}) && \text{Forget gate} \\
o^{(t)} &= \sigma_o(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + w_{bo}) && \text{Output/Exposure gate} \\
\tilde{c}^{(t)} &= \sigma_c(W_{cx}x^{(t)} + W_{ch}h^{(t-1)} + w_{bc}) && \text{New memory cell} \\
c^{(t)} &= f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{Final memory cell} \\
h^{(t)} &= o^{(t)} \circ \sigma_h(c^{(t)})
\end{aligned}
$$

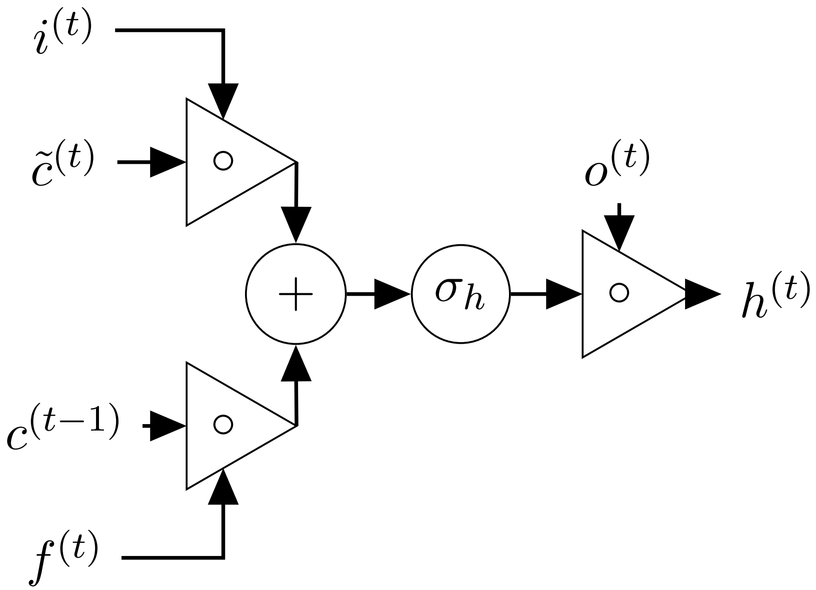
- $i^{(t)}$: Does $x^{(t)}$ matter?
- $f^{(t)}$: Should $c^{(t-1)}$ be forgotten?



Some Intuition.

$$
\begin{aligned}
i^{(t)} &= \sigma_i(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + w_{bi}) && \text{Input gate} \\
f^{(t)} &= \sigma_f(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + w_{bf}) && \text{Forget gate} \\
o^{(t)} &= \sigma_o(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + w_{bo}) && \text{Output/Exposure gate} \\
\tilde{c}^{(t)} &= \sigma_c(W_{cx}x^{(t)} + W_{ch}h^{(t-1)} + w_{bc}) && \text{New memory cell} \\
c^{(t)} &= f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{Final memory cell} \\
h^{(t)} &= o^{(t)} \circ \sigma_h(c^{(t)})
\end{aligned}
$$

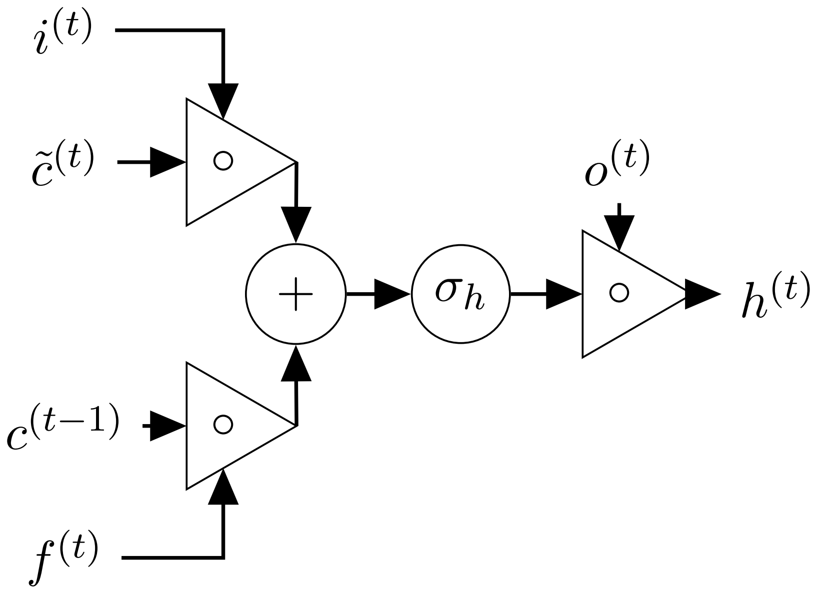
- $i^{(t)}$: Does $x^{(t)}$ matter?
- $f^{(t)}$: Should $c^{(t-1)}$ be forgotten?
- $o^{(t)}$: How much $c^{(t)}$ should be exposed?



Some Intuition.

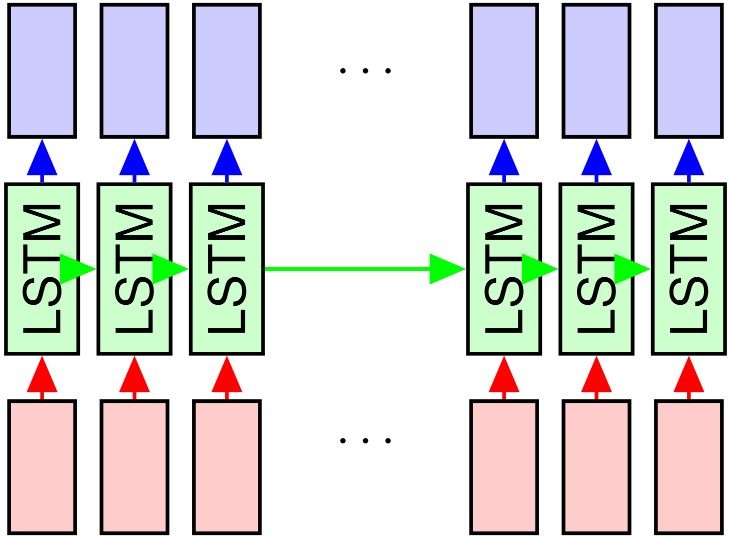
$$
\begin{aligned}
i^{(t)} &= \sigma_i(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + w_{bi}) && \text{Input gate}\\
f^{(t)} &= \sigma_f(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + w_{bf}) && \text{Forget gate}\\
o^{(t)} &= \sigma_o(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + w_{bo}) && \text{Output/Exposure gate}\\
\tilde{c}^{(t)} &= \sigma_c(W_{cx}x^{(t)} + W_{ch}h^{(t-1)} + w_{bc}) && \text{New memory cell}\\
c^{(t)} &= f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)} && \text{Final memory cell}\\
h^{(t)} &= o^{(t)} \circ \sigma_h(c^{(t)})
\end{aligned}
$$

- $i^{(t)}$: Does $x^{(t)}$ matter?
- $f^{(t)}$: Should $c^{(t-1)}$ be forgotten?
- $o^{(t)}$: How much $c^{(t)}$ should be exposed?
- $\tilde{c}^{(t)}$: Compute new memory



Putting components together.

- Long short term memory (LSTM) can be interpreted as a block in a neural net i.e. more complex $h^{(t)}$



Chung et al. (2014)

Gated recurrent unit (GRU)

Chung et al. (2014)

Gated recurrent unit (GRU)

- Performance similar to LSTM

Gated recurrent unit (GRU)

- Performance similar to LSTM
- Fewer parameters compared to LSTM (no output gate)

Gated recurrent unit (GRU)

- Performance similar to LSTM
- Fewer parameters compared to LSTM (no output gate)

Structure of hidden state:
($\circ$ denotes Hadamard product)

Gated recurrent unit (GRU)

- Performance similar to LSTM
- Fewer parameters compared to LSTM (no output gate)

Structure of hidden state:
($\circ$ denotes Hadamard product)

$$z^{(t)} = \sigma_z(W_{zx}x^{(t)} + W_{zh}h^{(t-1)} + w_{bz}) \qquad \text{Update gate}$$

Gated recurrent unit (GRU)

- Performance similar to LSTM
- Fewer parameters compared to LSTM (no output gate)

Structure of hidden state:
($\circ$ denotes Hadamard product)

$$\begin{aligned} z^{(t)} &= \sigma_z(W_{zx}x^{(t)} + W_{zh}h^{(t-1)} + w_{bz}) && \text{Update gate} \\ r^{(t)} &= \sigma_r(W_{rx}x^{(t)} + W_{rh}h^{(t-1)} + w_{br}) && \text{Reset gate} \end{aligned}$$

Chung et al. (2014)

Gated recurrent unit (GRU)

- Performance similar to LSTM
- Fewer parameters compared to LSTM (no output gate)

Structure of hidden state:
($\circ$ denotes Hadamard product)

$$
\begin{aligned}
z^{(t)} &= \sigma_z(W_{zx}x^{(t)} + W_{zh}h^{(t-1)} + w_{bz}) && \text{Update gate} \\
r^{(t)} &= \sigma_r(W_{rx}x^{(t)} + W_{rh}h^{(t-1)} + w_{br}) && \text{Reset gate} \\
\tilde{h}^{(t)} &= \sigma_h(W_{hx}x^{(t)} + W_{rwh}(r^{(t)} \circ h^{(t-1)}) + w_{bh}) && \text{New memory cell}
\end{aligned}
$$

Gated recurrent unit (GRU)

- Performance similar to LSTM
- Fewer parameters compared to LSTM (no output gate)

Structure of hidden state:
($\circ$ denotes Hadamard product)

$$
\begin{array}{rcll}
z^{(t)} & = & \sigma_z(W_{zx}x^{(t)} + W_{zh}h^{(t-1)} + w_{bz}) & \text{Update gate} \\
r^{(t)} & = & \sigma_r(W_{rx}x^{(t)} + W_{rh}h^{(t-1)} + w_{br}) & \text{Reset gate} \\
\tilde{h}^{(t)} & = & \sigma_h(W_{hx}x^{(t)} + W_{rwh}(r^{(t)} \circ h^{(t-1)}) + w_{bh}) & \text{New memory cell} \\
h^{(t)} & = & (1 - z^{(t)}) \circ \tilde{h}^{(t)} + z^{(t)} \circ h^{(t-1)} & \text{Hidden state}
\end{array}
$$

Gated recurrent unit (GRU)

- Performance similar to LSTM
- Fewer parameters compared to LSTM (no output gate)

Structure of hidden state:
($\circ$ denotes Hadamard product)

$$\begin{aligned}
z^{(t)} &= \sigma_z(W_{zx}x^{(t)} + W_{zh}h^{(t-1)} + w_{bz}) && \text{Update gate} \\
r^{(t)} &= \sigma_r(W_{rx}x^{(t)} + W_{rh}h^{(t-1)} + w_{br}) && \text{Reset gate} \\
\tilde{h}^{(t)} &= \sigma_h(W_{hx}x^{(t)} + W_{rwh}(r^{(t)} \circ h^{(t-1)}) + w_{bh}) && \text{New memory cell} \\
h^{(t)} &= (1 - z^{(t)}) \circ \tilde{h}^{(t)} + z^{(t)} \circ h^{(t-1)} && \text{Hidden state}
\end{aligned}$$

This can again be interpreted as a block in the computation graph (replaces the hidden state block).

Some intuition.

$$\begin{aligned}
z^{(t)} &= \sigma_z(W_{zx}x^{(t)} + W_{zh}h^{(t-1)} + w_{bz}) && \text{Update gate} \\
r^{(t)} &= \sigma_r(W_{rx}x^{(t)} + W_{rh}h^{(t-1)} + w_{br}) && \text{Reset gate} \\
\tilde{h}^{(t)} &= \sigma_h(W_{hx}x^{(t)} + W_{rwh}(r^{(t)} \circ h^{(t-1)}) + w_{bh}) && \text{New memory cell} \\
h^{(t)} &= (1 - z^{(t)}) \circ \tilde{h}^{(t)} + z^{(t)} \circ h^{(t-1)} && \text{Hidden state}
\end{aligned}$$

Some intuition.

$$
\begin{aligned}
z^{(t)} &= \sigma_z(W_{zx}x^{(t)} + W_{zh}h^{(t-1)} + w_{bz}) && \text{Update gate} \\
r^{(t)} &= \sigma_r(W_{rx}x^{(t)} + W_{rh}h^{(t-1)} + w_{br}) && \text{Reset gate} \\
\tilde{h}^{(t)} &= \sigma_h(W_{hx}x^{(t)} + W_{rwh}(r^{(t)} \circ h^{(t-1)}) + w_{bh}) && \text{New memory cell} \\
h^{(t)} &= (1 - z^{(t)}) \circ \tilde{h}^{(t)} + z^{(t)} \circ h^{(t-1)} && \text{Hidden state}
\end{aligned}
$$

- $r^{(t)}$: Include $h^{(t-1)}$ in new memory?

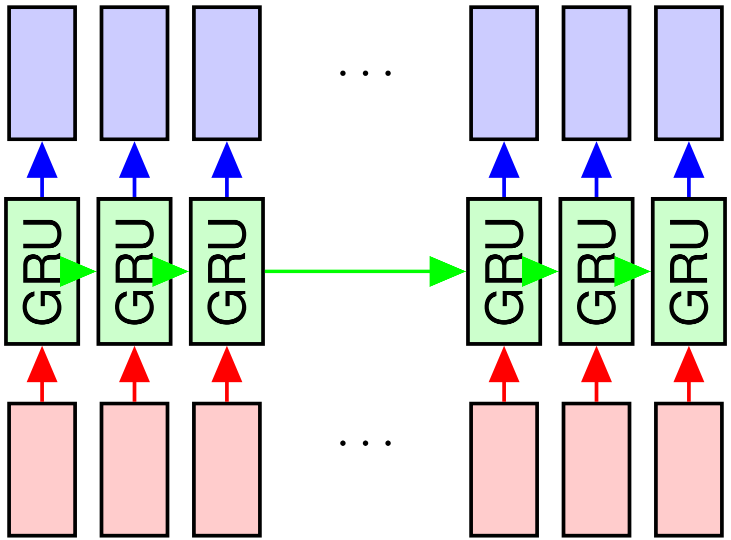
Some intuition.

$$\begin{aligned} z^{(t)} &= \sigma_z(W_{zx}x^{(t)} + W_{zh}h^{(t-1)} + w_{bz}) && \text{Update gate} \\ r^{(t)} &= \sigma_r(W_{rx}x^{(t)} + W_{rh}h^{(t-1)} + w_{br}) && \text{Reset gate} \\ \tilde{h}^{(t)} &= \sigma_h(W_{hx}x^{(t)} + W_{rwh}(r^{(t)} \circ h^{(t-1)}) + w_{bh}) && \text{New memory cell} \\ h^{(t)} &= (1 - z^{(t)}) \circ \tilde{h}^{(t)} + z^{(t)} \circ h^{(t-1)} && \text{Hidden state} \end{aligned}$$

- $r^{(t)}$: Include $h^{(t-1)}$ in new memory?
- $z^{(t)}$: How much $h^{(t-1)}$ in next state?

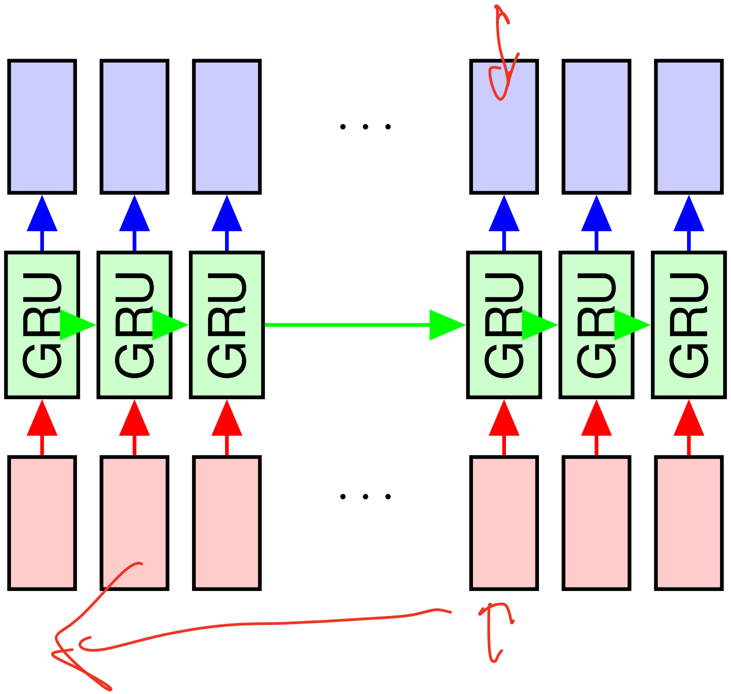
Putting components together.

- GRU can be interpreted as a block in a neural net i.e. more complex $h^{(t)}$



Putting components together.

- GRU can be interpreted as a block in a neural net i.e. more complex $h^{(t)}$



Lots of additional variants:

- e.g. Bi-directional LSTMs [Schuster&Paliwal (1997), Graves&Schmidhuber (2005)]

Learning.

How do we learn the parameters in the network?

Learning.

How do we learn the parameters in the network?

$$p(y_1, \ldots, y_T) = \prod_{i=1}^{T} p(y_i | y_1, \ldots y_{i-1})$$

$p(y_1) p(y_2 | y_1)$

$p(y_3 | y_2, y_1) \ldots$

Learning.

How do we learn the parameters in the network?

$$p(y_1, \ldots, y_T) = \prod_{i=1}^{T} p(y_i | y_1, \ldots y_{i-1})$$

The loss function is defined via maximum (log-)likelihood

Learning.

How do we learn the parameters in the network?

$$p(y_1, \ldots, y_T) = \prod_{i=1}^{T} p(y_i | y_1, \ldots y_{i-1})$$

The loss function is defined via maximum (log-)likelihood

~~Relation to structured models?~~

Training via gradient descent:

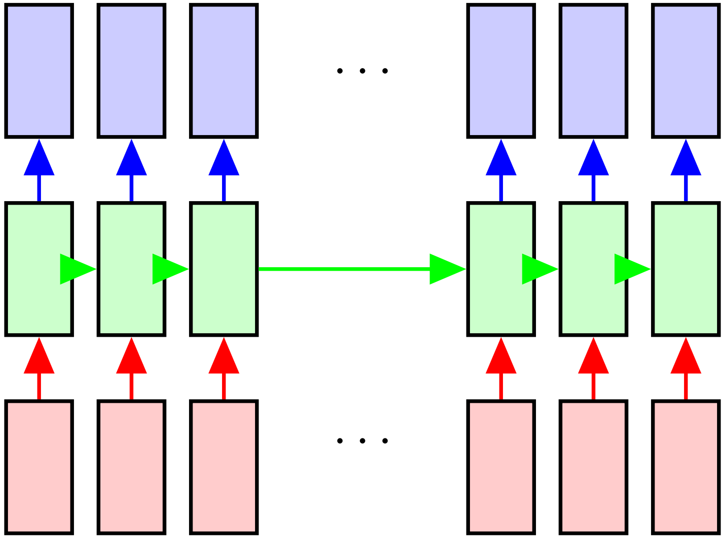
- Useful to think of unrolled model as a feedforward DNN with weight sharing

Training via gradient descent:

- Useful to think of unrolled model as a feedforward DNN with weight sharing
- Train using standard backpropagation.

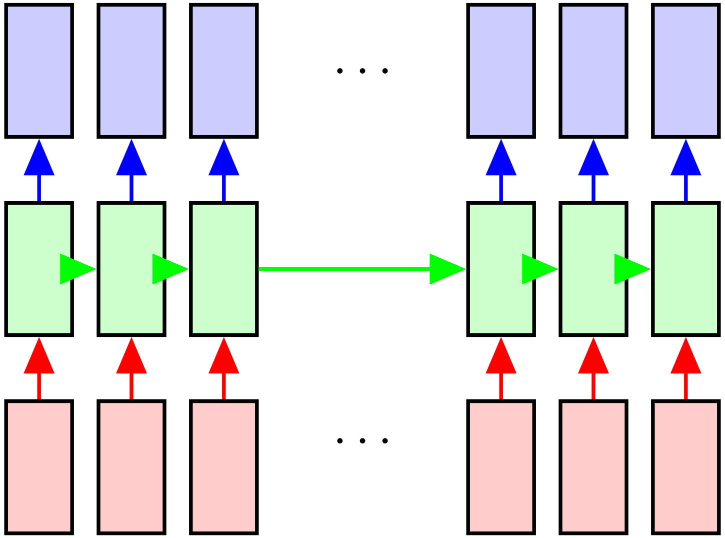
Training via gradient descent:

- Useful to think of unrolled model as a feedforward DNN with weight sharing
- Train using standard backpropagation.
- What information do we need to store?



Training via gradient descent:

- Useful to think of unrolled model as a feedforward DNN with weight sharing
- Train using standard backpropagation.
- What information do we need to store?



Backpropagation through time (BPTT)

Example: Application to image completion.

Pixel Recurrent Neural Networks

Example: Application to image completion.

Pixel Recurrent Neural Networks

- Pick an ordering, vectorize the image as a sequence.
- Image completion as sequence prediction.
- Can also be used for synthesis (How?)

Example: Simple RNN code from Andrej Karpathy

https://gist.github.com/karpathy/d4dee566867f8291f086

Quiz:

- Describe the prediction process for an RNN?

Quiz:

- Describe the prediction process for an RNN?
- Describe the training process for RNNs?

Important topics of this lecture

- Getting to know RNNs and some variants
- Pretraining and some applications of pre-trained models

Important topics of this lecture

- Getting to know RNNs and some variants
- Pretraining and some applications of pre-trained models

What's next:

Decision Trees