# CS 446/ECE 449: Machine Learning

A. G. Schwing

University of Illinois at Urbana-Champaign, 2020

L27: Autoregressive Methods (RNNs/LSTMs/GRUs)

**Goals of this lecture**

- Getting to know Recurrent Neural Nets (RNNs)
- Getting to know Long short term memory (LSTM)
- Getting to know Gated recurrent unit (GRU)
- Getting to know Graph convolutional nets (GCNs)
- Seeing how to apply them

**Reading Material**

- Goodfellow et al.; Deep Learning; Chapter 10
- Papers cited on the slides

# Pixel Recurrent Neural Networks

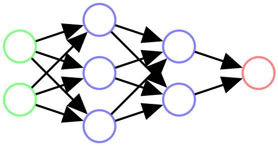
**Recap:** Our models so far

- Discriminative

$$p(\boldsymbol{y}|x)$$

- Generative

$$p(x)$$

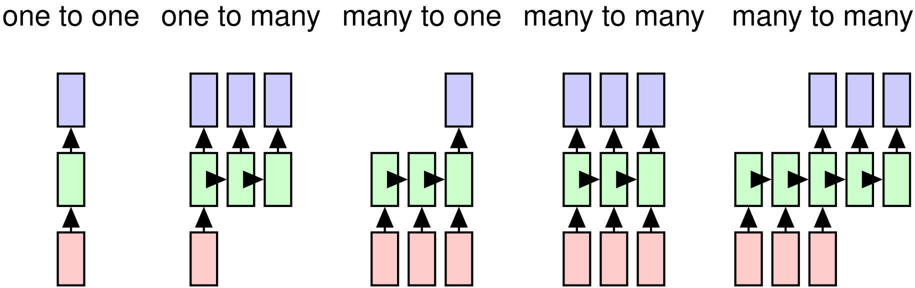


What's missing?

More flexibility regarding inputs and outputs:

- Sequences of inputs
- Sequences of outputs

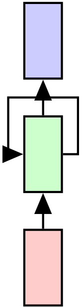
Length of sequences may vary



Recurrent Neural Nets (RNNs)

- input depends on previous output

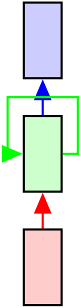
$$h^{(t)} = f(h^{(t-1)}, x^{(t)}, \boldsymbol{w})$$



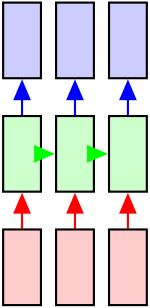
Applications:

- Natural language processing
- Speech recognition
- Image processing
- Video processing

## Important concept: Parameter sharing



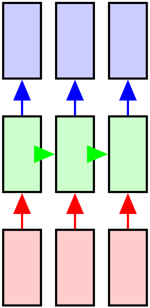
$\rightarrow$



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

unfolded/unrolled network
performs identical operations
easier to understand

General structure for recurrence:



Mathematical description in general:

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

Note that $f$ and $g$ are independent of time

What are $f$ and $g$?

Any differentiable function can be used

Useful functions:

- Original recurrent nets
- LSTM nets
- GRU nets

Original recurrent nets (Elman network):

(Jordan network is slightly different)

Generally:

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

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

Specifically:

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

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

What is $\sigma_h$ and $\sigma_y$?

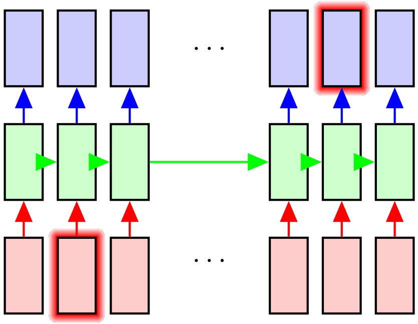
Activation functions: `tanh`, sigmoid

Affine transformations and point-wise non-linearity

What are the problems?

Problems with classical recurrent neural nets:

- Vanishing gradients
- Long term dependency



Long short term memory (LSTM)

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

Generally:

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

Specifically: ($\circ$ denotes Hadamard product; $\sigma$ is activation function)

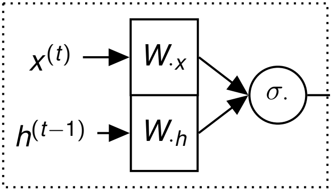
$$\begin{aligned} i^{(t)} &= \sigma_i(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + w_{bi}) && \text{Input gate} \\ f^{(t)} &= \sigma_f(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + w_{bf}) && \text{Forget gate} \\ o^{(t)} &= \sigma_o(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + w_{bo}) && \text{Output/Exposure gate} \\ \tilde{c}^{(t)}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

Equations:

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

Intuition:

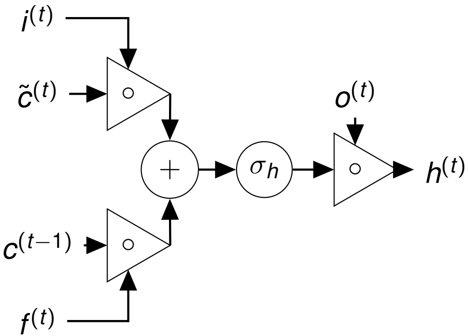
- $i^{(t)}$: Does $x^{(t)}$ matter?
- $f^{(t)}$: Should $c^{(t-1)}$ be forgotten?
- $o^{(t)}$: How much $c^{(t)}$ should be exposed?
- $\tilde{c}^{(t)}$: Compute new memory



Equations:

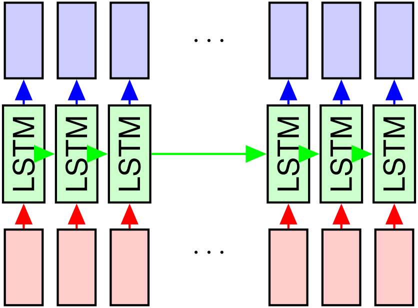
$$
\begin{array}{rcll}
i^{(t)} & = & \sigma_i(W_{ix}x^{(t)} + W_{ih}h^{(t-1)} + w_{bi}) & \text{Input gate} \\
f^{(t)} & = & \sigma_f(W_{fx}x^{(t)} + W_{fh}h^{(t-1)} + w_{bf}) & \text{Forget gate} \\
o^{(t)} & = & \sigma_o(W_{ox}x^{(t)} + W_{oh}h^{(t-1)} + w_{bo}) & \text{Output/Exposure gate} \\
\tilde{c}^{(t)}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

Intuition:



Long short term memory (LSTM):

- Can be interpreted as a block in a neural net



Gated recurrent unit (GRU):

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

Equations: ($\circ$ denotes Hadamard product)

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

Can again be interpreted as a block in the computation graph

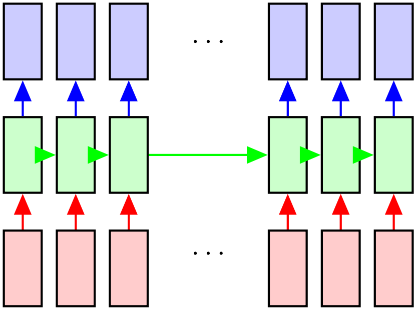
Equations: ($\circ$ denotes Hadamard product)

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

Intuition:

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

Recurrent nets generally:



Other variants:

- Bi-directional LSTMs [Schuster&Paliwal (1997), Graves&Schmidhuber (2005)]
- Continuous time RNNs

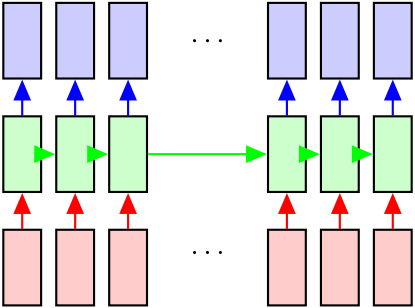
How do we learn the parameters in the network?

$$p(x_1, \ldots, x_T) = \prod_{i=1}^{T} p(x_i | x_1, \ldots x_{i-1})$$

Maximum likelihood specifies loss function

Training via gradient descent:

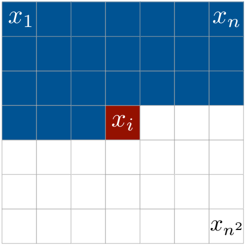
- How?
- What order?
- What information do we need to store?



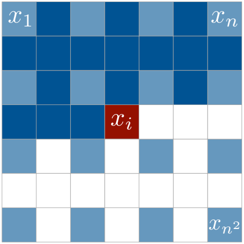

**Backpropagation through time (BPTT)**

# Pixel Recurrent Neural Networks

PixelRNN model (Autoregressive model):




Context



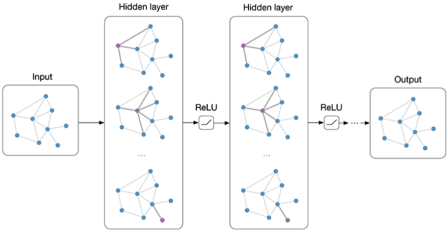

Multi-scale context

Generative models overview:

- Variational Auto-encoders (VAEs):
  - Pro: probabilistic graphical model interpretation
  - Con: slightly blurry examples
- Generative Adversarial Nets (GANs):
  - Pro: generate sharp images
  - Con: difficult to optimize (unstable)
- Autoregressive models (RNNs):
  - Pro: stable training & good likelihoods
  - Con: inefficient sampling & no low-dimensional codes

Very active research area

## Graph Convolutional Neural Nets

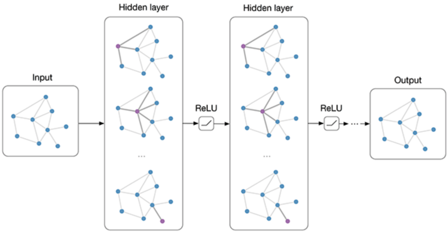



Operation:

$$H^{(l+1)} = f(H^{(l)}, A)$$

- Representation: $H^{(l)}$; $H^{(0)} = X$
- Graph adjacency matrix: $A$
- Nonlinearity: $f(\cdot, \cdot)$

## Graph Convolutional Neural Nets




Example:

$$H^{(l+1)} = \sigma(AH^{(l)}W^{(l)})$$

Note: more complex incarnations exist

**Quiz:**

- Describe the prediction process for an RNN?
- Describe the training process for RNNs?
- Contrast generative modeling techniques?
- Why graph convolutional nets?

**Important topics of this lecture**

- Getting to know RNNs and its variants
- Getting to know their use
- Contrasting RNNs to generative models
- Graph convolutional nets

**Next up:**

Reinforcement learning