Sequence Modeling: Recurrent and Recursive Nets

Sargur Srihari
srihari@buffalo.edu

Topics

- Recurrent Neural Networks

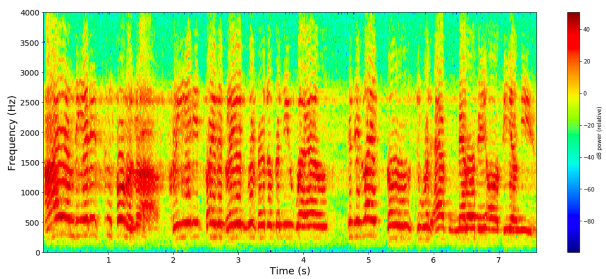
1. Unfolding Computational Graphs
2. Recurrent Neural Networks
3. Bidirectional RNNs
4. Encoder-Decoder Sequence-to-Sequence Architectures
5. Deep Recurrent Networks
6. Recursive Neural Networks
7. The Challenge of Long-Term Dependencies
8. Echo-State Networks
9. Leaky Units and Other Strategies for Multiple Time Scales
10. LSTM and Other Gated RNNs
11. Optimization for Long-Term Dependencies
12. Explicit Memory

RNNs process sequential data

- Recurrent Neural Networks are a family of neural networks for processing sequential data
- RNN and CNN are both specialized architectures
- Just as CNN is specialized for processing grid of values, e.g., image
  - RNN is specialized for processing a sequence of values $\boldsymbol{x}^{(1)},..,\boldsymbol{x}^{(\tau)}$
- Just as CNNs can readily scale images with large width/height and process variable size images
  - RNNs can scale to much longer sequences than would be practical for networks without sequence-based specialization
  - RNNs can also process variable-length sequences

Examples of Sequential Data and Tasks

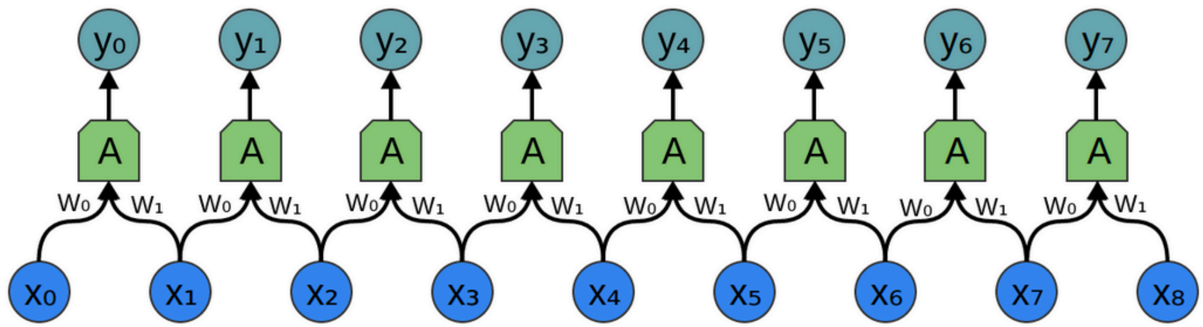
- Sequence data: sentences, speech, stock market, signal data
- Sequence-to-sequence Tasks
  - Speech recognition
    - decompose sound waves into frequency and amplitude using Fourier transforms yielding a spectrogram shown



  - Named Entity Recognition
    - Input: Jim bought 300 shares of Acme Corp. in 2006
    - NER: $[\text{Jim}]_{\text{Person}}$ bought 300 shares of $[\text{Acme Corp.}]_{\text{Organization}}$ in $[2006]_{\text{Time}}$
- Sequence-to-symbol
  - Sentiment
  - Speaker recognition

Neural network for 1-D convolution

$f\,[t]$



Kernel $g(t)$:

$[\ldots 0, w_1, w_0, 0\ldots]$

Equations for outputs of this network:

$$\boxed{\begin{aligned} y_0 &= \sigma(W_0 x_0 + W_1 x_1 - b) \\ y_1 &= \sigma(W_0 x_1 + W_1 x_2 - b) \end{aligned}}$$

etc. upto y_8

Note that kernel gets flipped in convolution

We can also write the equations in terms of elements of a general 8×8 weight matrix W as:

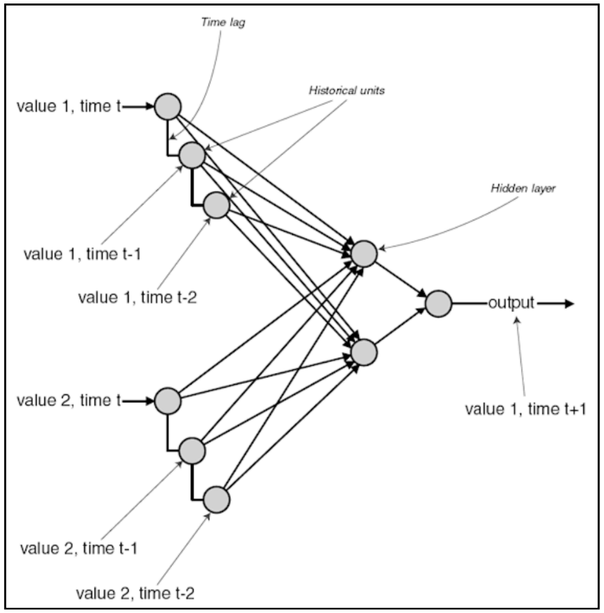
$$\boxed{\begin{aligned} y_0 &= \sigma(W_{0,0} x_0 + W_{0,1} x_1 + W_{0,2} x_2 \ldots) \\ y_1 &= \sigma(W_{1,0} x_0 + W_{1,1} x_1 + W_{1,2} x_2 \ldots) \end{aligned}}$$

where $W = \begin{bmatrix} w_0 & w_1 & 0 & 0 & \ldots \\ 0 & w_0 & w_1 & 0 & \ldots \\ 0 & 0 & w_0 & w_1 & \ldots \\ 0 & 0 & 0 & w_0 & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \end{bmatrix}$

http://colah.github.io/posts/2014-07-Understanding-Convolutions/

Time Delay Neural Networks

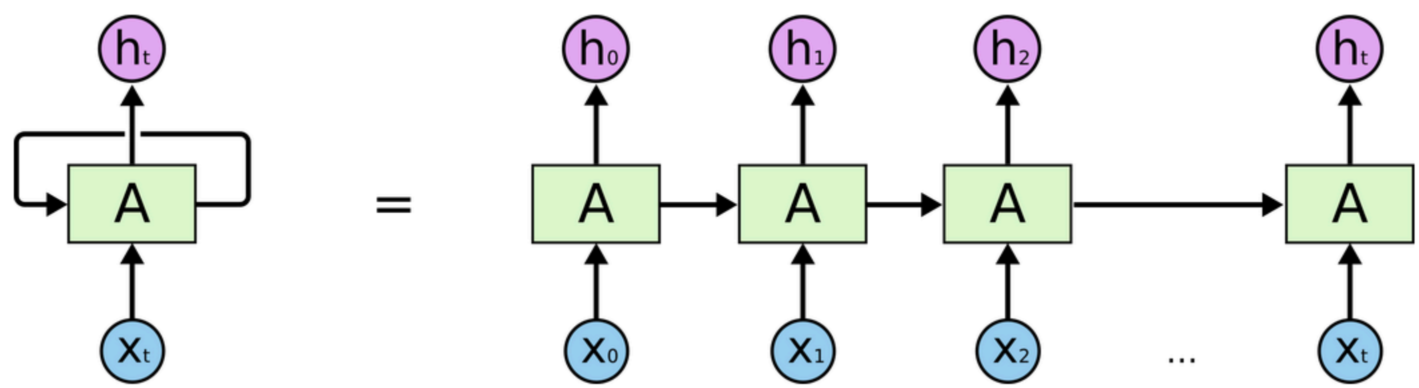
- Time-delay neural networks perform convolution across 1-D temporal sequence
  - Convolution operation allows a network to share parameters across time, but is shallow
    - Each member of output is dependent upon a small no. of neighboring members of the input
    - Parameter sharing manifests in the application of the same convolutional kernel at each time step



A TDNN remembers the previous few training examples and uses them as input into the network. The network then works like a feed-forward, back propagation network.

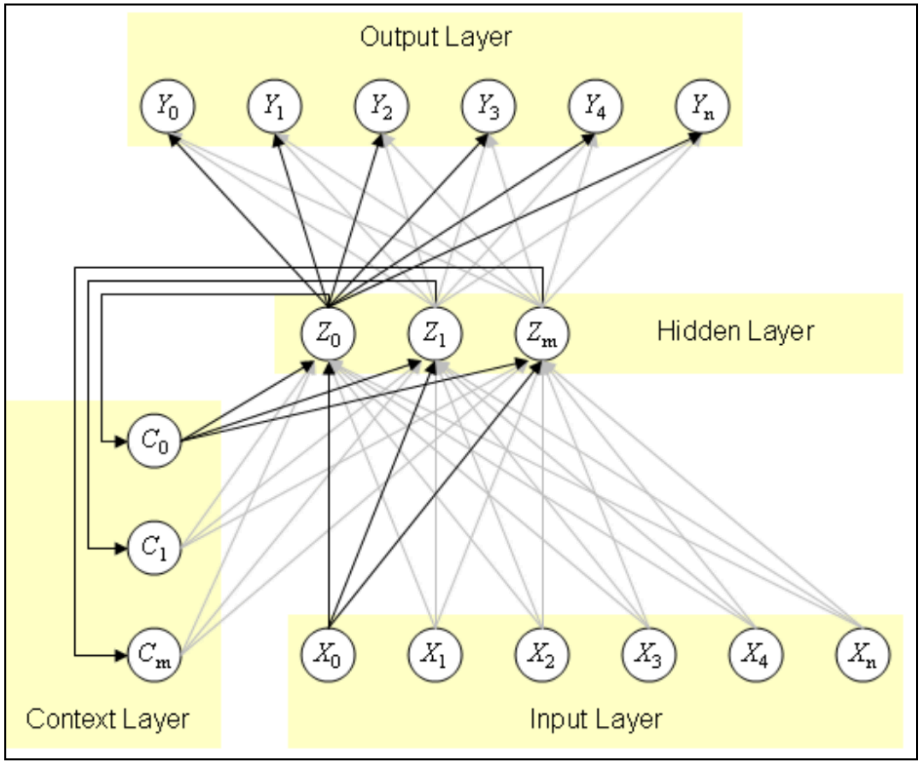
RNN vs. TDNN

- RNNs share parameters in a different way
  - Each member of output is a function of previous members of output
  - Each output produced using same update rule applied to previous outputs
  - This recurrent formulation results in sharing of parameters through a very deep computational graph
- An unrolled RNN



RNN as a network with cycles

- An RNN is a class of neural networks where connections between units form a directed cycle
- This creates an internal state of the network which allows it to exhibit dynamic temporal behavior
- The internal memory can be used to process arbitrary sequences of inputs



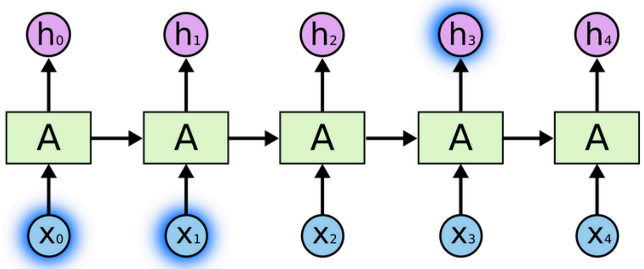
Three layer network with input $\boldsymbol{x}$, hidden layer $\boldsymbol{z}$ and output $\boldsymbol{y}$
Context units $\boldsymbol{c}$ maintain a copy of the previous value of the hidden units

RNNs share same weights across Time Steps

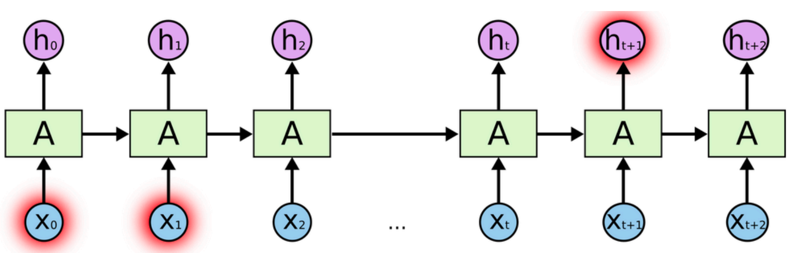
- To go from multi-layer networks to RNNs:
  - Need to share parameters across different parts of a model
  - Separate parameters for each value of cannot generalize to sequence lengths not seen during training
  - Share statistical strength across different sequence lengths and across different positions in time
- Sharing important when information can occur at multiple positions in the sequence
  - Given "$I\ went\ to\ Nepal\ in\ 1999$" and "$In\ 1999,\ I\ went\ to\ Nepal$", an ML method to extract year, should extract 1999 whether in position 6 or 2
  - A feed-forward network that processes sentences of fixed length would have to learn all of the rules of language separately at each position
  - An RNN shares the same weights across several time steps

Problem of Long-Term Dependencies

- Easy to predict last word in "the clouds are in the *sky*,"
  - When gap between relevant information and place that it's needed is small, RNNs can learn to use the past information



- "I grew up in France… I speak fluent *French*."
  - We need the context of France, from further back.
  - Large gap between relevant information and point where it is needed



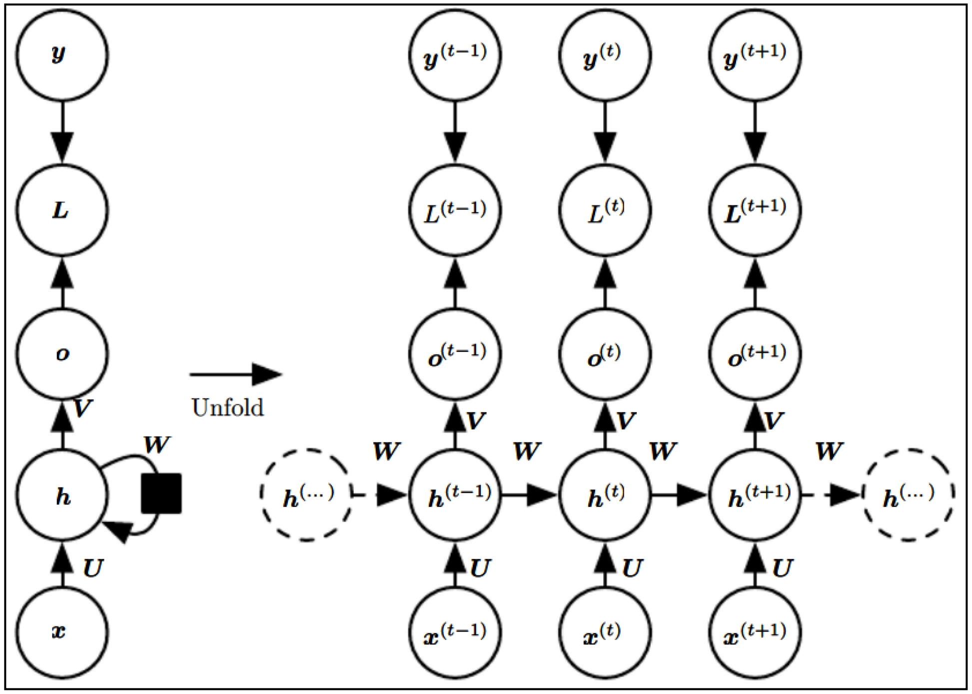
- In principle RNNs can handle it, but fail in practice
  - LSTMs offer a solution

RNN operating on a sequence

- RNNs operate on a sequence that contain vector $\boldsymbol{x}^{(t)}$ with time step index t, ranging from 1 to τ
  - Sequence: $\boldsymbol{x}^{(1)},..,\boldsymbol{x}^{(\tau)}$
  - RNNs operate on minibatches of sequences of length τ
- Some remarks about sequences
  - The steps need not refer to passage of time in the real world
  - RNNs can be applied in two-dimensions across spatial data such as image
  - Even when applied to time sequences, network may have connections going backwards in time, provided entire sequence is observed before it is provided to network

Computational Graphs for RNNs

- We extend computational graphs to include cycles
  - Cycles represent the influence of the present value of a variable on its own value at a future time step
  - In a Computational graph nodes are variables/operations
  - RNN to map input sequence of $\boldsymbol{x}$ values to output sequence of $\boldsymbol{o}$ values
    - Loss $\boldsymbol{L}$ measures how far each output $\boldsymbol{o}$ is from the training target $\boldsymbol{y}$



- Forward propagation is given as follows:

  For each time step t, $\mathrm{t}=1$ to $\mathrm{t}=\tau$

  Apply the following equations

  $\boldsymbol{o}^{(\mathrm{t})}=\boldsymbol{c}+V\boldsymbol{h}^{(\mathrm{t})}$

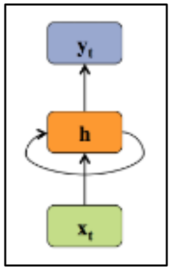
  $\boldsymbol{h}^{(\mathrm{t})}=\tanh(\boldsymbol{a}^{(\mathrm{t})})$

  $\boldsymbol{a}^{(\mathrm{t})}=\boldsymbol{b}+W\boldsymbol{h}^{(\mathrm{t-1})}+U\boldsymbol{x}^{(\mathrm{t})}$

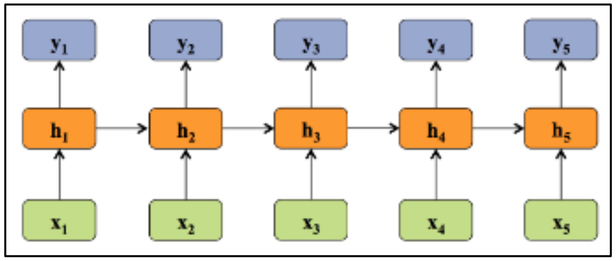
Summary of Neural Sequential Models

Recurrent Neural Network

RNN



Unrolled RNN



Definition

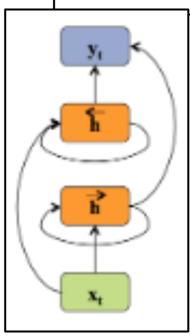
$$\begin{aligned} \text{inputs} &: x = (x_1, x_2, ..., x_T), x_i \in \mathbb{R}^I \\ \text{hidden units} &: h = (h_1, h_2, ..., h_T), h_i \in \mathbb{R}^J \\ \text{outputs} &: y = (y_1, y_2, ..., y_T), y_i \in \mathbb{R}^K \\ \text{nonlinearity} &: \mathcal{H} \end{aligned}$$

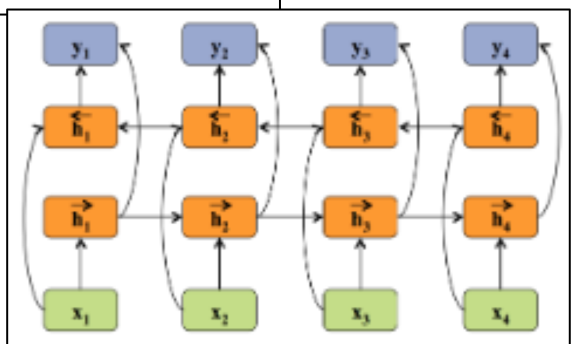
Activation Functions

$$h_t = \mathcal{H}(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$
$$y_t = W_{hy}h_t + b_y$$

Bidirectional RNN

Two types of hidden layers: one with forward loop other with backward loop

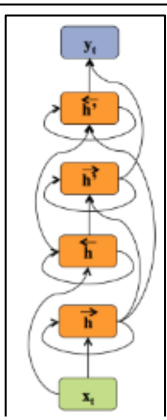




$$\begin{aligned} \text{inputs} &: x = (x_1, x_2, ..., x_T), x_i \in \mathbb{R}^I \\ \text{hidden units} &: \overrightarrow{h} \text{ and } \overleftarrow{h} \\ \text{outputs} &: y = (y_1, y_2, ..., y_T), y_i \in \mathbb{R}^K \\ \text{nonlinearity} &: \mathcal{H} \end{aligned}$$

$$\overrightarrow{h}_t = \mathcal{H}(W_{x\overrightarrow{h}}x_t + W_{\overrightarrow{h}\overrightarrow{h}}\overrightarrow{h}_{t-1} + b_{\overrightarrow{h}})$$
$$\overleftarrow{h}_t = \mathcal{H}(W_{x\overleftarrow{h}}x_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t-1} + b_{\overleftarrow{h}})$$
$$y_t = W_{\overleftarrow{h}y}\overleftarrow{h}_t + W_{\overrightarrow{h}y}\overrightarrow{h}_t + b_y$$

Deep Bidirectional RNN

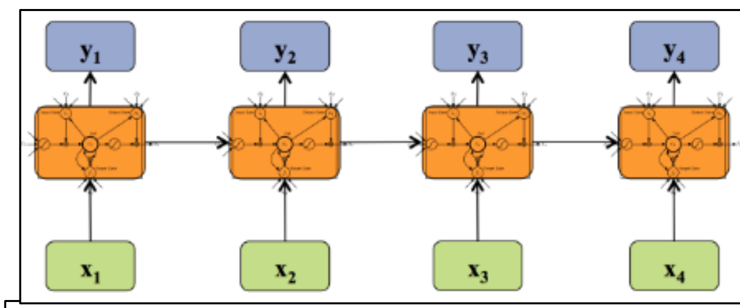


RNNs and Bidirectional RNNs with several hidden layers

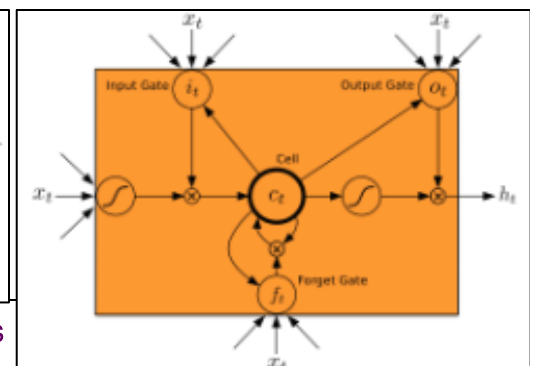
Source: http://www.cs.cmu.edu/~epxing/Class/10708-16/note/10708_scribe_lecture27.pdf

LSTM

Unrolled LSTM



LSTM Hidden Unit



Input gate i_t which masks out standard RNN inputs
Forget gate f_t which masks out the previous cell
Cell c_t combines input with forget mask to learn to keep current state of LSTM
Output gate o_t masks out values of next hidden input