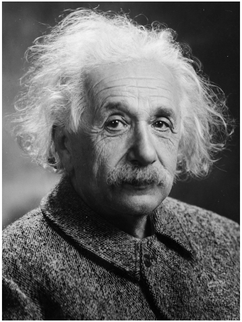
CS-568 Deep Learning

Recurrent Neural Networks

Nazar Khan
Department of Computer Science
University of the Punjab



Everything should be made as simple as possible, but no simpler.

Albert Einstein

Understanding Recurrent Neural Networks requires some effort and a correct perspective. Do not expect them to be as simple as linear regression.

Static vs. Dynamic Inputs

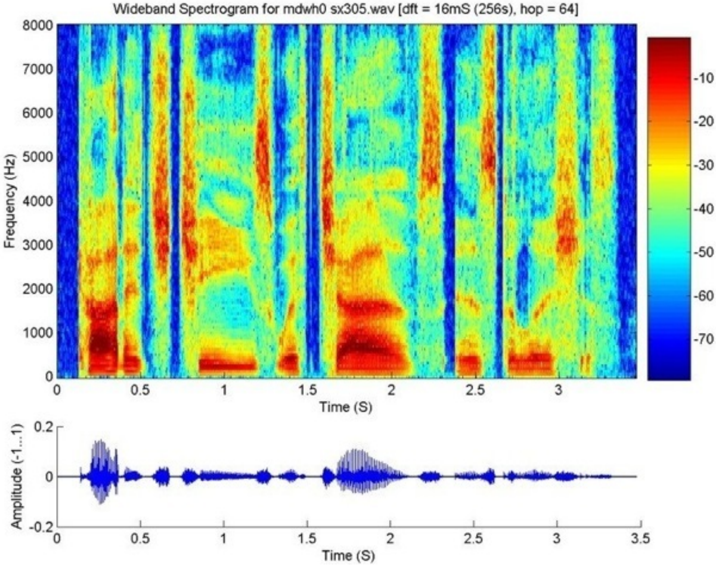
- *Static* signals, such as an image, do not change over time.
  - Ordered with respect to space.
  - Output depends on current input.
- *Dynamic* signals, such as text, audio, video or stock price change over time.
  - Ordered with respect to time.
  - Output depends on current input as well as past (or even future) inputs.
  - Also called *temporal*, *sequential* or *time-series* data.

Context in Text

The Taj ____ was commissioned by Shah Jahan in 1631, to be built in the memory of ___ wife Mumtaz Mahal, who died on 17 June that *year, giving birth to* their *14th child, Gauhara Begum.* Construction *started in 1632, and the mausoleum was completed ___ 1643.*

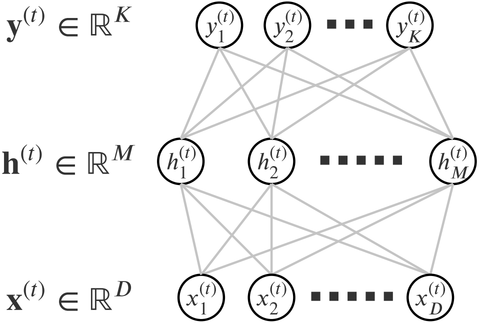
Context in Video

Context in Audio



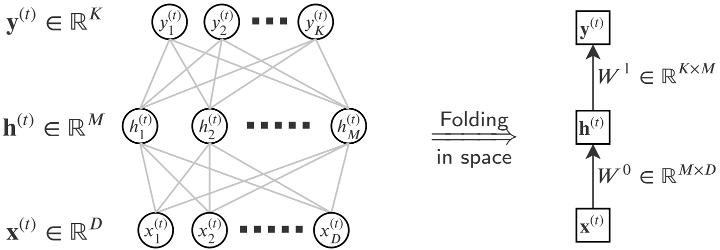
Time-series Data

- A *single* input will be a *series of vectors* $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^T$.



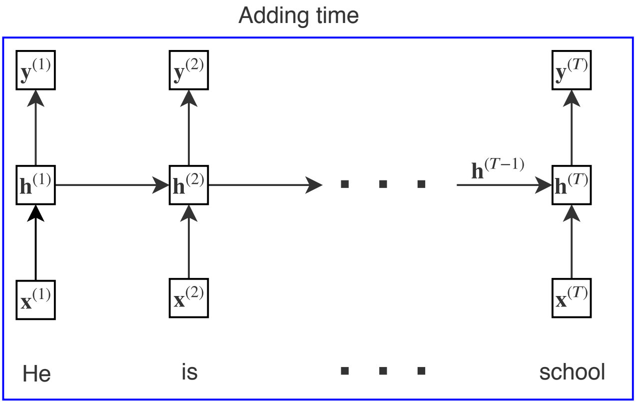
Input component at time t forward propagated through a network.

Representational Shortcut 1 – Space Folding



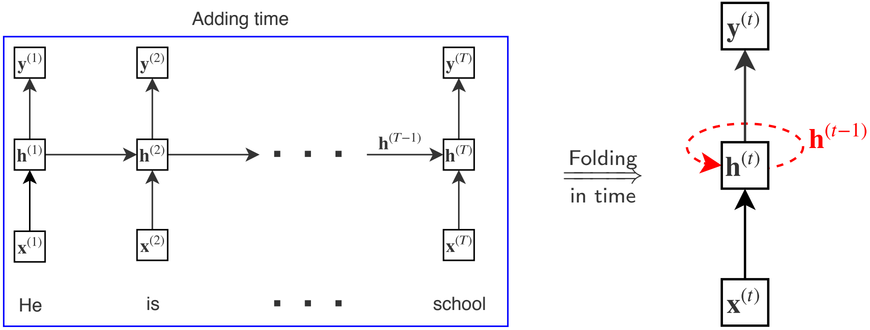
Each box represents a layer of neurons.

Recurrent Neural Networks



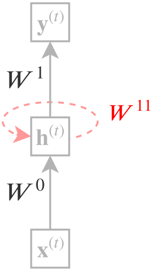
- A recurrent neural network (RNN) makes hidden state at time t directly dependent on the hidden state at time $t-1$ and therefore indirectly *on all previous times*.
- Output $\mathbf{y}_t$ depends on all that the network has already seen so far.

Representational Shortcut 2 – Time Folding



Recurrent Neural Networks

3 sets of weights

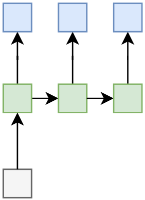


$$\mathbf{y}^{(t)} = f(\overbrace{W^1\mathbf{h}^{(t)} + \mathbf{b}_1}^{\mathbf{a}^{1(t)}})$$

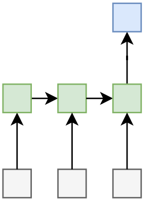
$$\mathbf{h}^{(t)} = \tanh(\underbrace{W^0\mathbf{x}^{(t)} + W^{11}\mathbf{h}^{(t-1)} + \mathbf{b}_0}_{\mathbf{a}^{0(t)}})$$

Sequence Mappings

One-to-many



Many-to-one



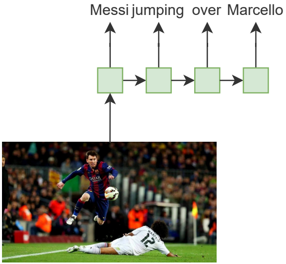
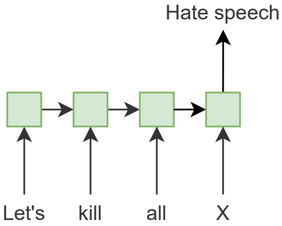
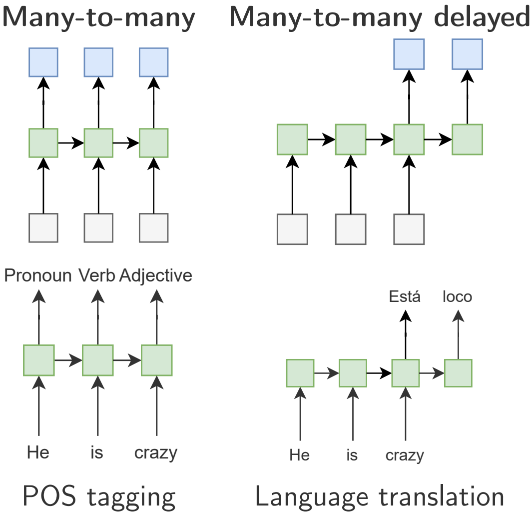


Image caption generation



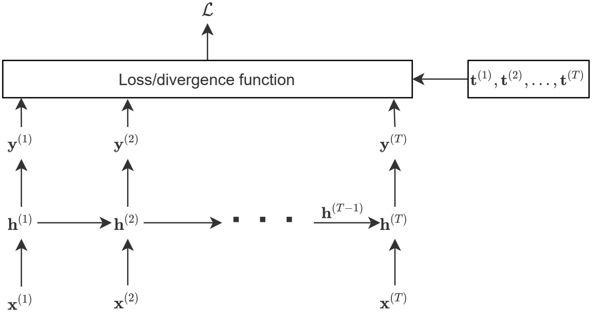
Sentiment classification

Sequence Mappings



Loss Functions for Sequences

- For recurrent nets, loss is between *series* of output and target vectors. That is $\mathcal{L}(\{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(T)}\}, \{\mathbf{t}^{(1)}, \ldots, \mathbf{t}^{(}T)\})$.

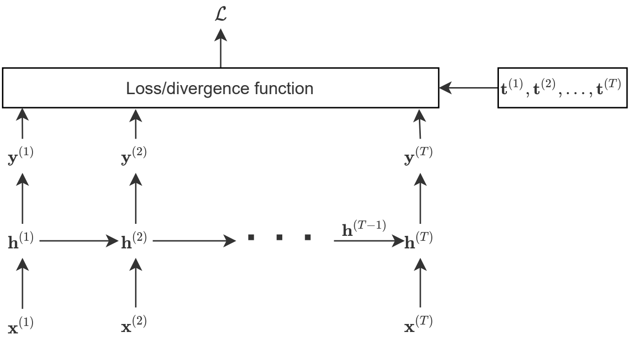


Forward propagation in an RNN unfolded in time.

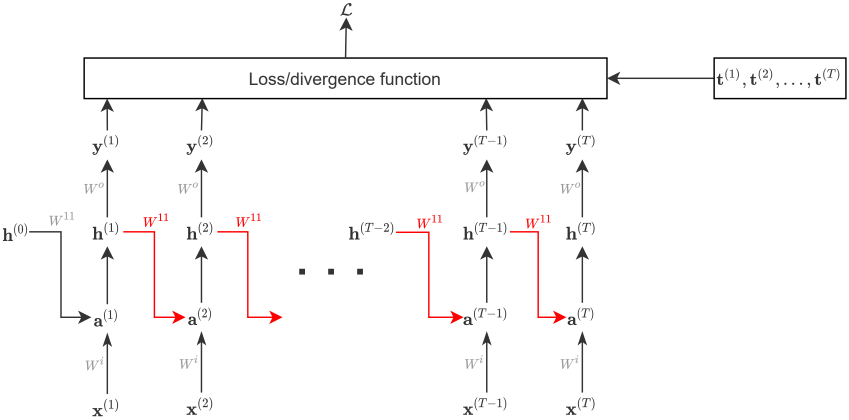
- Notice that loss $\mathcal{L}$ can be computed only after $\mathbf{y}^{(T)}$ has been computed.

Loss Functions for Sequences

- Loss is *not necessarily* decomposable.
- In the following, we will assume decomposable loss $\mathcal{L} = \sum_{t=1}^{T} \mathcal{L}(\mathbf{y}^{(t)}, \mathbf{t}^{(t)})$.
- In both cases, as long as $\frac{\partial L}{\partial \mathbf{y}^{(t)}}$ has been computed, backpropagation can proceed.

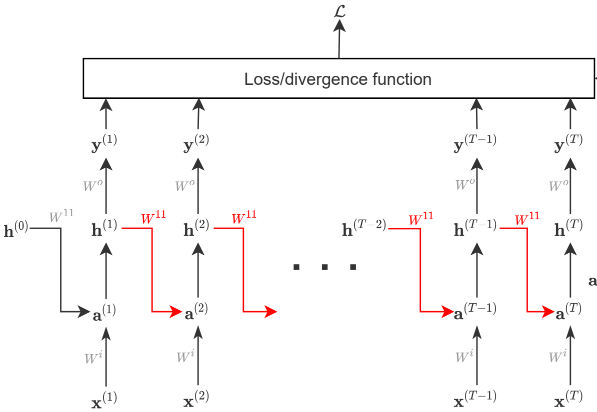


Forward Propagation Through Time



Forward propagation in an RNN unfolded in time. Recurrence between hidden states through pre-activation $\mathbf{a}^{(t)}$ is shown in red.

Forward Propagation Through Time



$$\mathbf{y}^{(T)} = f(W^o\mathbf{h}^{(T)} + \mathbf{b}^o)$$

$$\mathbf{h}^{(T)} = \tanh(\mathbf{a}^{(T)})$$

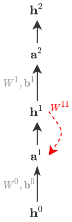
$$\mathbf{a}^{(T)} = W^i\mathbf{x}^{(T)} + W^{11}\mathbf{h}^{(T-1)} + \mathbf{b}^i$$

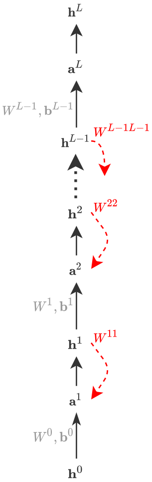
Forward propagation in an RNN unfolded in time. Recurrence between hidden states through pre-activation $\mathbf{a}^{(t)}$ is shown in red.

Notational Clarity

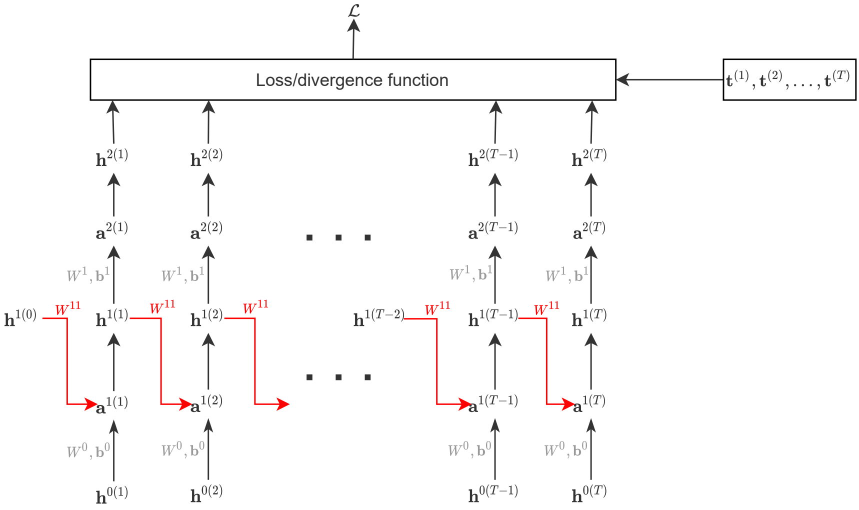
- At layer l, we will denote the pre-activation by $\mathbf{a}^l$ and activation by $\mathbf{h}^l$.
- So output layer $\mathbf{y}$ will be denoted by $\mathbf{h}^L$ in an L-layer network.
- Input will be denoted by $\mathbf{h}^0$.
- So forward propagation entails $\mathbf{h}^0 \to \underbrace{\mathbf{a}^1 \to \mathbf{h}^1}_{\text{layer } 1} \cdots \to \underbrace{\mathbf{a}^{L-1} \to \mathbf{h}^{L-1}}_{\text{layer } L-1} \to \underbrace{\mathbf{a}^L \to \mathbf{h}^L}_{\text{layer } L}$.
- For 2 layer network

$$\begin{aligned}
\mathbf{h}^{2,T} &= f(\mathbf{a}^{2,T}) \\
\mathbf{a}^{2,T} &= W^1\mathbf{h}^{1,T} + \mathbf{b}^1 \\
\mathbf{h}^{1,T} &= \tanh(\mathbf{a}^{1,T}) \\
\mathbf{a}^{1,T} &= W^0\mathbf{h}^{0,T} + {\color{red}W^{11}\mathbf{h}^{1,T-1}} + \mathbf{b}^0
\end{aligned}$$

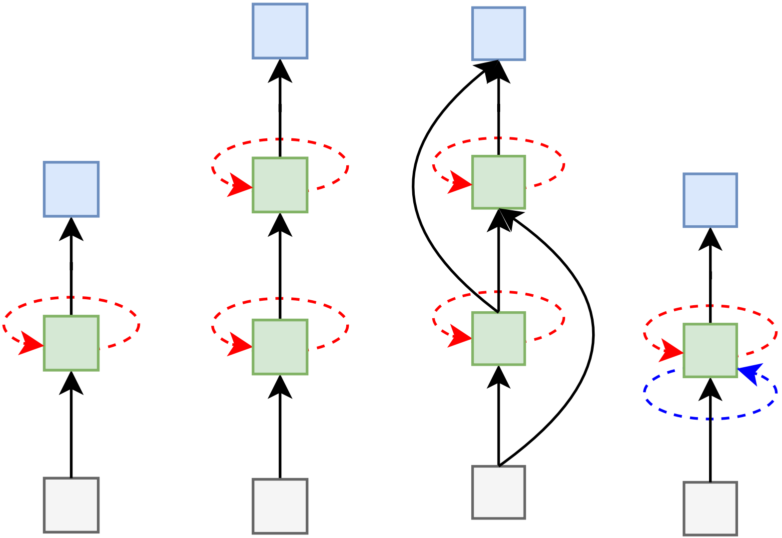




Notational Clarity



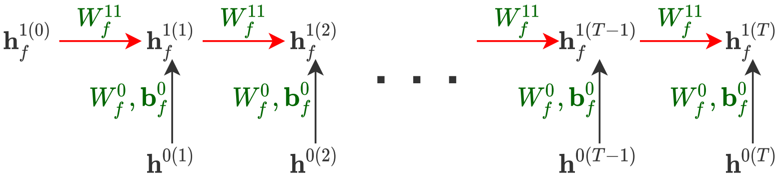
RNN Variations



1 hidden state   2 hidden states   Skip connections   Bidirectional

Bidirectional RNN

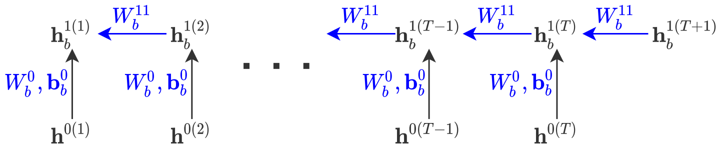
Step 1: Forward propagation into the future



$$fprop\left(\underbrace{\mathbf{h}^0(1), \mathbf{h}^0(2), \dots, \mathbf{h}^0(T)}_{\text{input sequence}}; \underbrace{\mathbf{h}_f^1(0)}_{\text{init}}, \underbrace{W_f^0, \mathbf{b}_f^0, W_f^{11}, W_f^1, \mathbf{b}_f^1}_{\text{parameters}}\right)$$

Bidirectional RNN

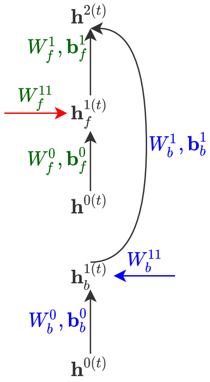
Step 2: Forward propagation into the past



$$fprop\left(\underbrace{\mathbf{h}^0(T), \mathbf{h}^0(T-1), \ldots, \mathbf{h}^0(1)}_{\text{input sequence}}; \underbrace{\mathbf{h}_b^1(T+1)}_{\text{init}}, \underbrace{W_b^0, \mathbf{b}_b^0, W_b^{11}, W_b^1, \mathbf{b}_b^1}_{\text{parameters}}\right)$$

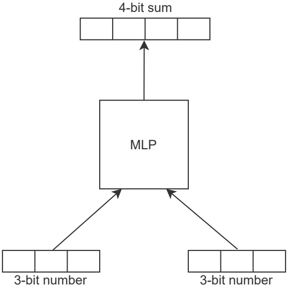
Bidirectional RNN

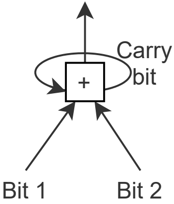
Step 3: Fusion of forward and backward hidden states



$$\mathbf{h}^{2(t)} = \tanh(\mathbf{a}^{2(t)})$$
$$\mathbf{a}^{2(t)} = W_f^1 \mathbf{h}_f^{1(t)} + \mathbf{b}_f^1 + W_b^1 \mathbf{h}_b^{1(t)} + \mathbf{b}_b^1$$

Benefit of Recurrent Architectures
n-bit addition

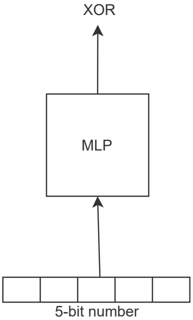




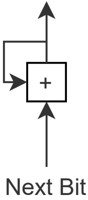
- Only for n-bit numbers
- Training set exponential in n
- Training errors

- Iterative application to numbers of arbitrary size
- Exact answers

Benefit of Recurrent Architectures
n-bit XOR



- Only for *n*-bit numbers
- Training set exponential in *n*
- Training errors



- Iterative application to numbers of arbitrary size
- Exact answer
- Will be 1 for odd number of ones in input.

Stability issues

- Even a 1-hidden layer RNN is a very deep network.
- Viewed in time, an RNN is as deep as the number of time steps.
- Suffers from vanishing gradients.
- Also suffers from *exploding gradients*.

Exploding Gradients

- Consider input $\mathbf{x}^{(1)}$ at time 1 and *assume linear* hidden layer.
- At time t, the RNN carries a term of the form

$$W^{11} \dots W^{11} W^{0} \mathbf{x}^{(1)} = \left(W^{11}\right)^{t-1} W^{0} \mathbf{x}^{(1)}$$

  which is an M-dimensional vector.
- Magnitude of this vector depends on largest eigenvalue λ_{max} of W^{11}.
  - $\lambda_{\text{max}} > 1 \implies$ magnitude of $\left(W^{11}\right)^{t-1} W^{0} \mathbf{x}^{(1)}$ keeps increasing.
  - $\lambda_{\text{max}} < 1 \implies$ magnitude of $\left(W^{11}\right)^{t-1} W^{0} \mathbf{x}^{(1)}$ keeps decreasing.

Exploding Gradients

- Even during forward propagation, depending on the largest eigenvalue of the recurrent weight matrix W^{11}, input at time t
  - is either forgotten very soon,
  - or explodes to very large values.
- Similar case for backpropagation.
- Notice that this has nothing to do with the choice of activation function.
- *Information will explode or vanish through time*.
- Similar behaviour for non-linear neurons.
- So, in practice, RNNs do not have long-term memory. Solution: LSTM.

Summary

- Dynamic signals that change over time can be modeled by RNNs.
- RNNs can represent mappings of one-to-many, many-to-one, many-to-many, and many-to-many delayed sequences.
- Previous hidden layer output influences subsequent output.
- Forward propagation is through space as well as time.
- Forward propagation can be into the future and/or into the past.
- Recurrent connection implies really deep network.
- Information can vanish or even explode through time.