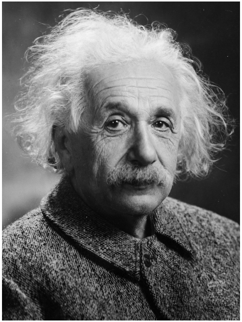
CS-568 Deep Learning

Nazar Khan

PUCIT

Recurrent Neural Networks

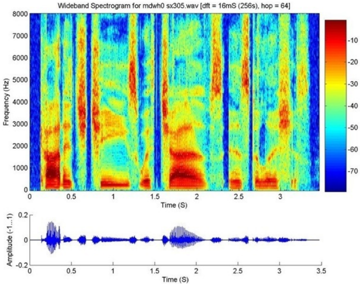


Everything should be made as simple as possible, but no simpler.
Albert Einstein

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.

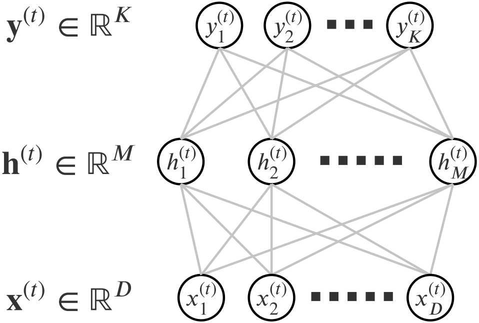


Context in Text

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.*

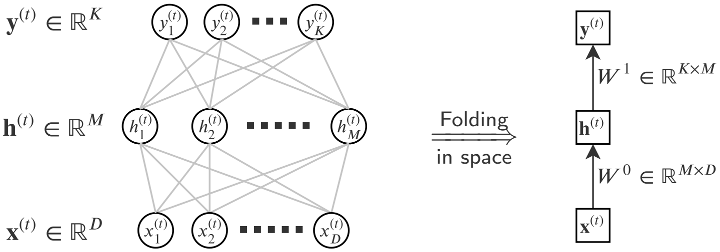
Time-series Data

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



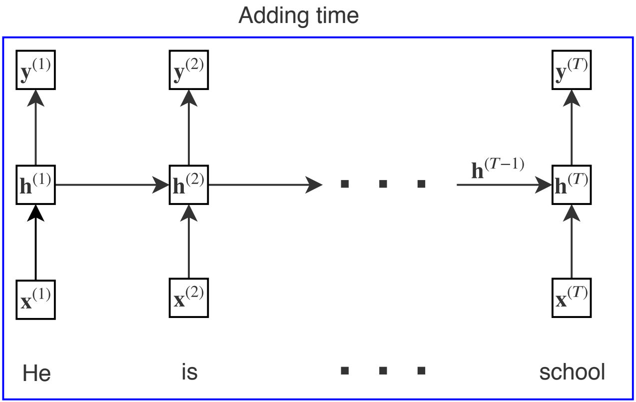
Input component at time t forward propagated through a network.

Representational Shortcut 1 – Space Folding



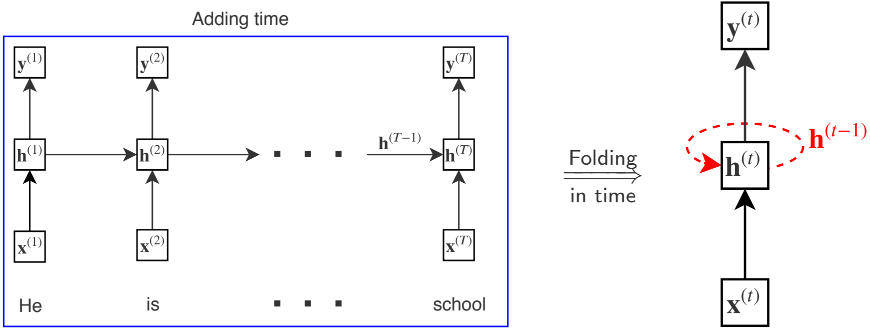
Each box represents a layer of neurons.

Recurrent Neural Networks



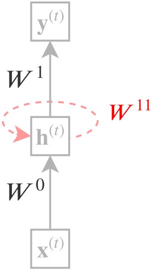
- A recurrent neural network (RNN) makes hidden state at time t directly dependent on the hidden state at time $t-1$ and therefore indirectly *on all previous times*.
- Output $\mathbf{y}_t$ depends on all that the network has already seen so far.

Representational Shortcut 2 – Time Folding



Recurrent Neural Networks

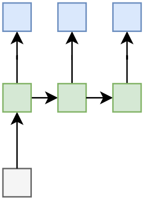
3 sets of weights



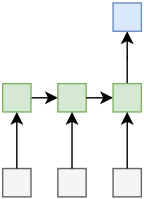
$$\mathbf{y}^{(t)} = f(\overbrace{W^1\mathbf{h}^{(t)} + \mathbf{b}_1}^{\mathbf{z}^{1(t)}})$$

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

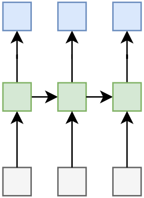
Sequence Mappings



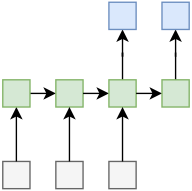
One-to-many



Many-to-one



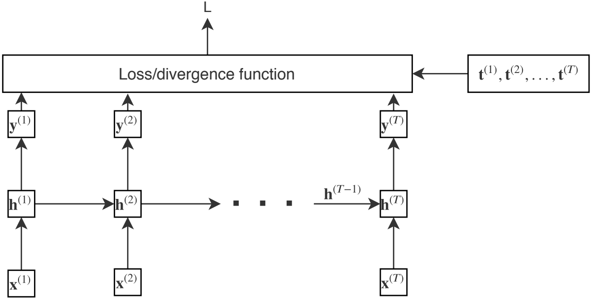
Many-to-many



Many-to-many delayed

Loss Functions for Sequences

- For recurrent nets, loss is between *series* of output and target vectors. That is $\mathcal{L}(\{\mathbf{y}_1, \ldots, \mathbf{y}_T\}, \{\mathbf{t}_1, \ldots, \mathbf{t}_T\})$.

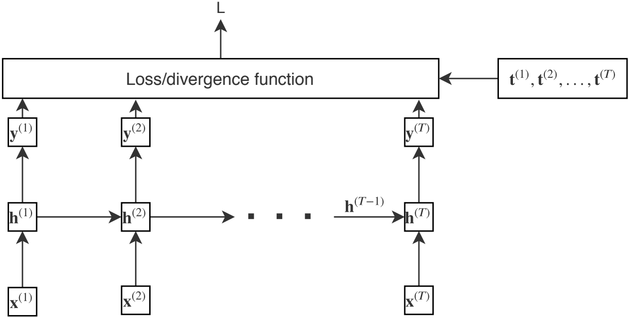


Forward propagation in an RNN unfolded in time.

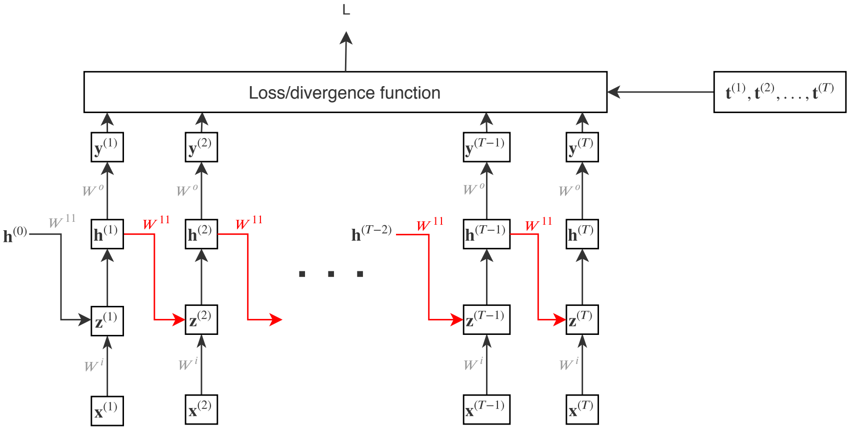
- Notice that loss $\mathcal{L}$ can be computed only after $\mathbf{y}_T$ has been computed.

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.

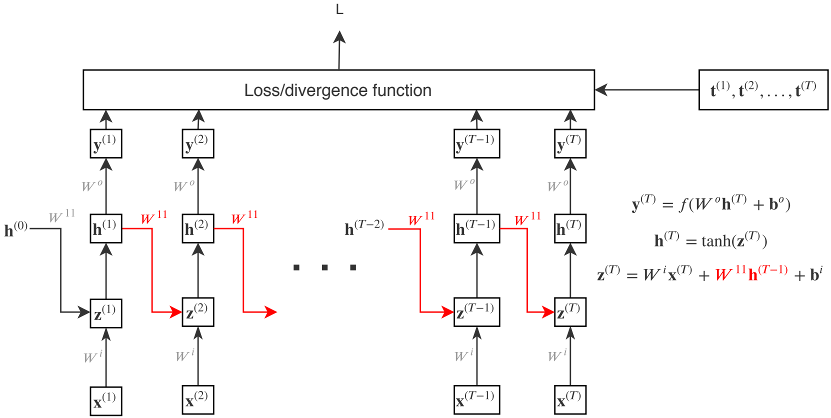


Backpropagation Through Time (BPTT)



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

BPTT

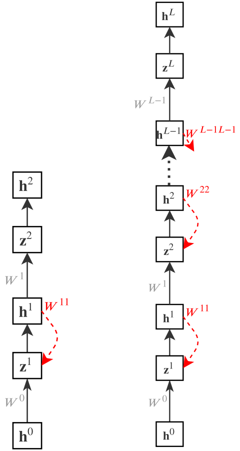


Forward propagation in an RNN unfolded in time. Recurrence between hidden states through pre-activation $\mathbf{z}^{(t)}$ is shown in red.

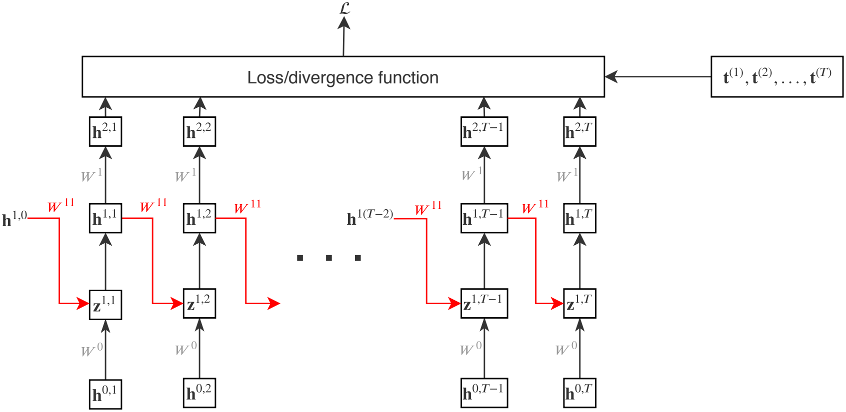
BPTT – Notational Clarity

- For notational clarity, at layer l, we will denote the pre-activation by $\mathbf{z}^l$ 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 $\mathbf{h}^0 \rightarrow \mathbf{z}^1 \rightarrow \mathbf{h}^1 \cdots \rightarrow \mathbf{z}^{L-1} \rightarrow \mathbf{h}^{L-1} \rightarrow \mathbf{z}^L \rightarrow \mathbf{h}^L$.
- For 2 layer network

$$\mathbf{h}^{2,T} = f(W^1\mathbf{h}^{1,T} + \mathbf{b}^1)$$
$$\mathbf{h}^{1,T} = \tanh(\mathbf{z}^{1,T})$$
$$\mathbf{z}^{1,T} = W^0\mathbf{h}^{0,T} + W^{11}\mathbf{h}^{1,T-1} + \mathbf{b}^0$$



BPTT – Notational Clarity



Multivariate Chain Rule

- The chain rule of differentiation states

$$\frac{df(u(x))}{dx} = \frac{df}{du}\frac{du}{dx}$$

  when f depends on x through $u()$.
- The *multivariate* chain rule of differentiation states

$$\frac{df(u(x), v(x))}{dx} = \frac{\partial f}{\partial u}\frac{du}{dx} + \frac{\partial f}{\partial v}\frac{dv}{dx}$$

  when f depends on x through $u()$ *and* through $v()$.
- Backpropagation is just an application of the multivariate chain rule.

BPTT

- We need 5 derivatives:
  1. $\nabla_{W^1}\mathcal{L} \in \mathbb{R}^{M\times K}$
  2. $\nabla_{\mathbf{b}^1}\mathcal{L} \in \mathbb{R}^{1\times K}$
  3. $\nabla_{W^{11}}\mathcal{L} \in \mathbb{R}^{M\times M}$
  4. $\nabla_{W^0}\mathcal{L} \in \mathbb{R}^{D\times M}$
  5. $\nabla_{\mathbf{b}^0}\mathcal{L} \in \mathbb{R}^{1\times M}$

BPTT

Derivative number 1 : $\nabla_{W^1}\mathcal{L}$

- Notice that W^1 affects loss $\mathcal{L}$ through $\mathbf{z}^{(2,t)}$ at each time t.

$$\mathcal{L}(\underbrace{\mathbf{z}^{(2,1)}(W^1)}_{t=1}, \underbrace{\mathbf{z}^{(2,2)}(W^1)}_{t=2}, \dots, \underbrace{\mathbf{z}^{(2,T)}(W^1)}_{t=T})$$

- Influence diagram
- Using the multivariate chain rule over time

$$\underbrace{\nabla_{W^1}\mathcal{L}}_{M\times K} = \sum_{t=1}^{T} \underbrace{\nabla_{\mathbf{z}^{(2,t)}}\mathcal{L}}_{1\times K} \underbrace{\nabla_{W^1}\mathbf{z}^{(2,t)}}_{K\times(M\times K)}$$

$$= \sum_{t=T}^{1} \underbrace{\mathbf{h}^{(1,t)}}_{M\times 1} \underbrace{\nabla_{\mathbf{z}^{(2,t)}}\mathcal{L}}_{1\times K}$$

- Computation of $\nabla_{\mathbf{z}^{(2,t)}}\mathcal{L}$ is described next.

BPTT
$\nabla_{\mathbf{z}^{(2,t)}}\mathcal{L}$

- The derivatives of loss $\mathcal{L}$ w.r.t pre-activations $\mathbf{z}^{(2,t)}$ can be computed as

$$\underbrace{\nabla_{\mathbf{z}^{(2,t)}}\mathcal{L}}_{1\times K} = \underbrace{\nabla_{\mathbf{h}^{(2,t)}}\mathcal{L}}_{1\times K}\underbrace{\nabla_{\mathbf{z}^{(2,t)}}\mathbf{h}^{(2,t)}}_{K\times K} = \nabla_{\mathbf{h}^{(2,t)}}\mathcal{L}\underbrace{\begin{bmatrix} \partial_{z_1}h_1 & \partial_{z_2}h_1 & \dots & \partial_{z_K}h_1 \\ \partial_{z_1}h_2 & \partial_{z_2}h_2 & \dots & \partial_{z_K}h_2 \\ \vdots & \vdots & \ddots & \vdots \\ \partial_{z_1}h_K & \partial_{z_2}h_K & \dots & \partial_{z_K}h_K \end{bmatrix}^{(2,t)}}_{\text{Jacobian matrix}}$$

- The Jacobian matrix is the derivative of outputs with respect to inputs.
- In 1D, the term $\frac{dy}{dx}$ is the 1×1 Jacobian matrix of $y = f(x)$.
- For scalar activation functions such as logistic sigmoid, tanh, ReLU, the Jacobian matrix is diagonal.
- For vector activation functions such as softmax, the Jacobian matrix is full.

BPTT

Derivative number 2 : $\nabla_{\mathbf{b}^1}\mathcal{L}$

- Following the same reasoning as used for $\nabla_{W^1}\mathcal{L}$ above, we can compute

$$\underbrace{\nabla_{\mathbf{b}^1}\mathcal{L}}_{1\times K} = \sum_{t=T}^{1} \underbrace{\nabla_{\mathbf{z}^{(2,t)}}\mathcal{L}}_{1\times K}$$

where we have used the fact that $\nabla_{\mathbf{b}^1}\mathbf{z}^{(2,t)} = I_K$.

BPTT

Derivative number 3 : $\nabla_{W^{11}}\mathcal{L}$

- Notice that W^{11} affects loss $\mathcal{L}$ through $\mathbf{z}^{(1,t)}$ at each time t.

$$\mathcal{L}(\underbrace{\mathbf{z}^{(1,1)}(W^{11})}_{t=1}, \underbrace{\mathbf{z}^{(1,2)}(W^{11})}_{t=2}, \ldots, \underbrace{\mathbf{z}^{(1,T)}(W^{11})}_{t=T})$$

- Influence diagram
- Using the multivariate chain rule over time

$$\begin{aligned}\underbrace{\nabla_{W^{11}}\mathcal{L}}_{M\times M} &= \sum_{t=1}^{T} \underbrace{\nabla_{\mathbf{z}^{(1,t)}}\mathcal{L}}_{1\times M}\, \underbrace{\nabla_{W^{11}}\mathbf{z}^{(1,t)}}_{M\times(M\times M)} \\ &= \sum_{t=T}^{1} \underbrace{\mathbf{h}^{(1,t-1)}}_{M\times 1}\, \underbrace{\nabla_{\mathbf{z}^{(1,t)}}\mathcal{L}}_{1\times M}\end{aligned}$$

- Computation of $\nabla_{\mathbf{z}^{(1,t)}}\mathcal{L}$ is described next.

BPTT

$\nabla_{\mathbf{z}^{(1,t)}}\mathcal{L}$

- The derivatives of loss $\mathcal{L}$ w.r.t pre-activations $\mathbf{z}^{(1,t)}$ can be computed as

$$\underbrace{\nabla_{\mathbf{z}^{(1,t)}}\mathcal{L}}_{1\times M} = \underbrace{\color{red}{\nabla_{\mathbf{h}^{(1,t)}}\mathcal{L}}}_{1\times M}\underbrace{\color{blue}{\nabla_{\mathbf{h}^{(1,t)}}\mathbf{z}^{(1,t)}}}_{M\times M} = \color{red}{\nabla_{\mathbf{h}^{(1,t)}}\mathcal{L}}\color{blue}{\underbrace{\begin{bmatrix} \partial_{z_1}h_1 & \partial_{z_2}h_1 & \dots & \partial_{z_M}h_1 \\ \partial_{z_1}h_2 & \partial_{z_2}h_2 & \dots & \partial_{z_M}h_2 \\ \vdots & \vdots & \ddots & \vdots \\ \partial_{z_1}h_M & \partial_{z_2}h_M & \dots & \partial_{z_M}h_M \end{bmatrix}^{(1,t)}}_{\text{Jacobian matrix}}}$$

- Computation of $\color{red}{\nabla_{\mathbf{h}^{(1,t)}}\mathcal{L}}$ is described next.

BPTT
$\nabla_{\mathbf{h}^{(1,t)}}\mathcal{L}$

- Notice that $\mathbf{h}^{(1,t)}$ affects loss $\mathcal{L}$
  1. through $\mathbf{z}^{(2,t)}$ at each time t, and
  2. through $\mathbf{z}^{(1,t+1)}$ at each time $t+1$.
- Influence diagram
- Using the multivariate chain rule over these 2 time steps

$$\underbrace{\nabla_{\mathbf{h}^{(1,t)}}\mathcal{L}}_{1\times M} = \nabla_{\mathbf{z}^{(2,t)}}\mathcal{L}\nabla_{\mathbf{h}^{(1,t)}}\mathbf{z}^{(2,t)} + \underbrace{\nabla_{\mathbf{z}^{(1,t+1)}}\mathcal{L}\nabla_{\mathbf{h}^{(1,t)}}\mathbf{z}^{(1,t+1)}}_{\text{Not required when } t=T}$$

$$= \underbrace{\nabla_{\mathbf{z}^{(2,t)}}\mathcal{L}}_{1\times K}\underbrace{W^{1}}_{K\times M} + \underbrace{\nabla_{\mathbf{z}^{(1,t+1)}}\mathcal{L}}_{1\times M}\underbrace{W^{11}}_{M\times M}$$

BPTT

Derivative number 4 : $\nabla_{W^0}\mathcal{L}$

- Notice that W^0 affects loss $\mathcal{L}$ through $\mathbf{z}^{(1,t)}$ at each time t.

$$\mathcal{L}(\underbrace{\mathbf{z}^{(1,1)}(W^0)}_{t=1}, \underbrace{\mathbf{z}^{(1,2)}(W^0)}_{t=2}, \dots, \underbrace{\mathbf{z}^{(1,T)}(W^0)}_{t=T})$$

- Influence diagram
- Using the multivariate chain rule over time

$$\underbrace{\nabla_{W^0}\mathcal{L}}_{D\times M} = \sum_{t=1}^{T} \underbrace{\nabla_{\mathbf{z}^{(1,t)}}\mathcal{L}}_{1\times M}\, \underbrace{\nabla_{W^0}\mathbf{z}^{(1,t)}}_{M\times(D\times M)}$$

$$= \sum_{t=T}^{1} \underbrace{\mathbf{h}^{(0,t)}}_{D\times 1}\, \underbrace{\nabla_{\mathbf{z}^{(1,t)}}\mathcal{L}}_{1\times M}$$

BPTT

Derivative number 5 : $\nabla_{\mathbf{b}^0}\mathcal{L}$

- Following the same reasoning as used for $\nabla_{W^0}\mathcal{L}$ above, we can compute

$$\underbrace{\nabla_{\mathbf{b}^0}\mathcal{L}}_{1\times M} = \sum_{t=T}^{1} \underbrace{\nabla_{\mathbf{z}^{(1,t)}}\mathcal{L}}_{1\times M}$$

where we have used the fact that $\nabla_{\mathbf{b}^0}\mathbf{z}^{(1,t)} = I_M$.

Now we have all 5 derivatives required to train an RNN with 1 hidden layer.

Please note that all 5 derivatives will be transposed to obtain the gradients used in gradient descent.

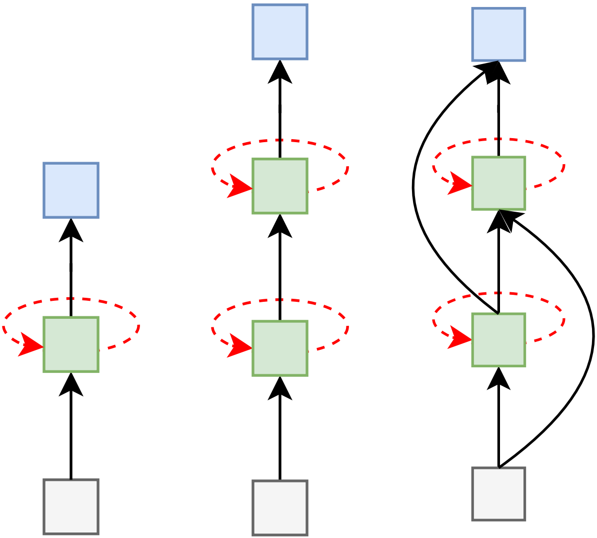
Satbility issues

- 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.
- Suffer from vanishing gradients.
- Also suffer from *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.
- So, in practice, RNNs do not have long-term memory. Solution: LSTM (next lecture).

Benefit of RNN over standard MLP

- N-bit addition (TBD)
- N-bit XOR (TBD)

RNN Variations



1 hidden state 2 hidden states Skip connections

Bidirectional RNN

TBD