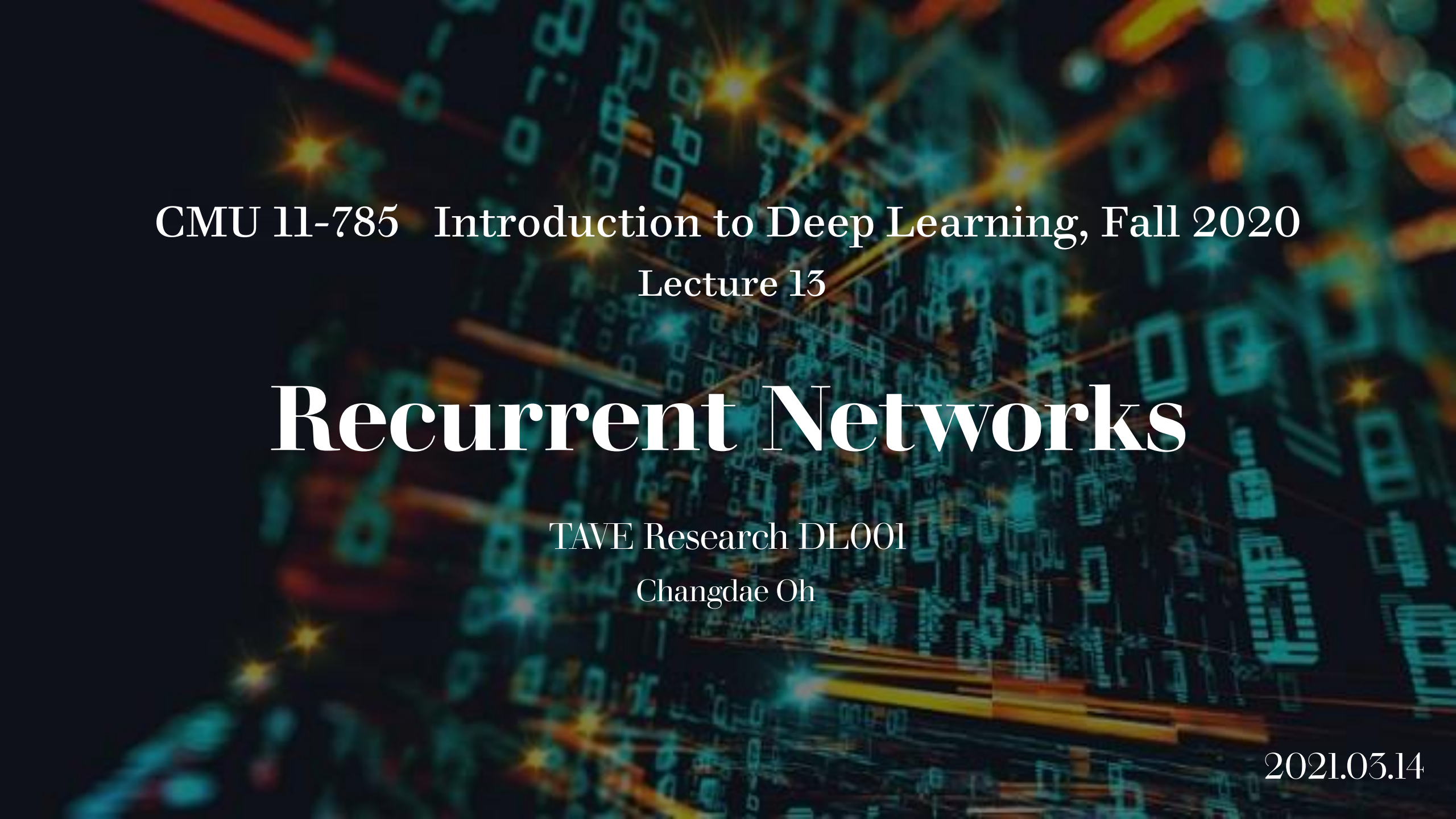


CMU 11-785 Introduction to Deep Learning, Fall 2020

Lecture 13

Recurrent Networks

TAVE Research DL001

Changdae Oh

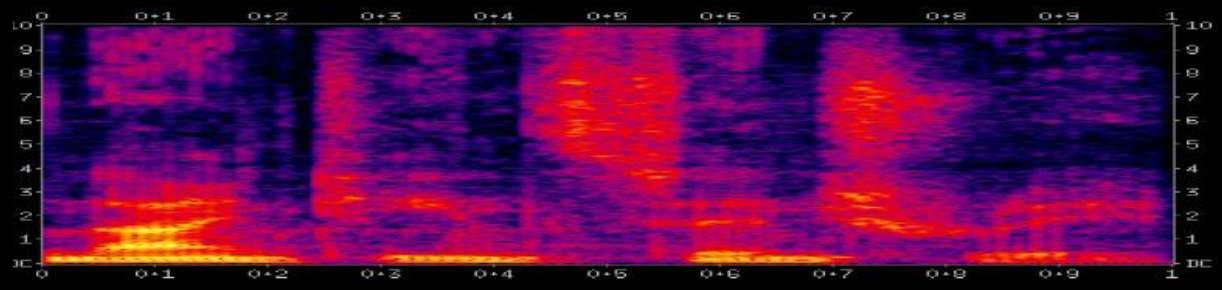
2021.03.14

Topics

- Sequence Modeling
- Recurrent Neural Networks
- Training Recurrent Networks
- Bidirectional Networks

Sequence Modeling

What is sequential data?



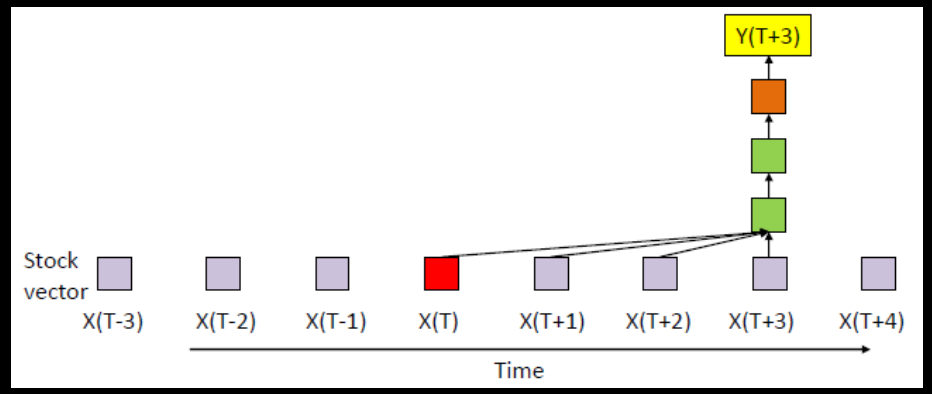
The Steelers, meanwhile, continue to struggle to make stops on defense. They've allowed, on average, 30 points a game, and have shown no signs of improving anytime soon.

context dependency ➡ Model must look at past inputs along with current input

Sequence Modeling

Finite-Response System

$$Y_t = f(X_t, X_{t-1}, \dots, X_{t-N})$$



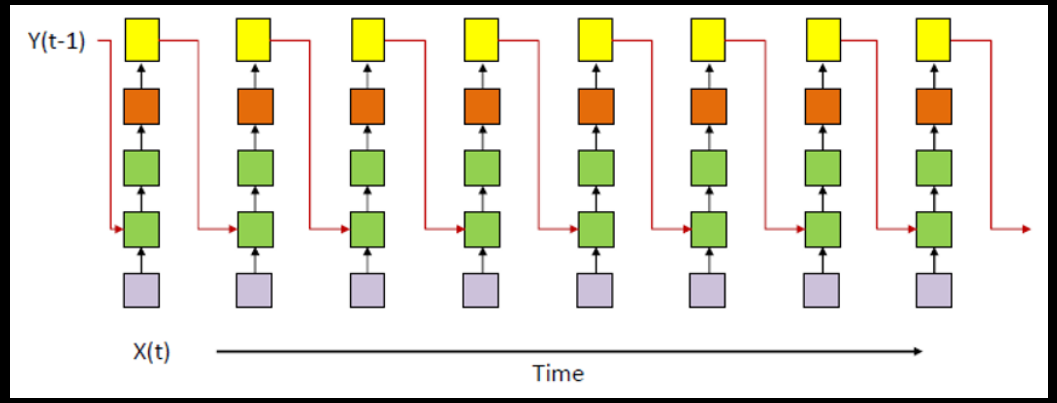
< Time-Delay Neural Network>

- Time t input only affects the output of the system for N days
- Long-term dependence of data cannot be reflected in the model

Need Recurrency!

Infinite-Response System

$$Y_t = f(X_t, Y_{t-1})$$



< NARX Network >

- Output contains information about the entire past

Sequence Modeling

More explicit memory

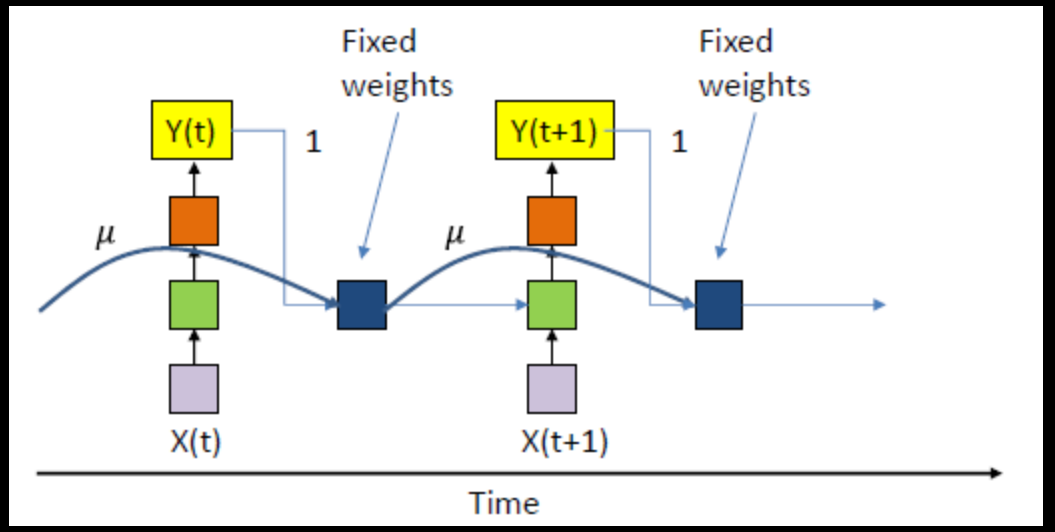
$$m_t = r(y_{t-1}, h_{t-1}, m_{t-1})$$
$$h_t = f(x_t, m_t)$$
$$y_t = g(h_t)$$

- m is a "memory" variable

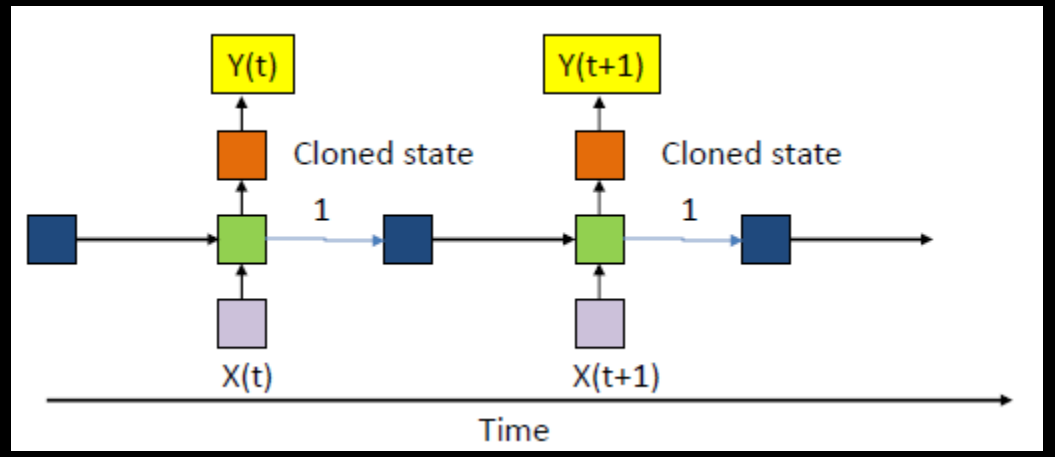
✓ "simple" recurrent networks

✓ During learning current error does not actually propagate to the past

< Jordan Network >



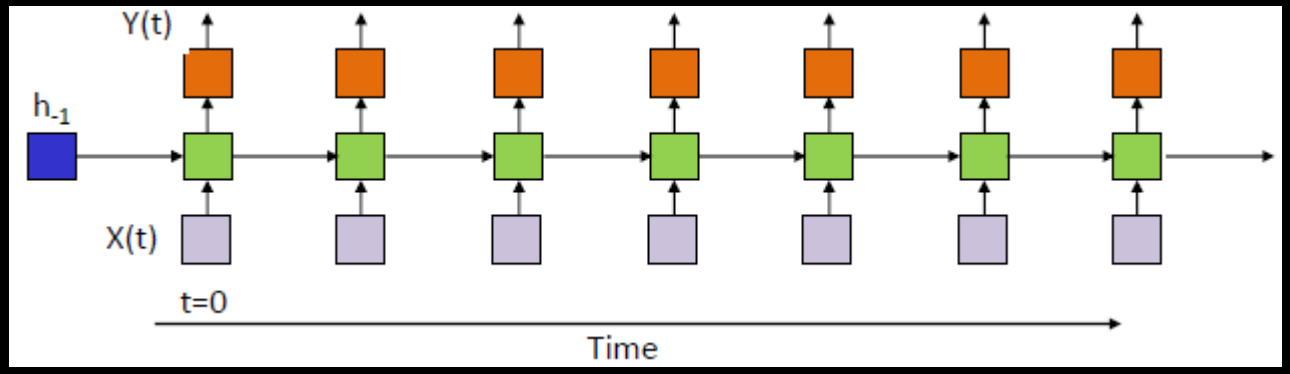
< Elman Network >



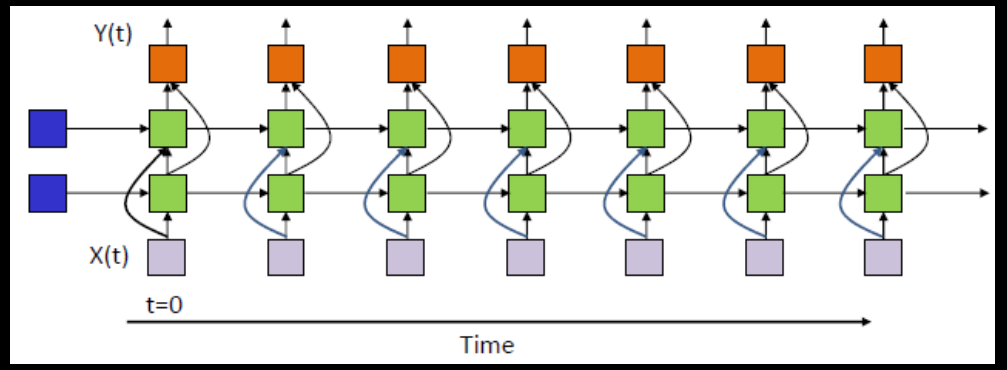
Recurrent Neural Networks

An alternate model for infinite response system : the state-space model

$$h_t = f(x_t, h_{t-1})$$
$$y_t = g(h_t)$$

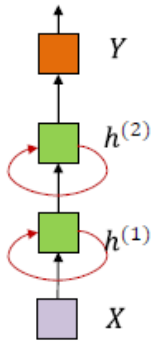


- h is the state of the network (need to define initial state)
- The state an be arbitrarily complex
- An input at t affects outputs forever



Recurrent Neural Networks

Equations



$$h_i^{(1)}(-1) = part\ of\ network\ parameters$$

$$h_i^{(2)}(-1) = part\ of\ network\ parameters$$

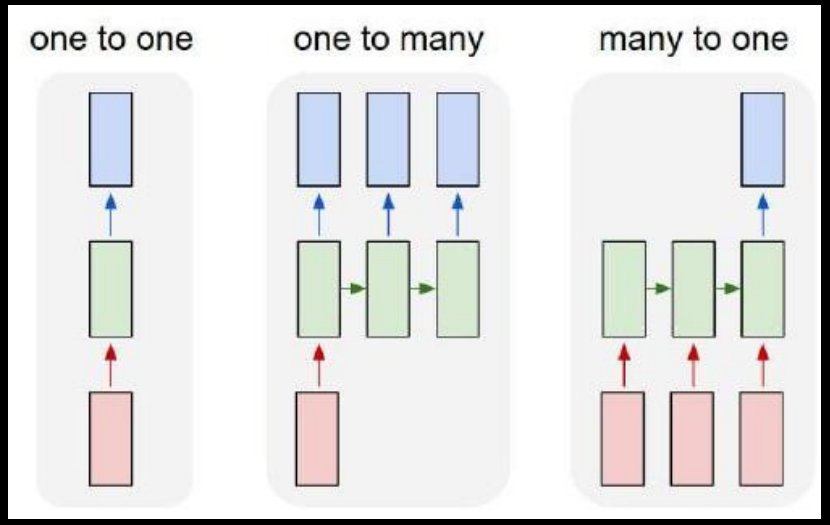
$$h_i^{(1)}(t) = f_1\left(\sum_j w_{ji}^{(1)} X_j(t) + \sum_j w_{ji}^{(11)} h_i^{(1)}(t-1) + b_i^{(1)}\right)$$

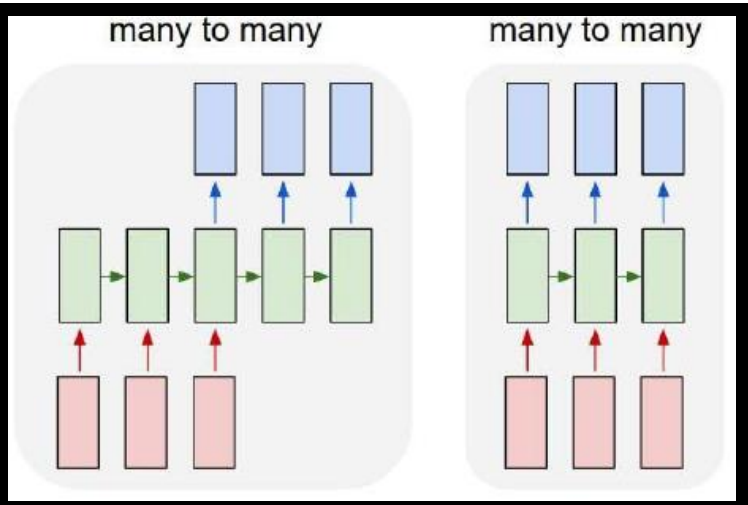
$$h_i^{(2)}(t) = f_2\left(\sum_j w_{ji}^{(2)} h_j^{(1)}(t) + \sum_j w_{ji}^{(22)} h_i^{(2)}(t-1) + b_i^{(2)}\right)$$

$$Y(t) = f_3\left(\sum_j w_{jk}^{(3)} h_j^{(2)}(t) + b_k^{(3)}, k = 1..M\right)$$

- Current weights / Recurrent weights
- Weights are shared over time

Variants



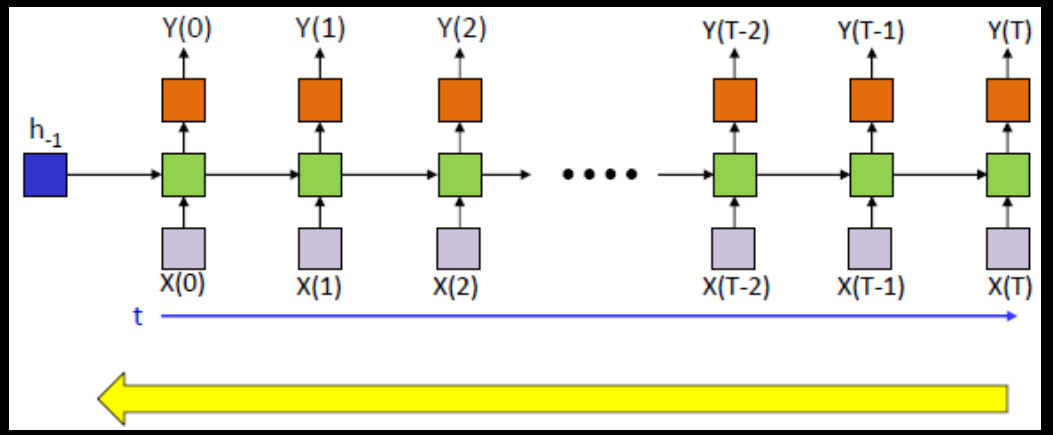


Training Recurrent Networks

- State-space models enable current error to update parameters in the past

How do we train the network?

  - Back propagation through time (BPTT)
  - Given a collection of sequence inputs (X_t, D_t)
  - Update parameters to minimize the error between the outputs of the network Y_t and the desired outputs D_t

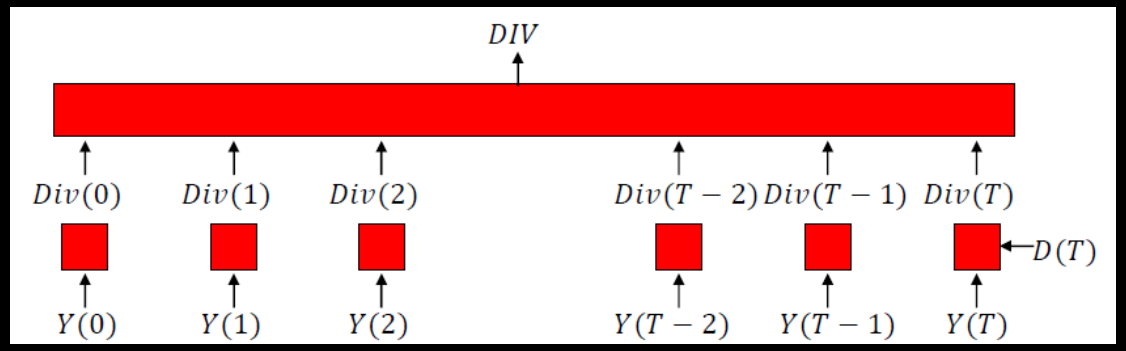


Forward
: pass the entire data sequence through the network, generate outputs

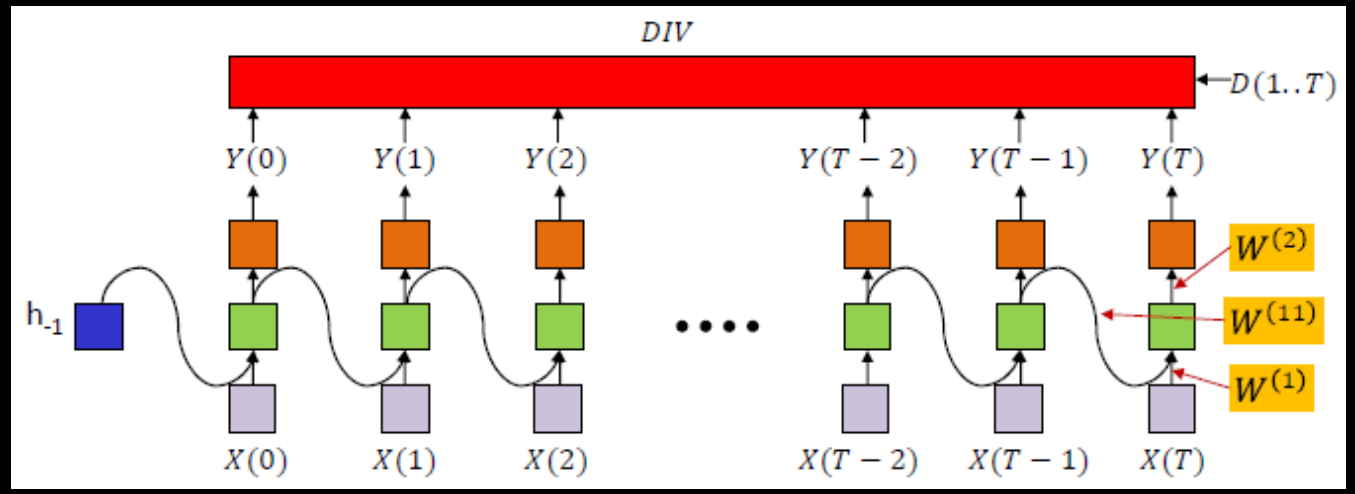
Backward
: compute gradients via backpropagation

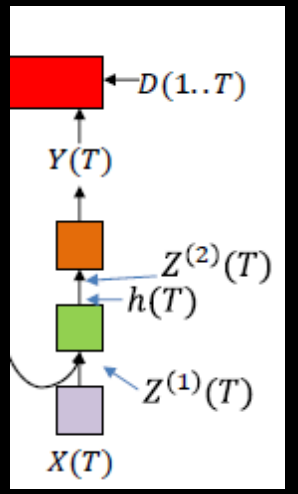
Training Recurrent Networks

BPTT



- Assume the overall divergence is a simple sum of local divergence at each time. (special case)





$$\frac{dDIV}{dY_i(t)} = \frac{dDiv(t)}{dY_i(t)}$$

$$\nabla_{Z^{(2)}(t)} DIV = \nabla_{Y(t)} DIV \, \nabla_{Z^{(2)}(t)} Y(t)$$

$$\nabla_{h(t)} DIV = \nabla_{Z^{(2)}(t)} DIV \, W^{(2)} + \nabla_{Z^{(1)}(t+1)} DIV \, W^{(11)}$$

$$\nabla_{Z^{(1)}(t)} DIV = \nabla_{h(t)} DIV \, \nabla_{Z^{(1)}(t)} h(t)$$

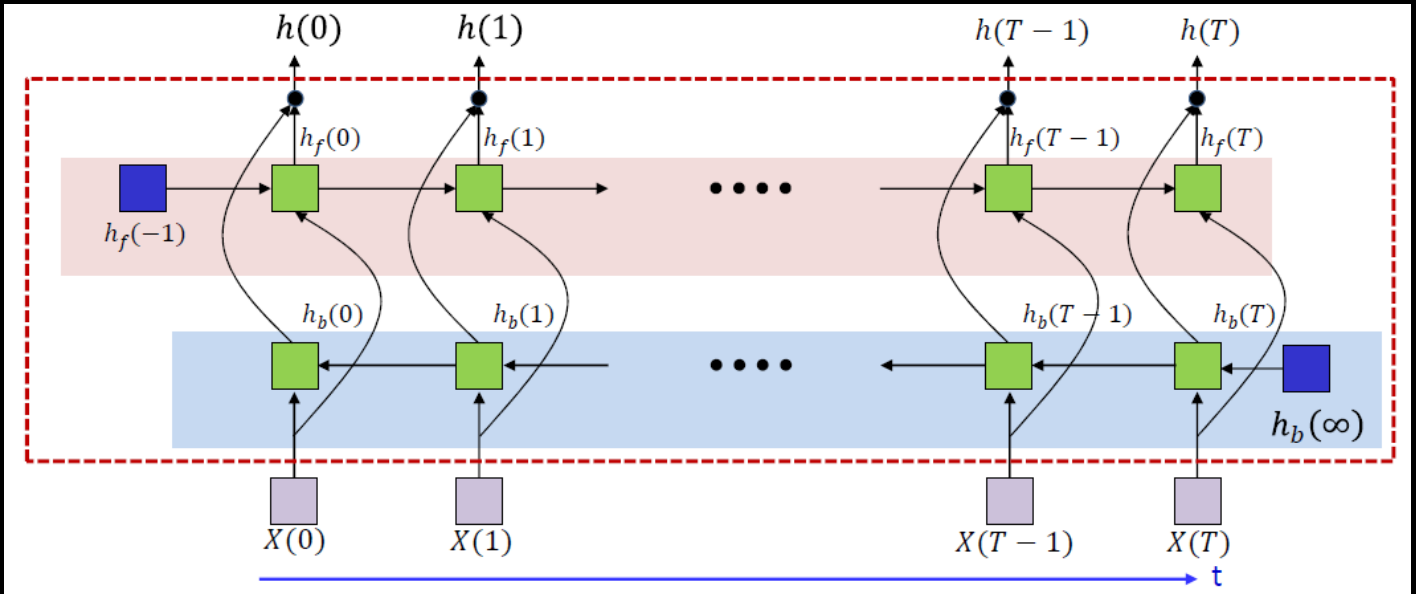
$$\nabla_{W^{(2)}} DIV \mathrel{+}= h(t) \nabla_{Z^{(2)}(t)} DIV$$

$$\nabla_{W^{(11)}} DIV \mathrel{+}= h(t-1) \nabla_{Z^{(1)}(t)} DIV$$

$$\nabla_{W^{(1)}} DIV \mathrel{+}= X(t) \nabla_{Z^{(1)}(t)} DIV$$

$$\nabla_{h_{-1}} DIV = \nabla_{Z^{(1)}(0)} DIV W^{(11)}$$

Bidirectional Networks



$$h(t) = [h_f(t); h_b(t)]$$

- Can be stacked independent bi-directional blocks
- Forward & backward nets may have several layers

- The forward net processes the data from t = 0 to t = T (only computing the hidden state values.)
- The backward net processes the data in reverse time (end to beginning)

✓ this is not an online process and requires the entire input data