Course number: 80240743

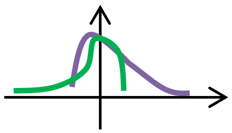
Deep Learning

Xiaolin Hu (胡晓林) & Jun Zhu (朱军)
Dept. of Computer Science and Technology
Tsinghua University

Last lecture review

Part 1

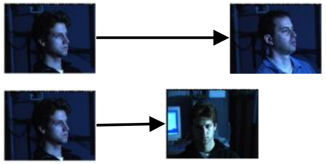
$$\hat{x}_i = \frac{x_i - \mathrm{E}[x_i]}{\sqrt{\mathrm{Var}[x_i]}} \qquad y_i = \gamma_i \hat{x}_i + \beta_i$$



Part 2

Multi-class classification

Face verification → Triplet loss / Contrastive loss



Part 3

R-CNN → Fast R-CNN → Faster R-CNN → YOLO

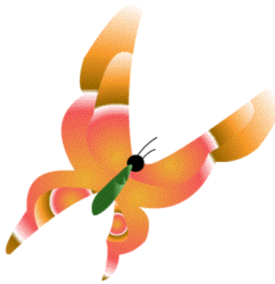
Lecture 7: Recurrent Neural Networks

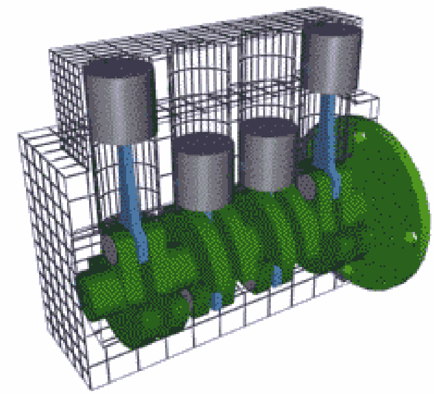
Xiaolin Hu
Dept. of Computer Science and Technology
Tsinghua University

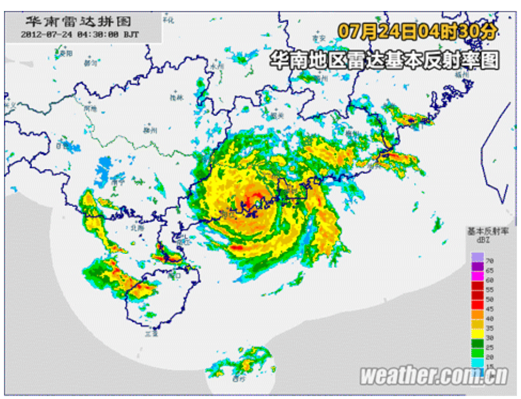
Outline

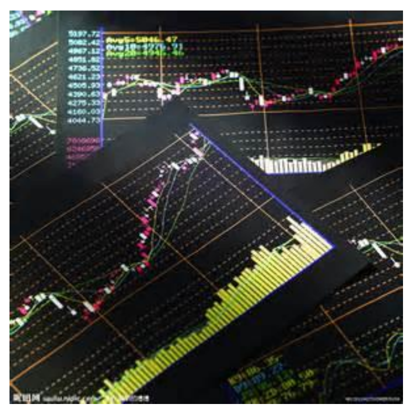
1. Dynamic systems
2. Simple RNNs
3. Gated RNNs
4. Applications to speech recognition
5. Summary

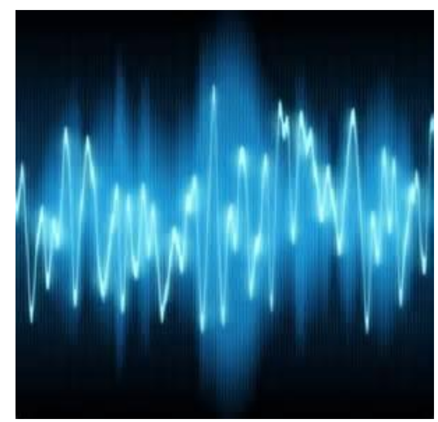
Dynamic systems





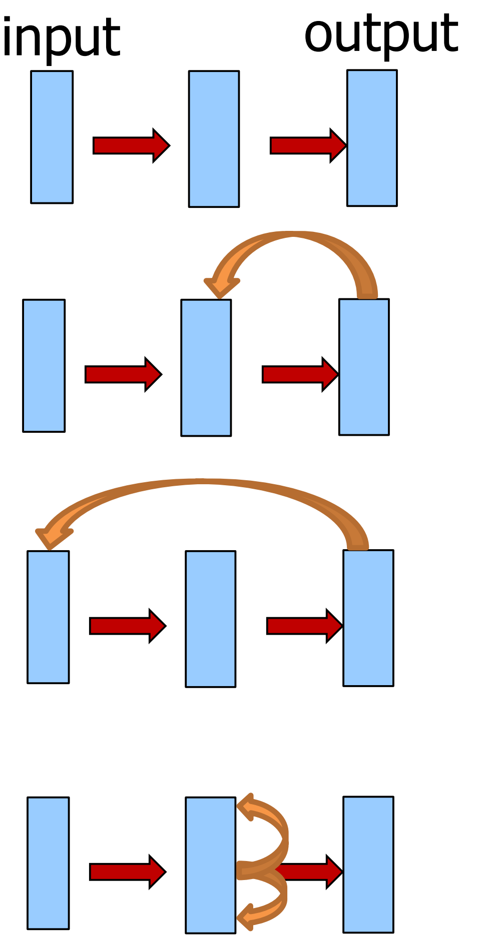






Feedback connections

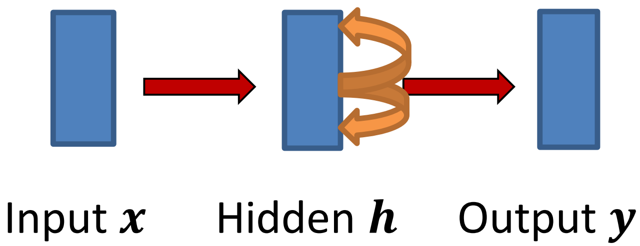
- Feedforward networks
  - No feedback
- Recurrent networks
  - Between-layer feedback
  - Within-layer feedback



With feedback connections, the state (and therefore output) of neurons will change over time

RNNs are dynamic systems

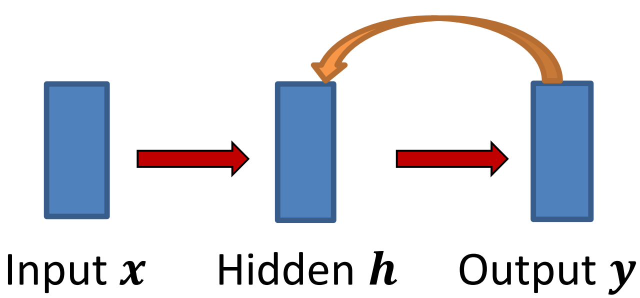
Elman network



$$\boldsymbol{h}(t) = \sigma_h(\boldsymbol{W}_h \boldsymbol{x}(t) + \boldsymbol{U}_h \boldsymbol{h}(t-1) + \boldsymbol{b}_h)$$
$$\boldsymbol{y}(t) = \sigma_y(\boldsymbol{W}_y \boldsymbol{h}(t) + \boldsymbol{b}_y)$$

where σ_h and σ_y are activation functions, $\boldsymbol{W}, \boldsymbol{U}, \boldsymbol{b}$ are parameters

Jordan network



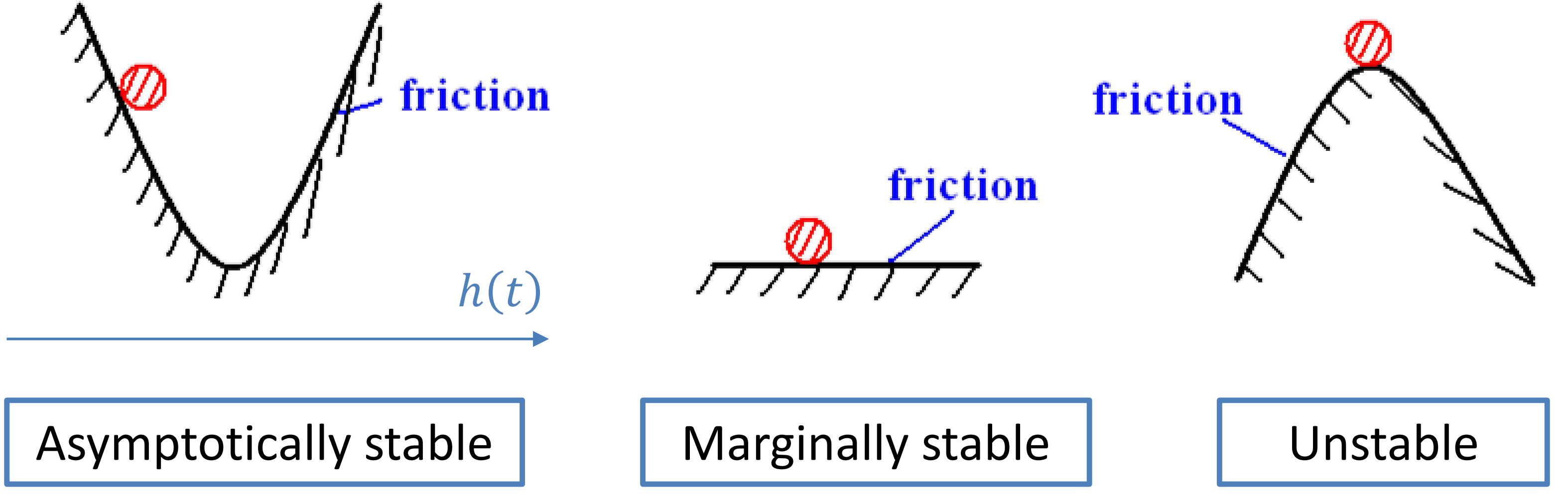
$$\boldsymbol{h}(t) = \sigma_h(\boldsymbol{W}_h \boldsymbol{x}(t) + \boldsymbol{U}_h \boldsymbol{y}(t-1) + \boldsymbol{b}_h)$$
$$\boldsymbol{y}(t) = \sigma_y(\boldsymbol{W}_y \boldsymbol{h}(t) + \boldsymbol{b}_y)$$

RNNs in general

- The states of the neurons in RNN evolve over time
$$\boldsymbol{h}(t+1) = f(\boldsymbol{h}(t), \boldsymbol{x}(t), \boldsymbol{y}(t))$$
  - $\boldsymbol{h}$ denotes the states of *all neurons*
  - $\boldsymbol{x}$ denotes input to the network
  - $\boldsymbol{y}$ denotes output of the network
  - f is a nonlinear function
- Often, the output neurons $\boldsymbol{y}$ are separated from the above equation
$$\boldsymbol{y}(t) = g(\boldsymbol{h}(t), \boldsymbol{x}(t))$$
  - where g denotes the output function
- Such systems are termed (discrete) dynamic systems
- Linear VS nonlinear: f is linear or nonlinear
- Non-autonomous VS autonomous: f depends on t explicitly or not

Properties of dynamic systems

- For autonomous system
$$\boldsymbol{h}(t+1) = f(\boldsymbol{h}(t))$$
- A point $\boldsymbol{h}^*$ satisfying $\boldsymbol{h}^* = f(\boldsymbol{h}^*)$ is called a fixed point or equilibrium point
- A system might be stable or not around its fixed points

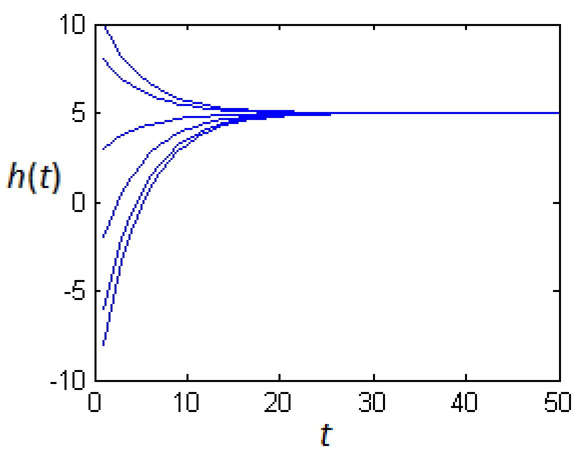


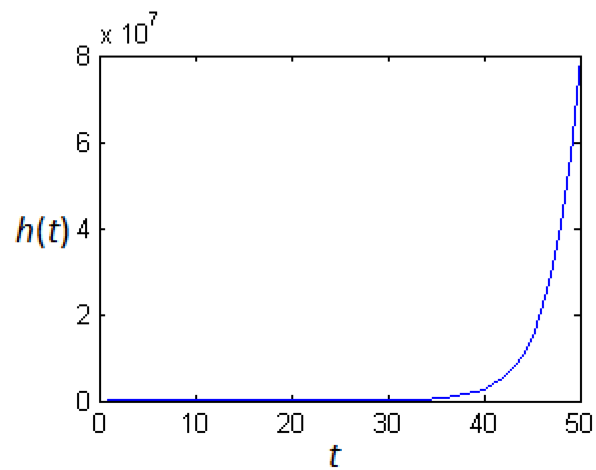
Examples about stability

- For 1D problem, by approximating f with a linear function, we get that a fixed point h^* is stable whenever $|f'(h^*)| < 1$
- Consider the linear system

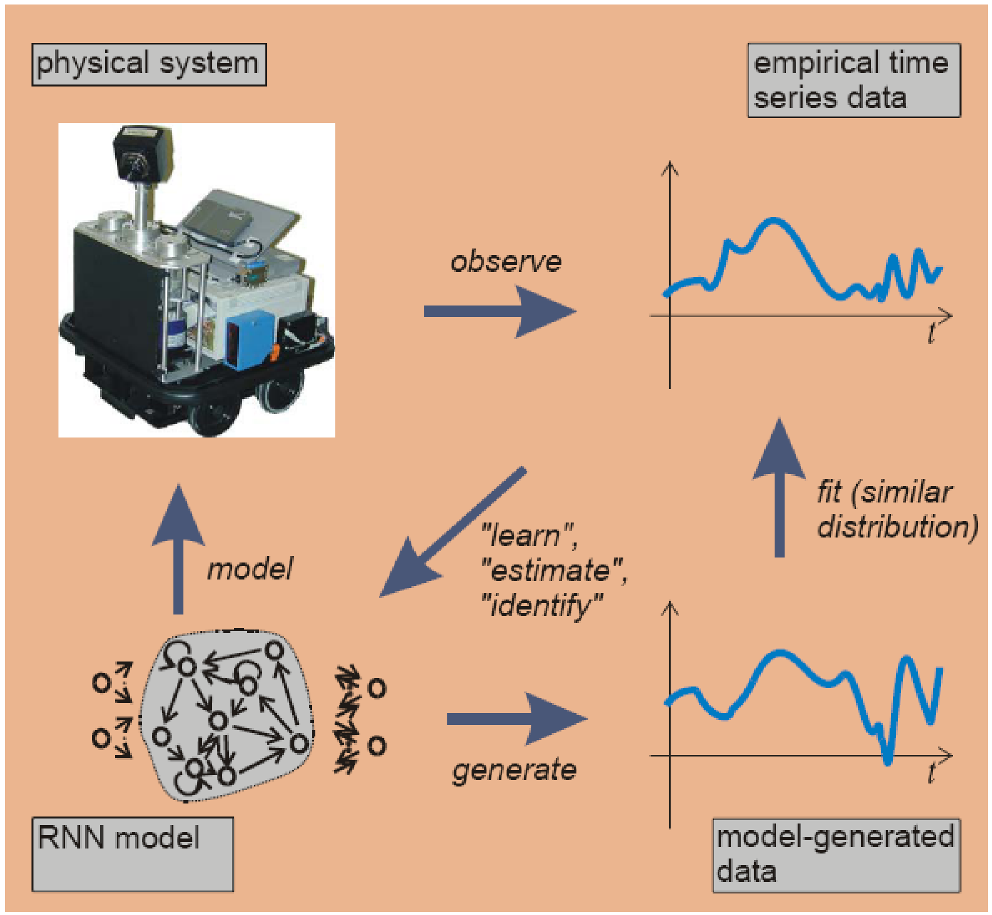
$$h(t+1) = 0.8h(t) + 1$$

  It is stable (globally converges to $h^* = 5$)
- But $h(t+1) = 1.4h(t) + 1$ is unstable





(1) Model dynamic systems with RNN



Why do we need RNN?

- Case 1: At every time step t you always have an input $\boldsymbol{x}(t)$ and the desired output $\boldsymbol{r}(t)$

  Can you train an MLP or CNN to do the prediction task?

- Case 2: You only have input at the beginning, but the desired output at different time is different

  Can you train an MLP or CNN to do the prediction task?

We use another dynamic system (RNN) to approximate the real dynamic system!

Why is it possible?

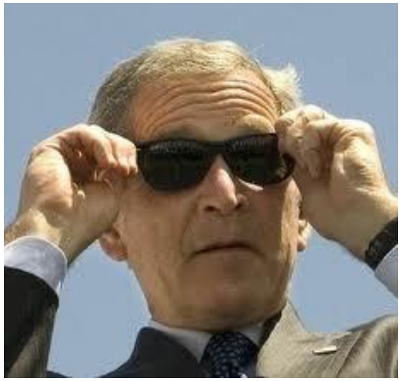
- The hidden states $\boldsymbol{h}$ in an RNN

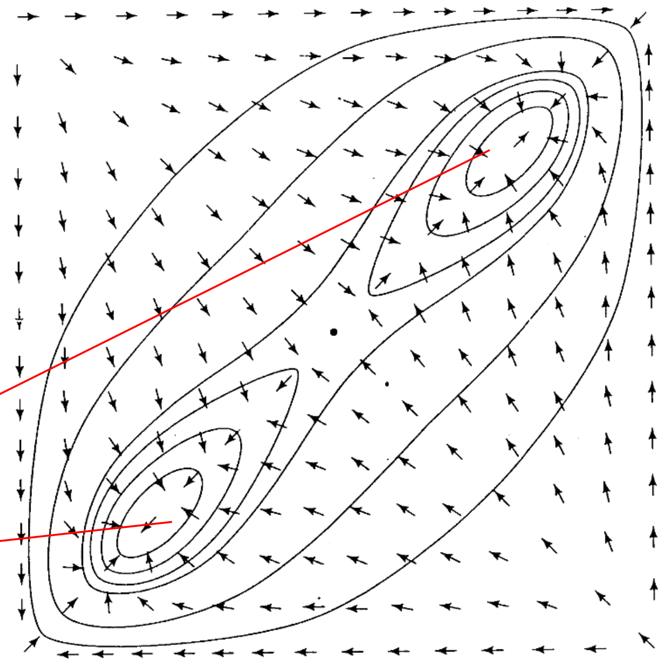
$$\boldsymbol{h}(t+1) = f(\boldsymbol{h}(t), \boldsymbol{x}(t), \boldsymbol{y}(t))$$

  have a memory of previous events
  - The hidden states are expected to capture the past information, or temporal dependence in the system that the RNN tries to model
  - This memory resembles short-term memory of animals
- Given many input-output pairs of the system, there exists efficient algorithms to learn the parameters of the RNN
  - Backpropagation through time (BPTT)

(2) Explain how the brain works*

- A partial or approximate representation of a stored item is used to recall the full item.
- This can be a dynamic process in the brain





attractors

Continuous Hopfield network*

The continuous version

$$\frac{d\mathbf{I}}{dt} = -\frac{\mathbf{I}}{\rho} + \mathbf{h} + \mathbf{M}\mathbf{F}(\mathbf{I}),$$

Same as type I model with $\rho = 1$

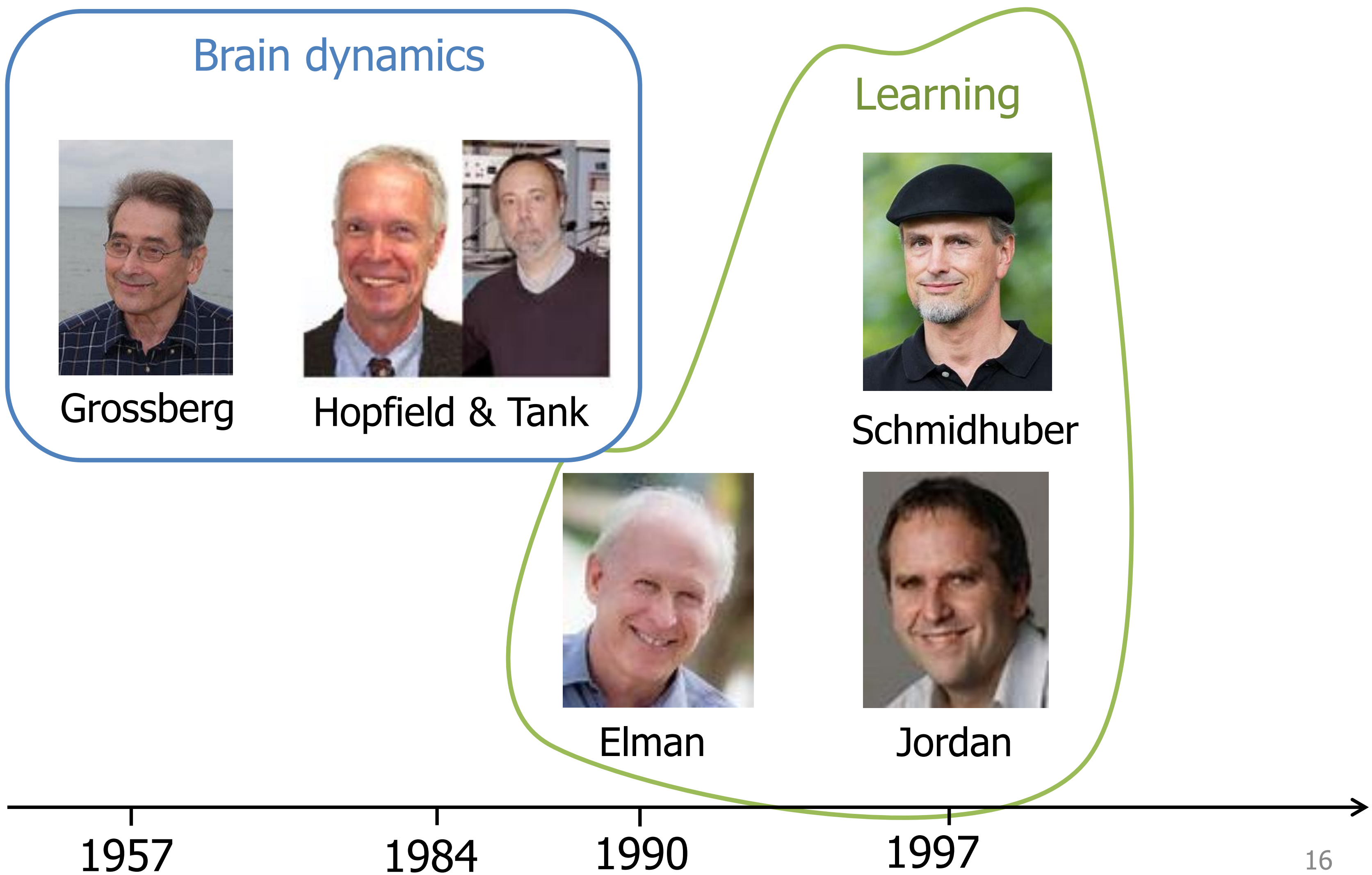
where $F'(I_i) > 0$ for all i. The energy function

$$E(\mathbf{I}) = \sum_{i=1}^{N_v} \left(\int_0^{I_i} dz_i \frac{z_i F'(z_i)}{\rho} - h_i F(I_i) - \frac{1}{2} \sum_{j=1}^{N_v} F(I_i) M_{ij} F(I_j) \right)$$

has the property $\frac{dE}{dt} = -\sum_i^{N_v} F'(I_i) \left(\frac{dI_i}{dt}\right)^2$.

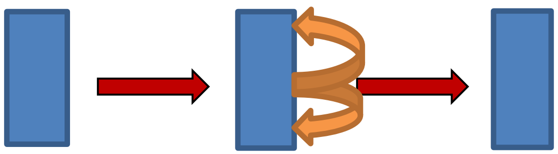
- If $F(I_i)$ is a sigmoid function, the model converges to $\mathbf{I} = \rho(\mathbf{h} + \mathbf{M}\mathbf{F}(\mathbf{I}))$.
- For high-gain sigmoid functions, $F(I_i) \approx \mathrm{sgn}(I_i)$ and the model behaves like a discrete-time model.

History of RNN

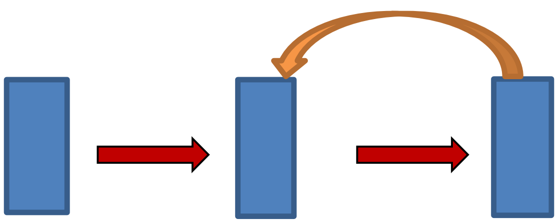


Summary of Part 1

Feedback results in dynamics



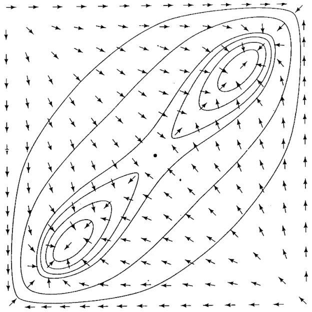
Input $\boldsymbol{x}$ Hidden $\boldsymbol{h}$ Output $\boldsymbol{y}$



Input $\boldsymbol{x}$ Hidden $\boldsymbol{h}$ Output $\boldsymbol{y}$

Model dynamic systems

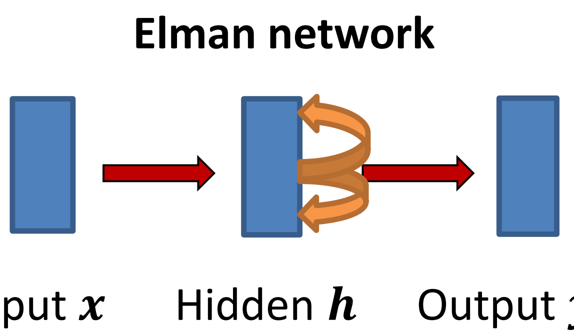
Model the brain*



Outline

1. Dynamic systems
2. Simple RNNs
3. Gated RNNs
4. Applications to speech recognition
5. Summary

Elman network



- Dynamic system:

$$\boldsymbol{h}(t) = \sigma_h(\boldsymbol{W}_h \boldsymbol{x}(t) + \boldsymbol{U}_h \boldsymbol{h}(t-1) + \boldsymbol{b}_h)$$

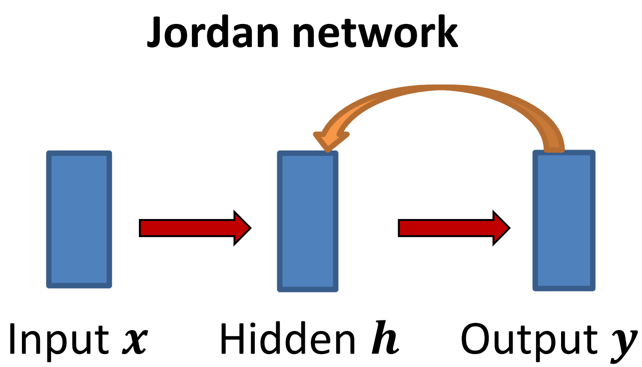
$$\boldsymbol{y}(t) = \sigma_y(\boldsymbol{W}_y \boldsymbol{h}(t) + \boldsymbol{b}_y)$$

where σ_h and σ_y are activation functions, $\boldsymbol{W}, \boldsymbol{U}, \boldsymbol{b}$ are parameters

January 22, 1948 – June 28, 2018

- Jeffrey Elman
  - BS in 1969 from Harvard University
  - Ph.D. in 1977 from the University of Texas at Austin
- Professor of cognitive science at the UCSD

Jordan network



- Dynamic system:

$$\boldsymbol{h}(t) = \sigma_h(\boldsymbol{W}_h \boldsymbol{x}(t) + \boldsymbol{U}_h \boldsymbol{y}(t-1) + \boldsymbol{b}_h)$$
$$\boldsymbol{y}(t) = \sigma_y(\boldsymbol{W}_y \boldsymbol{h}(t) + \boldsymbol{b}_y)$$

where σ_h and σ_y are activation functions, $\boldsymbol{W}, \boldsymbol{U}, \boldsymbol{b}$ are parameters

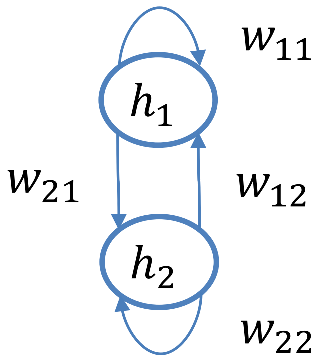
- Michael I Jordan
  - BS in Psychology in 1978 from the Louisiana State University,
  - MS in Mathematics in 1980 from Arizona State University
  - PhD in Cognitive Science in 1985 from the UCSD
- At UCSD, Jordan was a student of David Rumelhart
- Now at UC Berkeley

Back-propagation through time (BPTT)

Unfold the temporal operation of the network into a layered feedforward network, the topology of which grows by one layer at every time step.

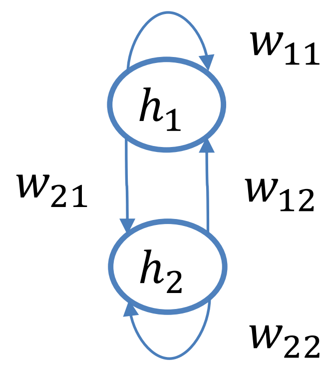
Consider a linear system without input:

$$h_1(t+1) = w_{11}h_1(t) + w_{12}h_2(t)$$
$$h_2(t+1) = w_{21}h_1(t) + w_{22}h_2(t)$$



Back-propagation through time (BPTT)

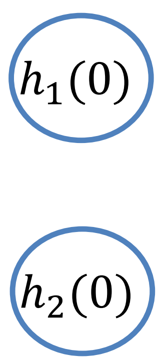
Unfold the temporal operation of the network into a layered feedforward network, the topology of which grows by one layer at every time step.



Consider a linear system without input:

$$h_1(t+1) = w_{11}h_1(t) + w_{12}h_2(t)$$
$$h_2(t+1) = w_{21}h_1(t) + w_{22}h_2(t)$$

Unfold through time:



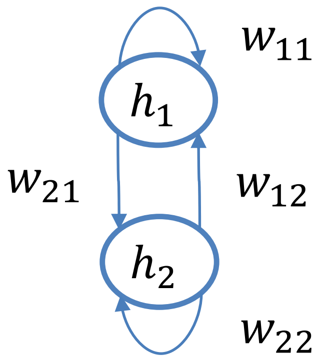
time 0

Back-propagation through time (BPTT)

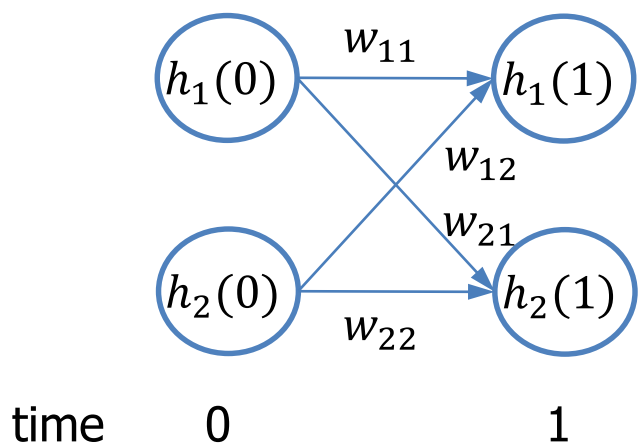
Unfold the temporal operation of the network into a layered feedforward network, the topology of which grows by one layer at every time step.

Consider a linear system without input:

$$h_1(t+1) = w_{11}h_1(t) + w_{12}h_2(t)$$
$$h_2(t+1) = w_{21}h_1(t) + w_{22}h_2(t)$$



Unfold through time:

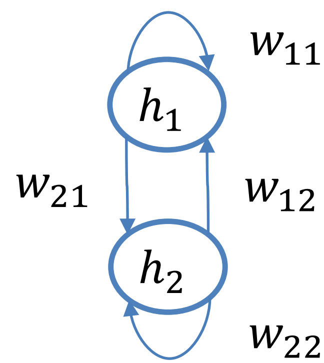


Back-propagation through time (BPTT)

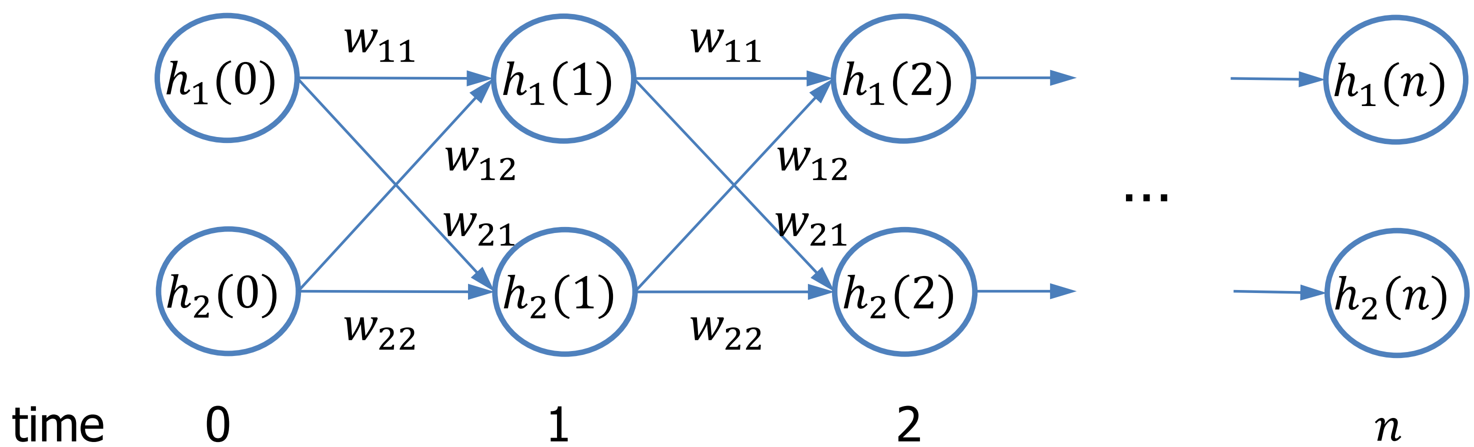
Unfold the temporal operation of the network into a layered feedforward network, the topology of which grows by one layer at every time step.

Consider a linear system without input:

$$h_1(t+1) = w_{11}h_1(t) + w_{12}h_2(t)$$
$$h_2(t+1) = w_{21}h_1(t) + w_{22}h_2(t)$$

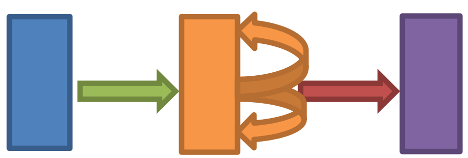


Unfold through time:



Unfold the Elman network

Unfold for q times

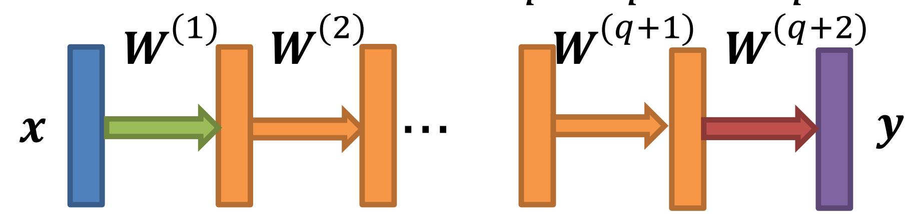


$$\boldsymbol{h}(t) = \sigma_h(\boldsymbol{W}_h \boldsymbol{x}(t) + \boldsymbol{U}_h \boldsymbol{h}(t-1) + \boldsymbol{b}_h)$$

$$\boldsymbol{y}(t) = \sigma_y(\boldsymbol{W}_y \boldsymbol{h}(t) + \boldsymbol{b}_y)$$

- Case 1:
  - $\boldsymbol{x}$ is only present at the first step
  - Label $\boldsymbol{r}$ is only present at the last step

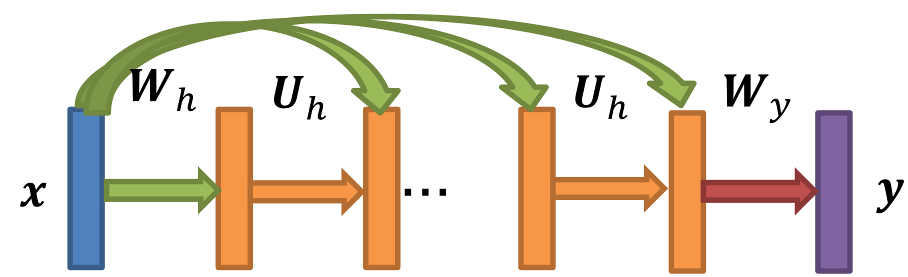
Layer l: 0 1 2 q $q+1$ $q+2$



$$\boldsymbol{W}^{(1)} = \boldsymbol{W}_h, \boldsymbol{W}^{(2)} = \cdots = \boldsymbol{W}^{(q+1)} = \boldsymbol{U}_h$$

$$\boldsymbol{W}^{(q+2)} = \boldsymbol{W}_y$$

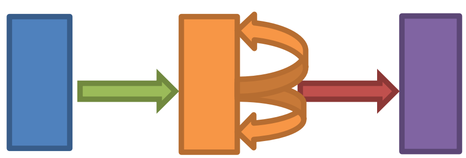
- Case 2:
  - $\boldsymbol{x}$ is fixed but present at all steps
  - Label $\boldsymbol{r}$ is only present at the last step
  - E.g., image classification (Liang, Hu, CVPR 2015)



(Arrows in the same color share weights)

Unfold the Elman network

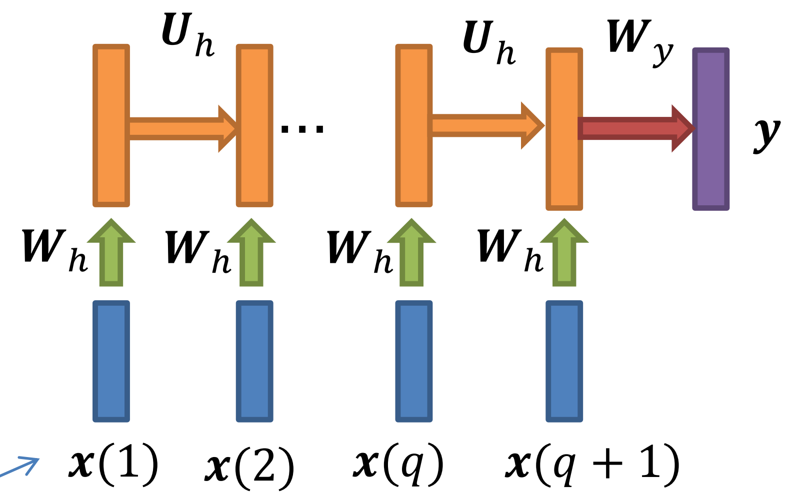
Unfold for q times



$$\boldsymbol{h}(t) = \sigma_h(\boldsymbol{W}_h \boldsymbol{x}(t) + \boldsymbol{U}_h \boldsymbol{h}(t-1) + \boldsymbol{b}_h)$$

$$\boldsymbol{y}(t) = \sigma_y(\boldsymbol{W}_y \boldsymbol{h}(t) + \boldsymbol{b}_y)$$

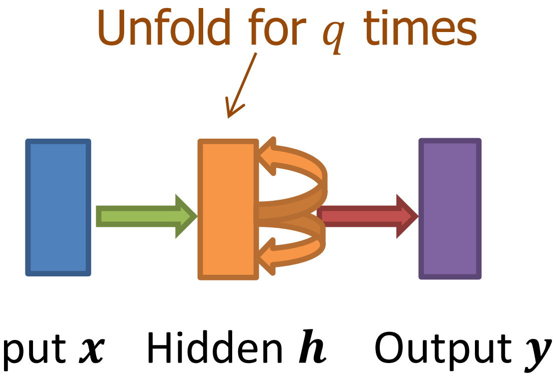
- Case 3:
  - $\boldsymbol{x}$ is time-varying
  - Label $\boldsymbol{r}$ is only present at the last step
  - E.g., sentence classification



This can be viewed as layer 0
attached to the orange backbone

(Arrows in the same color share weights)

Unfold the Elman network

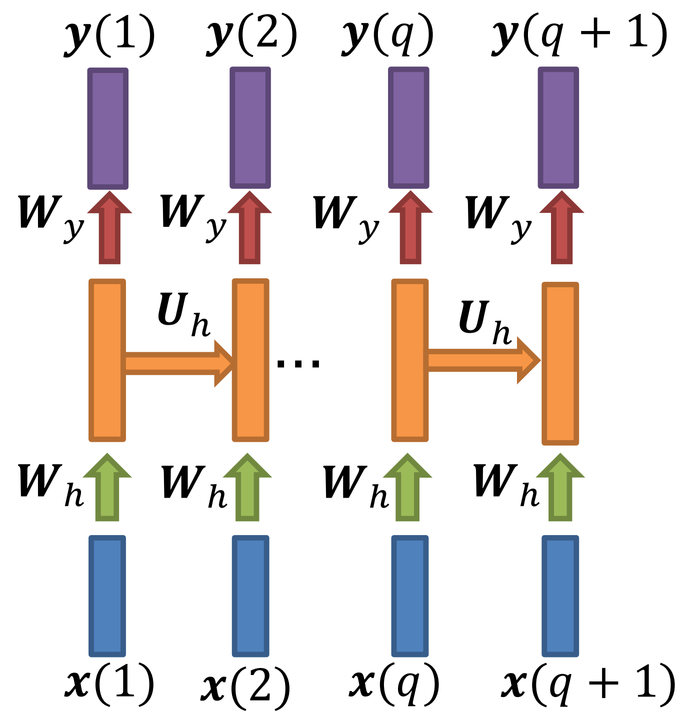


$$\boldsymbol{h}(t) = \sigma_h(\boldsymbol{W}_h\boldsymbol{x}(t) + \boldsymbol{U}_h\boldsymbol{h}(t-1) + \boldsymbol{b}_h)$$

$$\boldsymbol{y}(t) = \sigma_y(\boldsymbol{W}_y\boldsymbol{h}(t) + \boldsymbol{b}_y)$$

Do you know other cases?

- Case 4:
  - $\boldsymbol{x}$ is time-varying
  - Label $\boldsymbol{r}$ is present at all steps
  - E.g., speech recognition

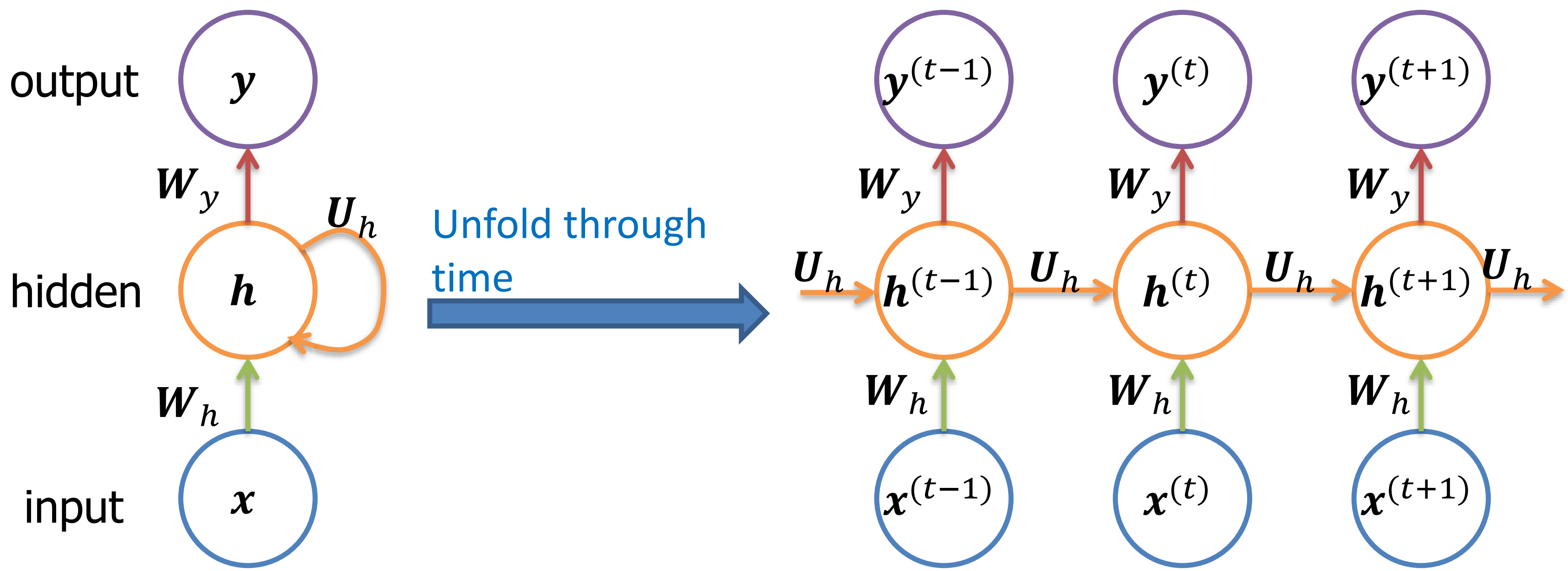


(Arrows in the same color share weights)

Simplified illustration (Elman network)

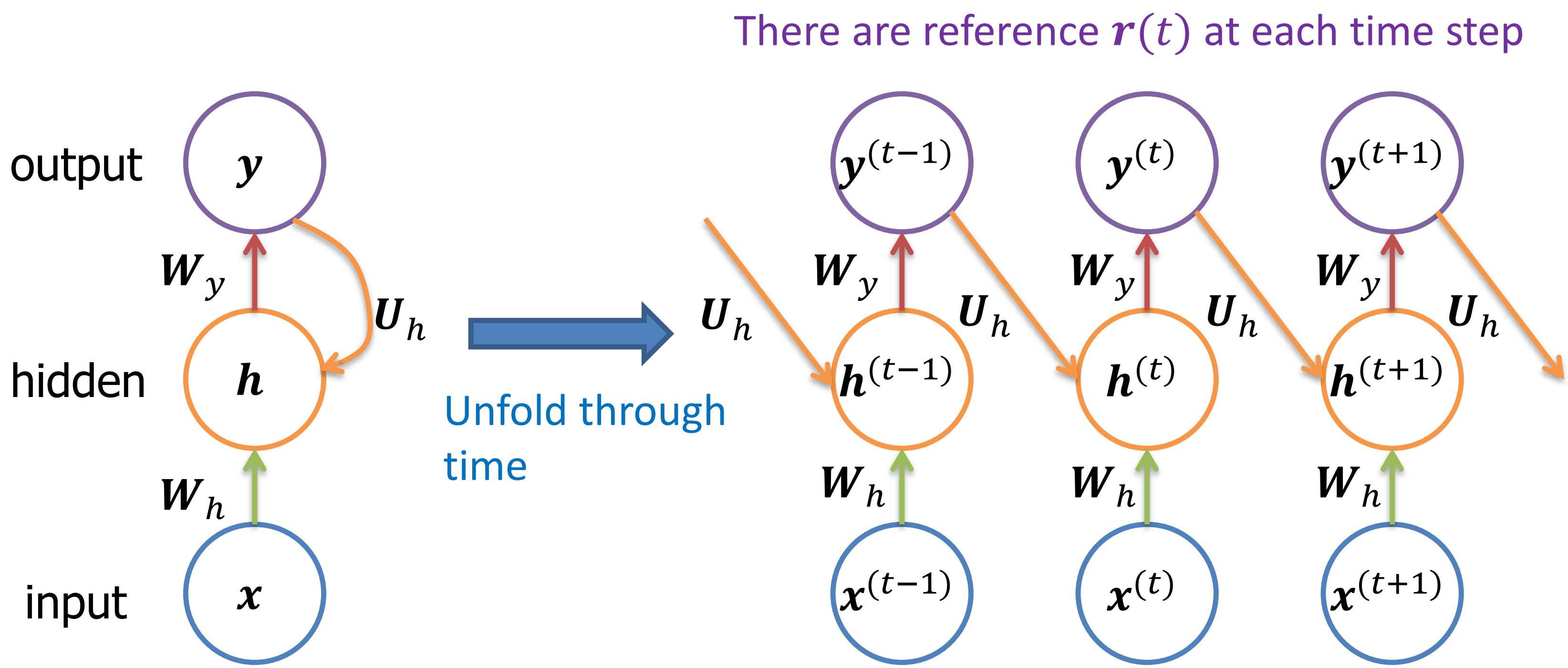
- Use circles to represent vectors (one circle one layer)
- Put time step into superscript

There are reference $\boldsymbol{r}(t)$ at each time step



- The forward propagation runtime is $O(q)$ and cannot be reduced

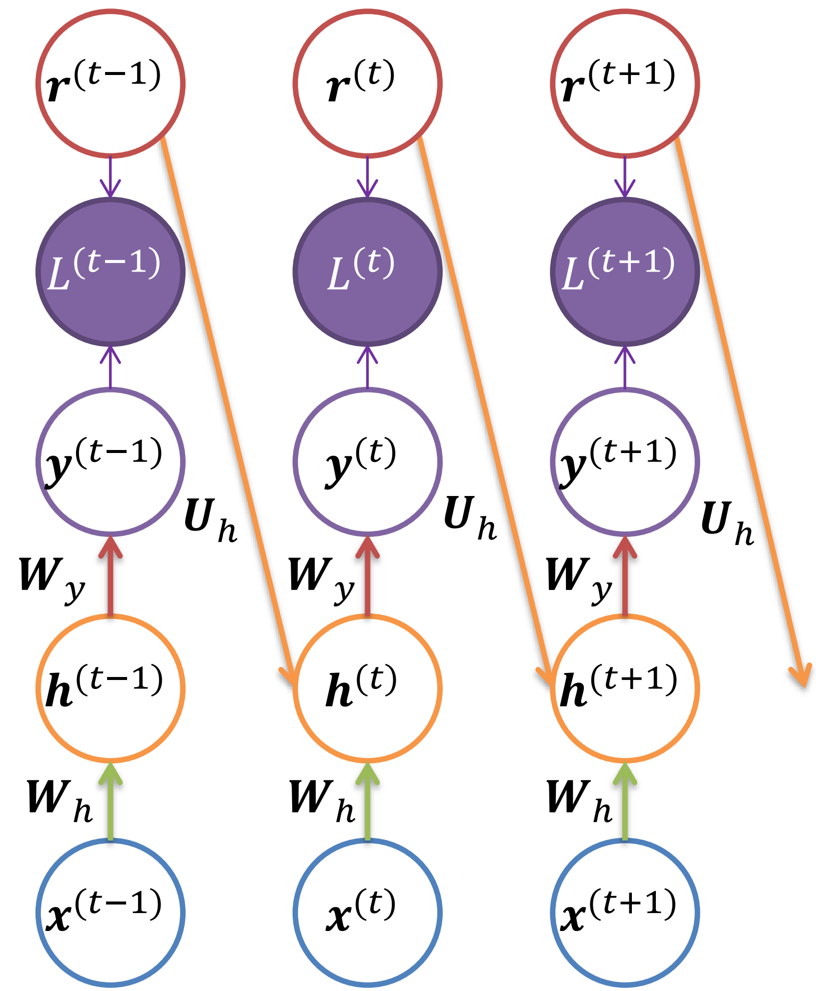
Unfold the Jordan network



- If the loss is based on comparing $\boldsymbol{y}(t)$ and $\boldsymbol{r}(t)$, all time steps are decoupled and training can be parallelized

Teacher forcing

- Some networks such as Jordan network, have connections from the output at one time step to values computed in the next time step
- Then, what should be input to the next time step to represent the output?
- Teacher forcing: in training, we use the reference signal



Teacher forcing

- However, in testing, there is no reference signal and we have to use the network's output at time t
- The kind of inputs that the network sees during training could be quite different from the kind of inputs that it will see at test time

Exposure bias

- To mitigate this problem:
  - Alternately use teacher-forced inputs and free-running inputs for a number of time steps
  - Randomly choose between the teacher-forced input and free-running input at every time step

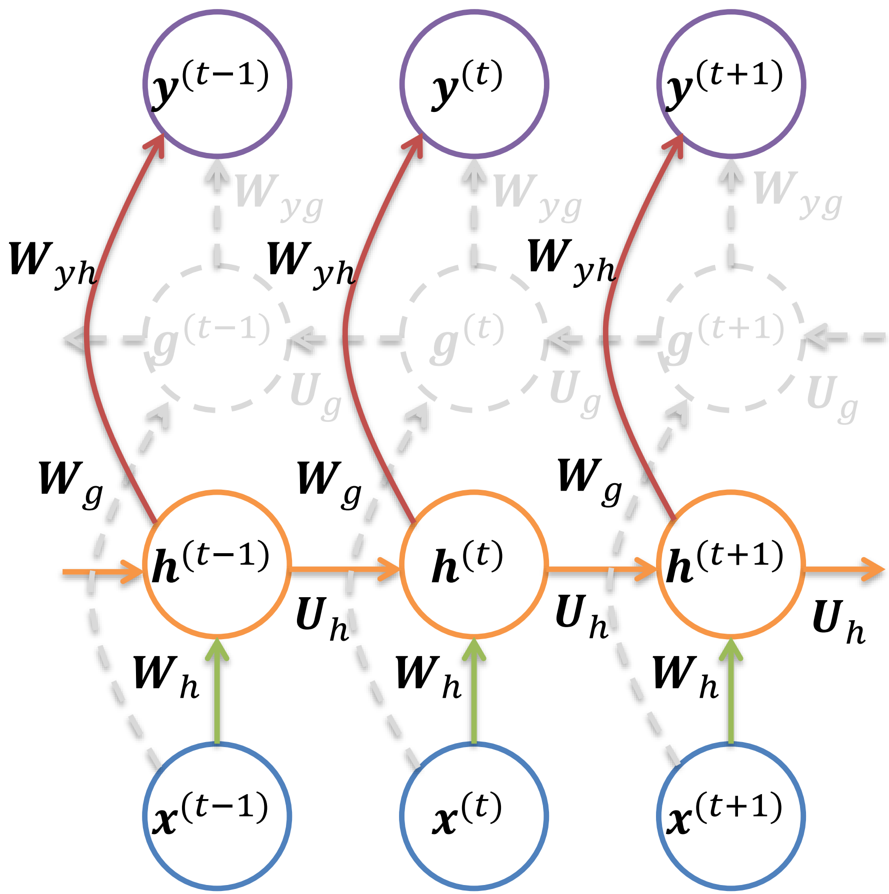
Bidirectional RNN

- In many applications, the prediction at the current time step may depend on the whole input sequence (both the past and the future)

Bank is the side of a river.
Bank provides various financial services.

- Bidirection RNNs combine an RNN that moves forward through time with another RNN that moves backward through time
- The output of the entire network at every time step then receives two inputs

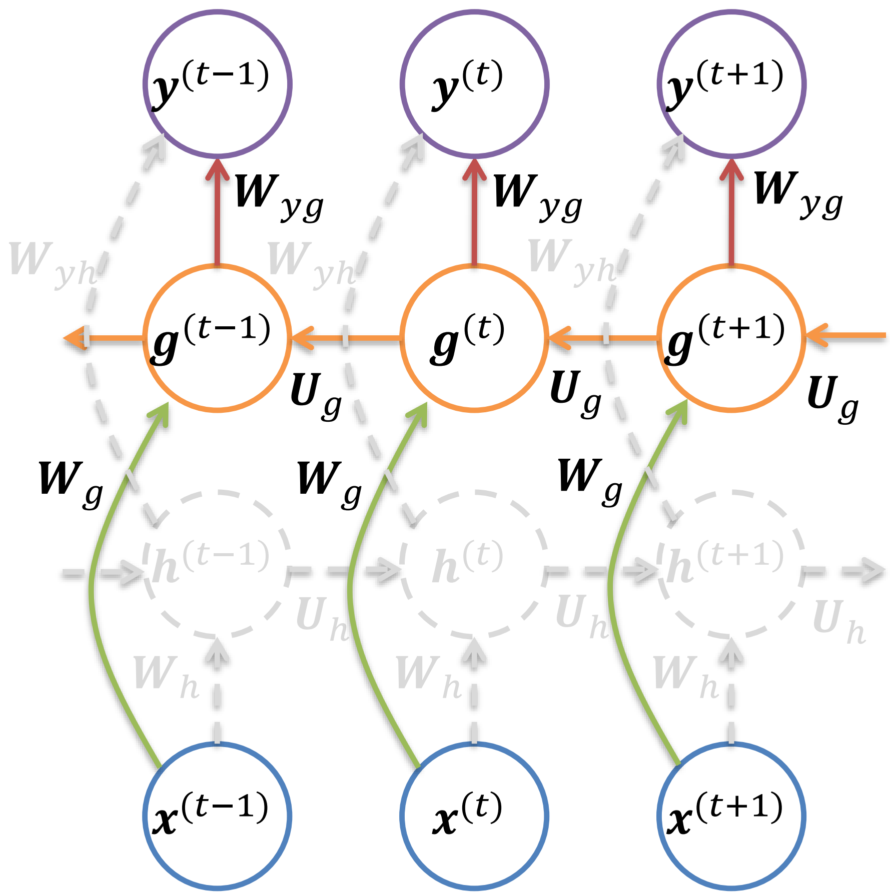
Bidirectional RNN



$\boldsymbol{W}_{yh}$: connection weights from the sub-RNN $\boldsymbol{g}$ to output

One sub-RNN moves forward, same as before

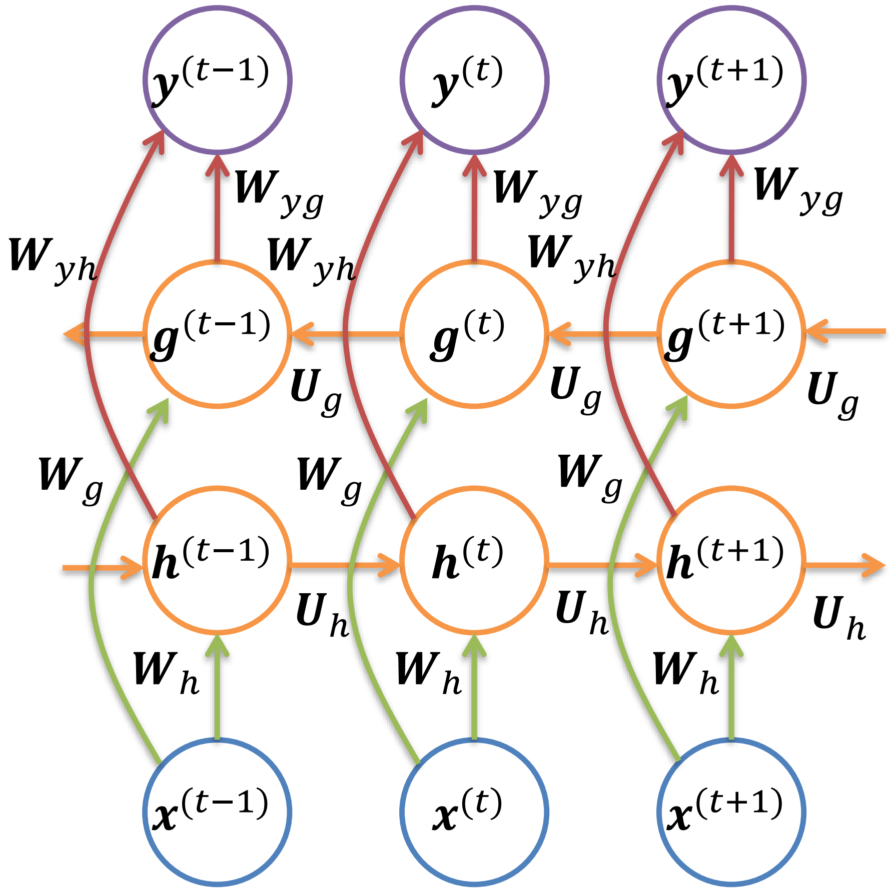
Bidirectional RNN



W_{yg}: connection weights from the sub-RNN g to output

One sub-RNN moves backward

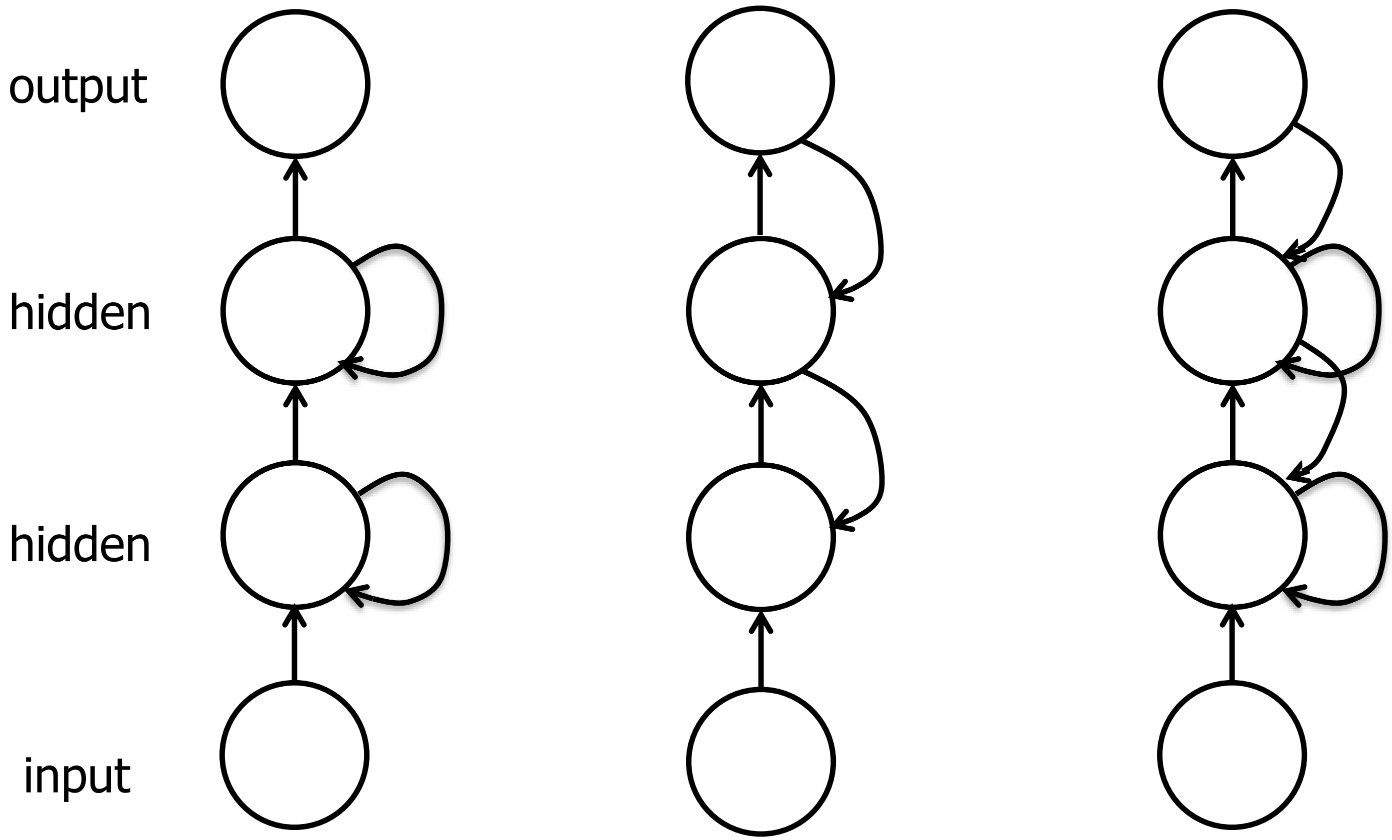
Bidirectional RNN



The output $\boldsymbol{y}^{(t)}$ recives input from both sub-RNNs via $\boldsymbol{W}_{yh}$ and $\boldsymbol{W}_{yg}$

Deep RNNs

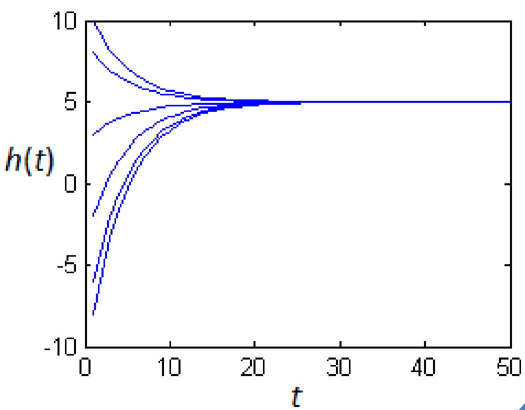
- Many ways to construct deep RNNs



Challenges

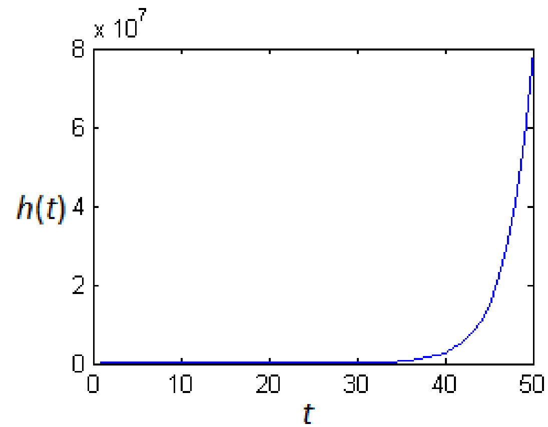
- Consider the 1D Elman network
$$h(t) = \sigma_h(Wx(t) + Uh(t-1) + b)$$
- Suppose σ_h is an identity mapping, $x(t)$ is a constant $\bar{x}$
$$h(t) = Uh(t-1) + W\bar{x} + b$$
  - If $U > 1$, $h(t)$ will approach infinity
  - If $U < 1$, $h(t)$ will converge to a fixed point

Autonomous system



$W\bar{x} + b$

$$h(t+1) = 0.8h(t) + 1$$



$$h(t+1) = 1.4h(t) + 1$$

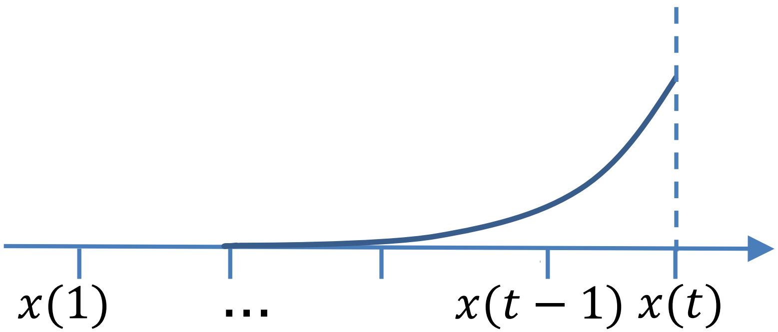
Challenges

- Even for time-varying $x(t)$, when $U > 1$, $h(t)$ will also approach infinity

$$h(t) = Uh(t-1) + Wx(t) + b$$

  - Let $b = 0$. After t steps from zero
    $h(t) = U^{t-1}Wx(1) + U^{t-2}Wx(2) + \cdots + Wx(t) + U^t h(0) \rightarrow \infty$
  - What if $U < 1$?

Contribution to $h(t)$



The contribution of $x(t-\tau)$ to $h(t)$ exponentially decays when $\tau > 0$ increases

$\boxed{h(t) \text{ is mainly determined by recent input}}$

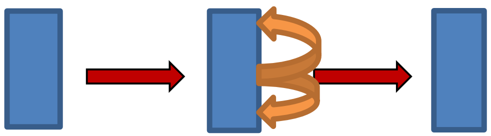
The short-term memory is too short!

Remarks

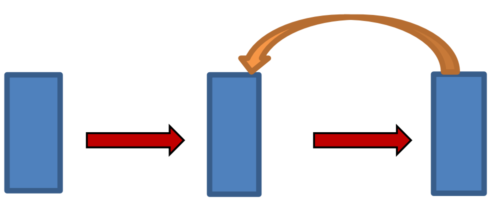
- Similar arguments hold for multi-dimensional Elman network
  - Some assumptions on U are needed, e.g., U is symmetric and has eigenvalue-eigenvector decomposition
- Same conclusions can be drawn on the Jordan network
  - You can express $\boldsymbol{y}(t)$ as a function of $\boldsymbol{y}(0)$ and $\boldsymbol{x}(1), \dots, \boldsymbol{x}(t)$

Summary of Part 2

Elman network



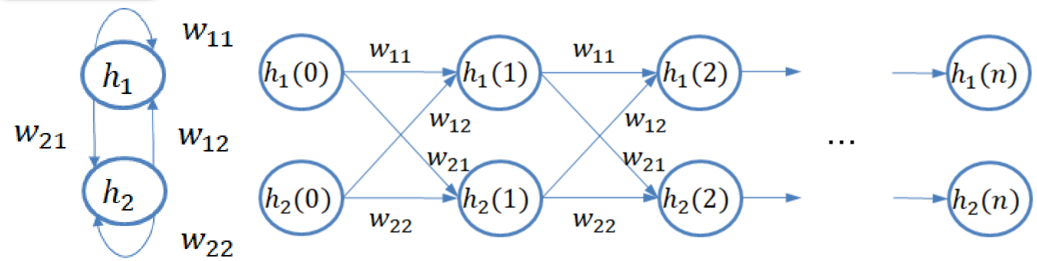
Jordan network



Bidirectional RNN

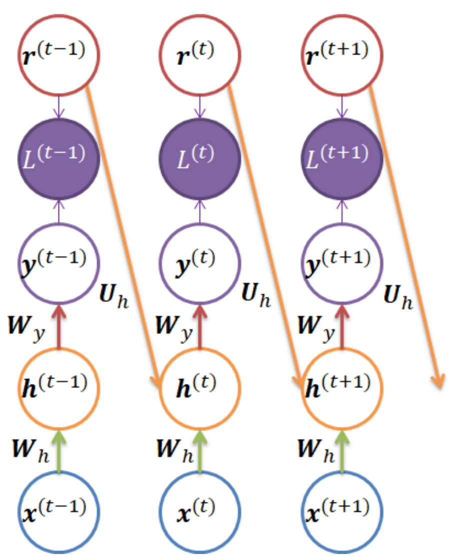
Deep RNN

BPTT



Different unfolding schemes

Teacher forcing



Multiple Choice(single) Points: 1

Setting

If you want your RNN’ s prediction depends on both history and future, which model would you choose?

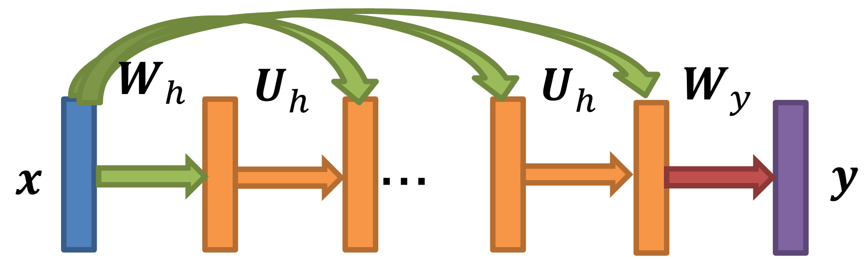
A Deep RNN

B Bidirectional RNN

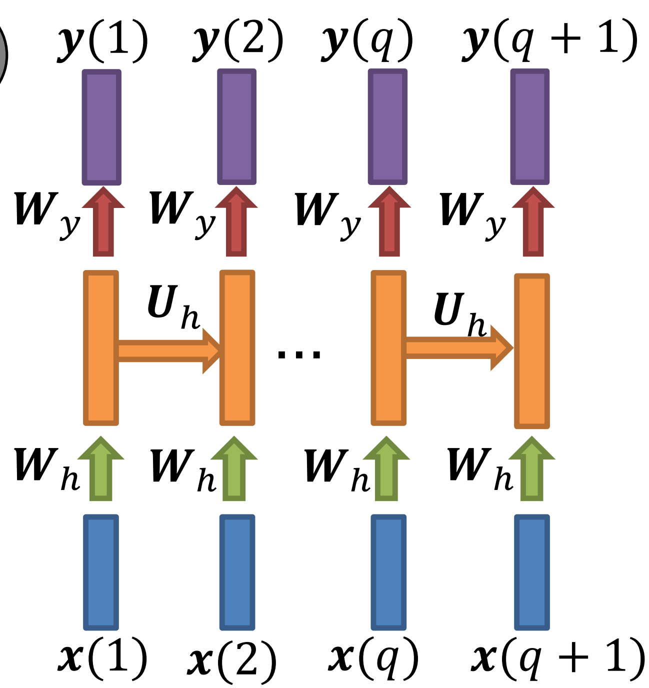
Submit

Consider using an RNN to do translation, which unfolding scheme suits this task?

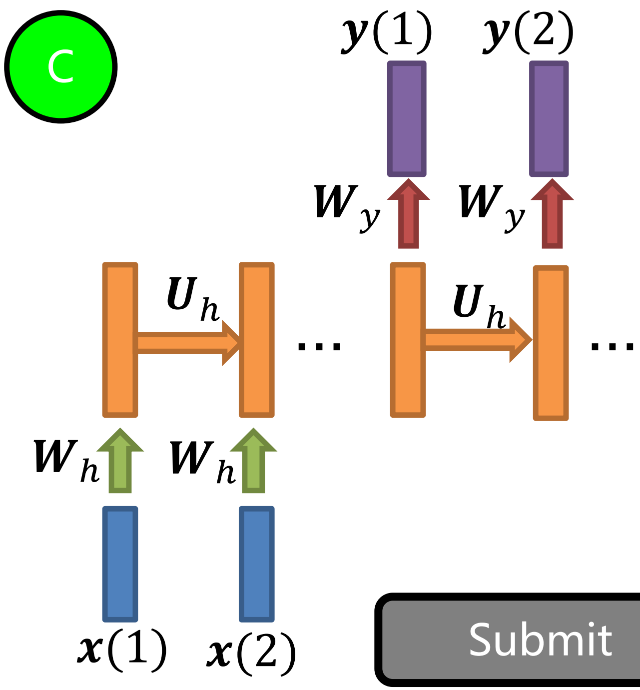
A



B



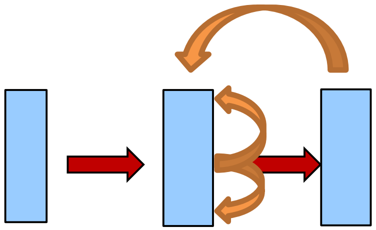
C



Submit

Setting

Can we use teacher forcing to train an RNN with both hidden-to-hidden and output-to-hidden recurrent connections?

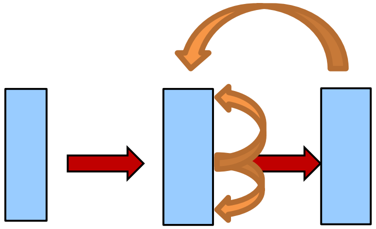


A Yes

B No

Submit

Consider training an RNN using teacher forcing, which has both hidden-to-hidden and output-to-hidden recurrent connections. Can we train all time steps in parallel?



A Yes

B No

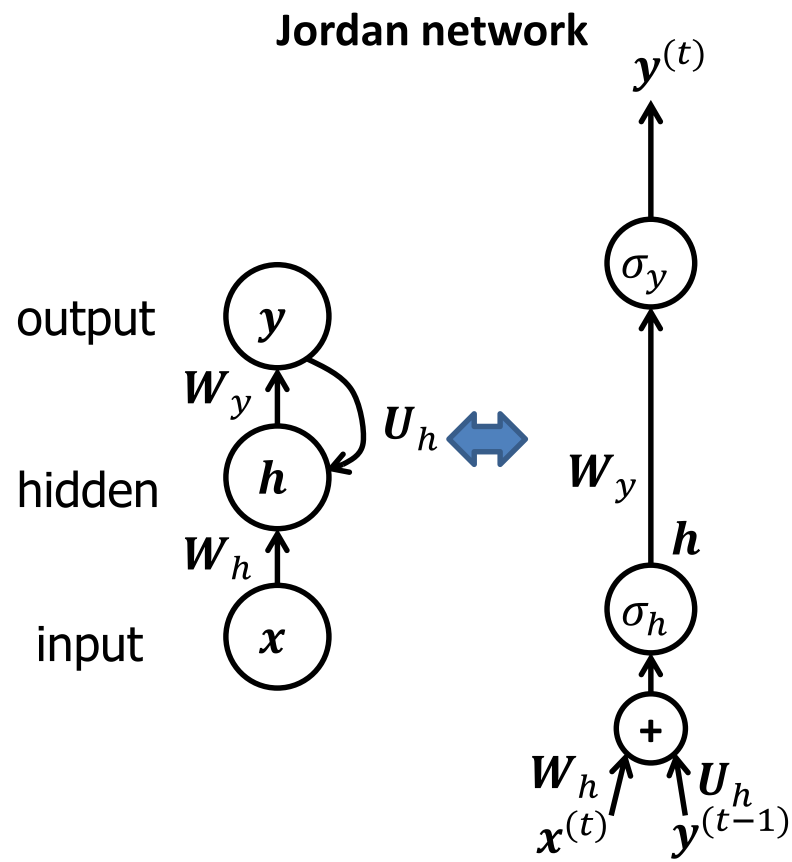
Submit

Outline

1. Dynamic systems
2. Simple RNNs
3. Gated RNNs
4. Applications to speech recognition
5. Summary

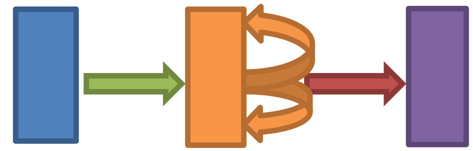
Long short-term memory (LSTM) cell

- It can be viewed as a combination of the Jordan network and the Elman network
  - The output is connect to the input
  - A self-loop is used to capture the information about the past
- Redraw the Jordan network
  - Use circles to denote operations
  - Variables are indicated on arrows
  - Bias $\boldsymbol{b}$ is ignored

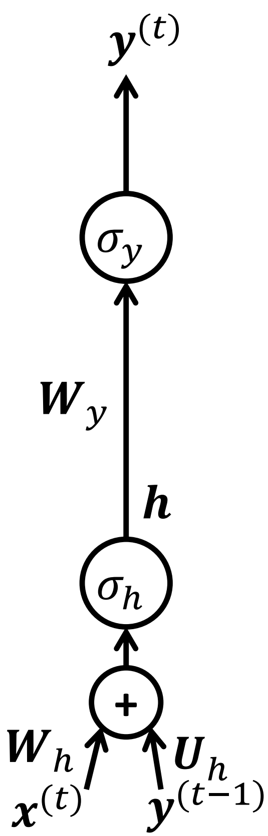


$$\boldsymbol{h}^{(t)} = \sigma_h\left(\boldsymbol{W}_h\boldsymbol{x}^{(t)} + \boldsymbol{U}_h\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_h\right)$$
$$\boldsymbol{y}^{(t)} = \sigma_y(\boldsymbol{W}_y\boldsymbol{h}^{(t)} + \boldsymbol{b}_y)$$

Step 1: add a self-loop



Jordan network

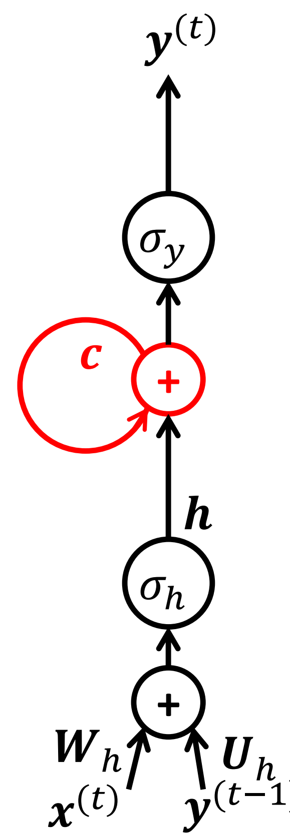


$$h^{(t)} = \sigma_h\left(W_h x^{(t)} + U_h y^{(t-1)} + b_h\right)$$
$$y^{(t)} = \sigma_y(W_y h^{(t)} + b_y)$$

$$h^{(t)} = \sigma_h\left(W_h x^{(t)} + U_h y^{(t-1)} + b_h\right)$$
$$c^{(t)} = c^{(t-1)} + h^{(t)}$$
$$y^{(t)} = \sigma_y(c^{(t)})$$

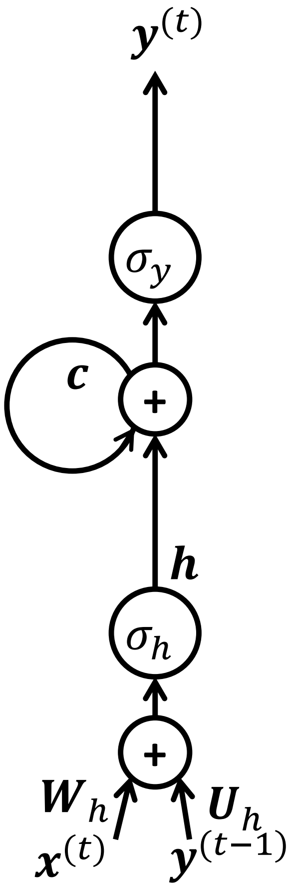
(We eliminate the linear transformation in the output)

- σ_h is either the logistic sigmoid function or tanh function
- σ_y is often tanh function



Step 2: add three gates

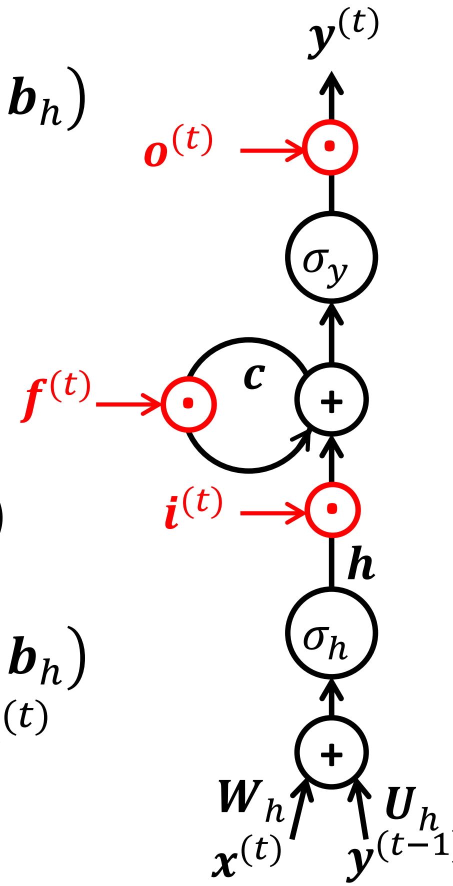
Gates are introduced to adaptively control the flow of information



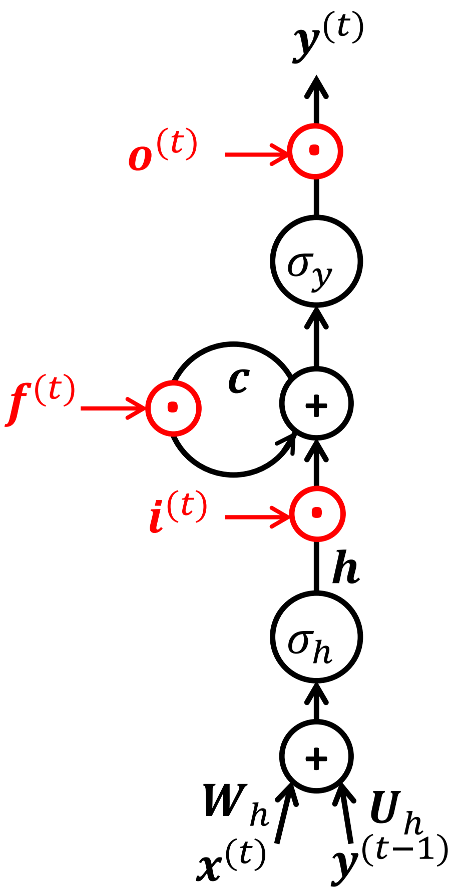
$$\boldsymbol{h}^{(t)} = \sigma_h\left(\boldsymbol{W}_h \boldsymbol{x}^{(t)} + \boldsymbol{U}_h \boldsymbol{y}^{(t-1)} + \boldsymbol{b}_h\right)$$
$$\boldsymbol{c}^{(t)} = \boldsymbol{c}^{(t-1)} + \boldsymbol{h}^{(t)}$$
$$\boldsymbol{y}^{(t)} = \sigma_y(\boldsymbol{c}^{(t)})$$

Forget gate $f^{(t)}$, input gate $i^{(t)}$, output gate $o^{(t)}$: between (0,1)

$$\boldsymbol{h}^{(t)} = \sigma_h\left(\boldsymbol{W}_h \boldsymbol{x}^{(t)} + \boldsymbol{U}_h \boldsymbol{y}^{(t-1)} + \boldsymbol{b}_h\right)$$
$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \boldsymbol{h}^{(t)}$$
$$\boldsymbol{y}^{(t)} = \boldsymbol{o}^{(t)} \odot \sigma_y(\boldsymbol{c}^{(t)})$$



What determine these gates?



$$\boldsymbol{h}^{(t)} = \sigma_h\left(\boldsymbol{W}_h\boldsymbol{x}^{(t)} + \boldsymbol{U}_h\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_h\right)$$
$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \boldsymbol{h}^{(t)}$$
$$\boldsymbol{y}^{(t)} = \boldsymbol{o}^{(t)} \odot \sigma_y(\boldsymbol{c}^{(t)})$$

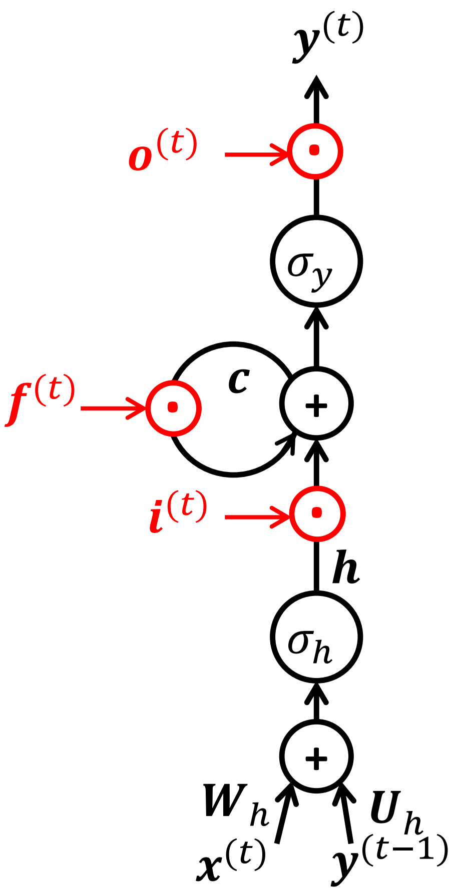
- All of the gates are determined by the input $\boldsymbol{x}^{(t)}$ and $\boldsymbol{y}^{(t-1)}$

$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{W}_f\boldsymbol{x}^{(t)} + \boldsymbol{U}_f\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_f\right)$$
$$\boldsymbol{i}^{(t)} = \sigma\left(\boldsymbol{W}_i\boldsymbol{x}^{(t)} + \boldsymbol{U}_i\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_i\right)$$
$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{W}_o\boldsymbol{x}^{(t)} + \boldsymbol{U}_o\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_o\right)$$

  where σ is the logistic sigmoid function

- Sometimes, they are also determined by $\boldsymbol{c}^{(t)}$ and $\boldsymbol{c}^{(t-1)}$: *peepholes*

Terminology



$$\boldsymbol{h}^{(t)} = \sigma_h\left(\boldsymbol{W}_h \boldsymbol{x}^{(t)} + \boldsymbol{U}_h \boldsymbol{y}^{(t-1)} + \boldsymbol{b}_h\right)$$
$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \boldsymbol{h}^{(t)}$$
$$\boldsymbol{y}^{(t)} = \boldsymbol{o}^{(t)} \odot \sigma_y(\boldsymbol{c}^{(t)})$$

- $\boldsymbol{x}^{(t)}$: input
- $\boldsymbol{y}^{(t)}$: output
- $\boldsymbol{h}^{(t)}$ and $\boldsymbol{c}^{(t)}$: hidden states
- $\boldsymbol{f}^{(t)}, \boldsymbol{i}^{(t)}$ and $\boldsymbol{o}^{(t)}$: gates

Note: sometimes, the output $\boldsymbol{y}$ is also called *hidden state of LSTM*, especially when LSTM is integrated into a larger system.

Multiple Choice(single) Points: 1

Setting

Can LSTM keep longer short-term memory than the Elman network?

A Yes

B No

Submit

Multiple Choice(single) Points: 1

Setting

$$\boldsymbol{h}^{(t)} = \sigma_h\left(\boldsymbol{W}_h \boldsymbol{x}^{(t)} + \boldsymbol{U}_h \boldsymbol{y}^{(t-1)} + \boldsymbol{b}_h\right)$$
$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \boldsymbol{h}^{(t)}$$
$$\boldsymbol{y}^{(t)} = \boldsymbol{o}^{(t)} \odot \sigma_y(\boldsymbol{c}^{(t)})$$

What' s the ideal case for keeping the memory $\boldsymbol{c}^{(n)}$ obtained at $t = n$ forever?

A $\boldsymbol{i}^{(t)} = 1$ and $\boldsymbol{o}^{(t)} = 1$ for $t \geq n + 1$

B $\boldsymbol{i}^{(t)} = 1$ and $\boldsymbol{f}^{(t)} = 0$ for $t \geq n + 1$

C $\boldsymbol{i}^{(t)} = 0$ and $\boldsymbol{f}^{(t)} = 1$ for $t \geq n + 1$

Submit

Advantage of LSTM

- The gates enable the model to keep the memory for a longer time than simple RNNs

"Long short-term memory network"

- Gates have been widely used in deep learning models, not only in RNNs but also CNNs
  - Highway Network (Srivastava et al., ICML 2015 Deep Learning workshop)
  - SEnet (Hu et al., CVPR 2018)
  - SKnet (Li et al., CVPR 2019)

It is used in a more broad sense: attention

Gated recurrent unit (GRU)

- In the Elman network, the hidden units $\boldsymbol{h}$ are used to capture the history information

$$\boldsymbol{h}^{(t)} = \sigma_h\left(\boldsymbol{W}_h \boldsymbol{x}^{(t)} + \boldsymbol{U}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_h\right)$$

- In an LSTM cell without gates, a new vector $\boldsymbol{c}$ is introduced for this purpose

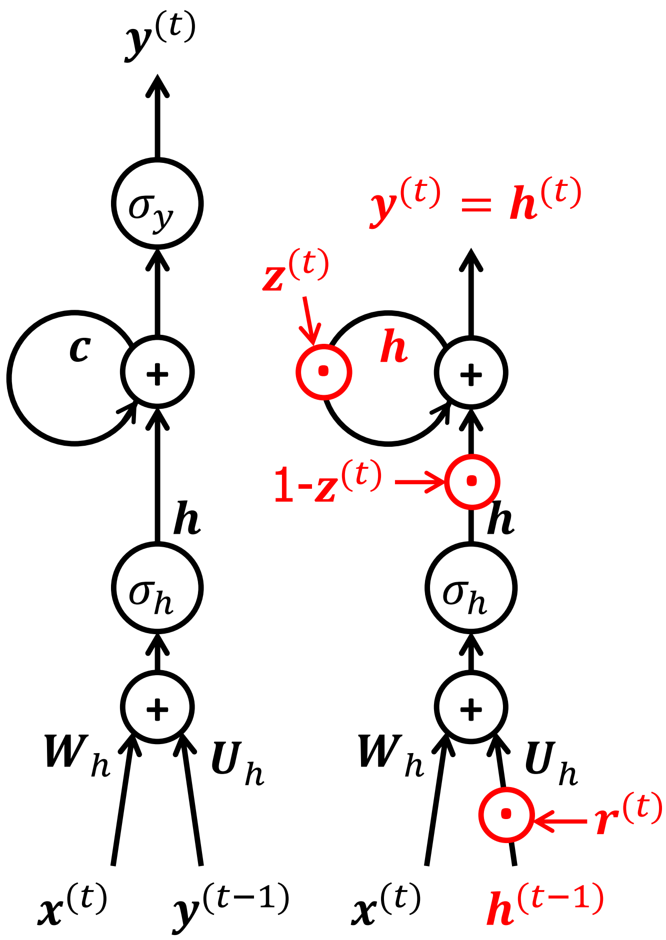
$$\boldsymbol{h}^{(t)} = \sigma_h\left(\boldsymbol{W}_h \boldsymbol{x}^{(t)} + \boldsymbol{U}_h \boldsymbol{y}^{(t-1)} + \boldsymbol{b}_h\right)$$
$$\boldsymbol{c}^{(t)} = \boldsymbol{c}^{(t-1)} + \boldsymbol{h}^{(t)}$$

- Why not use $\boldsymbol{h}$ directly?
- This is the 1st idea of GRU

$$\boldsymbol{h}^{(t)} = \boldsymbol{z}^{(t)} \odot \boldsymbol{h}^{(t-1)} + \left(1 - \boldsymbol{z}^{(t)}\right) \odot \tilde{\boldsymbol{h}}^{(t)}$$

where $\boldsymbol{z}^{(t)} \in (0,1)$ and

$$\tilde{\boldsymbol{h}}^{(t)} = \sigma_h\left(\boldsymbol{W}_h \boldsymbol{x}^{(t)} + \boldsymbol{U}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_h\right)$$



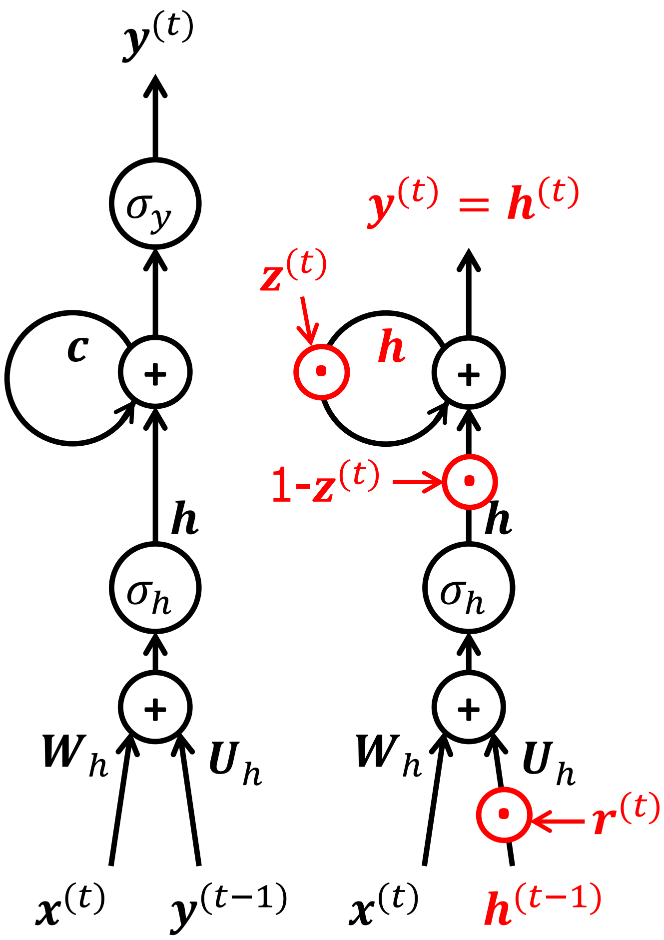
LSTM cell w/o gates GRU

Gated recurrent unit (GRU)

- The 2nd idea of GRU
  - Let output be equal to hidden states: $\boldsymbol{y}^{(t)} = \boldsymbol{h}^{(t)}$
- The 3rd idea of GRU
  - Use a gate to modulate the recurrent input

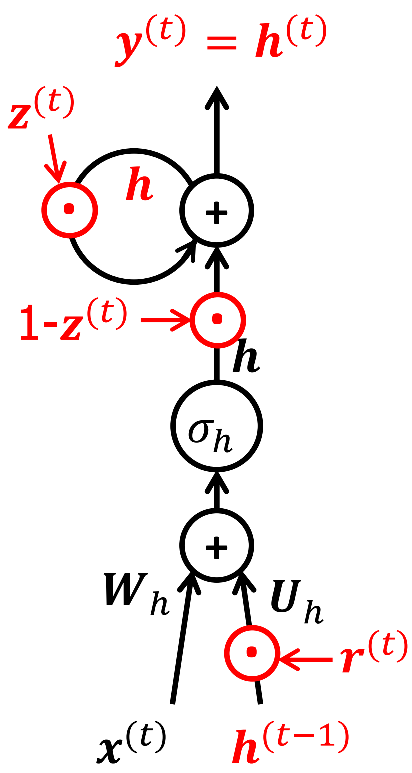
$$\widetilde{\boldsymbol{h}}^{(t)} = \sigma_h\left(\boldsymbol{W}_h\boldsymbol{x}^{(t)} + \boldsymbol{U}_h\left(\boldsymbol{r}^{(t)} \odot \boldsymbol{h}^{(t-1)}\right) + \boldsymbol{b}_h\right)$$

- The gates depend on input and hidden states
  - Update gate $\boldsymbol{z}^{(t)} = \sigma\left(\boldsymbol{W}_z\boldsymbol{x}^{(t)} + \boldsymbol{U}_z\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_z\right)$
  - Reset gate $\boldsymbol{r}^{(t)} = \sigma\left(\boldsymbol{W}_r\boldsymbol{x}^{(t)} + \boldsymbol{U}_r\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_r\right)$



LSTM cell w/o gates GRU

GRU in summary



- Dynamic equations

$$\boldsymbol{h}^{(t)} = \boldsymbol{z}^{(t)} \odot \boldsymbol{h}^{(t-1)} + \left(1 - \boldsymbol{z}^{(t)}\right) \odot \widetilde{\boldsymbol{h}}^{(t)}$$

$$\widetilde{\boldsymbol{h}}^{(t)} = \sigma_h\left(\boldsymbol{W}_h \boldsymbol{x}^{(t)} + \boldsymbol{U}_h\left(\boldsymbol{r}^{(t)} \odot \boldsymbol{h}^{(t-1)}\right) + \boldsymbol{b}_h\right)$$

  where σ_h is either the logistic sigmoid function or tanh function

- The gates

$$\boldsymbol{z}^{(t)} = \sigma\left(\boldsymbol{W}_z \boldsymbol{x}^{(t)} + \boldsymbol{U}_z \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_z\right)$$

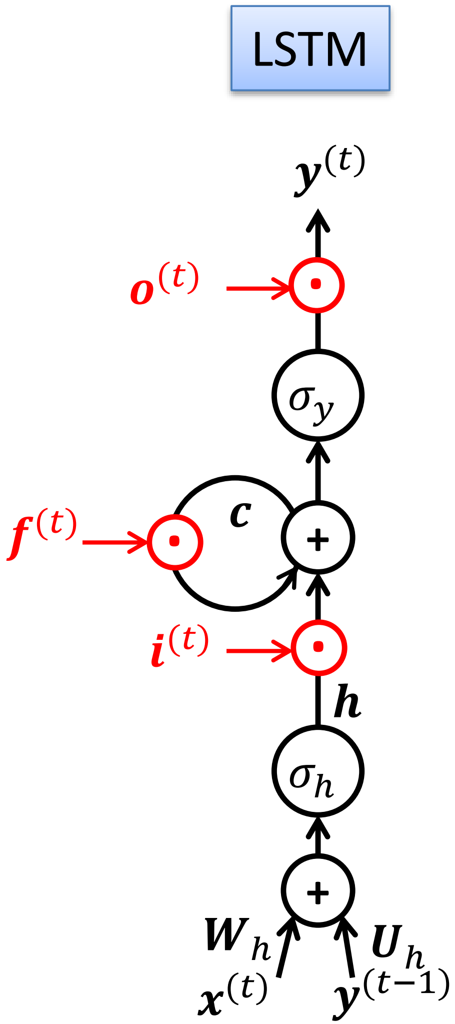
$$\boldsymbol{r}^{(t)} = \sigma\left(\boldsymbol{W}_r \boldsymbol{x}^{(t)} + \boldsymbol{U}_r \boldsymbol{h}^{(t-1)} + \boldsymbol{b}_r\right)$$

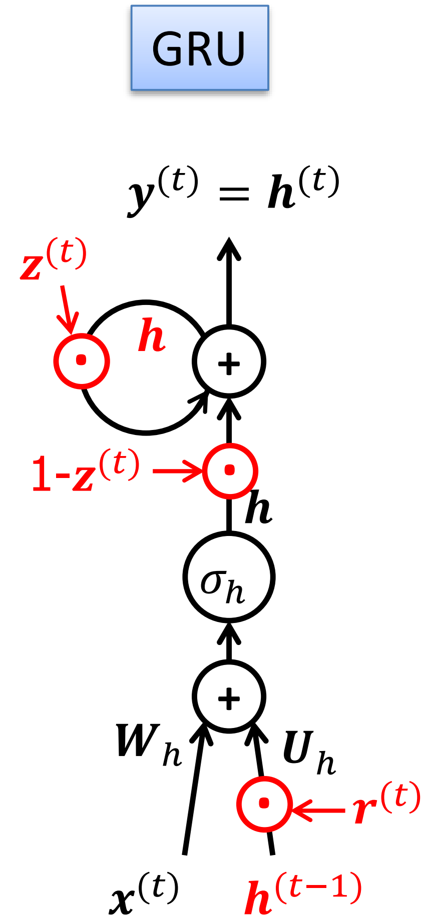
  σ is the logistic sigmoid function

Which is better?

- LSTM and GRU perform similarly on many tasks
- There are many other variants of LSTM and GRU, but none of them would clearly beat these two across a wide range of tasks (Greff et al., TNNLS 2017)

Summary of Part 3





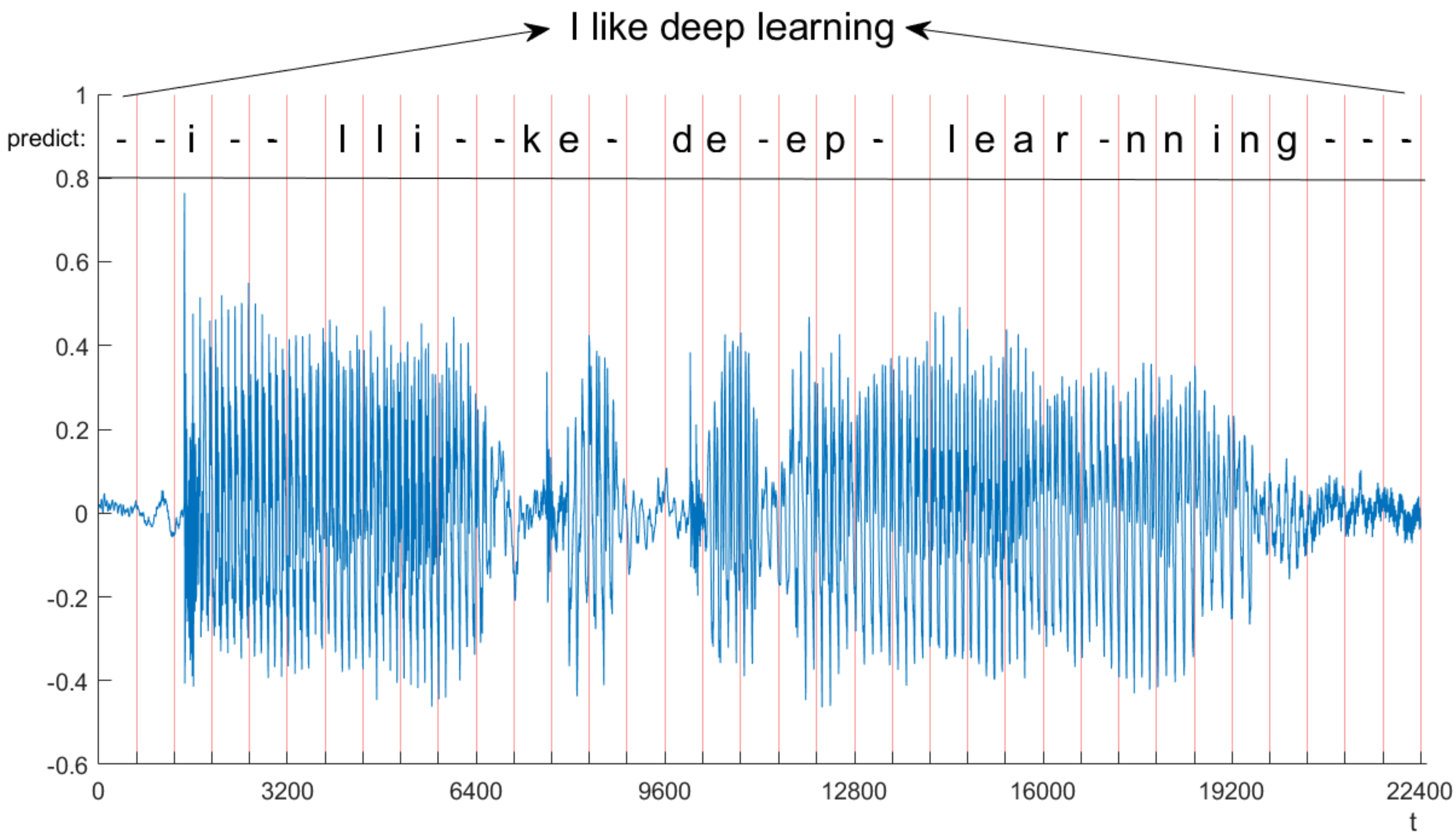
Outline

1. Dynamic systems
2. Simple RNNs
3. Gated RNNs
4. Applications to speech recognition
5. Summary

Speech recognition

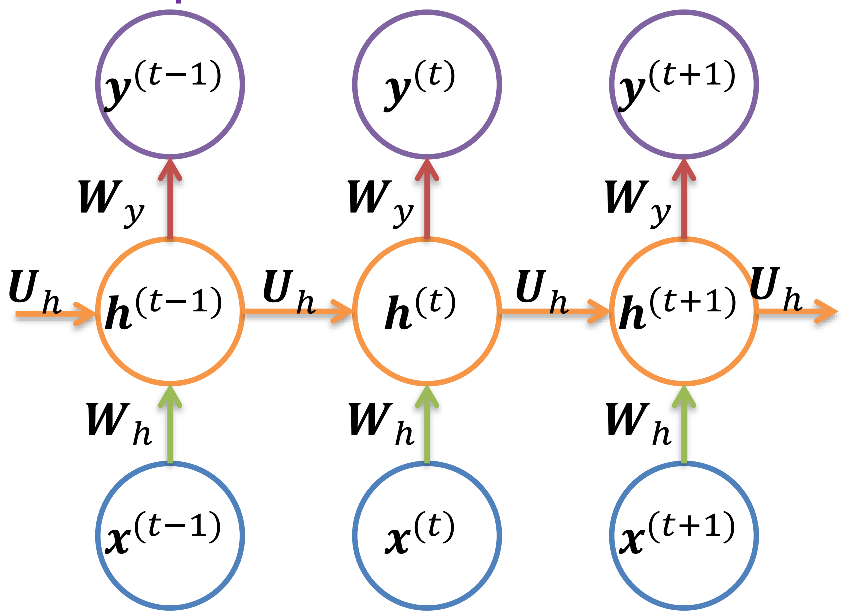
Apple Siri

Speech recognition



RNN setting

There is a reference $\boldsymbol{r}^{(t)}$ at each time step



- Suppose there is no act fun

$$\boldsymbol{y}^{(t)} = \boldsymbol{W}_y \boldsymbol{h}^{(t)} + \boldsymbol{b}_y$$

- At every time step, use a softmax fun to predict an output, e.g., a phoneme or a blank
  - There are $K+1$ classes at time t, where K is the number of phonemes, usually < 100

$$P\left(\mathrm{r}_k^{(t)} = 1 | \boldsymbol{h}^{(t)}\right) = \frac{\exp\left(y_k^{(t)}\right)}{\sum_{k=1}^{K+1} \exp\left(y_k^{(t)}\right)}$$

A random variable, not reference value

Objective function

- Maximize the prob of reference class at all time steps

$$\max_{\boldsymbol{\theta}} \sum_{t=1}^{T} \ln P\left(\mathrm{r}_k^{(t)} = r_k^{(t)} | \boldsymbol{h}^{(t)}\right)$$

Reference value which is 1

- This is equivalent to minimizing the cross-entropy error
  - The cross-entropy error at time t

$$-\sum_{k=1}^{K+1} r_k^{(t)} \ln p\left(\mathrm{r}_k^{(t)} = 1 \middle| \boldsymbol{h}^{(t)}\right) = -\ln p\left(\mathrm{r}_k^{(t)} = 1 \middle| \boldsymbol{h}^{(t)}\right)$$

    where k satisfies $r_k^{(t)} = 1$ because other elements of $\boldsymbol{r}^{(t)}$ are zeros

  - Sum the cross-entropy error over time

$$-\sum_{t=1}^{T} \ln p\left(\mathrm{r}_k^{(t)} = 1 \middle| \boldsymbol{h}^{(t)}\right)$$

Objective function

- Optimizing the objective function in the previous slide will result in T outputs

"learning"

ϕ ϕ /l/ /ə/ /ə/ /ə/ ϕ ϕ /n/ /i/ /i/ /ŋ/ ϕ

ϕ is blank

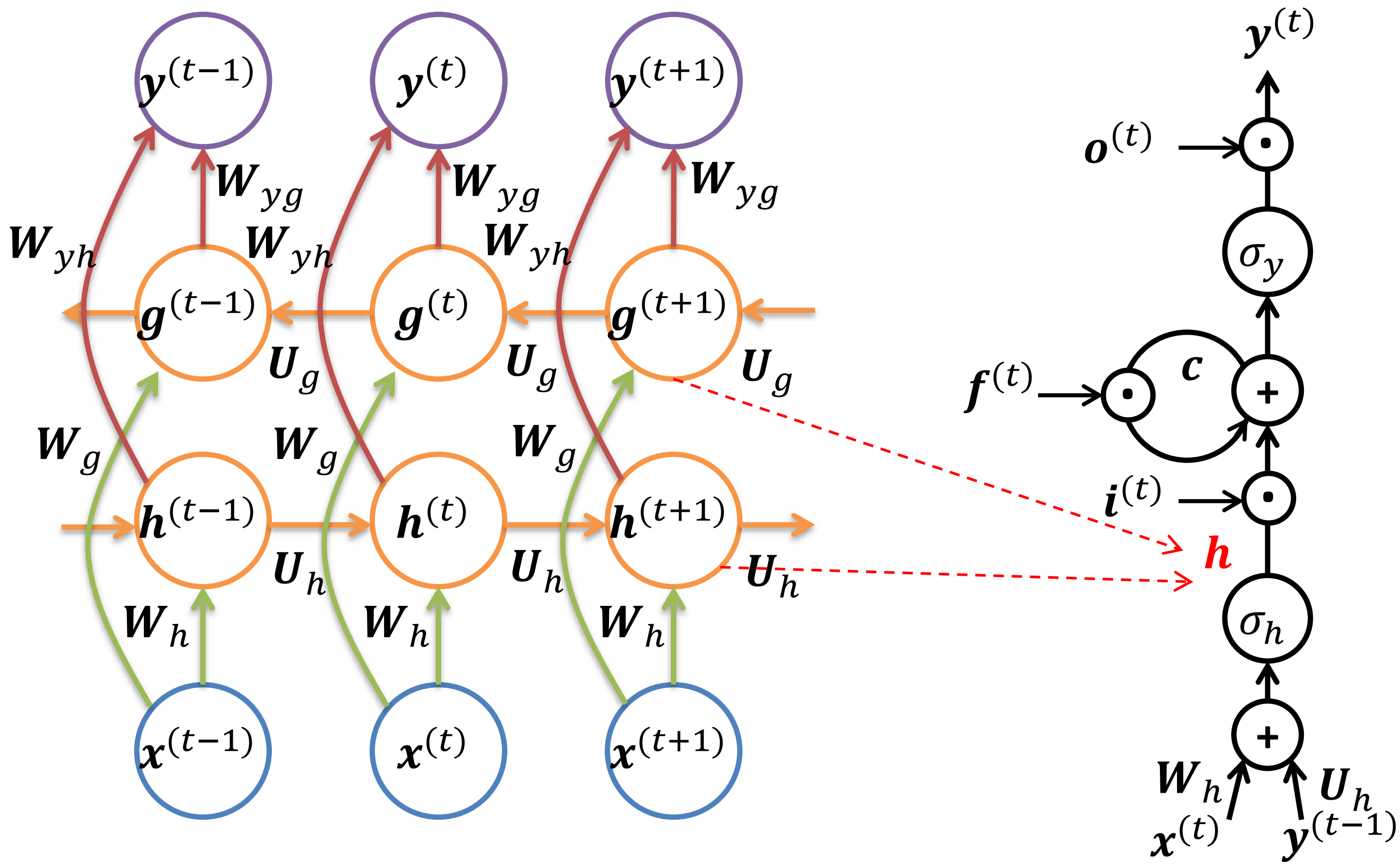
/l/ /ə/ /r/ /n/ /i/ /ŋ/

- But the reference sequence may be shorter or longer than it
- How to measure the difference and minimize the difference?
  – "edit-distance" is introduced: the minimum number of insertions, substitutions and deletions required to change seq p into seq q
  – A method called "Connectionist Temporal Classification (CTC)" is usually used (Graves et al., 2006)

Use bidirectional LSTM

Graves et al., 2013

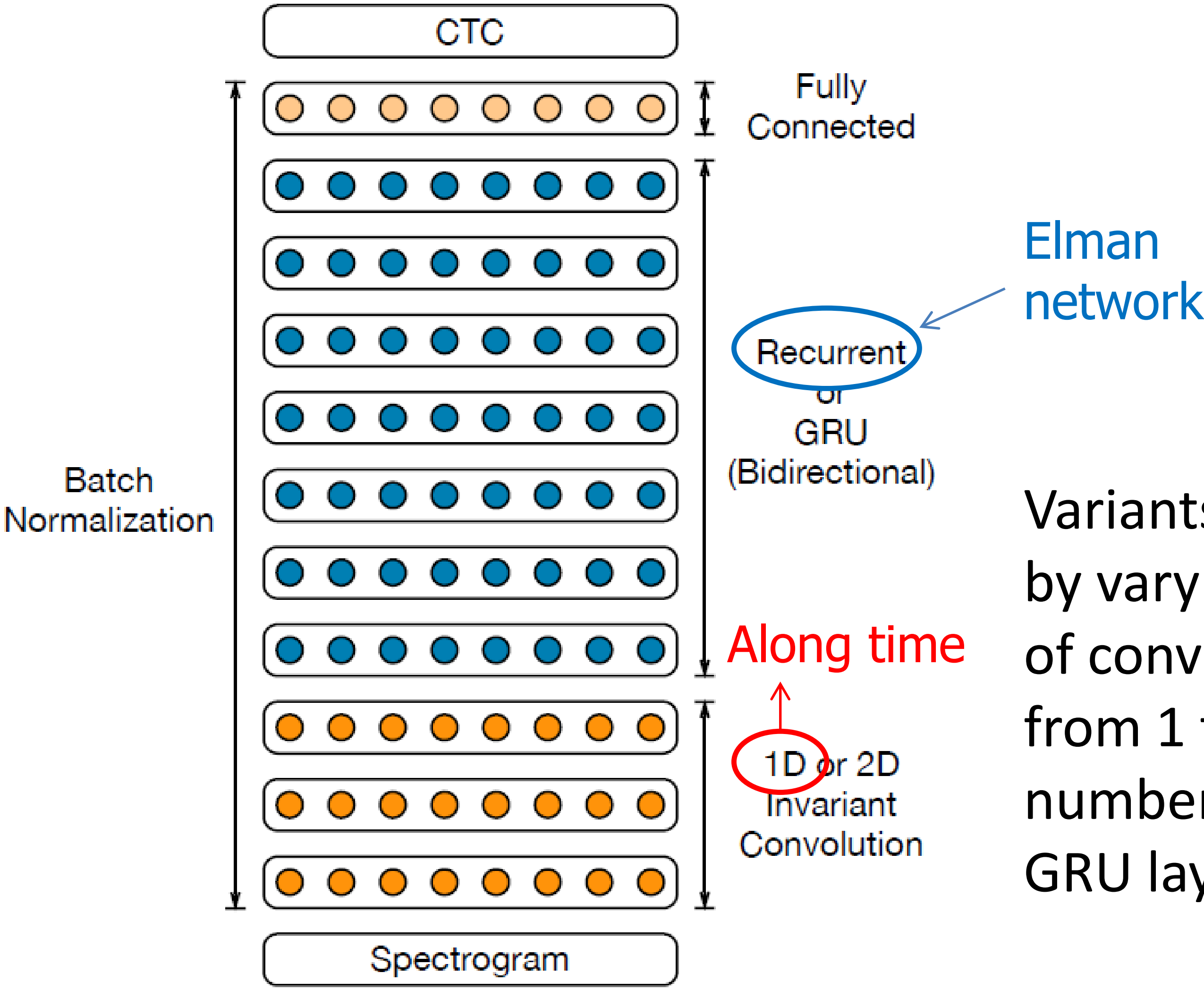
Each of the two RNNs are LSTMs



Define the obj fun the same as before based on: $y^{(t)} = W_{yh} h^{(t)} + W_{yg} g^{(t)} + b_y$

Deep RNN – Deep Speech 2 by Baidu

Amodei et al., 2015



Variants were explored by varying the number of convolutional layers from 1 to 3 and the number of recurrent or GRU layers from 1 to 7

Data preprocessing

- Usually, the input to the model is not raw wav signal, but the spectral-temporal signal
  - In (Graves et al., 2013), the audio data was encoded using a Fourier-transform-based filter-bank with 40 coefficients (plus energy) distributed on a mel-scale, together with their first and second temporal derivatives
  - Each input vector was therefore size 123
  - The data were normalized so that every element of the input vectors had zero mean and unit variance over the training set

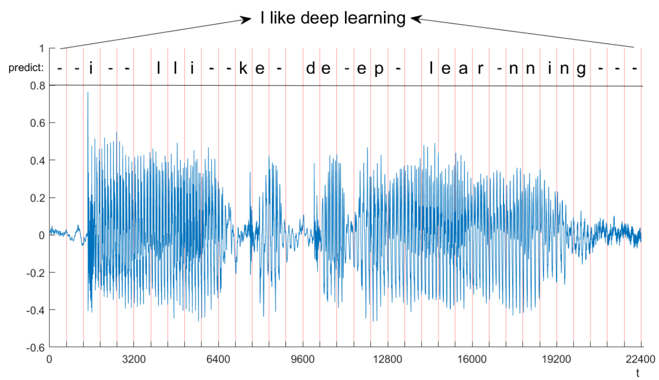
Benchmark datasets

- Benchmark datasets
  - TIMIT, small
  - Switchborad, 260 hours
  - LibriSpeech, 1000 hours
  - CHiME, with various enviroment noises
- Many benchmark datasets are only used for testing, and you need to use your own training set
  - Deep speech 2 the English system was trained on 11,940 hours of English speech, while the Mandarin system was trained on 9,400 hours. Data synthesis was used to further augment the data.
- Researchers tend to opensource their models but do not release the training set
  - This makes the evaluation of different models difficult

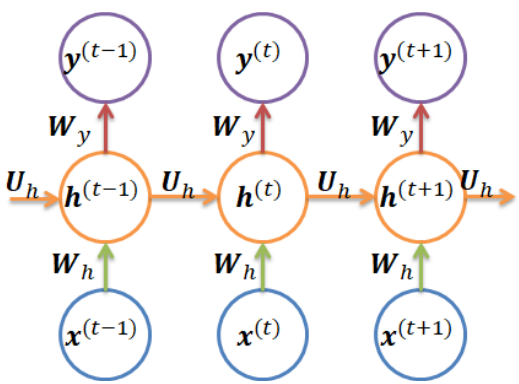
State of the art

- In 2017, Microsoft announced that their speech recognition system has achieved 5.1% word error rate (WER) on Switchboard
  - This is average level of professional transcribers
- However, all current models perform poorly on noisy data
  - The lowest WER in *CHiME 2018 Challenge* is about 50%. See results here: http://spandh.dcs.shef.ac.uk/chime_challenge/results.html

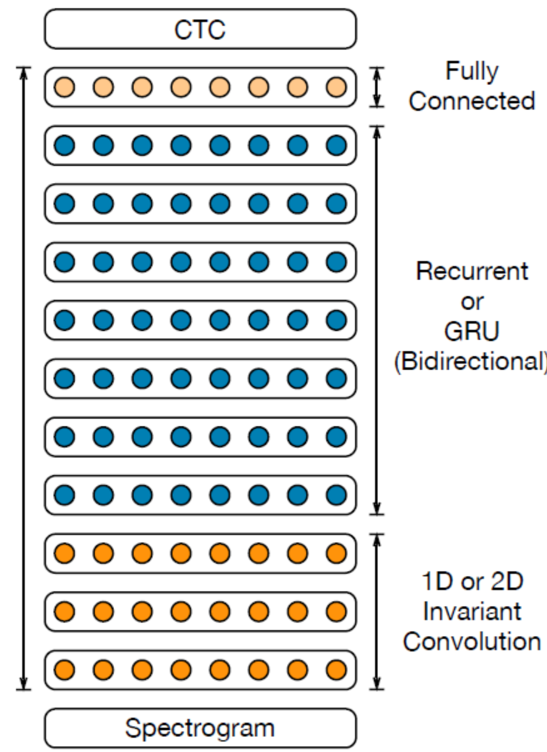
Summary of Part 4



Elman network



Deep Speech 2



Difficulty: difference between sequences

Outline

1. Dynamic systems
2. Simple RNNs
3. Gated RNNs
4. Applications to speech recognition
5. Summary

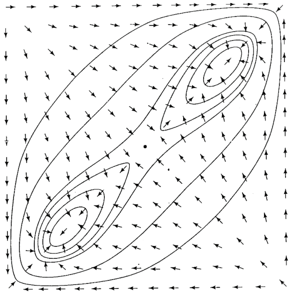
Summary of this lecture

Knowledge

1. Dynamic systems

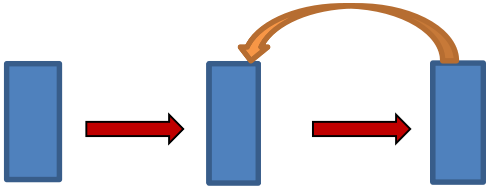
Model dynamic systems

Model the brain*

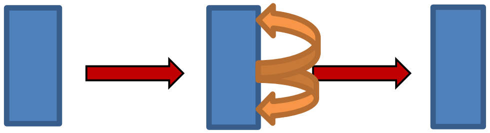


2. Simple RNNs

Jordan network



Elman network



Training

BPTT

Teacher forcing

Extensions

Bidirectional RNN

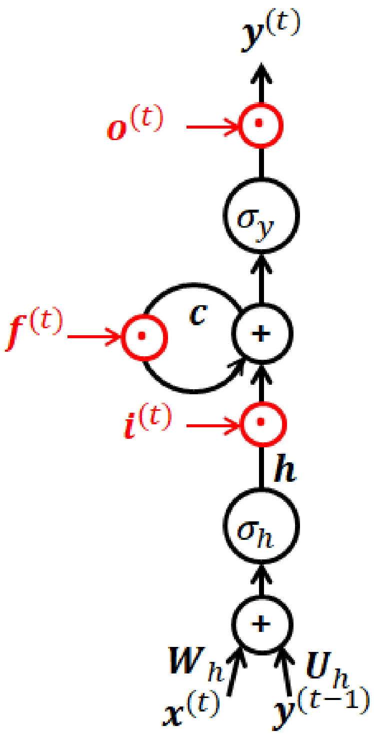
Deep RNN

Summary of this lecture

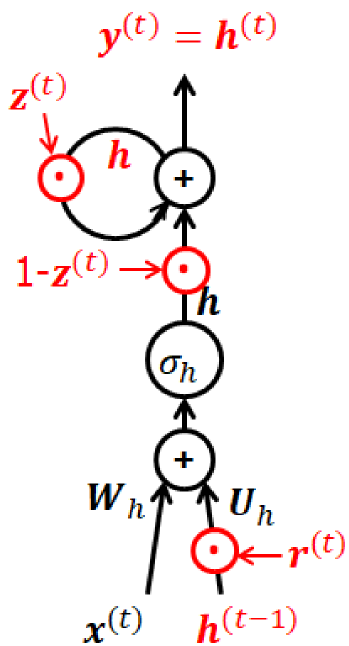
Knowledge

3. Gated RNNs

LSTM

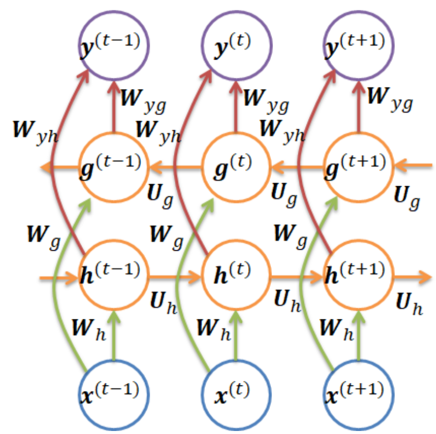


GRU

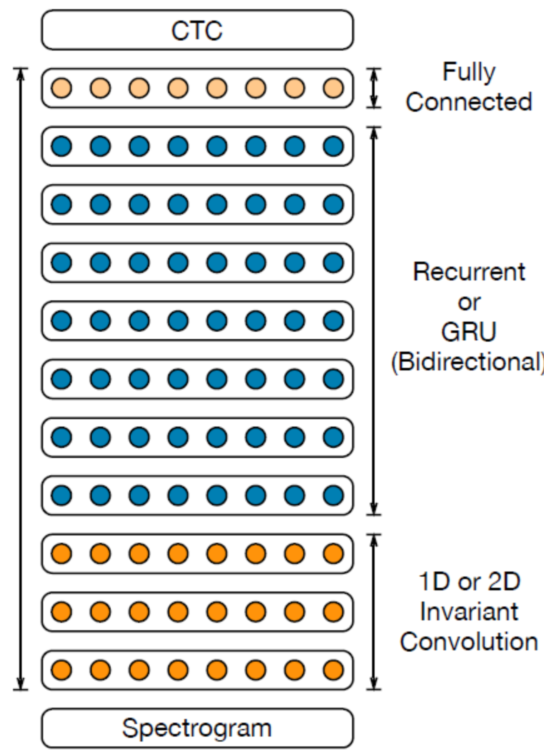


4. Applications to speech recognition

Bidirectional LSTM



Deep Speech 2

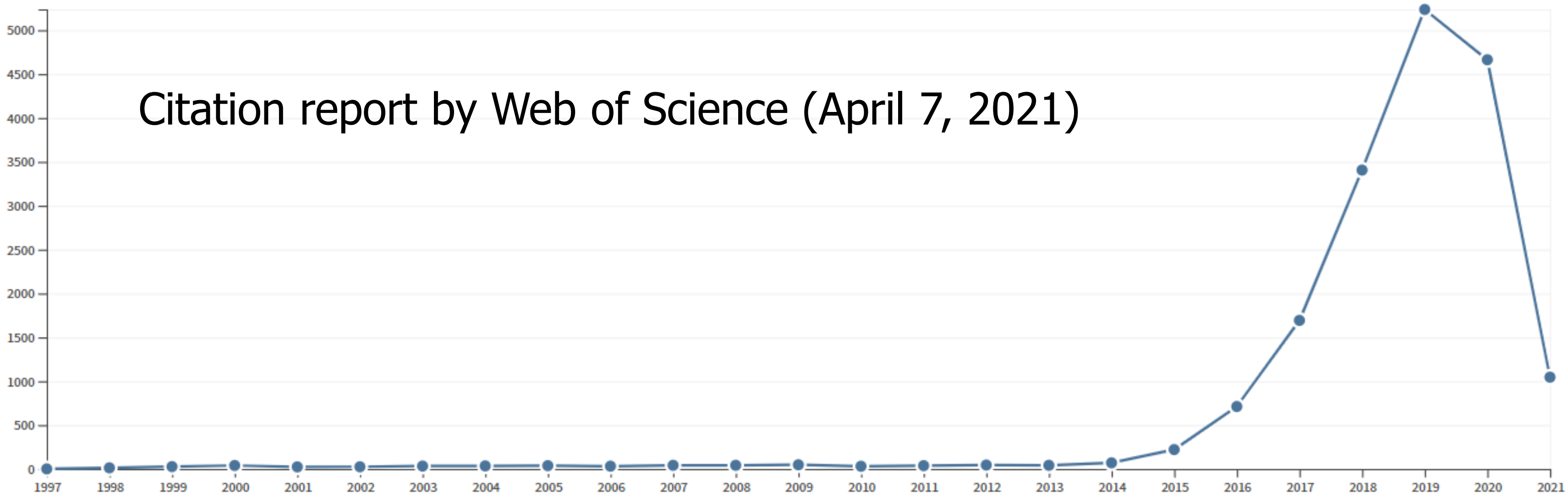


Summary of this lecture

Value

Hochreiter, S, Schmidhuber, J, Long short-term memory, Neural Computation, vol 9, no. 8, pp. 1735-1780, 1997

Jürgen Schmidhuber



A good work needs time to be recognized!

Recommended reading

- Goodfellow, Bengio and Courville, 2016
  Deep Learning, MIT Press, Chapters 10
- Understanding LSTM networks
  http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Graves, Mohamed, Hinton (2013)
  Speech recognition with deep recurrent neural networks
  IEEE ICASSP

Prepare for the next lecture

- Form groups of 2 and every group prepares a 5-minute presentation with slides for one of the following papers
  - Greff, Srivastava, Koutník, Steunebrink, Schmidhuber, “LSTM: a search space odyssey,” IEEE Trans. on Neural Networks and Learning Systems, 2017
  - Liang, Hu, “Recurrent convolutional neural network for object recognition,” CVPR 2015