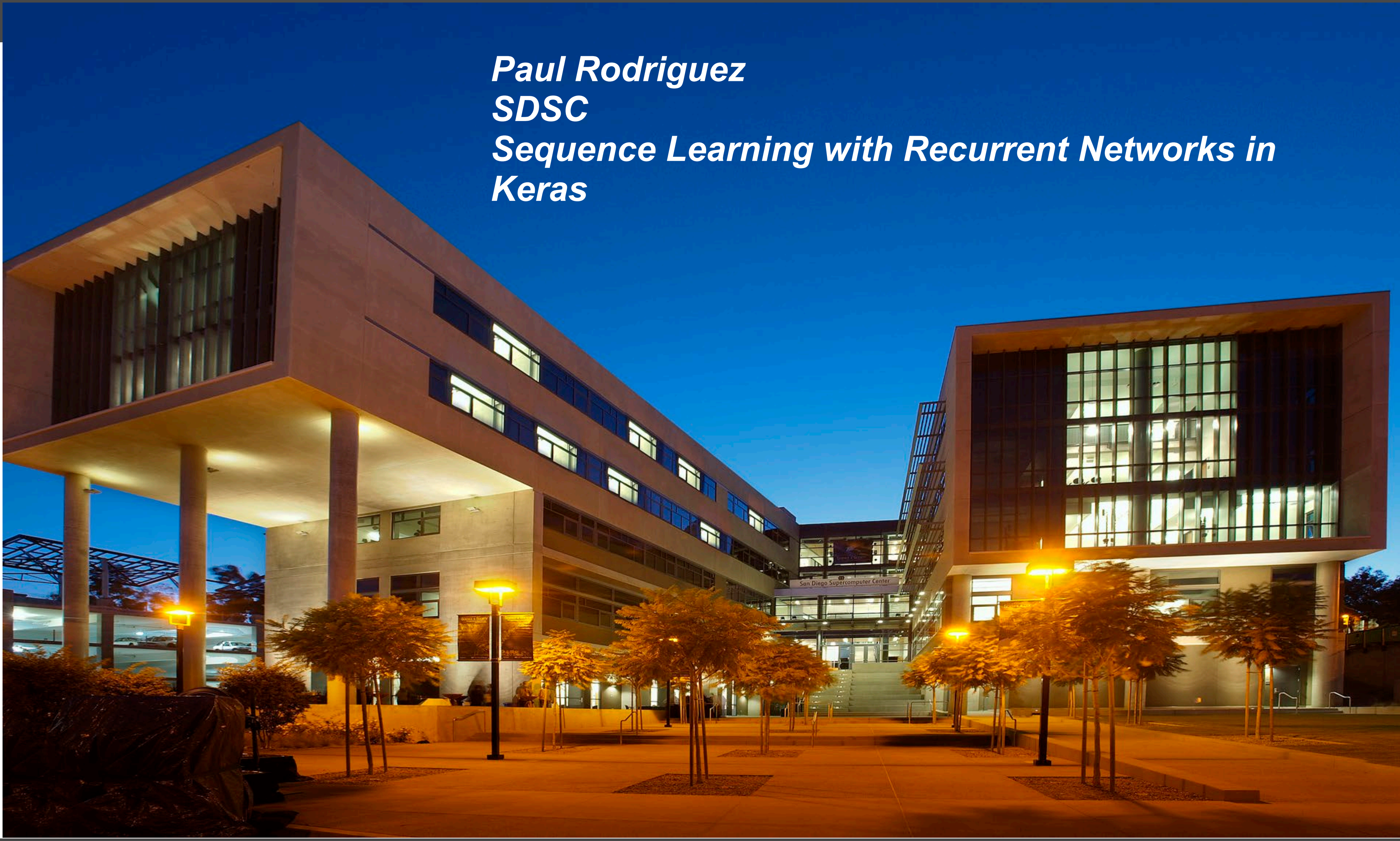
CIML



Paul Rodriguez
SDSC
Sequence Learning with Recurrent Networks in Keras

SDSC SAN DIEGO SUPERCOMPUTER CENTER

UC San Diego

Outline

Applications of Sequence Learning

Recurrent Networks and Memory Units

Some Varieties of Recurrent Models

RNNs in Keras and exercise

Sequence Learning

Language or grammar sequences

Time series, autoregressive models

Machine Translation

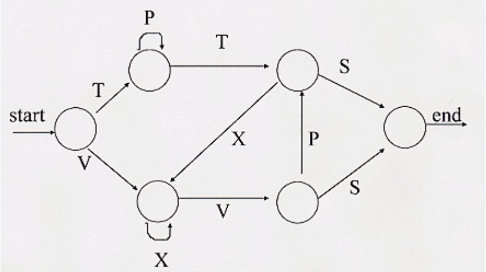
Video, Image sequences

many others…

All involve learning about dependencies in time, often with variable length input

Artificial Grammar Learning:

predict next letter and END-OF-SEQUENCE



generates:

TPPTSXVS - END
VXVPS - END
TTS – END
VXXVPXVS – END
etc…

Inputs and Outputs use 1 hot vectors of input

	P	S	T	V	X	END
T	0	0	**1**	0	0	0
P	**1**	0	0	0	0	0
P	**1**	0	0	0	0	0
…	…	..	…	…	…	…
END	0	0	0	0	0	**1**

Temporal Sequence Classification:

given a sequence of words can you classify the movie review sentiment

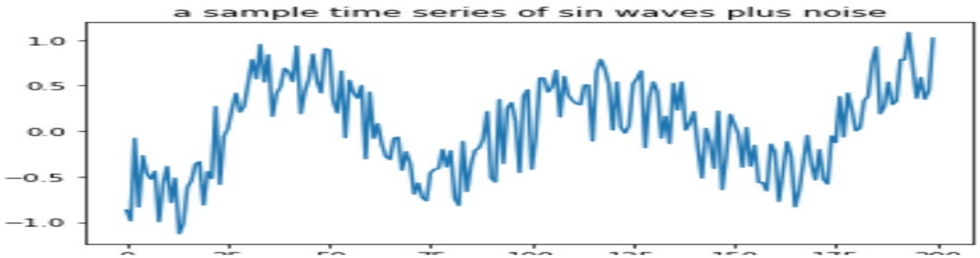
id	sentiment	review
4518_9	1	Adrian Pasdar is excellent is this film. He makes a fascinating woman.
874_1	0	Long, boring, blasphemous. Never have I been so glad to see ending credits roll.
3247_10	1	I don't know why I like this movie so well, but I never get tired of watching it.

1 hot encoding would be too large, so Inputs and Outputs use word embeddings (ie a reduced space of real values)

	H1	H2	H3	H4	H5	…
Long	0.3	0.1	0.01	0.94	0.1	0.43
boring	0.01	0.2	03	0.51	0.2	0.62
I	…	..	…	…	…	…
have						
…						

Time series, autoregressive models

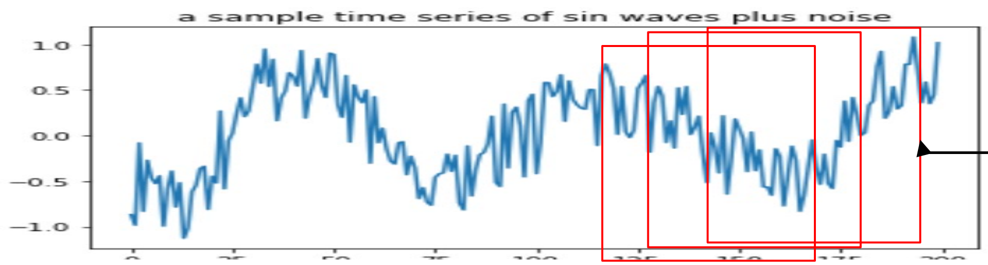
Predict rainfall for next hour, or next 24 hours (Kaggle)



Real valued input → output at each time step

Time series, autoregressive models

Predict rainfall for next hour, or next 24 hours (Kaggle)

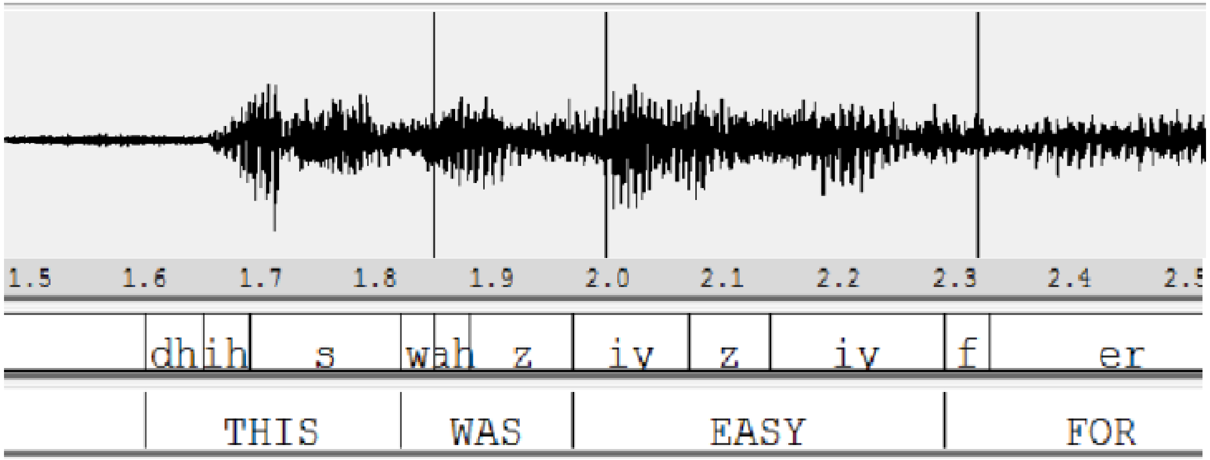


Real valued input → output at each time step

Or, you can feed in many time lags at once ("multi-step input")

This is like sliding a 1D convolution layer and getting a sequence of convolution outputs

Continuous speech phoneme/word recognition (TIMIT challenge)



RNNs with convolution were doing well, but lately (2020),

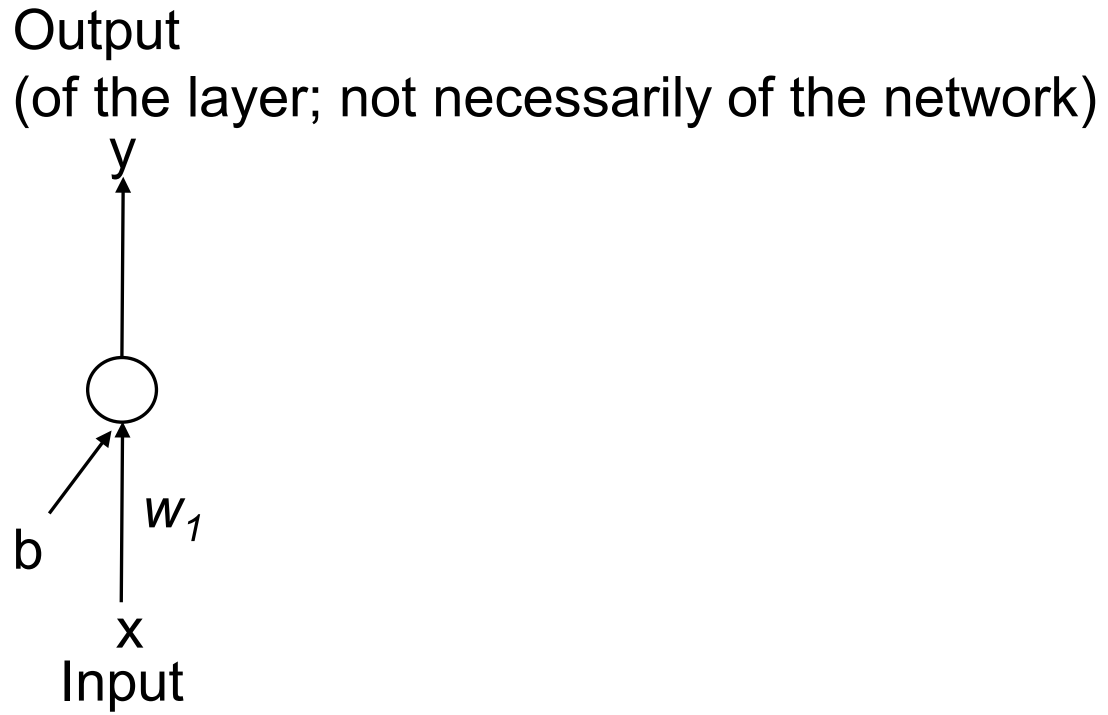
RNNs might be getting beat out by convolution over lagged inputs (of all time steps), with positional information.

But for more varied length input positional information might not generalize as well

What is a Recurrent Neural Network?

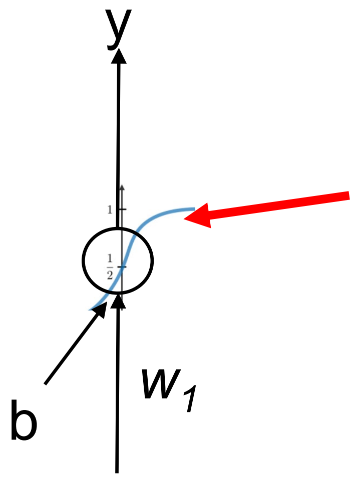
First, recall a Neural Network node

x,w,y are possibly vectors



First, recall a Neural Network node

Output
(of the layer; not necessarily of the network)

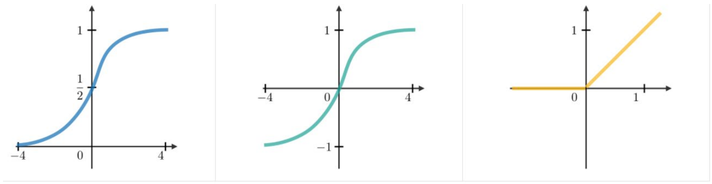


Input

An activation function,
usually one of these:

x,w,y are possibly vectors

Logistic Tanh RELU



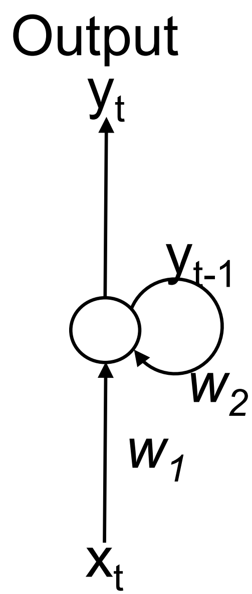
Add recurrent connection

Let add feedback and work with discrete time steps

This is a dynamical system –

Y(t) depends on Y(t-1)

How does it learn?

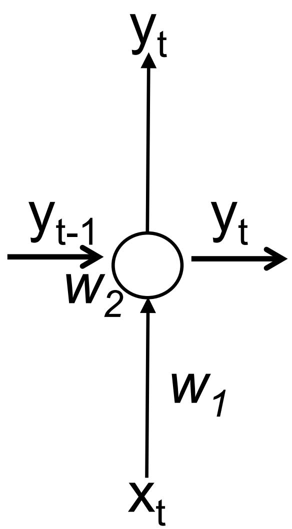


‘Unrolling’ the network

This is a dynamical system – how does it learn?

Let’s ‘unroll’ it in time by making copies.

Output at t

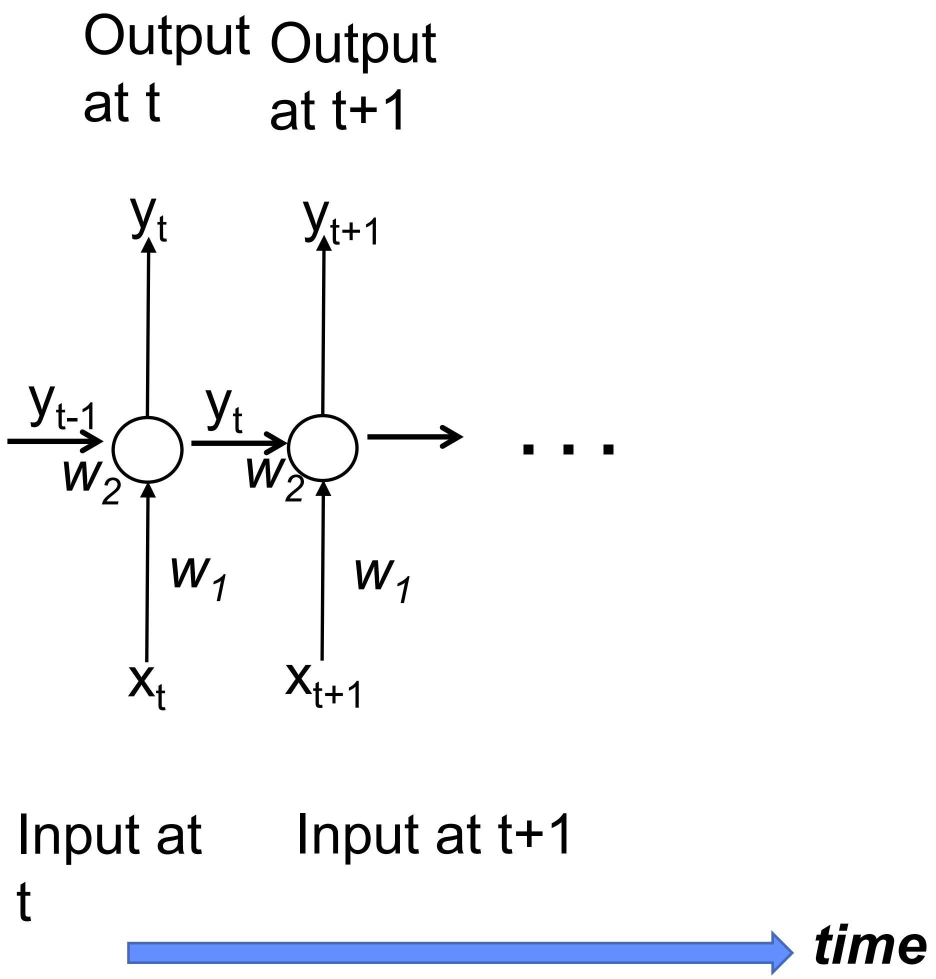


Input at t

‘Unrolling’ the network

This is a dynamical system – how does it learn?

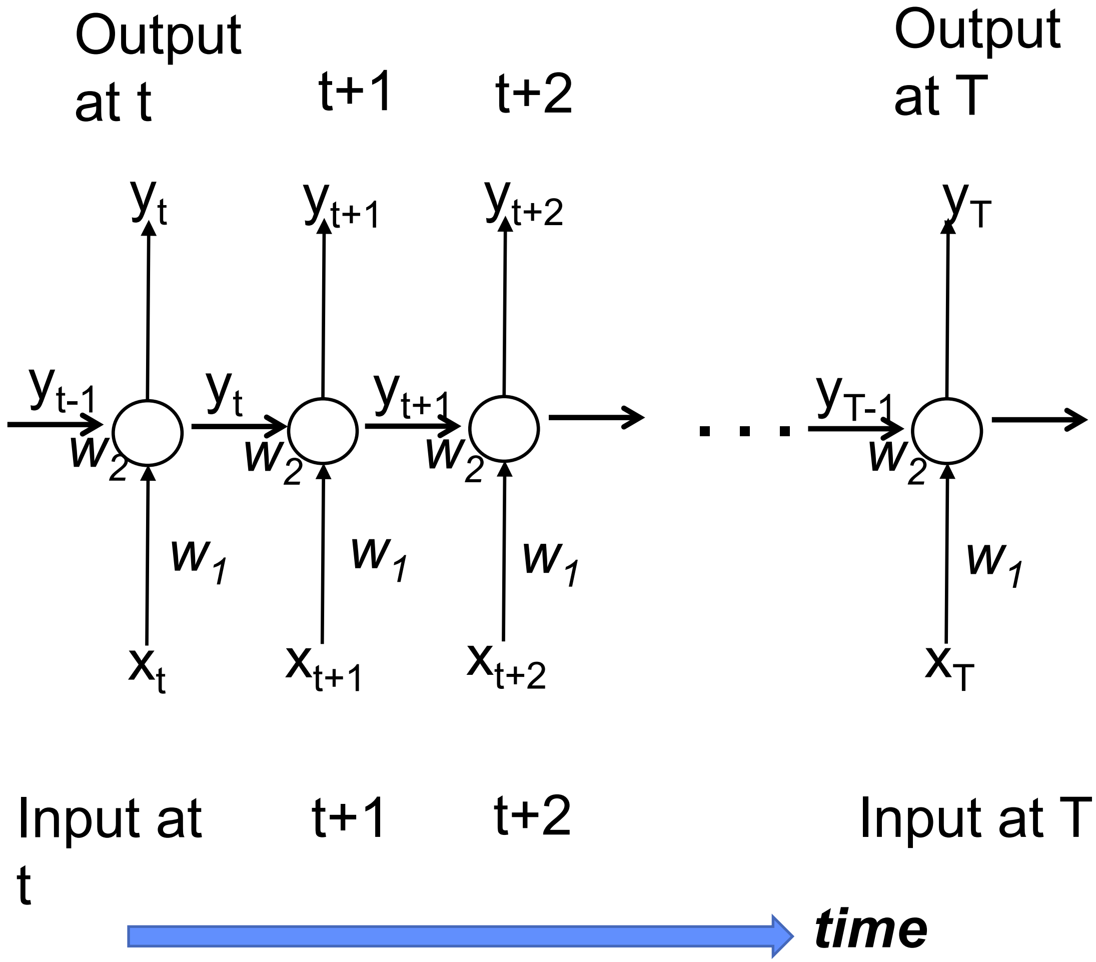
Let’s ‘unroll’ it in time by making copies



‘Unrolling’ the network

This is a dynamical system – how does it learn?

Let’s ‘unroll’ it in time by making copies

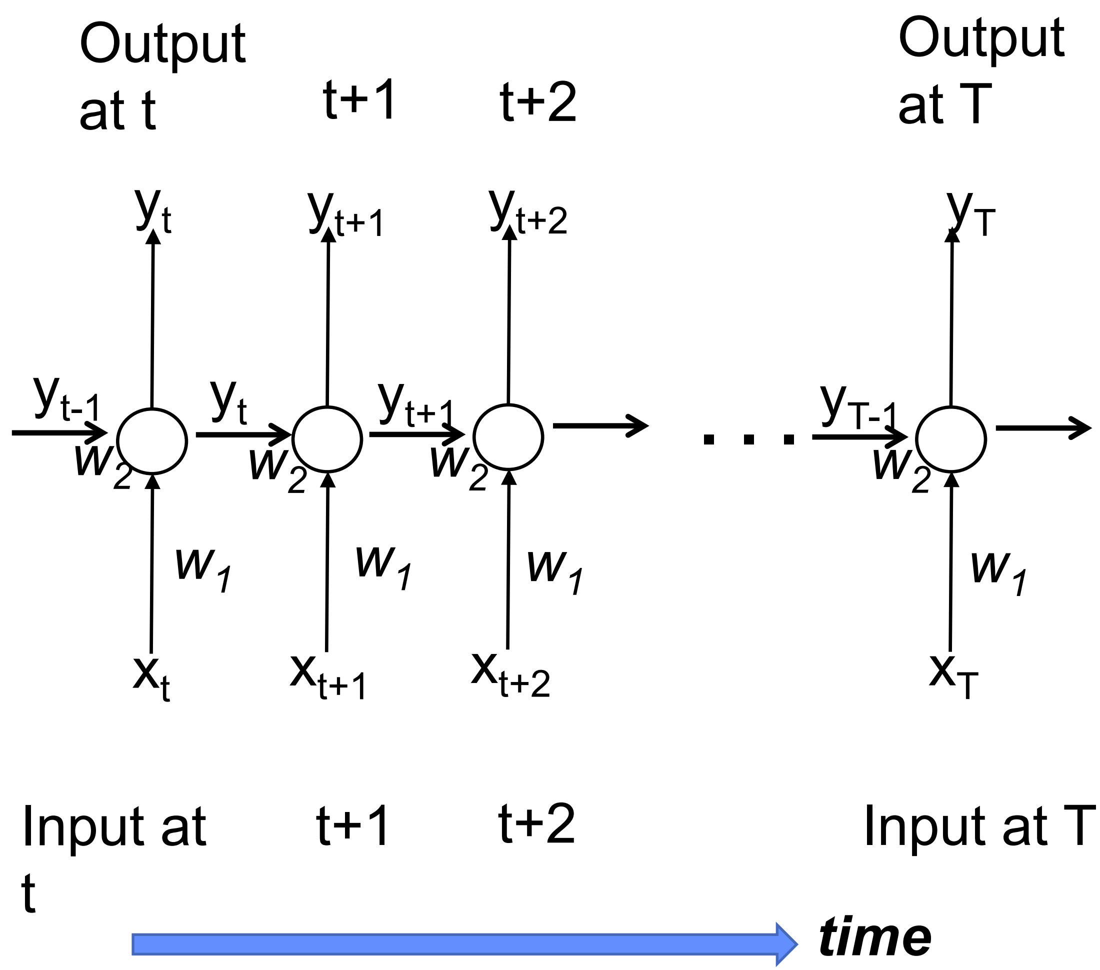


‘Unrolling’ the network

This is a dynamical system – how does it learn?

Let’s ‘unroll’ it in time by making copies

Can we see dependencies in time that backprop must learn?

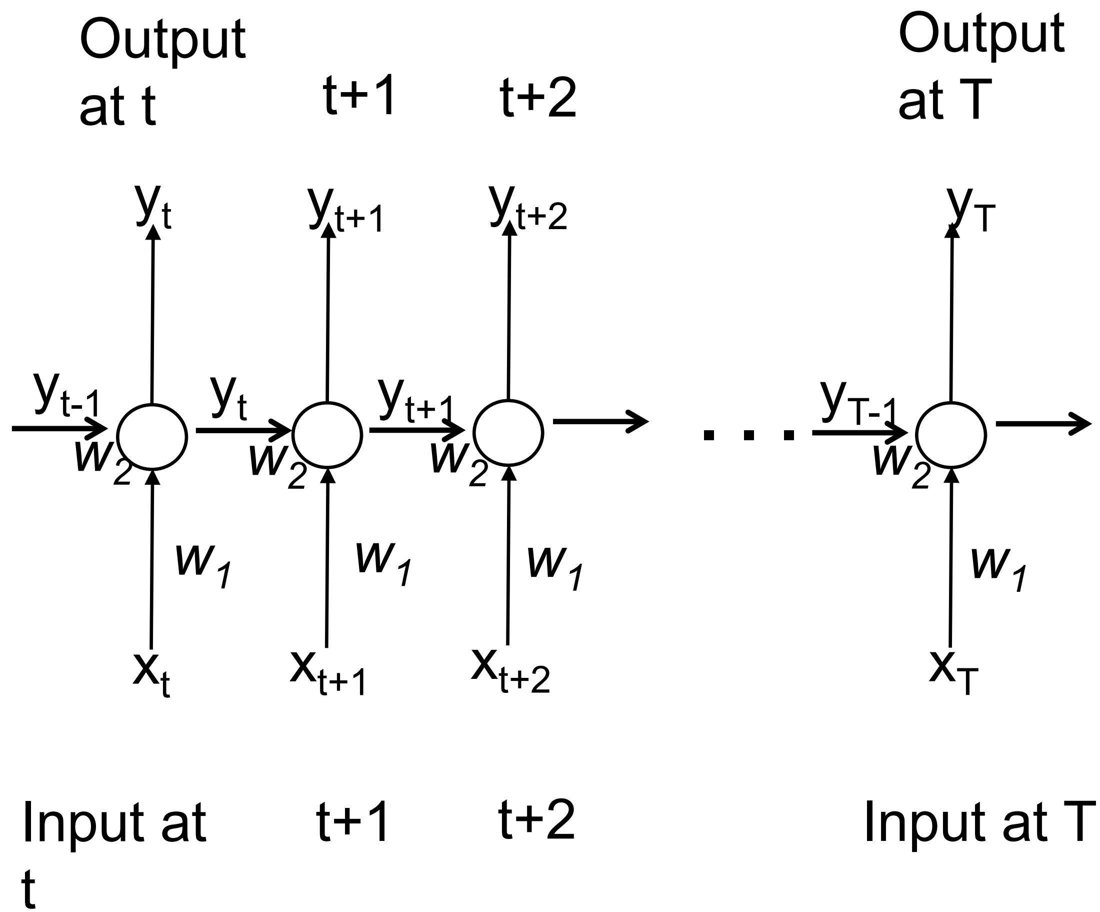


‘Unrolling’ the network

This is a dynamical system – how does it learn?

Let’s ‘unroll’ it in time by making copies

Can we see dependencies in time that backprop must learn?

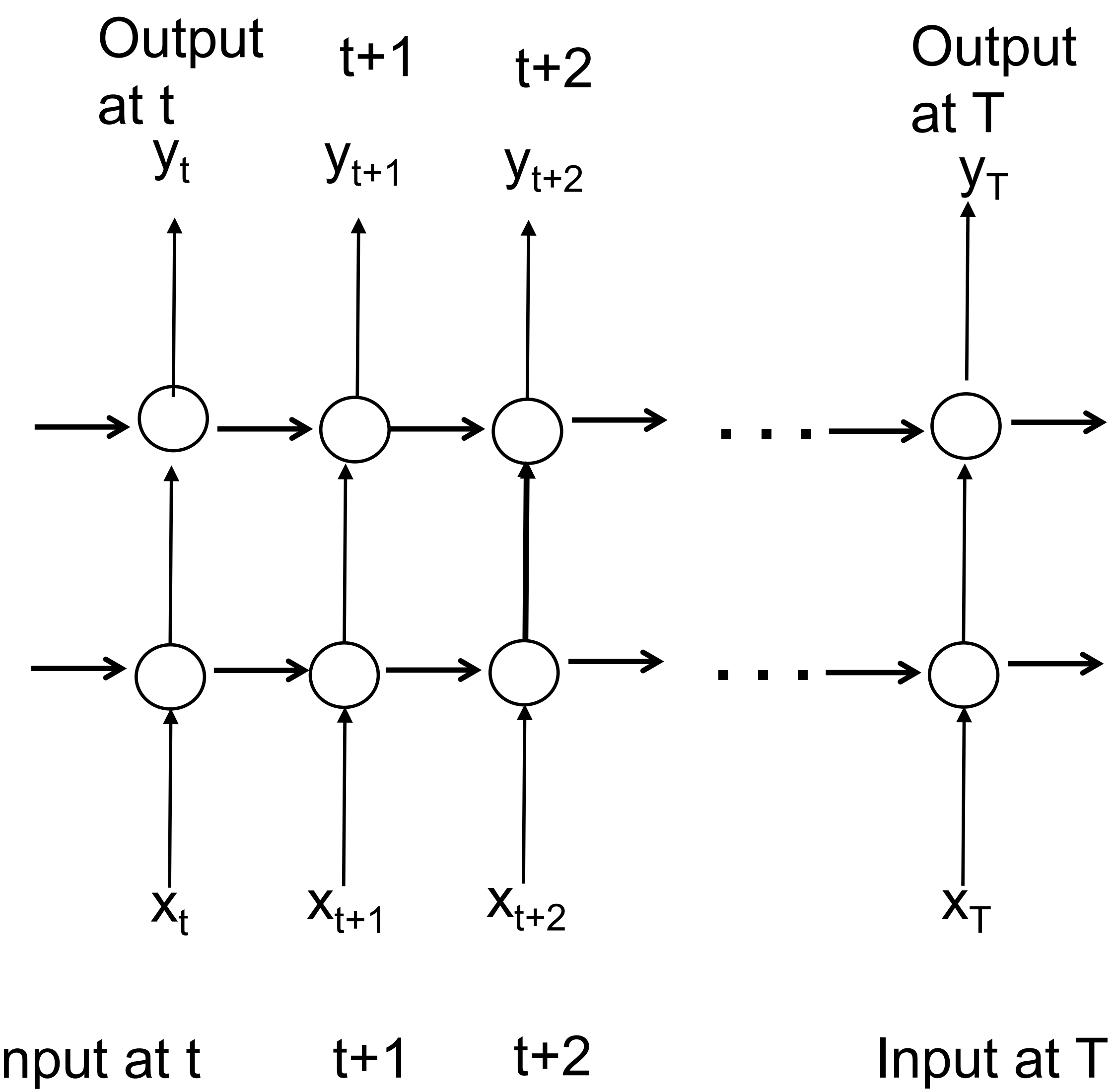


Backpropagation through time

Deep RNN

Layers can be added

Layers are ‘unrolled’ together



Can we make the feedback more like a memory unit?

What would a 'memory' unit look like?

Consider:

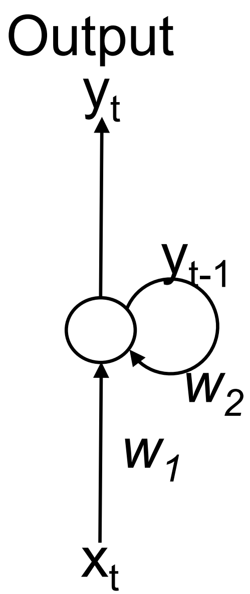
Given an initial value of y_{t-1},

Given:

$w1=0$,
$w2=1$,
$bias=0$,

If you use a linear activation function:

What is y_t?



What would a ‘memory’ unit look like?

Consider:

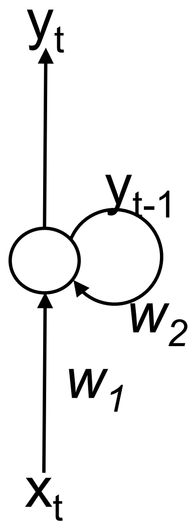
Given an initial value of y_{t-1},

Given:

$w1=0$,
$w2=1$,
$bias=0$,

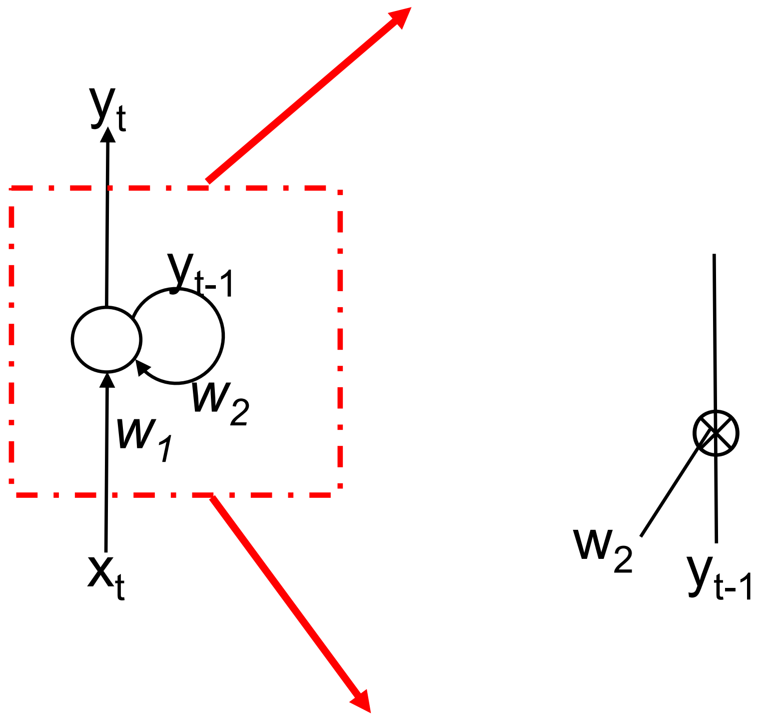
If you use a linear activation function:

What is y_t? ***Same as y_{t-1}***



Replace node with memory-like computation

Use w_2 as an 'update' gate on y



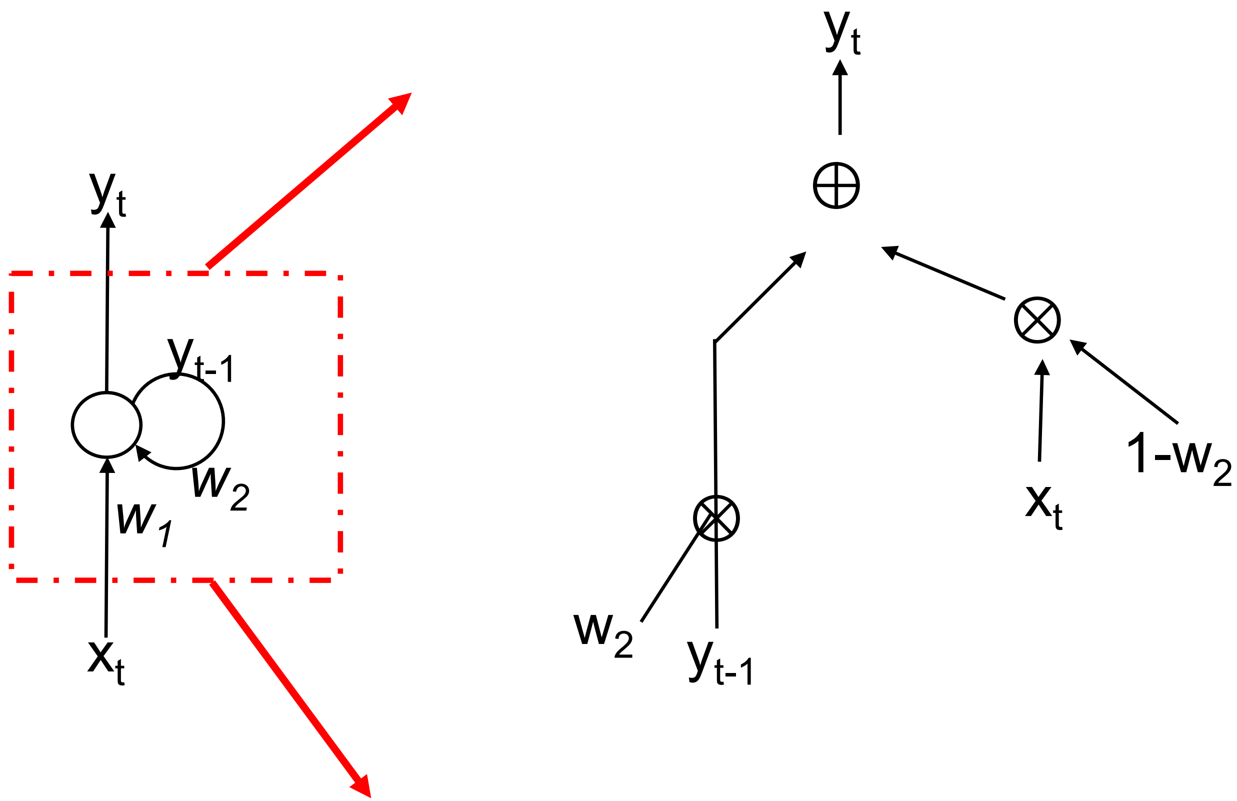
$\oplus = element\ wise\ add$

$\otimes = element\ wise\ multiply$

Replace node with memory-like computation

Use w_2 as an 'update' gate on y

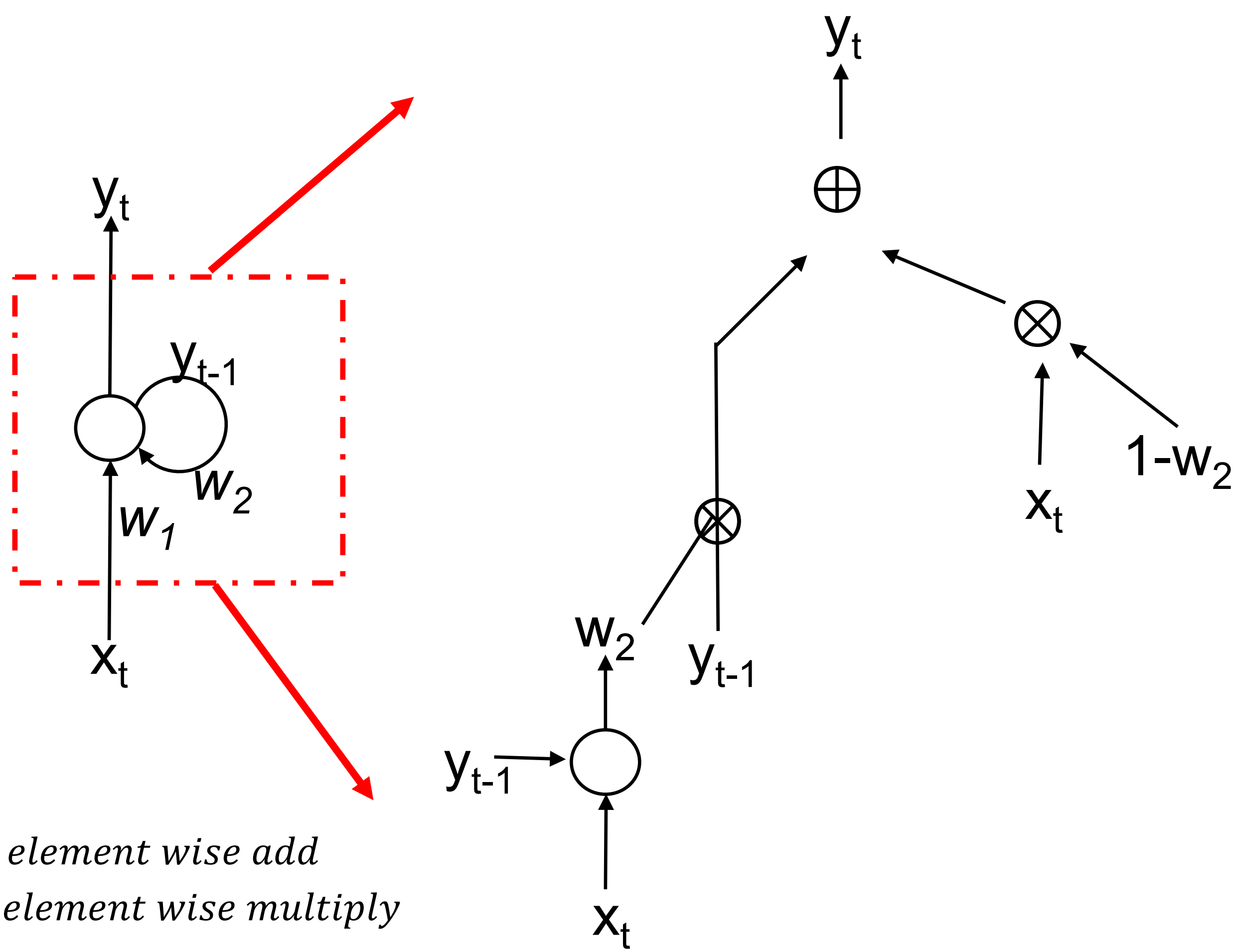
Let y_t be weighted sum of y_{t-1} and x_t



$\oplus$ = *element wise add*

$\otimes$ = *element wise multiply*

Replace node with memory-like computation



$\oplus$ = *element wise add*

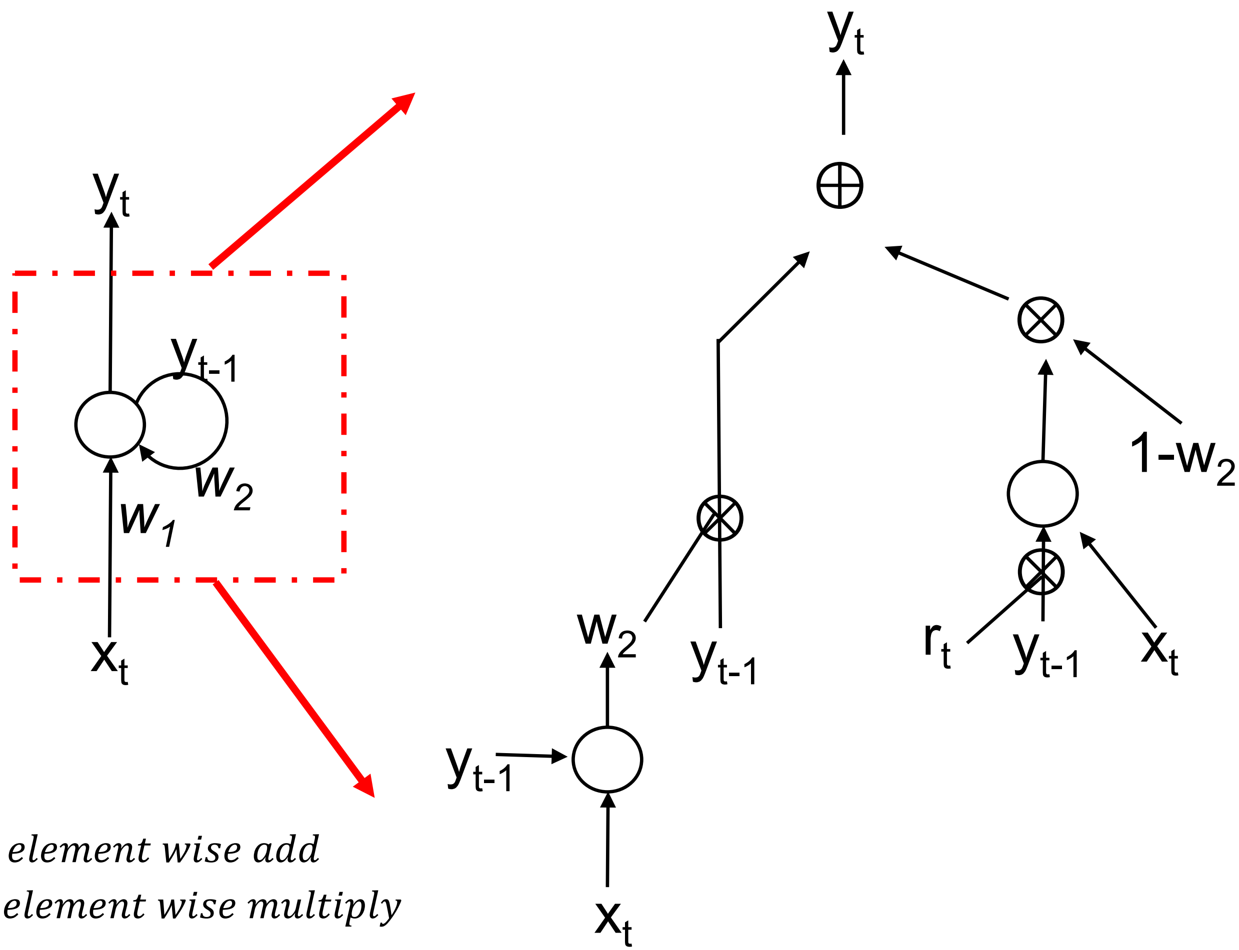
$\otimes$ = *element wise multiply*

Use w_2 as an 'update' gate on y

Let y_t be weighted sum of y_{t-1} and x_t

Use node to get w_2

Replace node with memory-like computation



$\oplus$ = *element wise add*

$\otimes$ = *element wise multiply*

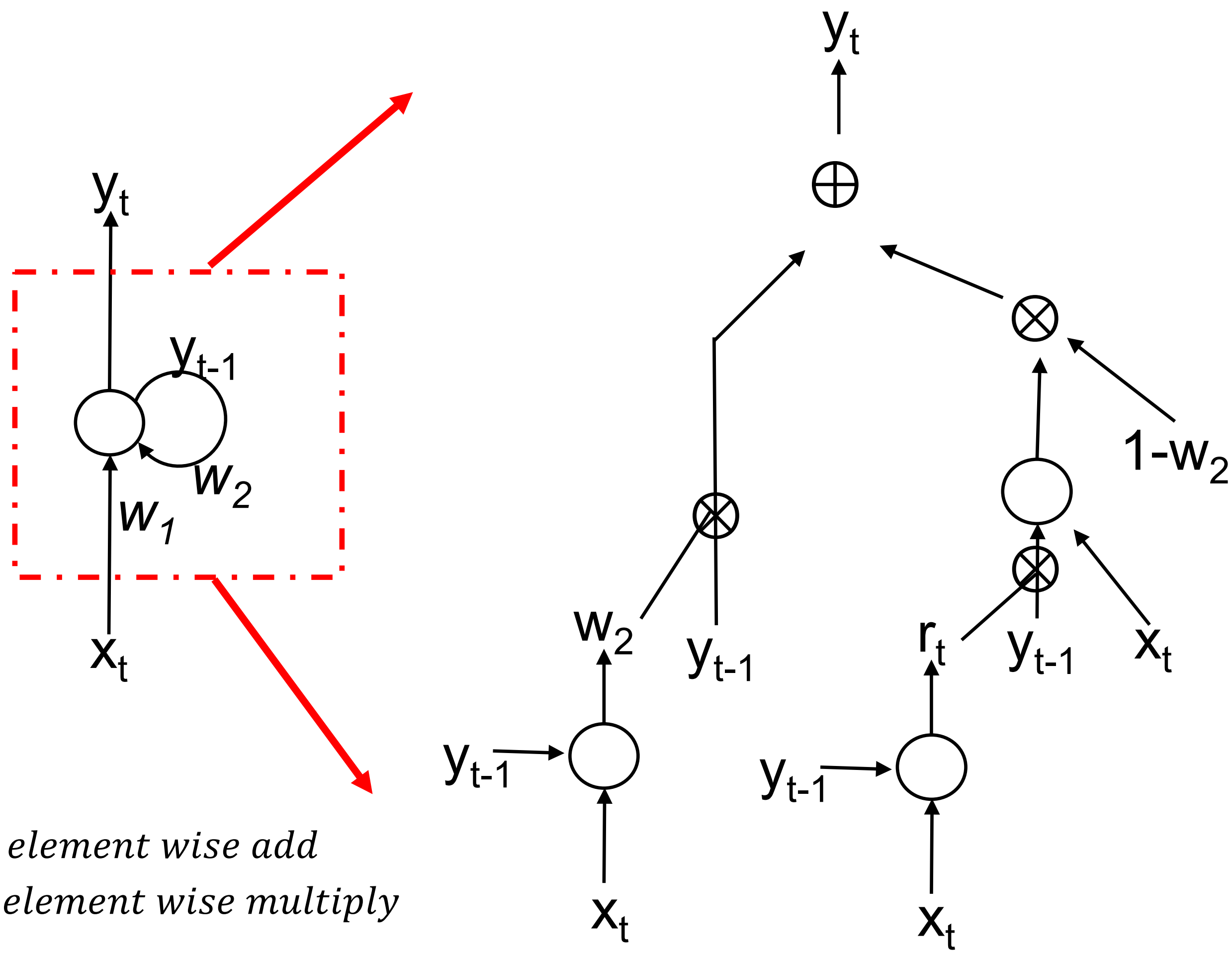
Use w_2 as an 'update' gate on y

Let y_t be weighted sum of y_{t-1} and x_t

Use node to get w_2

Use another node to combine y and x – with a 'relevance' gate on y_{t-1}

Replace node with memory-like computation



$\oplus = element\ wise\ add$

$\otimes = element\ wise\ multiply$

Use w_2 as an 'update' gate on y

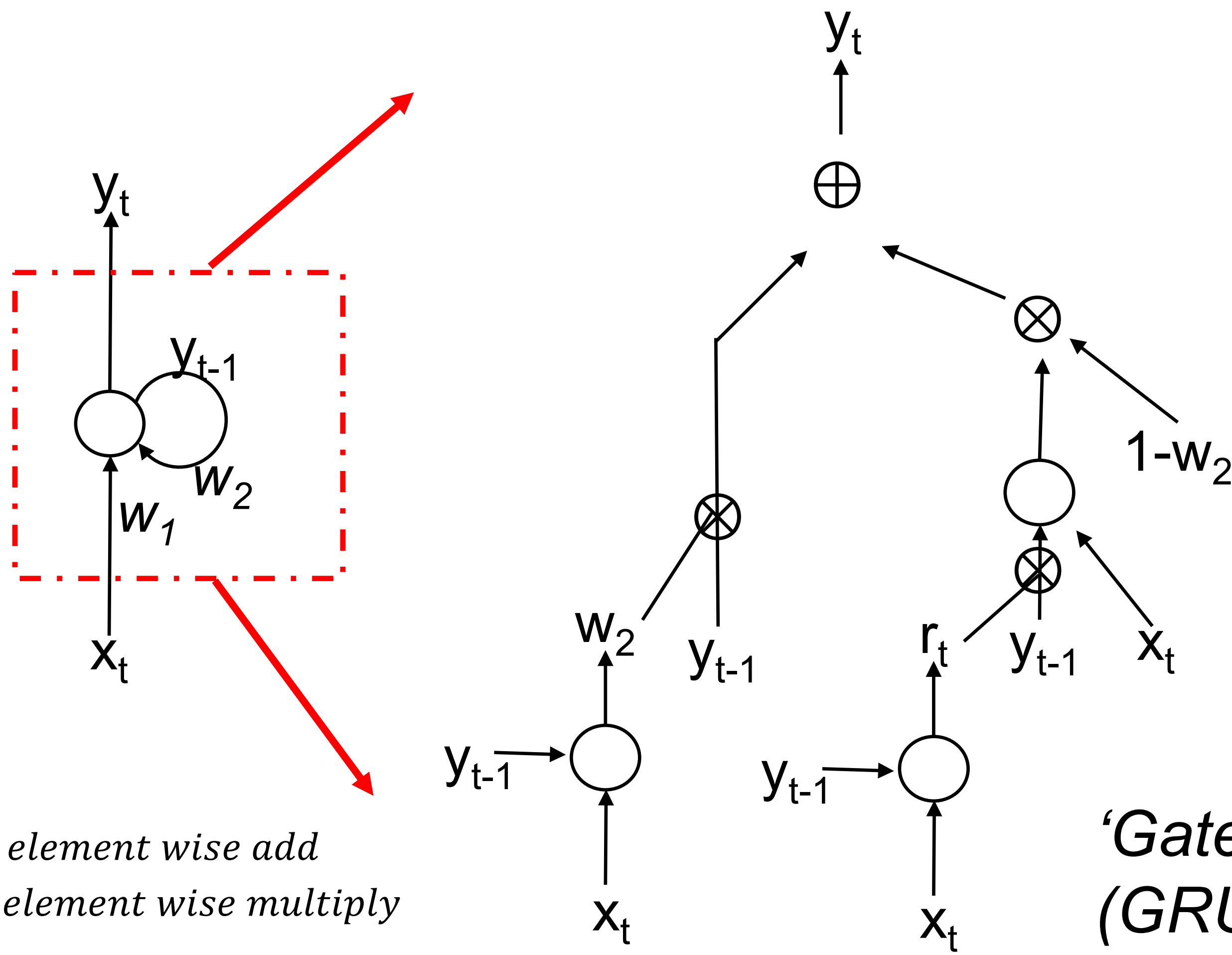
Let y_t be weighted sum of y_{t-1} and x_t

Use node to get w_2

Use another node to combine y and x – with a 'relevance' gate on y_{t-1}

Use node to get r_t

Replace node with memory-like computation



$\oplus = element\ wise\ add$

$\otimes = element\ wise\ multiply$

Use w_2 as an 'update' gate on y

Let y_t be weighted sum of y_{t-1} and x_t

Use node to get w_2

Use another node to combine y and x – with a 'relevance' gate on y_{t-1}

Use node to get r_t

'Gated Recurrent Unit' (GRU)

Cho, Bengio 2015

GRUs often drawn as circuit

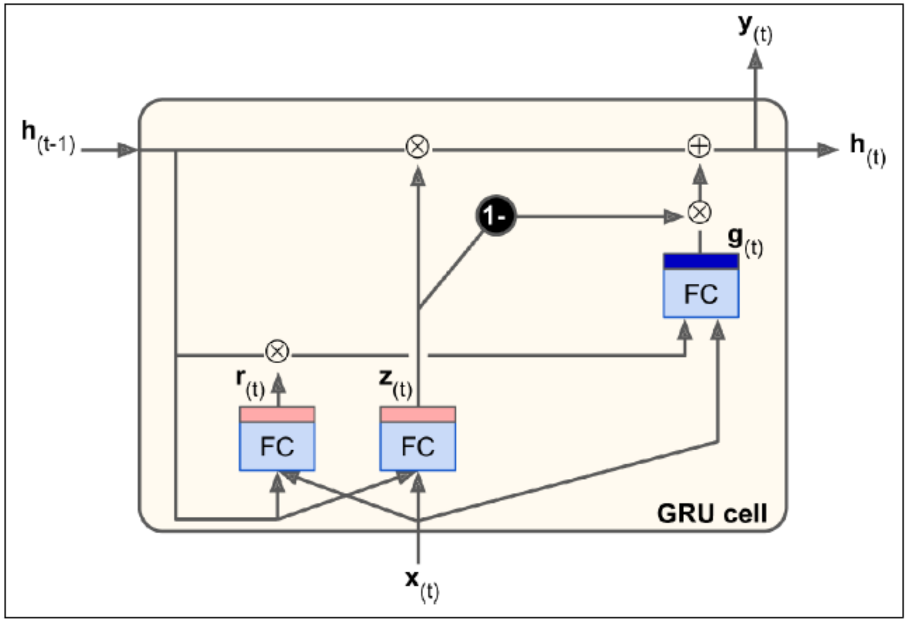


Figure 15-10. GRU cell

Image from Geron, 2019

Long Short Term Memory (LSTM) has more parameters and gates

(Schmidhuber et al, 1997)

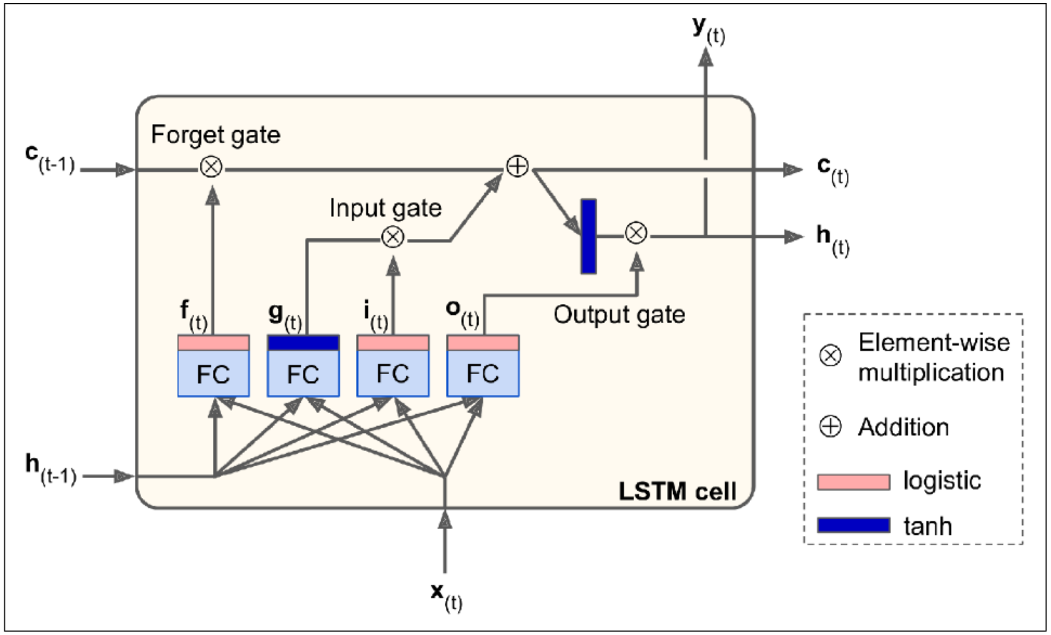
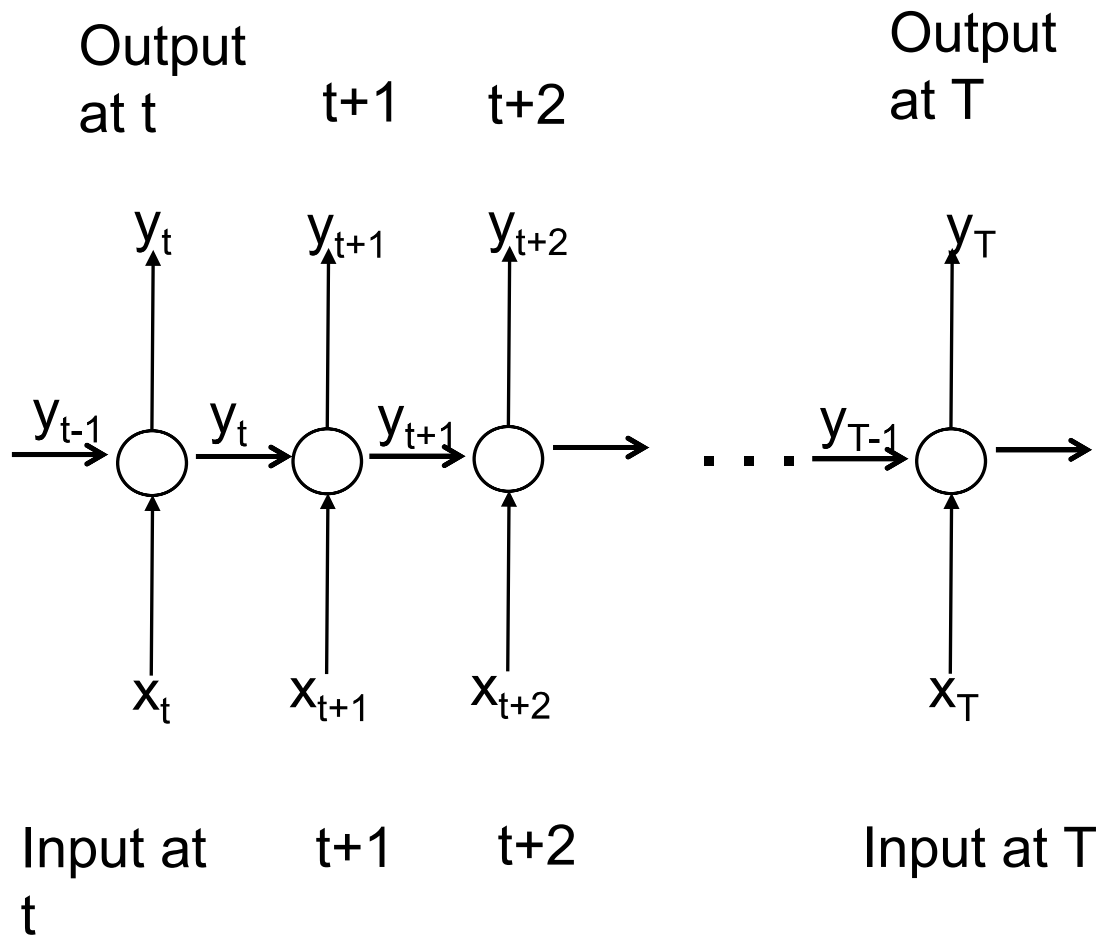


Image from Geron, 2019

Variations in RNN architectures

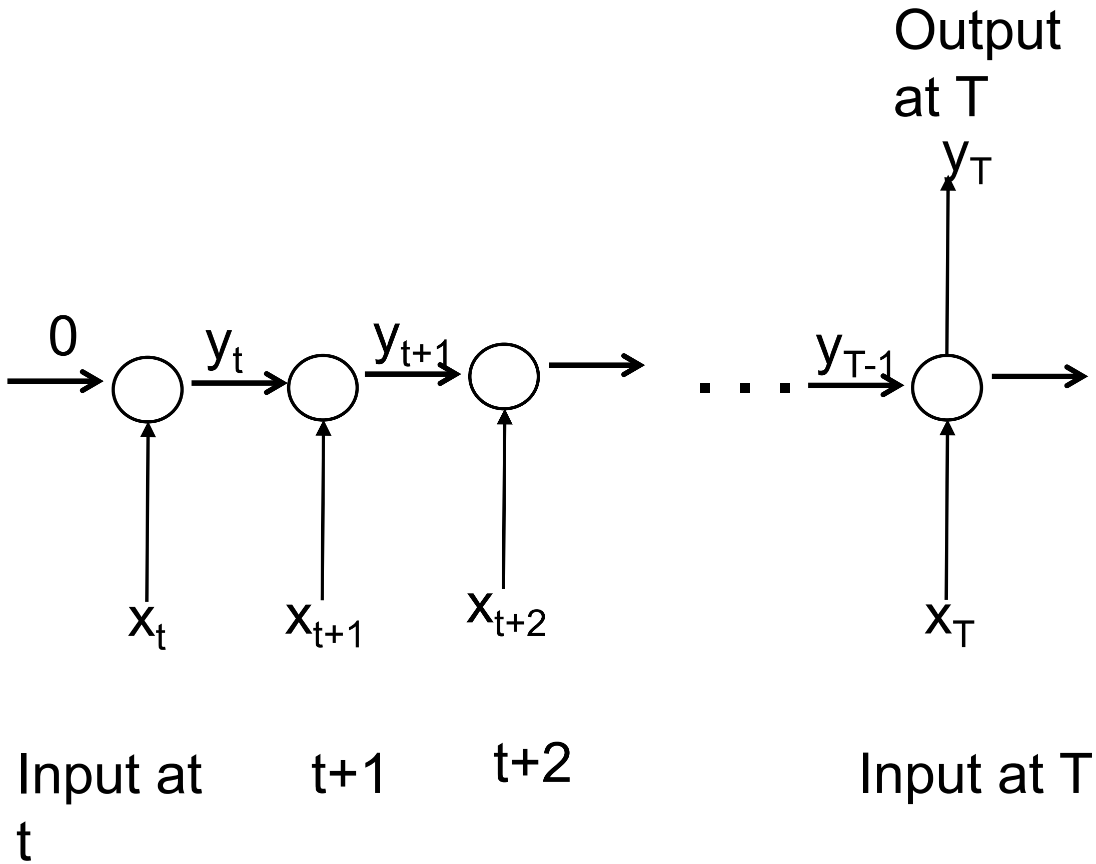
Input Sequence to Output Sequence at each step

Often used to predict next input (self-supervised)

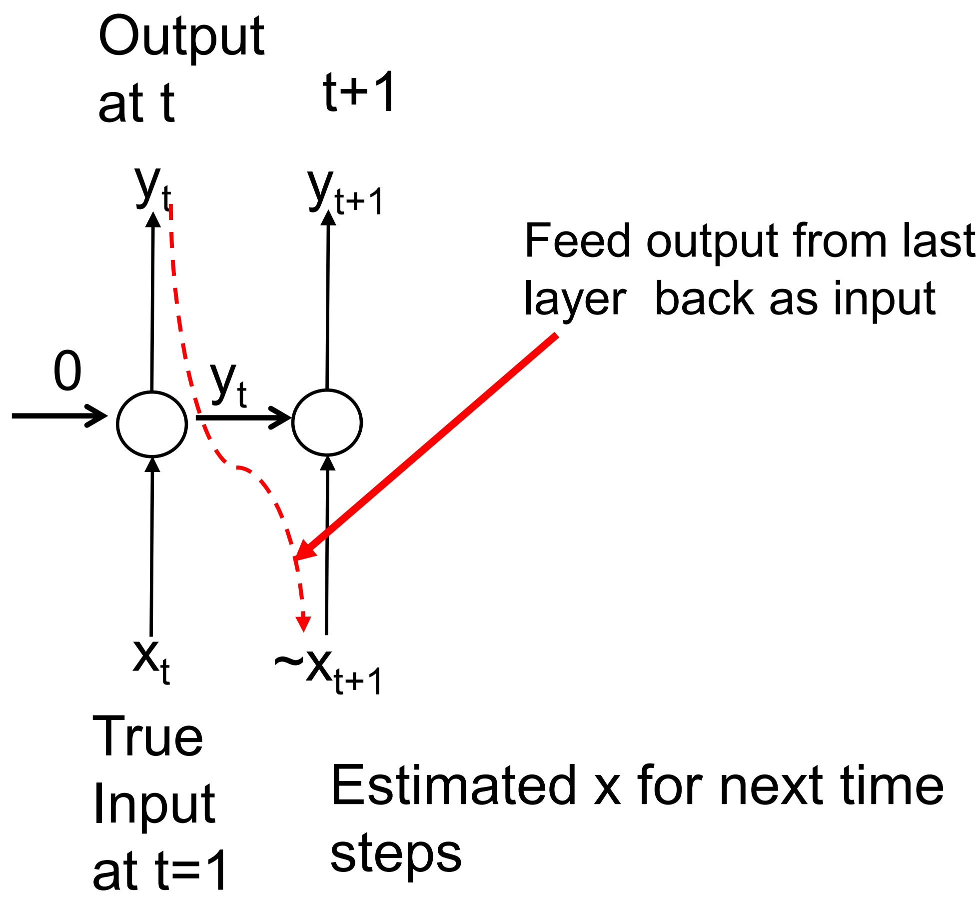


Sequence to 1 output at end

Often used for time series classification

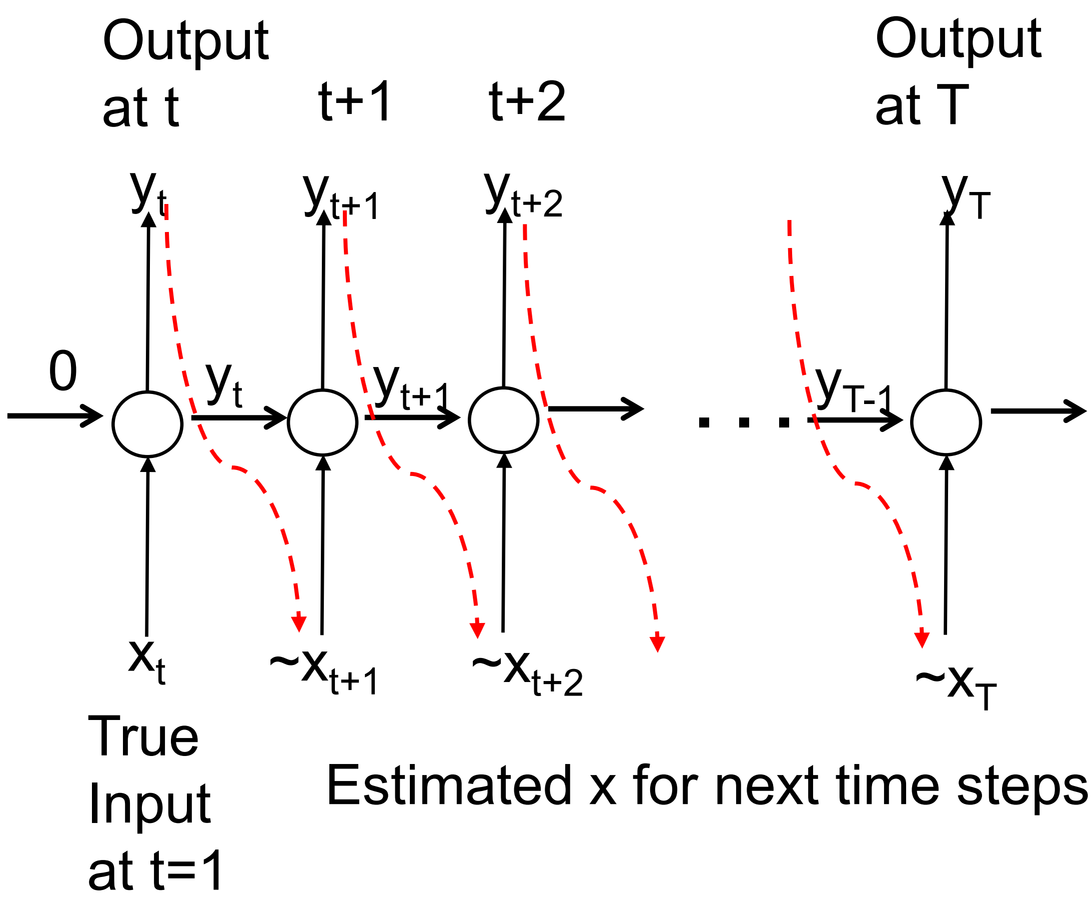


Sequence Generation from 1 input



Sequence Generation from 1 input

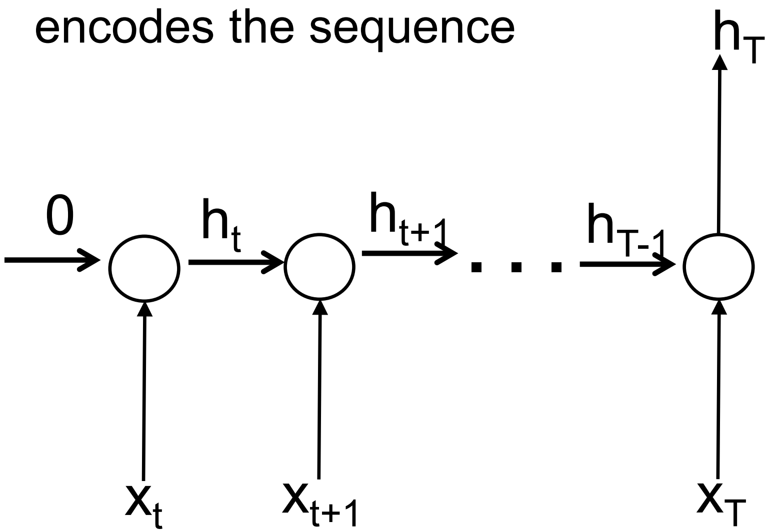
Used for long term predictions or generating styles



Encoding – Decoding sequence

Run a network forward in time to get a final state that encodes the sequence

State at T



t

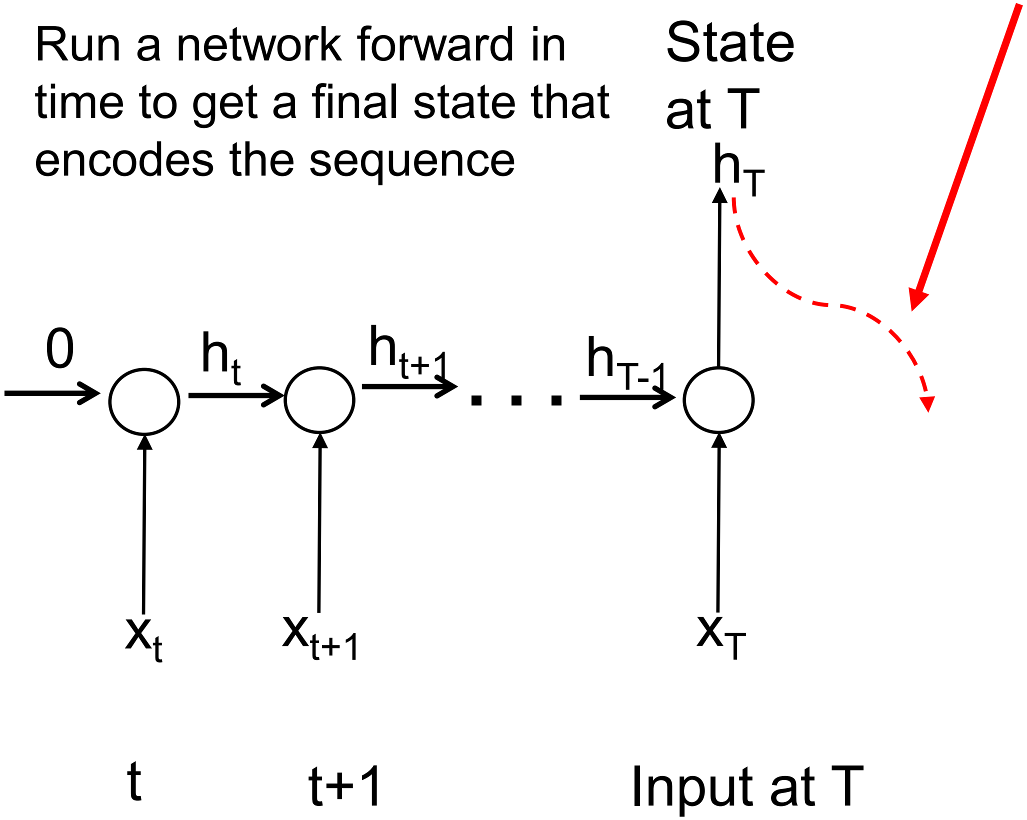
t+1

Input at T

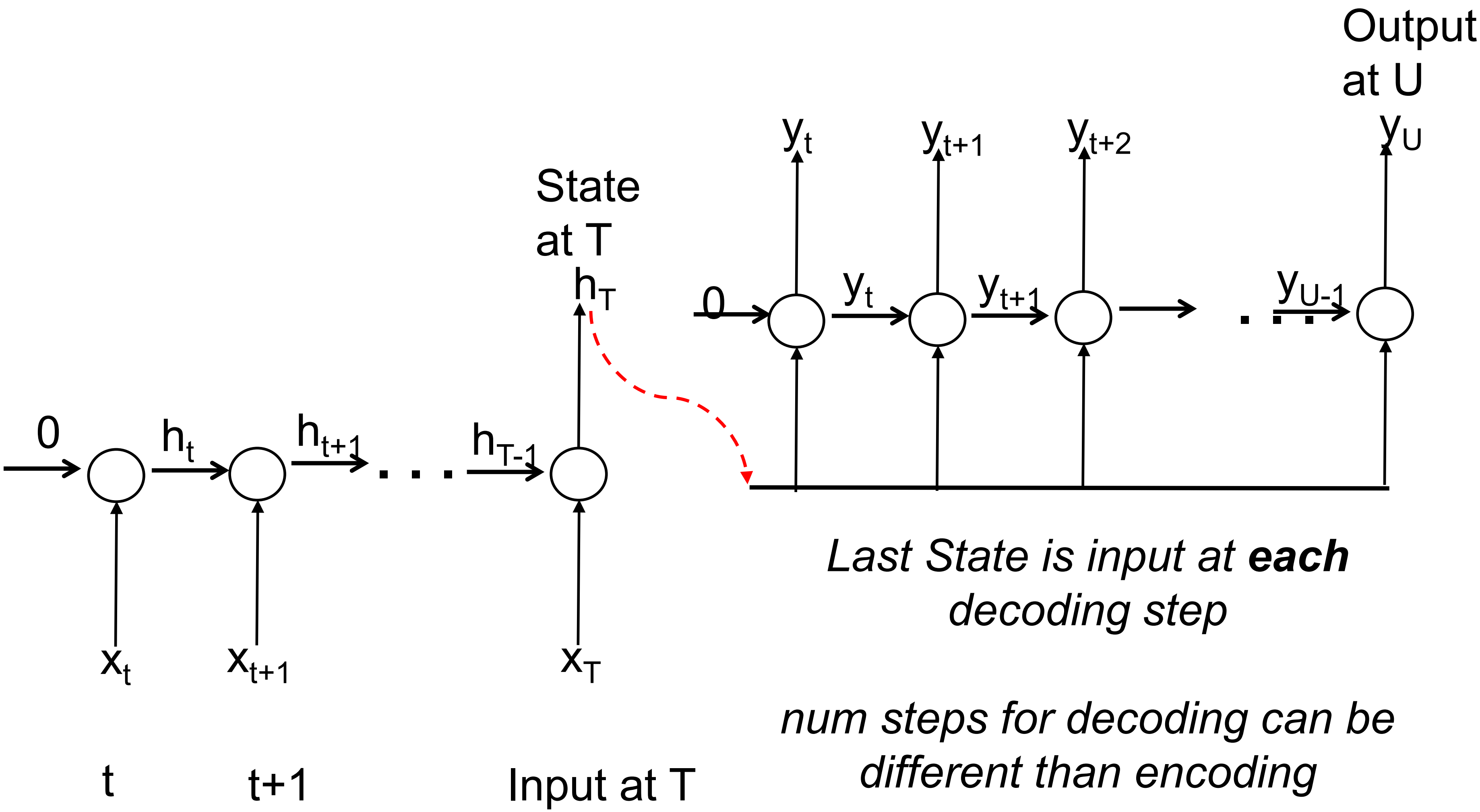
Encoding – Decoding sequence

Feed output from last layer as input to decoder

Run a network forward in time to get a final state that encodes the sequence



Encoding – Decoding sequence



*Last State is input at **each** decoding step*

num steps for decoding can be different than encoding

Example: Machine Translation

Summary

- RNNs can learn dependencies in time
- Long term dependencies can be hard because of vanishing gradients
- ‘Simple’ RNNs are unrolled in time
- ‘Memory Unit’ RNNs are built with “gate” multiplicative connections
- RNNs can be used for different input-output sequence mappings
- RNNs can be most useful for varying length sequences
- FeedForward NNs with lagged inputs compete with RNNs

RNN exercise – the task

A toy problem: add two numbers in time

Inputs:
1 real
1 binary
indicator

1	2	3	4	5	6	7	….		T
0.11	0.53	0.34	0.725	0.824	0.45	0.61			0.06
0	0	1	0	0	1	0			0

At the end,
output the sum

Output = 0.80

RNN exercise – the input

Generate Data as 3D array: N x T x 2

N samples

Inputs:
1 real
1 binary

1	2	3	4	5	6	7	….		T
0.11	0.53	0.34	0.725	0.824	0.45	0.61			0.06
0	0	1	0	0	1	0			0

RNNs with Keras

The Script parameters

```
# =======================options to change =================
samplesize_2use=500;  #pick a large enough size
nsteps_2use   =20;    #this is T stpes

num1        =3          #  choose number of 1s for binary variable
t_fixed     =True      #  are the times when it is 1 fixed
t_fix_inds =[5,10,15]  #  the times to set it to 1

#Now set up parameters to run model
numunits        =64
act2use         ='relu'  #or 'relu' or sigmoid or tanh
```

Inputs:
1 real
1 binary
indicator

1	….	5	…	…	10	…	15	…	T
0.11	…	0.34	…	…	0.45	…	0.61		0.06
0		1	…	…	1	…	1		0

Sum=0.34+0.45+0.61
= 1.41

RNNs with Keras

It needs to know this to set up the network

```
keras.layers.SimpleRNN(your-umber-of-units,
                       return_sequences= True of False,
                       input_shape      =  (None, your-number-of-variables))
```

RNNs with Keras

It needs to know this to set up the network

```
keras.layers.SimpleRNN(your-umber-of-units,
                       return_sequences= True of False,
                       input_shape      = (None, your-number-of-variables))
```

If TRUE then output at each time step
If FALSE then output only at T

Same options for:

keras.layers.GRU(…..
keras.layers.LSTM(…….

RNNs with Keras

```
[12]: #a Simple RNN setup
      #set return_sequences=True for all recurrent layers
      #except the last one, if you only care about the last output
      nvar          = P   #set number of variables to P

      mysrn_model = keras.models.Sequential([
          keras.layers.SimpleRNN(numunits, return_sequences=True, input_shape=[None,nvar]),
          keras.layers.SimpleRNN(numunits),
          keras.layers.Dense(1,activation='linear')])

      # keras.layers.TimeDistributed(keras.layers.Dense(1,activation='sigmoid'))])  for pred at each step
```

```
[13]: mysrn_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn (SimpleRNN)       (None, None, 128)         16768
_________________________________________________________________
simple_rnn_1 (SimpleRNN)     (None, 128)               32896
_________________________________________________________________
dense (Dense)                (None, 1)                 129
=================================================================
Total params: 49,793
Trainable params: 49,793
Non-trainable params: 0
_________________________________________________________________
```

First RNN layer outputs 3D tensor

RNNs with Keras

```
[12]: #a Simple RNN setup
      #set return_sequences=True for all recurrent layers
      #except the last one, if you only care about the last output
      nvar           = P   #set number of variables to P

      mysrn_model = keras.models.Sequential([
          keras.layers.SimpleRNN(numunits, return_sequences=True, input_shape=[None,nvar]),
          keras.layers.SimpleRNN(numunits),
          keras.layers.Dense(1,activation='linear')])

      # keras.layers.TimeDistributed(keras.layers.Dense(1,activation='sigmoid'))])  for pred at each step
```

```
[13]: mysrn_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn (SimpleRNN)       (None, None, 128)         16768
_________________________________________________________________
simple_rnn_1 (SimpleRNN)     (None, 128)               32896
_________________________________________________________________
dense (Dense)                (None, 1)                 129
=================================================================
Total params: 49,793
Trainable params: 49,793
Non-trainable params: 0
_________________________________________________________________
```

First RNN layer outputs 3D tensor

Second RNN layer outputs 2D tensor

Last layer is class output

RNNs with Keras

Exercise
1 Run the script

2. Run with different number-of-1's in binary variable or different ttime steps

3. Try different number of hidden units and/or layers

4. Compare LSTM and/or GRU

keras.layers.GRU(…..
keras.layers.LSTM(…….

```
]: numepochs=100

my_model=mysrn_model   #<<<<<<<<<----- you can change the model here and run these cells
#my_model=mygru_model

my_model.compile(optimizer='adam',          #or just use 'adam' to get defaults
```