Recurrent Neural Networks

DL-U08

Outline

- Sequence Labelling
- Recurrent Neural Network
- LSTM
- Prepare data for recurrent net

Sequence Labelling

Motivation

- Feed-forward network assume that all inputs and outputs are independent of each other
- While application like: language/speech processing
  - Predicting the next word in a sentence depends on the entire sequence of words before the current word
  - Example: “The man who wore a wig on his head went inside”
  - Who went inside? Man or wig?

Sequence Labelling

- Sequence Labelling encompasses all tasks where sequences of data are transcribed with sequences of discrete labels.
  - speech and handwriting recognition
  - protein secondary structure prediction
  - Part-of-speech tagging
  - …

Three class of sequence labelling

- Sequence classification
- Segment classification
- Temporal classification

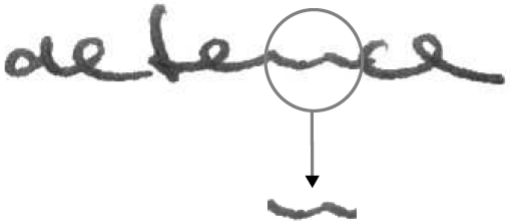
Sequence Labelling – Sequence

- Sequence classification
  - where each input sequence is assigned a single class, i.e. label sequence is constrained to be length one
  - It is a special case of segment classification

- Examples
  - a single spoken work
  - the recognition of an individual handwritten letter

Sequence Labelling – Segment

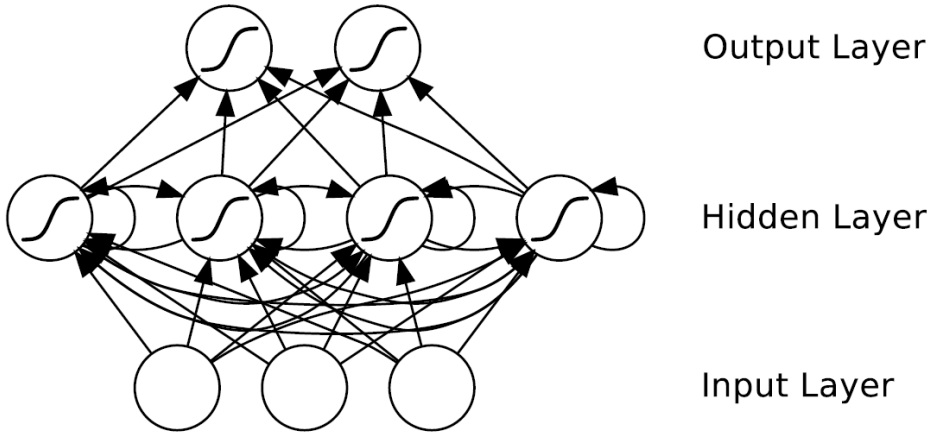
- Segment classification
  - Segment classification refers to those tasks where the target sequences consist of multiple labels, but the locations of the labels are known in advance.
  - It is a special case of temporal classification



- Examples
  - the recognition of a handwritten word
  - Phoneme recognition
  - framewise phoneme classification

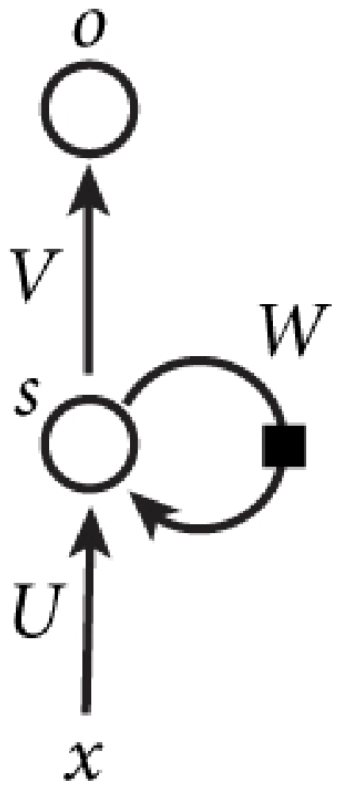
Sequence Labelling – Temporal

- Temporal classification
  - It is the most general case
  - nothing can be assumed about the label sequences except that their length is less than or equal to that of the input sequences.
- Examples
  - phoneme classification
  - the task is not to phonetically label individual frames, but instead to output the complete phonetic transcription of the sequence



Recurrent Network

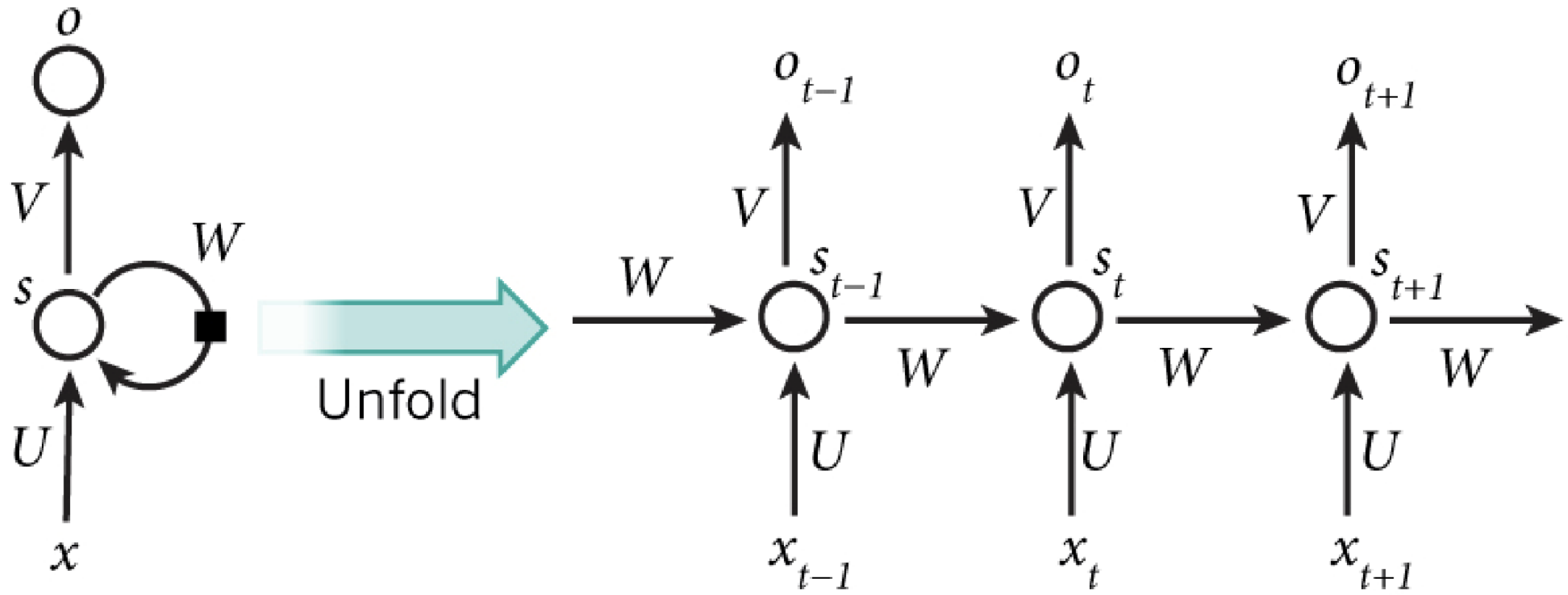
Recurrent network



- x: input
- s: Neural network
- o: output
- V, U, W: weights

- There is a loop → allow us introduce t represent time/step
- Signals/info to be passed from one step to the next

Unfolding RNN

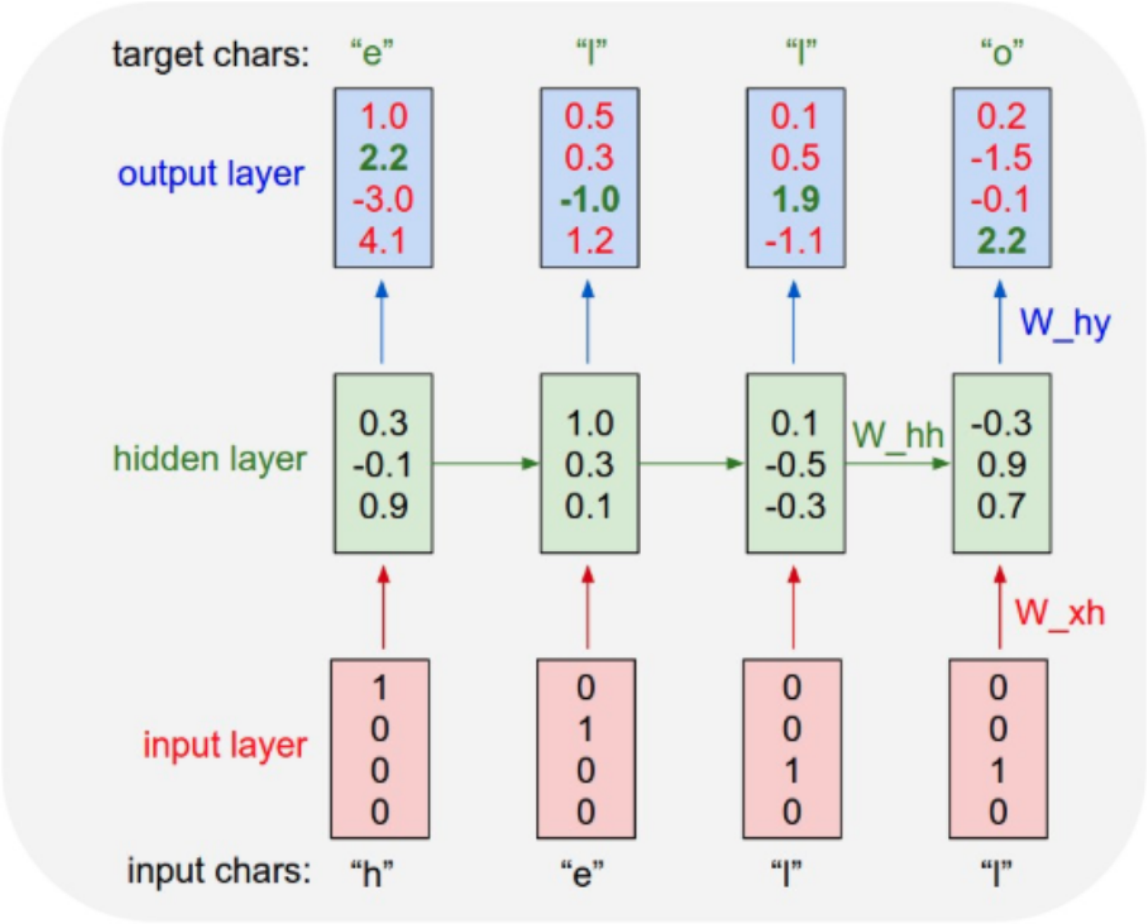


RNN shares same parameters (U,V,W) across all steps

- x_t - input at time t
  - E.g., One-hot vector corresponding to word of a sentence.
- s_t - hidden state at time t
  - "memory" of the network
  - Calculated based on the previous hidden state and the input at the current step
  - $s_t = f(Ux_t + Ws_{t-1})$, where f is the activation function
  - s_{-1} typically initialized to all zeroes, to calculate state s_0
- o_t - outputat time t
  - E.g., if we want to predict the next word in a sentence, it would be a vector of probabilities across our vocabulary.
  - o_t = softmax(Vs_t)

Use of RNN

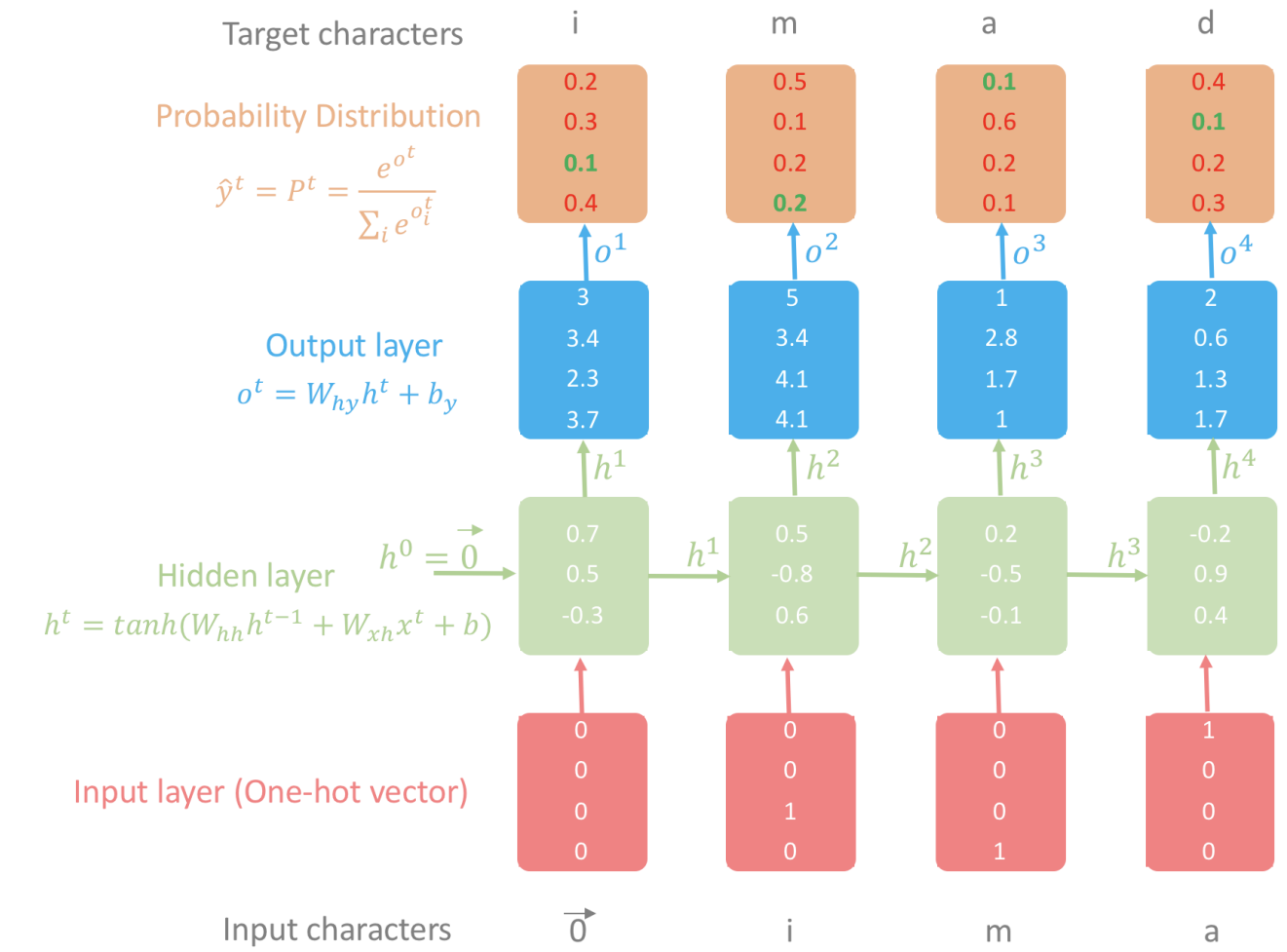
- Character level language model
  - The main task is to predict the next character given all previous characters in a sequence of data
  - Use RNN to model the probability distribution of the next character in the sequence given a sequence of previous characters.



https://towardsdatascience.com/character-level-language-model-1439f5dd87fe

An example

- Feed forward step in training
- Training purpose:
  - to make the green numbers in output layer as big as we can and
  - the red numbers as small as we can



https://towardsdatascience.com/character-level-language-model-1439f5dd87fe

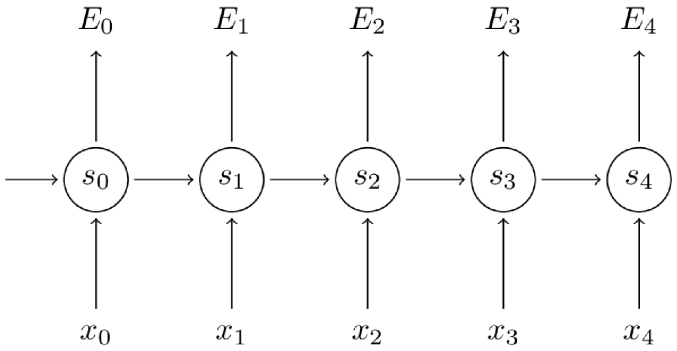
One-hot vector: [a d i m]

Loss function

- Feed forward computing

$$s_t = \tanh(Ux_t + Ws_{t-1})$$

$$\hat{y} = o_t = \text{softmax}(Vs_t)$$



- Cross-Entropy Loss

$$E = \sum_t E_t = -\sum_t y_t \log \hat{y}_t$$

Train RNN algorithm - BPTT

- Back Propagation Through Time: SGD

$$E = \sum_t E_t = -\sum_t y_t \log \hat{y}_t$$

- Gradient

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$

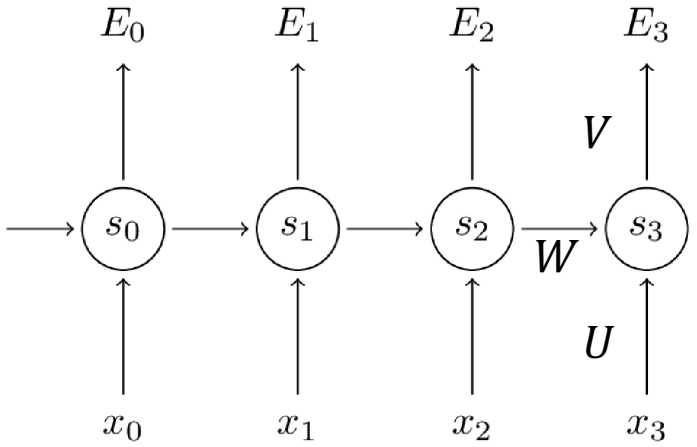
BPTT – gradient of E_3 wrt. V

- Chain Rule

$$E_3 = y_3 \log \hat{y}_3$$

$$\hat{y}_3 = softmax(z_3)$$

$$z_3 = Vs_3$$



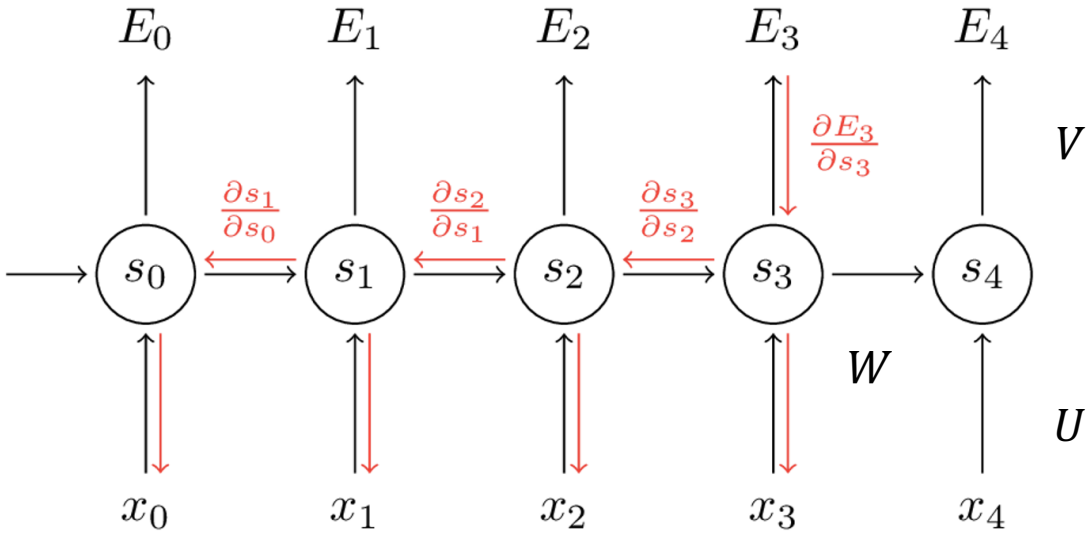
$$\frac{\partial E_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial z_3}\frac{\partial z_3}{\partial V} = (\hat{y}_3 - y_3) \otimes s_3$$

where $\otimes$ is outer product

BPTT – gradient of E_3 wrt. W

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial W} \quad \text{where}$$

$$s_3 = \tanh(Ux_3 + Ws_2)$$



Chain Rule is needed again !

Sequences can be long → **value of W will either decay to zero exponentially fast, or growth exponentially fast. (if w is not 1)** It causes gradient vanishing or gradient exploding

BPTT – gradient vanishing/exploding

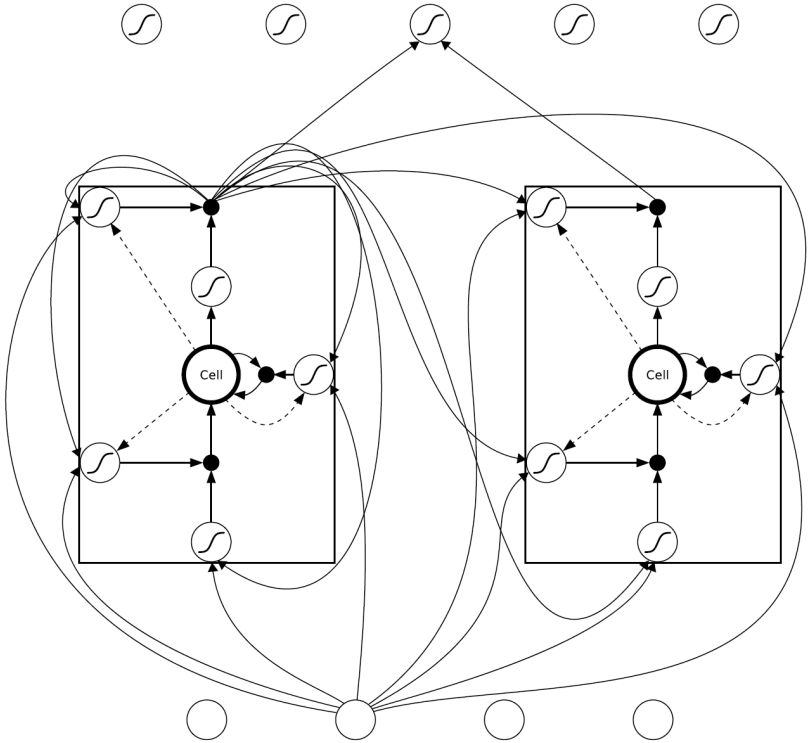
$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$

$$= \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \left(\prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial W}$$

$$\prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}} = W^{3-j} \prod_{m=1}^{3-j} sigmoid'(W s_{3-\mathrm{m}})$$

Decay/growth exponentially fast, if W is not all 1s

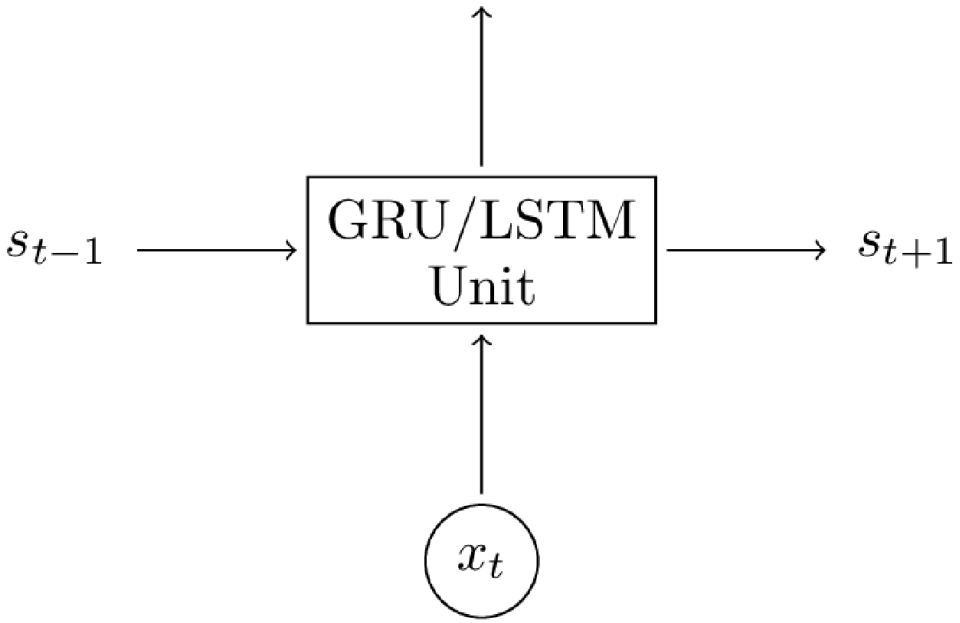
LSTM



How to overcome

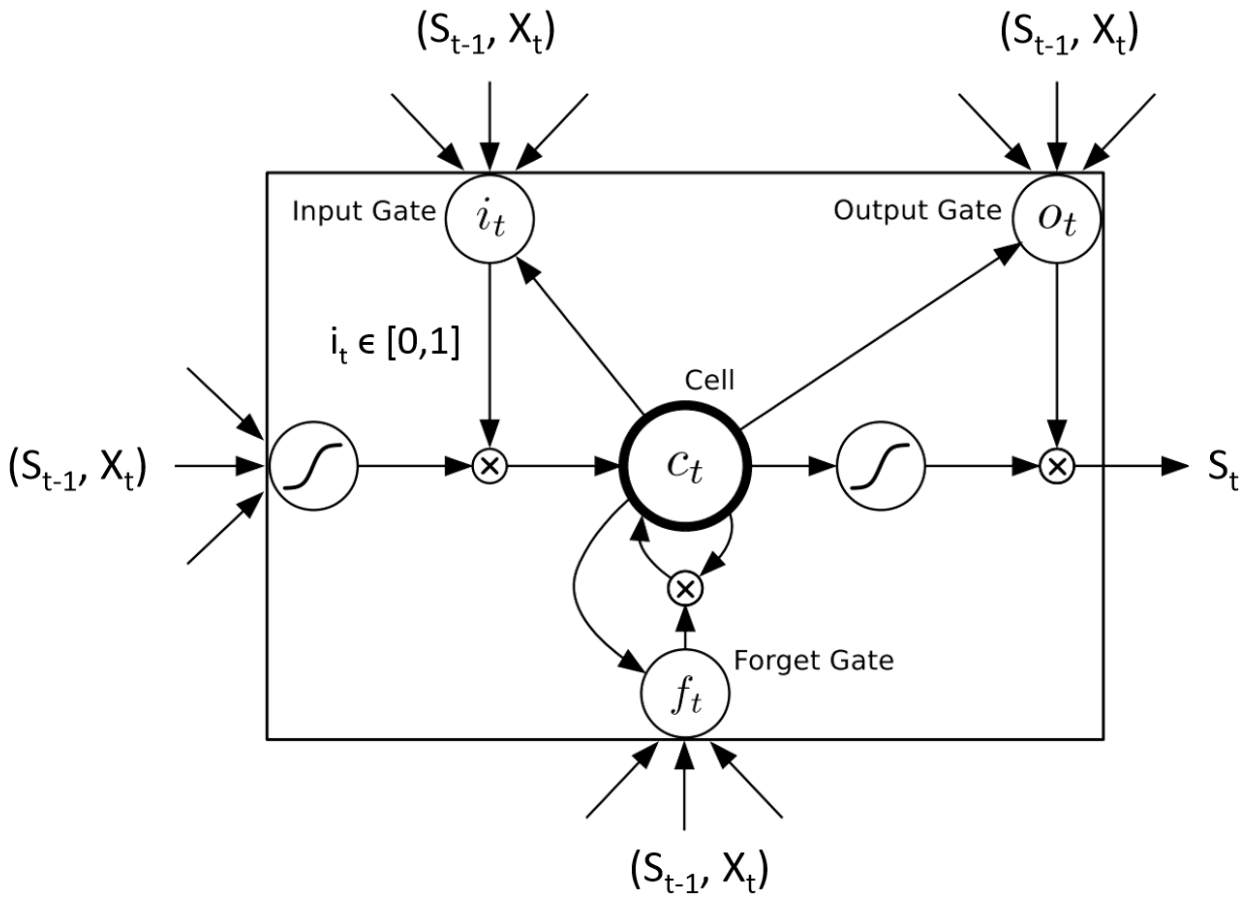
LSTM – Long Short Term Memory

GRU – Variant of LSTM

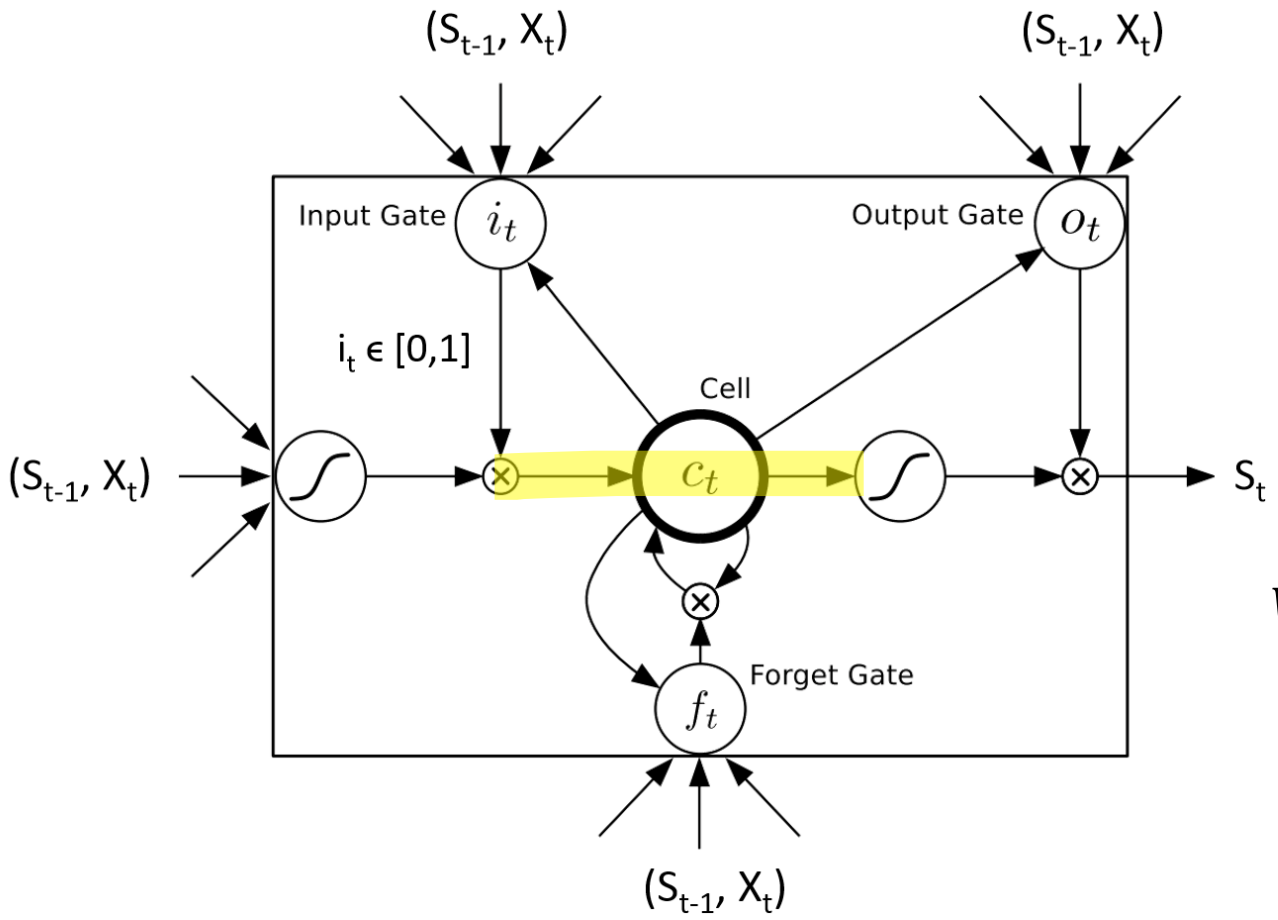


LSTM block with one cell

The three gates are nonlinear summation units that collect activations from inside and outside the block, and control the activation of the cell via multiplications (⊗)



LSTM Unit – Carry state

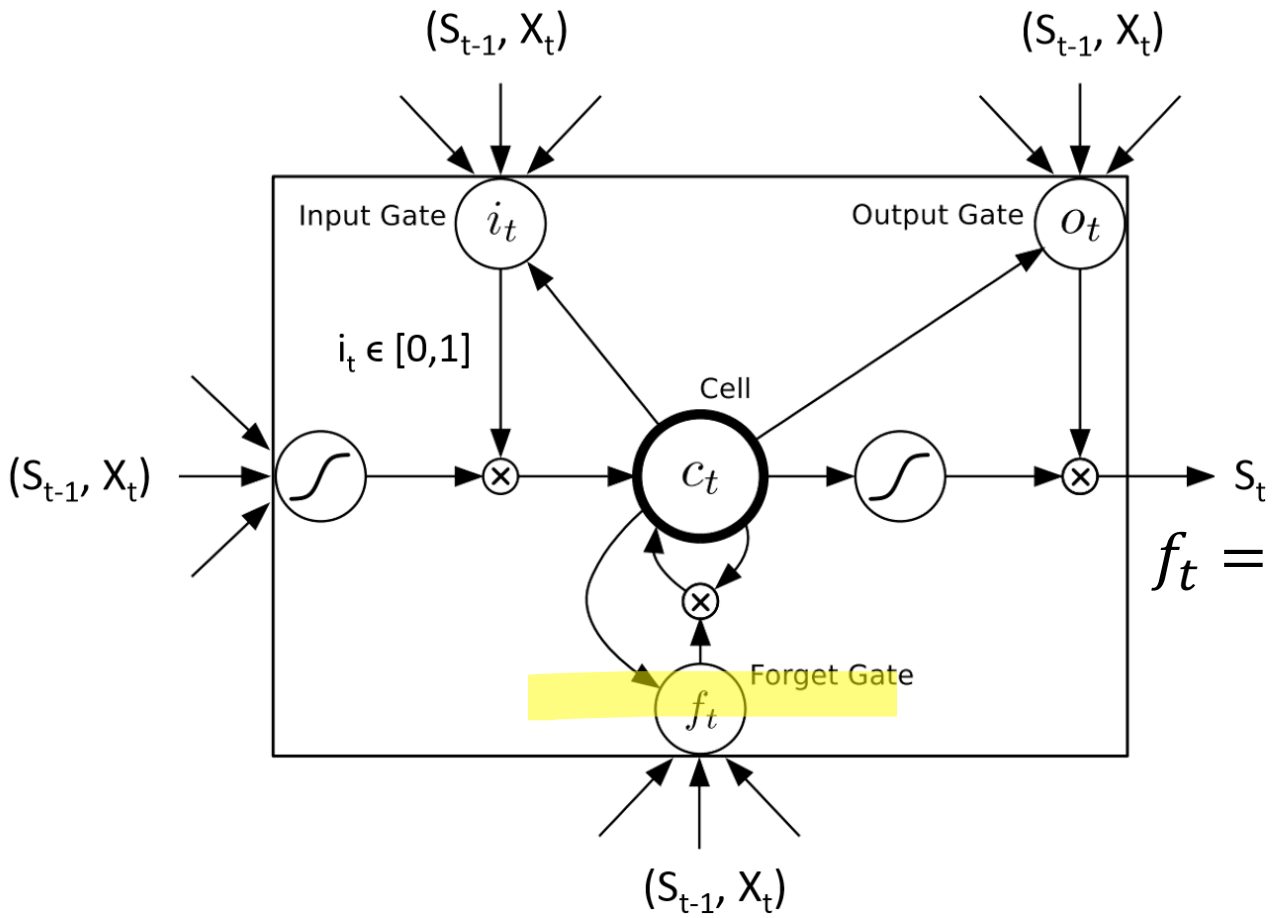


$$c_t = c_{t-1} \circ f_t + g \circ i_t$$

c_t : carry (selected long term memory)

$$where\ g = \tanh(x_t U^g + s_{t-1} W^g + b_c)$$

LSTM Unit – forget gate

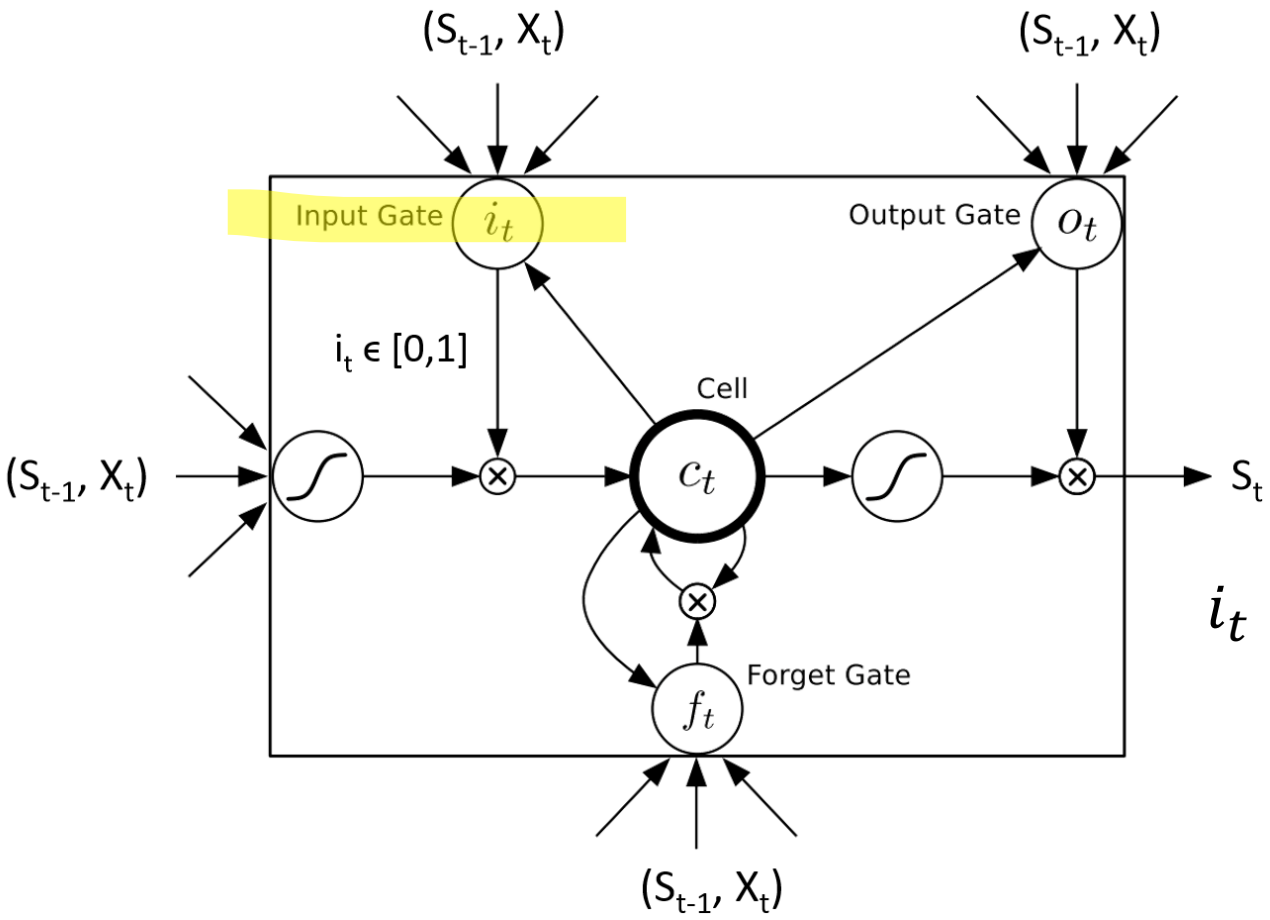


- Forget Gate – scales old cell (reset)

$$f_t = sigmoid\left(x_t U^f + s_{t-1} W^f + c_{t-1} V^f + b_f\right)$$

1 – gate open – remember
0 – gate closed – forget

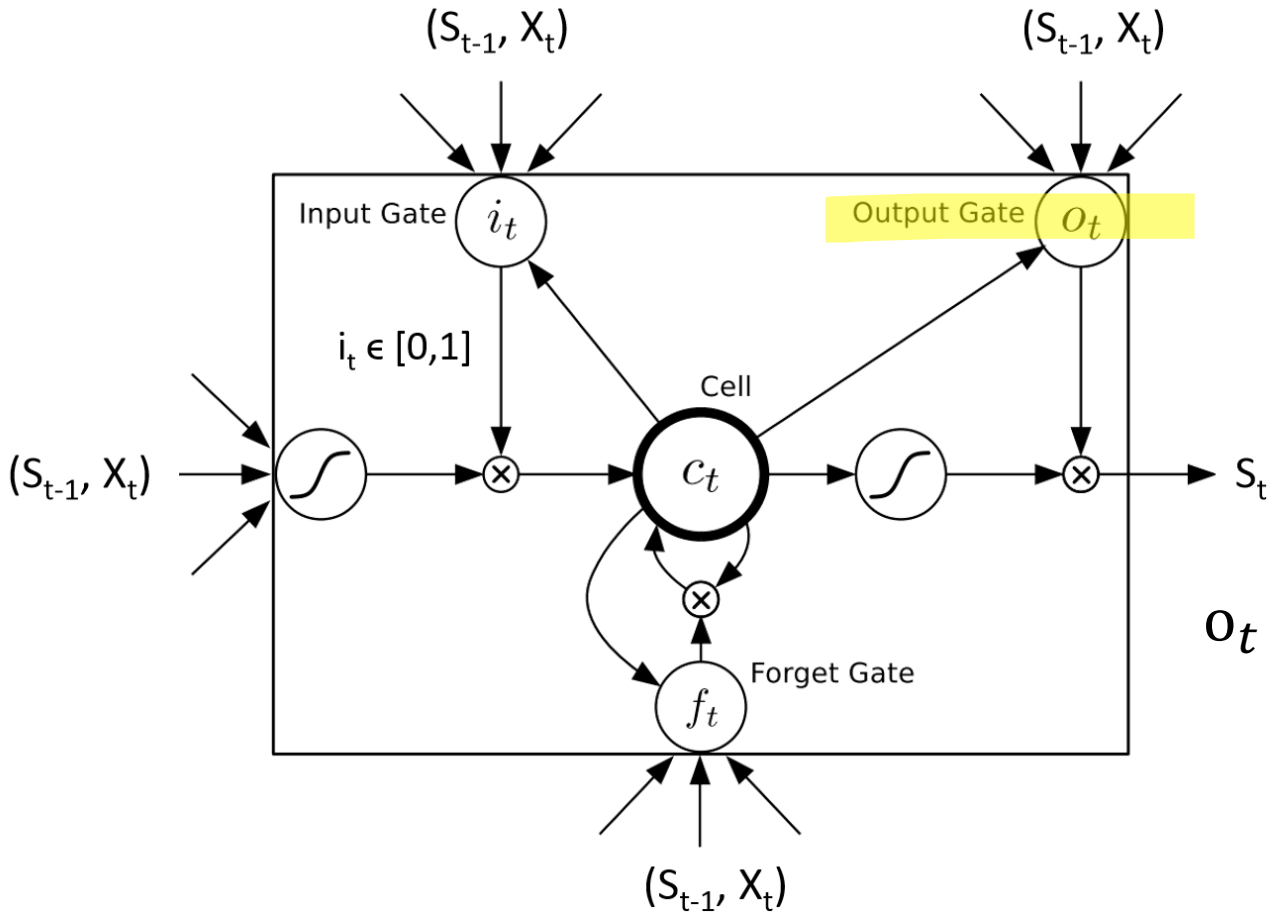
LSTM Unit – input gate



- Input Gate – scales input to cell (write)

$$i_t = sigmoid\left(x_t U^i + s_{t-1} W^i + c_{t-1} V^i + b_i\right)$$

LSTM Unit – output gate

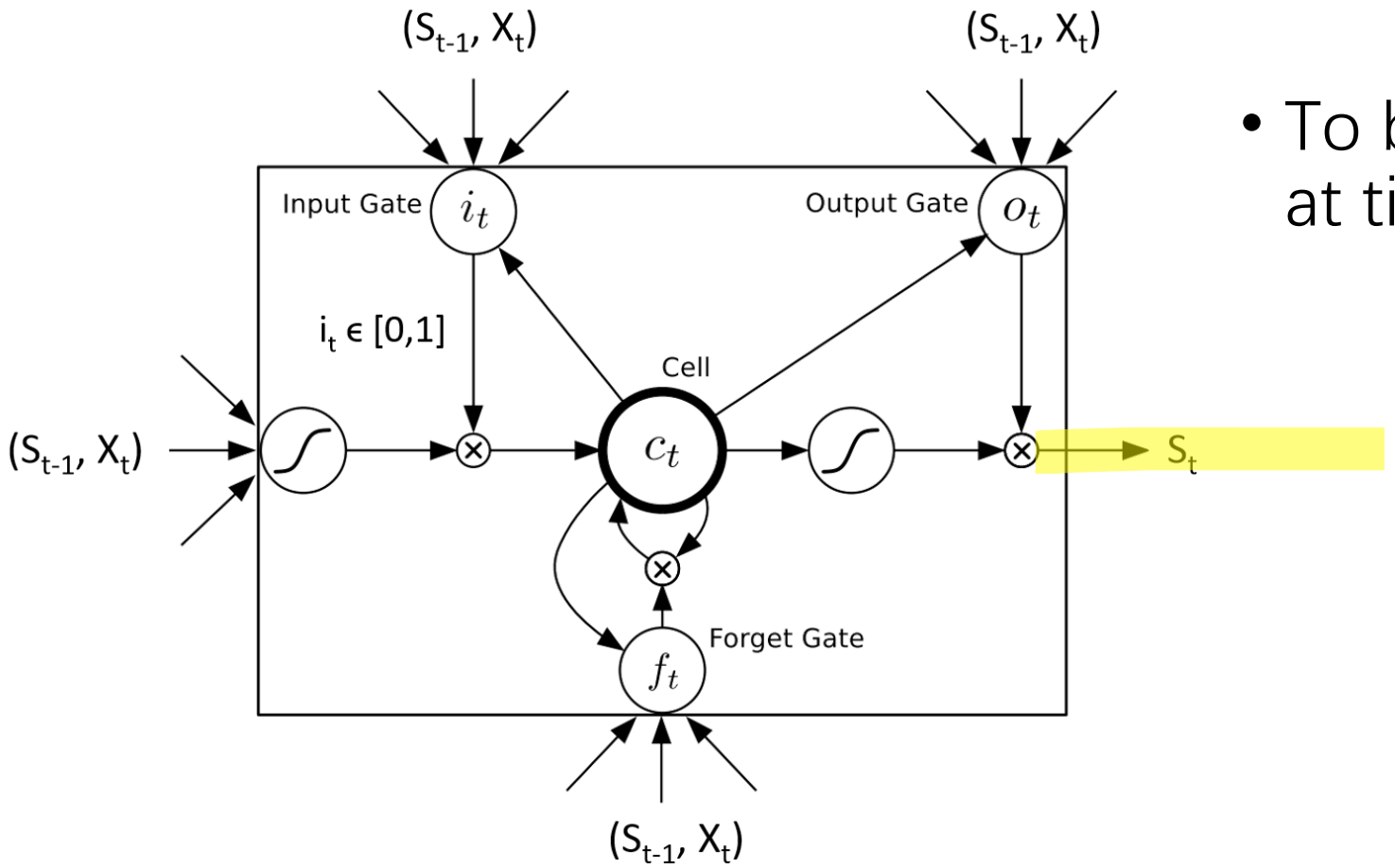


- output Gate – scales output to cell (read)

$$o_t = sigmoid(x_t U^o + s_{t-1} W^o + c_t V^o + b_o)$$

x_t: new input (new info)
s_{t-1}: state at time t-1 (short term memory)
c_t : carry (selected long term memory)

LSTM Unit – hidden state output



- To be passed onto next cell at time $t + 1$

$$s_t = \tanh(c_t) \circ o_t$$

LSTM Unit

- What happens if f is always 1 or 0?
- What happens if input gate is fixed as all 1s, forget gate as all 0s and output gate as all 1s?

How does LSTM avoid gradient vanishing/exploding?

- Depends only on forget gate!

$$\frac{\partial S_3}{\partial S_j} = \prod_{k=j}^{3} sigmoid\left(x_k U^f + s_{k-1} W^f + b_f\right) = \prod_{k=j}^{3} f_k$$

there is no exponentially fast decaying factor involved

However, because that the product-sums have a sigmoid term,
LSTM will suffer from vanishing gradients as well, but not nearly as much as the vanilla RNN

Data Preparing

Raw Data

Uni-variate Time Series

Time	Measure
1	10
2	20
3	30
…	…
1000	10000

Multi-variate Time Series

Date Time	p (mbar)	T (degC)	Tpot (K)	Tdew (degC	rh (%)	VPmax (mb	VPact (mba	VPdef (mba	sh (g/kg)	H2OC (mm	rho (g/m**	wv (m/s)	max. wv (m	wd (deg)
01.01.2009 00:10:00	996.52	-8.02	265.4	-8.9	93.3	3.33	3.11	0.22	1.94	3.12	1307.75	1.03	1.75	152.3
01.01.2009 00:20:00	996.57	-8.41	265.01	-9.28	93.4	3.23	3.02	0.21	1.89	3.03	1309.8	0.72	1.5	136.1
01.01.2009 00:30:00	996.53	-8.51	264.91	-9.31	93.9	3.21	3.01	0.2	1.88	3.02	1310.24	0.19	0.63	171.6
01.01.2009 00:40:00	996.51	-8.31	265.12	-9.07	94.2	3.26	3.07	0.19	1.92	3.08	1309.19	0.34	0.5	198
01.01.2009 00:50:00	996.51	-8.27	265.15	-9.04	94.1	3.27	3.08	0.19	1.92	3.09	1309	0.32	0.63	214.3
01.01.2009 01:00:00	996.5	-8.05	265.38	-8.78	94.4	3.33	3.14	0.19	1.96	3.15	1307.86	0.21	0.63	192.7
01.01.2009 01:10:00	996.5	-7.62	265.81	-8.3	94.8	3.44	3.26	0.18	2.04	3.27	1305.68	0.18	0.63	166.5
01.01.2009 01:20:00	996.5	-7.62	265.81	-8.36	94.4	3.44	3.25	0.19	2.03	3.26	1305.69	0.19	0.5	118.6
01.01.2009 01:30:00	996.5	-7.91	265.52	-8.73	93.8	3.36	3.15	0.21	1.97	3.16	1307.17	0.28	0.75	188.5
01.01.2009 01:40:00	996.53	-8.43	264.99	-9.34	93.1	3.23	3	0.22	1.88	3.02	1309.85	0.59	0.88	185
01.01.2009 01:50:00	996.62	-8.76	264.66	-9.66	93.1	3.14	2.93	0.22	1.83	2.94	1311.64	0.45	0.88	183.2
01.01.2009 02:00:00	996.62	-8.88	264.54	-9.77	93.2	3.12	2.9	0.21	1.81	2.91	1312.25	0.25	0.63	190.3
01.01.2009 02:10:00	996.63	-8.85	264.57	-9.7	93.5	3.12	2.92	0.2	1.82	2.93	1312.11	0.16	0.5	158.3
01.01.2009 02:20:00	996.74	-8.83	264.58	-9.68	93.5	3.13	2.92	0.2	1.83	2.93	1312.15	0.36	0.63	184.8
01.01.2009 02:30:00	996.81	-8.66	264.74	-9.46	93.9	3.17	2.98	0.19	1.86	2.99	1311.37	0.33	0.75	155.9
01.01.2009 02:40:00	996.81	-8.66	264.74	-9.5	93.6	3.17	2.97	0.2	1.85	2.98	1311.38	0.07	0.5	272.4
01.01.2009 02:50:00	996.86	-8.7	264.7	-9.55	93.5	3.16	2.95	0.21	1.85	2.96	1311.64	0.32	0.63	219.2

Prediction task – univariate

Uni-variate Time Series

Time	Measure
1	10
2	20
3	30
…	…
1000	10000

Given data going as far back as **lookback** timesteps

Can you predict the data in **delay** timesteps?

Prediction task – univariate

Uni-variate Time Series

Time	Measure
1	10
2	20
3	30
4	40
5	50
6	60
7	70
8	80
9	90

lookback = 3 **delay** = 2

Series	Target - Y
10,20,30	50
20,30,40	60
30,40,50	70
40,50,60	80
50,60,70	90

Shape of input data

- **Samples**. One sequence is one sample. A batch is comprised of one or more samples.
- **Time Steps**. One time step is one point of observation in the sample. (=lookback)
- **Features**. One feature is one observation at a time step.

- (batch_size, 3, 1)

Series	Target - Y
10,20,30	50
20,30,40	60
30,40,50	70
40,50,60	80
50,60,70	90

Prediction task – multivariate

Uni-variate Time Series

Time	Measure1	Measure2
1	10	0.1
2	20	0.2
3	30	0.3
…	…	…
1000	10000	100

Given data going as far back as **lookback** timesteps

Can you predict the data in **delay** timesteps?

Prediction task – multivariate

multi-variate Time Series

Time	Measure1	Measure2
1	10	0.1
2	20	0.2
3	30	0.3
4	40	0.4
5	50	0.5
6	60	0.6
7	70	0.7
8	80	0.8
9	90	0.9

lookback = 3 **delay** = 2

Series	Y(Measure1)
(10 0.1),(20,0.2),(30,0.3)	50
(20 0.2),(30,0.3),(40,0.4)	60
(30 0.3),(40,0.4),(50,0.5)	70
(40 0.4),(50,0.5),(60,0.6)	80
(50 0.5),(60,0.6),(70,0.7)	90

Shape of input data

- **Samples**. One sequence is one sample. A batch is comprised of one or more samples.
- **Time Steps**. One time step is one point of observation in the sample. (=lookback)
- **Features**. One feature is one observation at a time step.

- (batch_size, 3, 2)

Series	Y(Measure1)
(10 0.1),(20,0.2),(30,0.3)	50
(20 0.2),(30,0.3),(40,0.4)	60
(30 0.3),(40,0.4),(50,0.5)	70
(40 0.4),(50,0.5),(60,0.6)	80
(50 0.5),(60,0.6),(70,0.7)	90