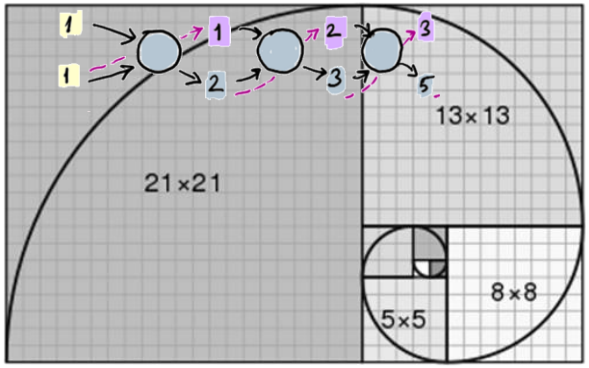
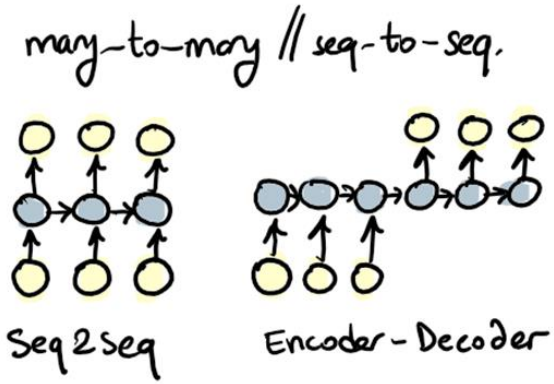
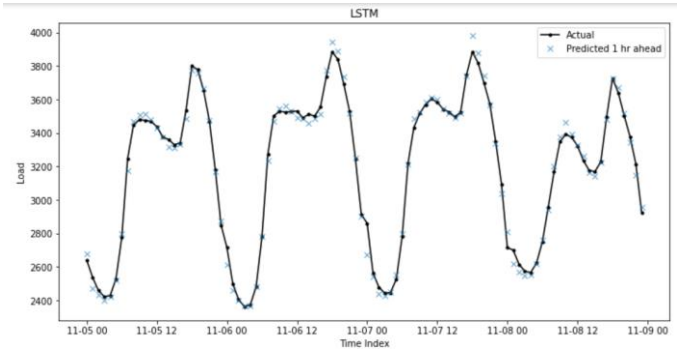
Data Driven Engineering I: Machine Learning for Dynamical Systems

Analysis of Dynamical Datasets II: Time Series

Institute of Thermal Turbomachinery
Prof. Dr.-Ing. Hans-Jörg Bauer





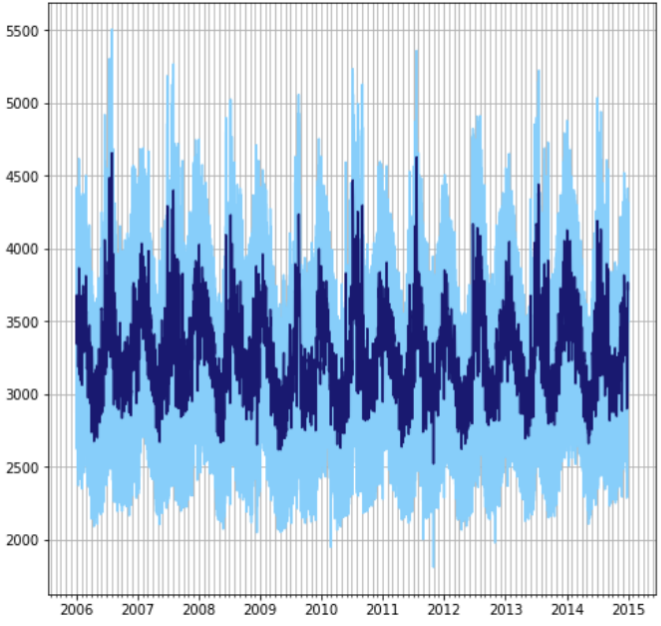


Dynamical Datasets I: Time Series

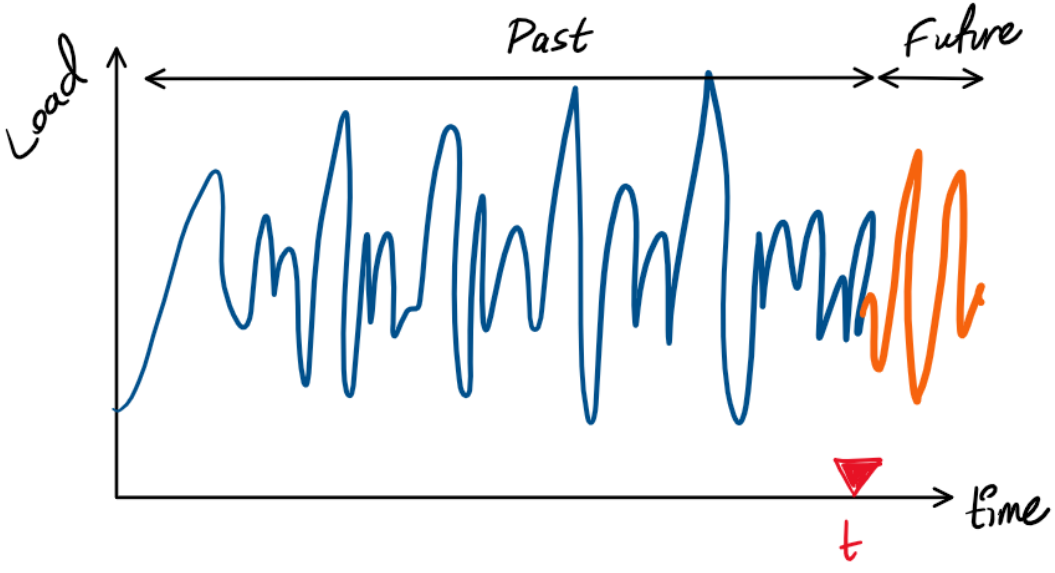
- Time Series: Overview
- Statistical Models for time series
- State space models ⇒ DDE II
- Machine Learning Part I
- Machine Learning Part II

Case: Energy Demand Forecasting

Typical	STLF	LTLF
Horizon	1 hr – 2 days	⩾ 1 months
Granularity	~ hr	~ hr – day
History Range	~ 2 years	~ ⩾ 5 years
Accuracy	⩽ 5% error	⩽ 25% error
Forecasting freq.	~ hr to day	⩾ month



how can we use ML algorithms ?



In M.L.:

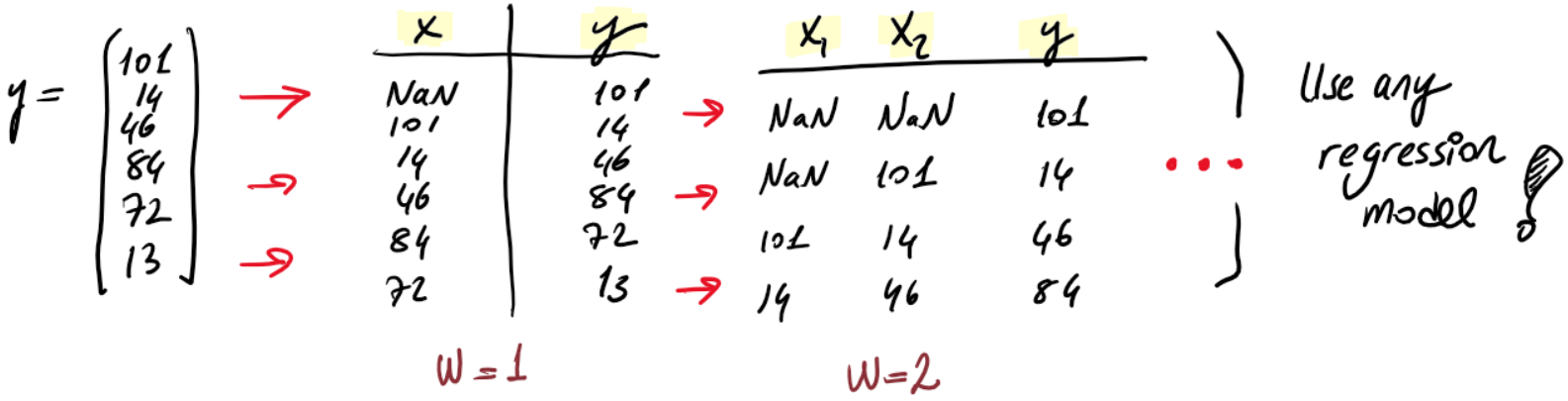
* [x] → [y]

time	Load
0	321
1	316
2	314
3	
4	318
...	

y(t)

how can we use ML algorithms?

* Time Series ⇒ "Supervised learning task" [batch // real-time]



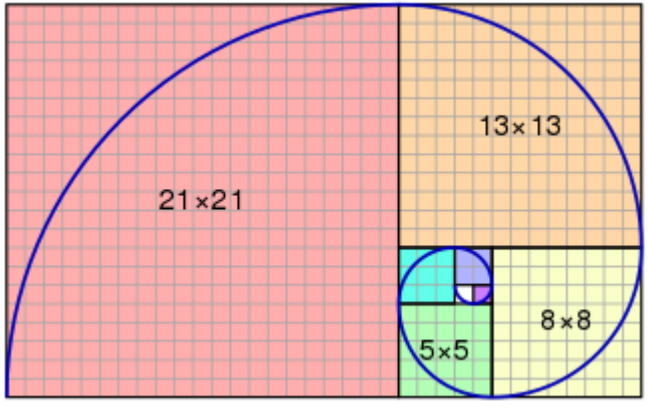
Ateliers & Saveurs in Montreal

colab

Recurrent Neural Networks

Data → N.N. → Rules
Label →

} No inherent seq. arch. ⇒ How can we design a temporal graph?



Eg: Fibonacci Sequence:

1	1	2	3	5	8	13	21	34	55	...
X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	

Rule: $X_n = X_{n-1} + X_{n-2}$

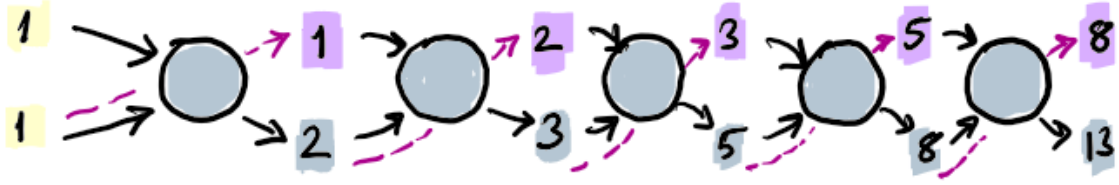
Recurrent Neural Networks

Fibonacci Sequence:

1 1 2 3 5 8 13 21 34 55 ...

X_0 X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8 X_9

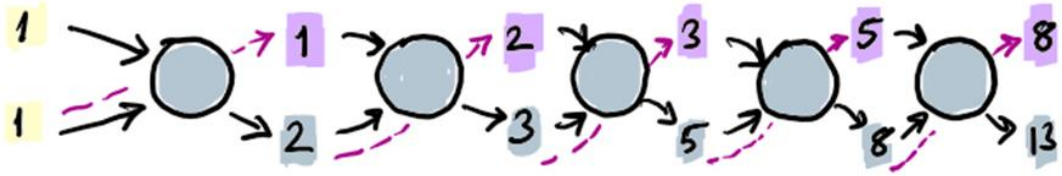
Graph:

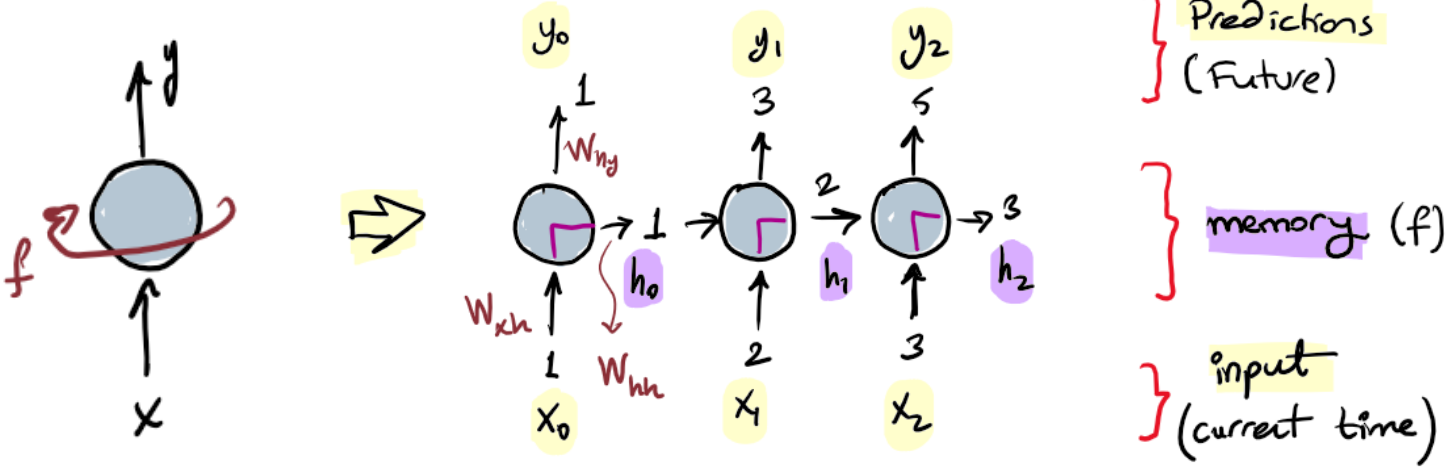


W=2

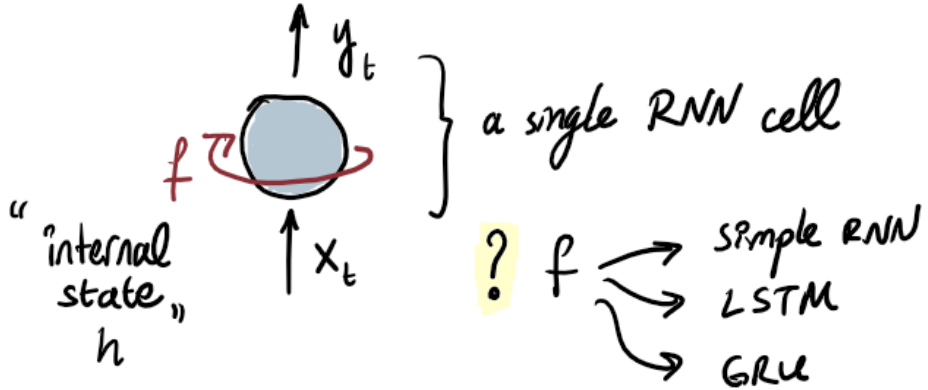
→ Recurrance of information

Recurrent Neural Networks





Recurrent Neural Networks



$$h_t = f_W(h_{t-1}, x_t)$$ "weights"

- h_t: updated state
- h_{t-1}: previous state
- x_t: input @ t

Eg $f \Rightarrow \tanh$;

① $h_t = \tanh\left(W_{hh}\, h_{t-1} + W_{xh}\, x_t + \text{bias}\right)$

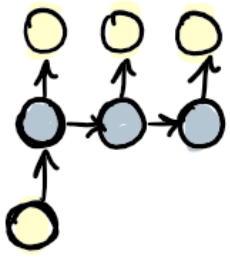
② $y_t = W_{hy}\, h_t$

$$y_t = g(x_t, x_{t-1}, x_{t-2}, \ldots, x_0)$$

"memory"

Feeding a RNN

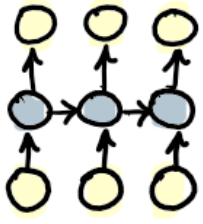
One-to-many // vector-to-seq.



Feature extraction:

$[X] \rightarrow$ info I
$\hookrightarrow$ info II
Exp $\hookrightarrow$ info III

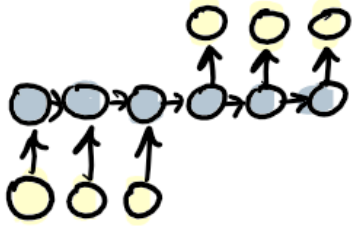
many-to-many // seq-to-seq.



Seq2seq

Sequence Modeling

$[T, C_x] \rightarrow (r_A, \xi)$

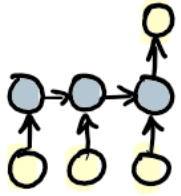


Encoder-Decoder

NLP // Machine Translation

$[t_0 - t_5 \rightarrow t_6 - t_{10}]$

many-to-one // seq-to-vector



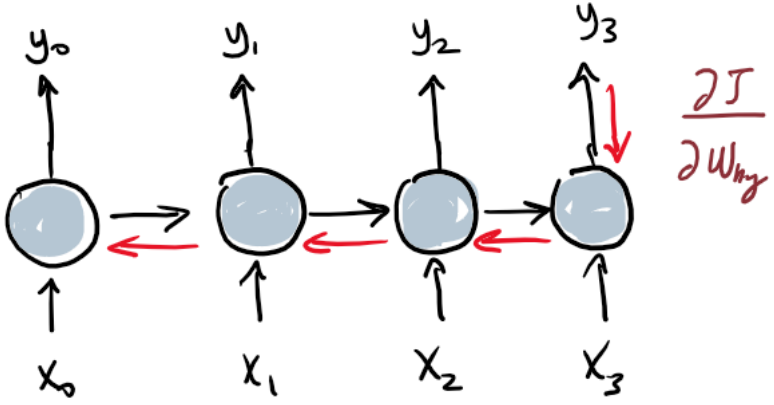
Sentiment Analysis

$[X] \rightarrow \begin{pmatrix} \checkmark \\ 0 \end{pmatrix} \parallel \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$

Training Your RNN

* Back prop. th. time → unroll the nodes in time

* tanh / sigmoids ⇒ vanishing grad.
  ↓
  Test with ReLU (?)
  ↳ Using LSTM & GRU.
  ↳ Tune hyperparameters



Multivariate Time Series Forecasting

[X] → load temp.
... ...

median (w)	std (w)	Temp.	Load
NaN	NaN	50	2500
NaN	NaN	51	2004
NaN	NaN	50	2302
2300	200	54	2280

w=4

? how many previous steps will you pass

? how many future steps will you predict at once

} Graphical architecture

Multivariate Time Series Forecasting

Other Important issues:

* Data Scaling ⇒ [-1, 1] ; mean = 0.0

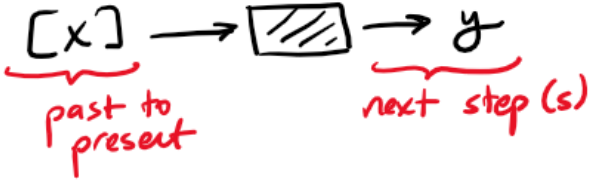
↳ past → min, max, std → ? how valid !

make sure that it is large enough.

You should not leak info. about future.

* Data Processing ⇒ use indices a lot !

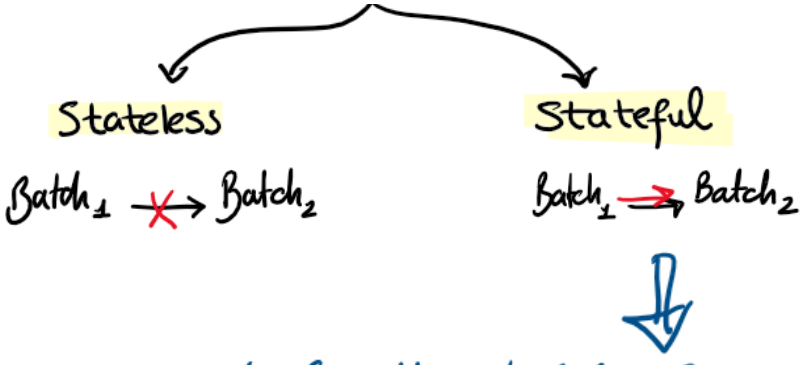
↳ Time stamp management with sliding windows



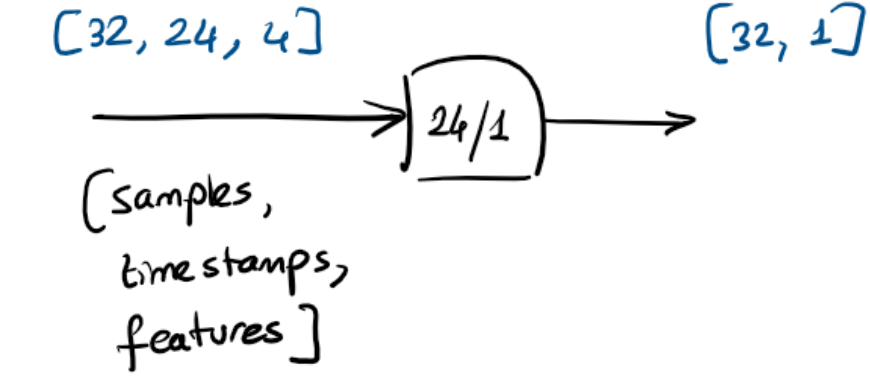
Other Important issues:

* Training & Predicting with TF

= Initialization =



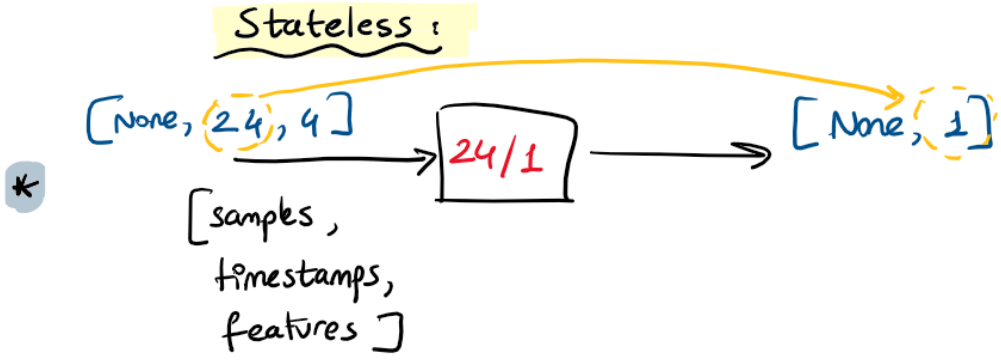
You must fix the batch size.



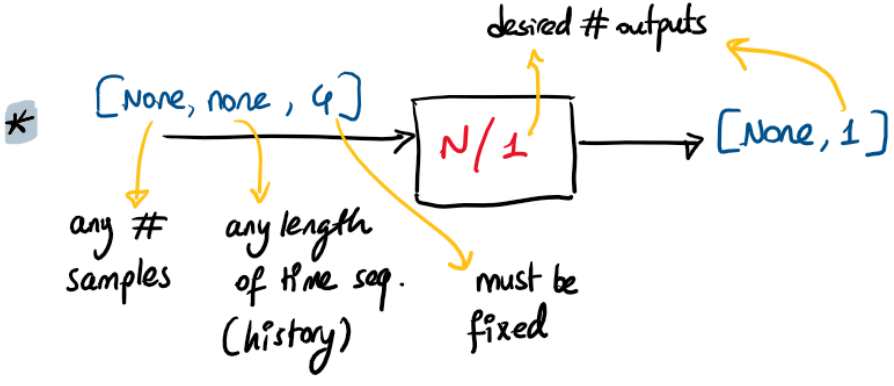
Eg// [1, 24, 4] → ✗

Train it to predict 32 samples at once.

Other Important issues:



X.shape → [1, 24, 4]



Sample 1 → $[x^1, x^2, x^3, x^4, x^5]$ → y_p

Sample 2 → $[x^1, x^2, x^3]$ → y_p

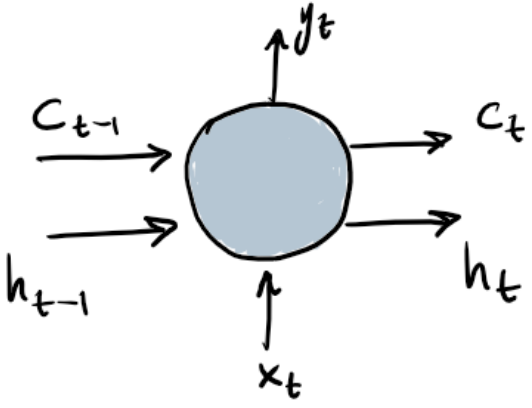
Ateliers & Saveurs in Montreal

colab

Extending the Limits of Memory: LSTM

* Long Short Term Memory

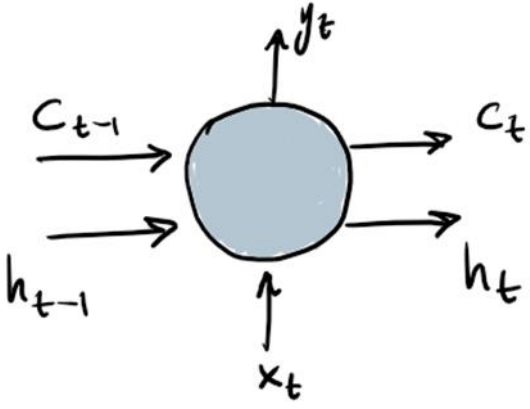
* Recurrance formula is more complex
  - hidden state (h) ⇨ Short Term
  - cell state (c) ⇨ Long Term

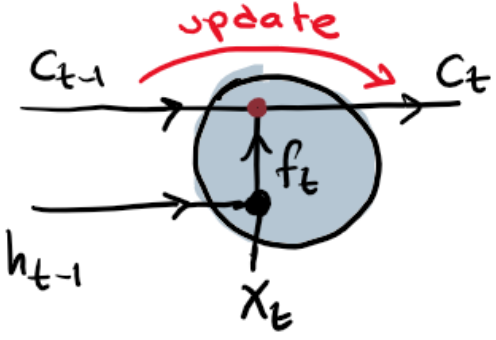


Binary Gates → sigmoid (diff.)

Extending the Limits of Memory: LSTM

① What information should I keep (forget)?





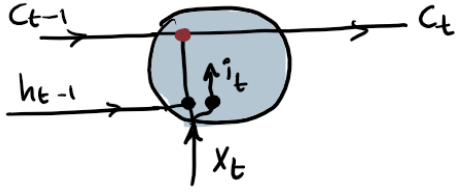
⇒ how much we keep

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad \Rightarrow [0, 1]$$

Extending the Limits of Memory: LSTM

(2) How much of the new information should I add?

(i)

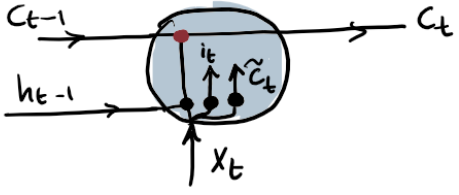


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, X_t] + b_i\right)$$

$[0,1]$ → fraction to keep

} Probability

(ii)



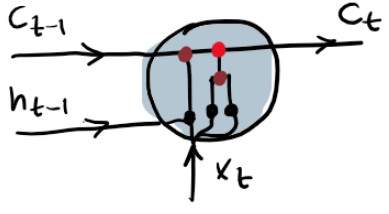
Candidate information:

$$\tilde{C}_t = g\left(W_c \cdot [h_{t-1}, X_t] + b_c\right)$$

[tanh]

} ~ scaled ~ Possibility Space

(iii)



$$C_t = C_{t-1} \cdot f_t + i_t \cdot \tilde{C}_t$$

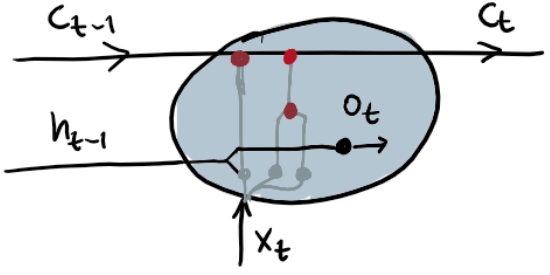
$C_{t-1} \cdot f_t$: forget some fraction of past

$i_t \cdot \tilde{C}_t$: Degree of update → new candidate values

Extending the Limits of Memory: LSTM

③ What should I give as the output (h) ?

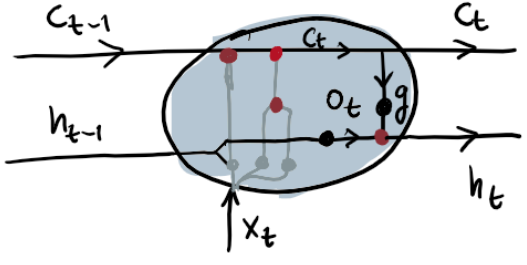
(i)



$$O_t = \sigma\left(W_o \cdot [h_{t-1}, X_t] + b_o\right)$$

$[0,1] \Rightarrow$ probability of past passing to future

(ii)



$$h_t = O_t \cdot g(C_t)$$

new hidden state

how cell state influence output

next time interation

calculation of the prediction y_t

Ateliers & Saveurs in Montreal

colab

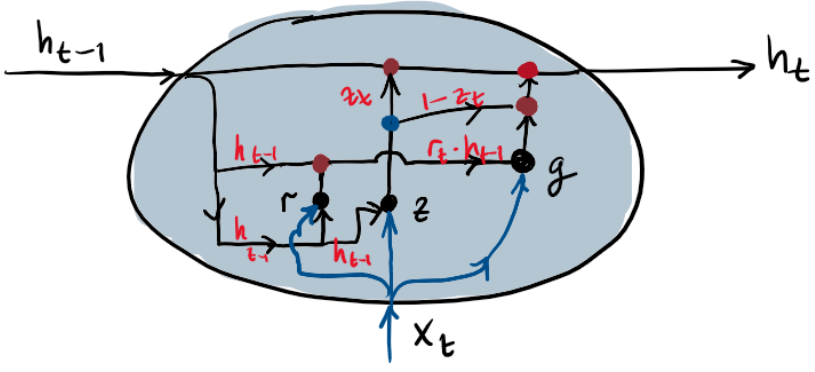
Gated Recurrent Unit (GRU):

* Simplified LSTM → h only (no c)

* Forget, Input } update gate
* Reset Gate } how much past is needed?

z ⇒ update gate ↗ forget ↘ input

r ⇒ Probability of passing to future state



(i) $z_t = \sigma\left(W_{xz}^T \cdot X_t + W_{hz}^T h_{t-1} + b_z\right)$

(ii) $r_t = \sigma\left(W_{xr}^T X_t + W_{hr}^T h_{t-1} + b_r\right)$

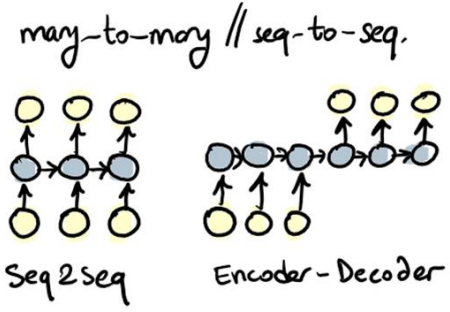
(iii) $g_t = \tanh\left(W_{xg}^T X_t + W_{hg}^T (r_t \cdot h_{t-1}) + b_g\right)$

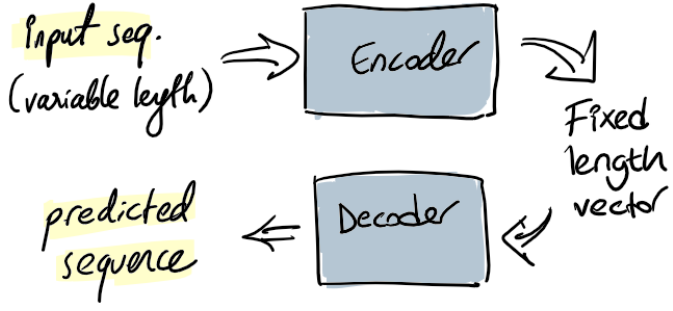
(iv) $h_t = (z_t \cdot h_{t-1}) + (1 - z_t) \cdot g_t$

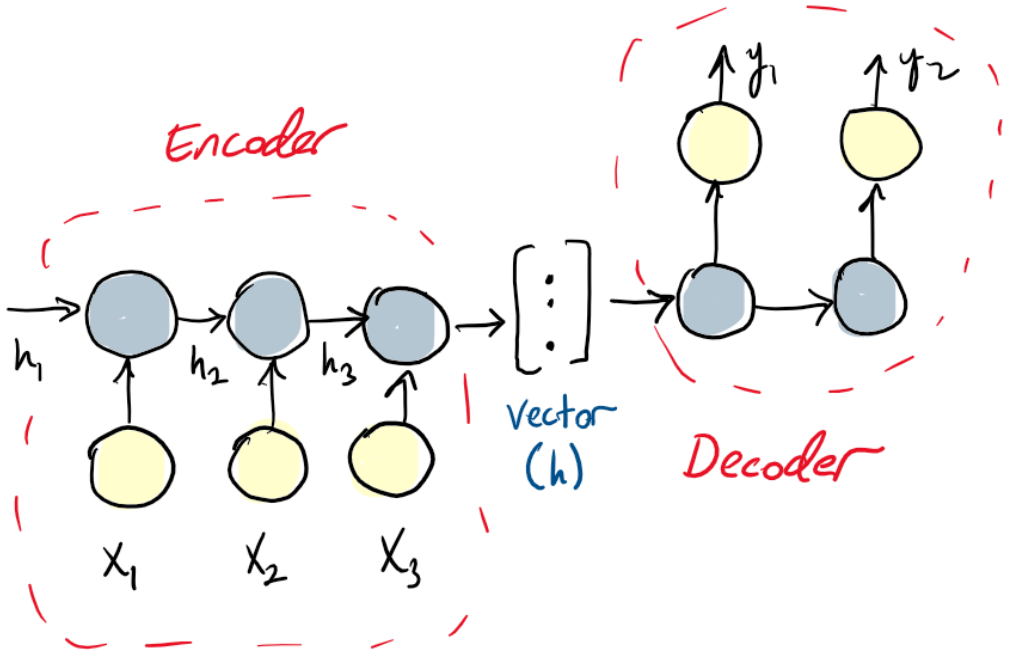
Ateliers & Saveurs in Montreal

colab

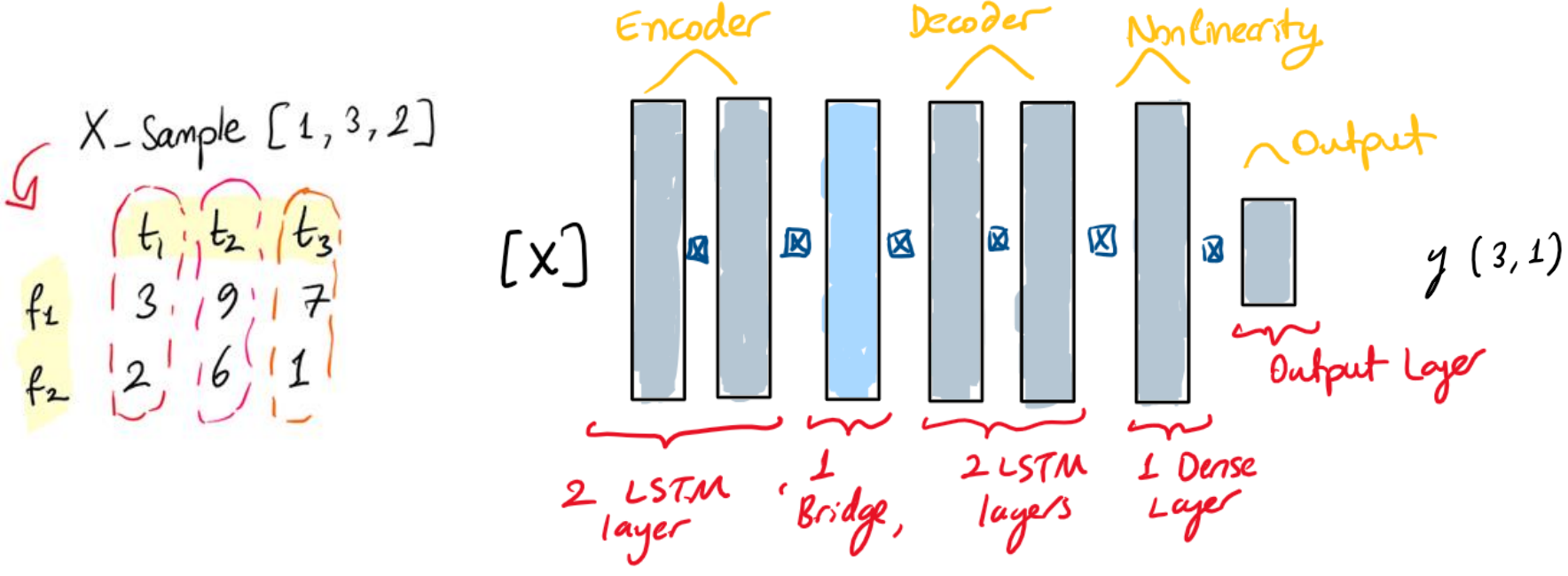
Encoder - Decoder + LSTM







Encoder – Decoder + LSTM

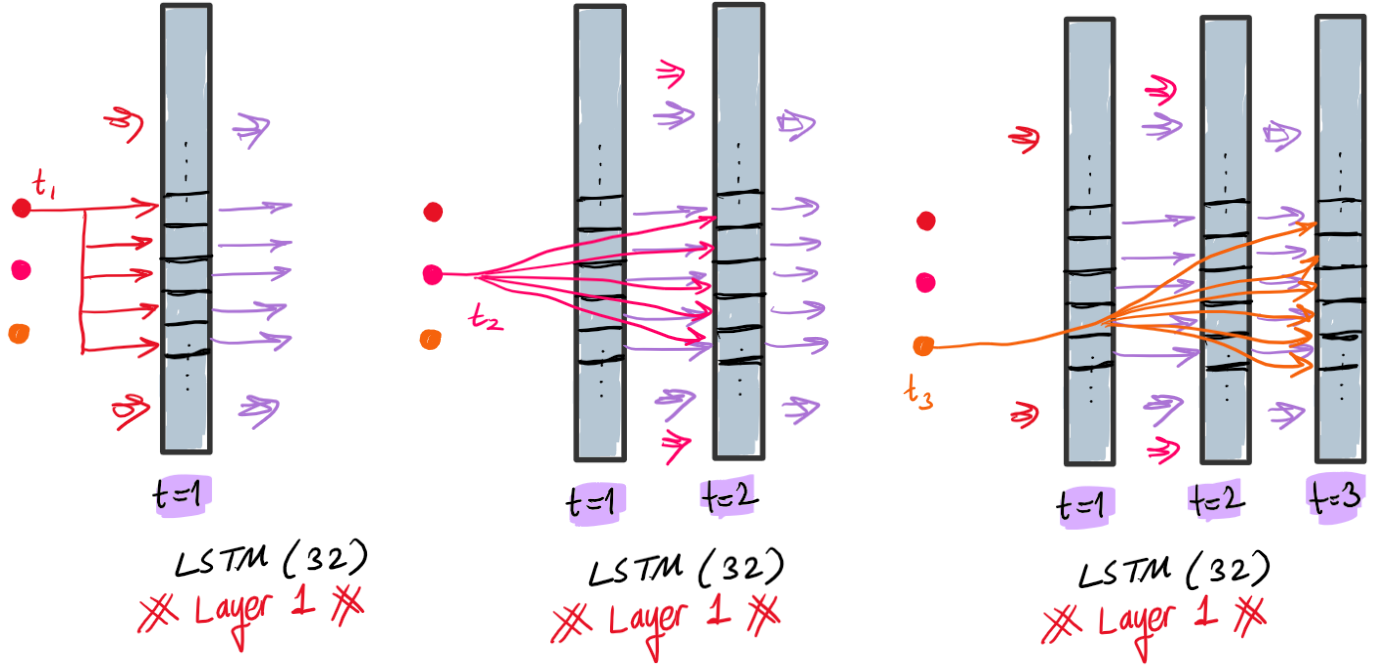


Encoder-Decoder+LSTM

Encoding...

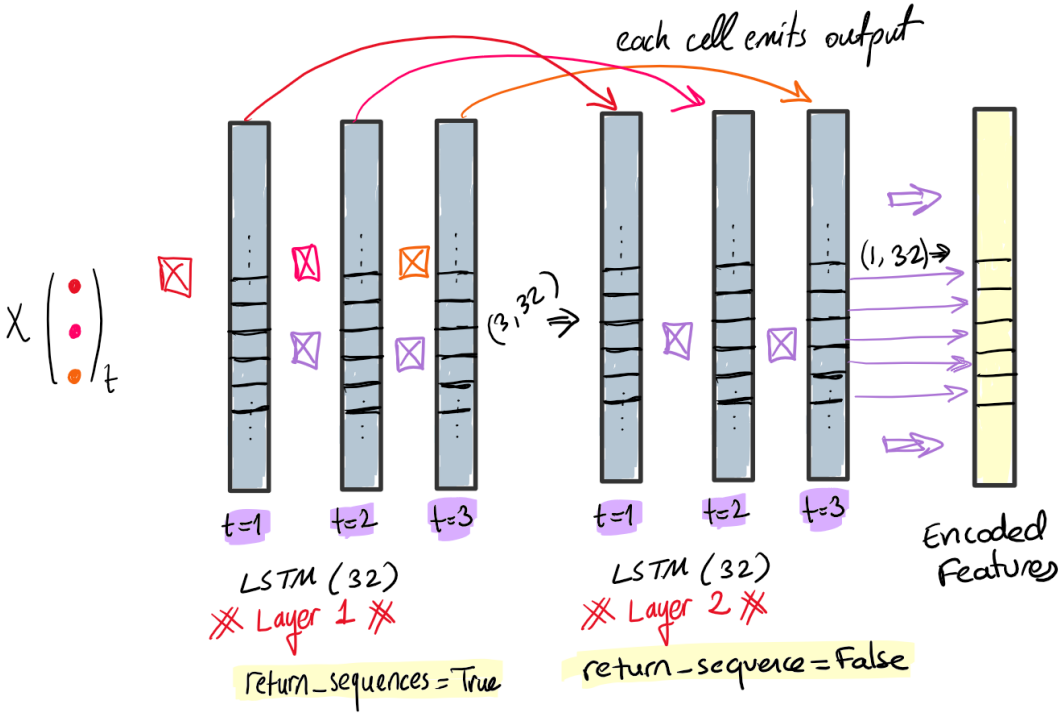
X_Sample [1, 3, 2]

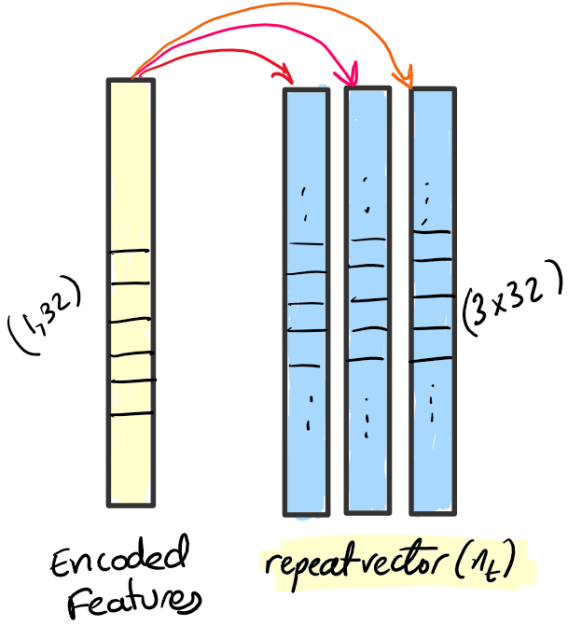
	t_1	t_2	t_3
f_1	3	9	7
f_2	2	6	1



Encoder-Decoder+LSTM

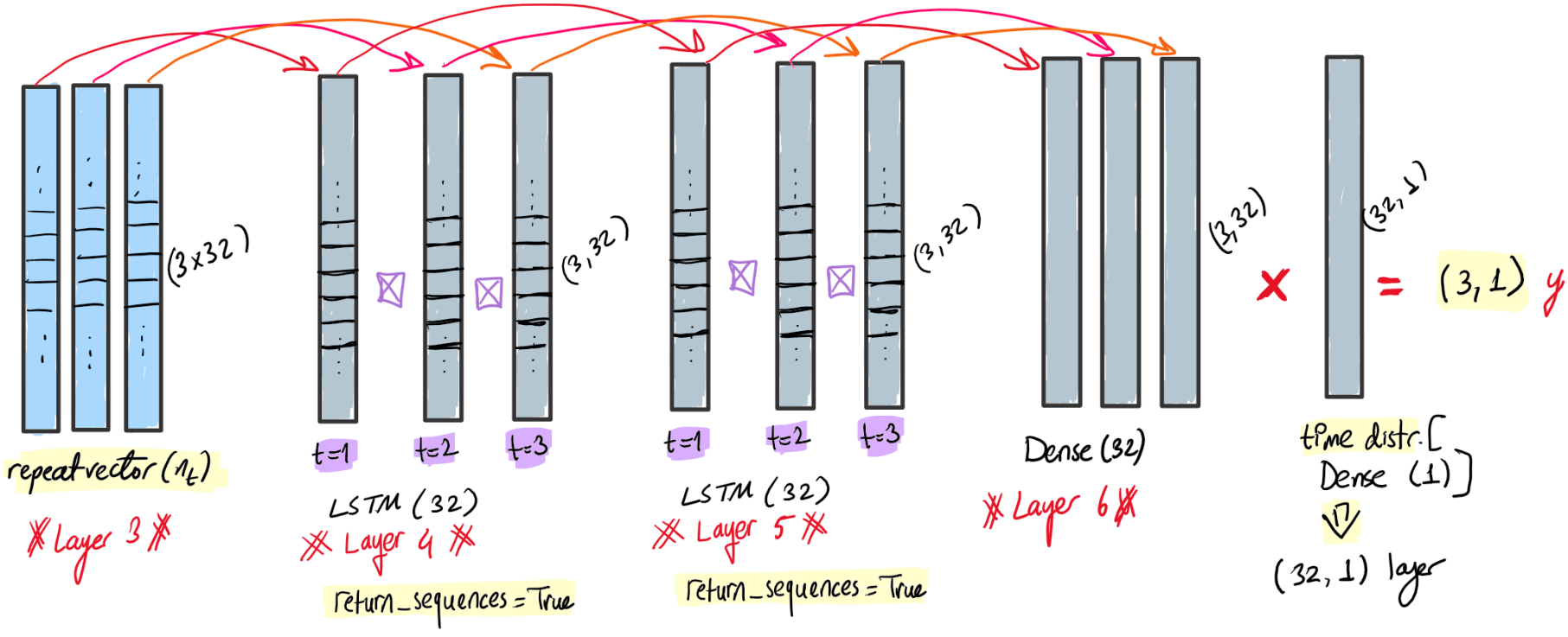
Encoding...





Encoder-Decoder+LSTM

Decoding ...



Additional Notes