Day 4: Recurrent Neural Networks (RNNs) & Application to Sequence Modelling

RNN = special kind of NN for processing sequential data.

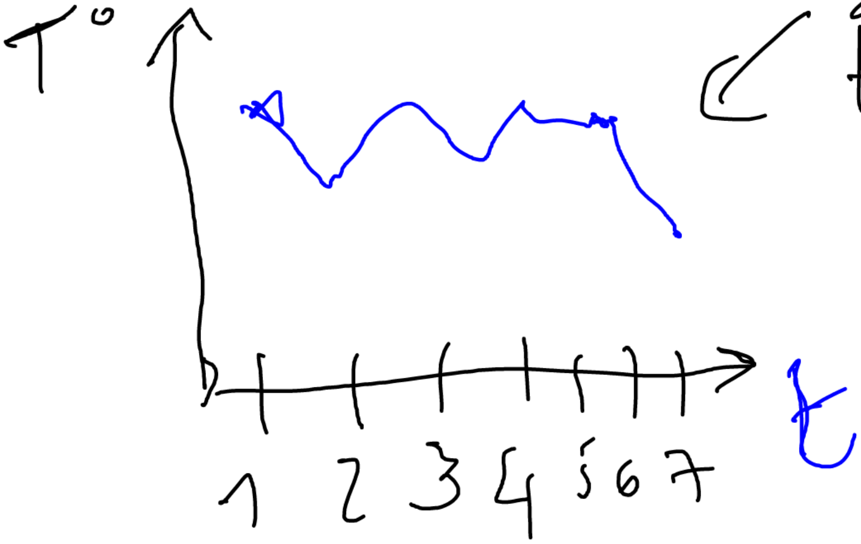
$$\boxed{X = [x^{(1)} \ldots x^{(T)}]}$$

each point $x^{(i)}$ of the sequence is dependant of the rest of the sequence points $x^{(j)}$

$x^{(t)}$ dependant of the $x^{(t-1)}, x^{(t-2)}, \ldots$

- Timeseries (Séries temporelles)
- Langage Naturel → Natural Language Processing

⊗ time series



série temporelle 1D.

↳ Multivariate time-series (séries temporelles multivariées)

variables:

(T°, P, humidity rate, % de précipitation)

air pressure,

→ X (N, T, F)

input data

nombre de pas de temps

nombre d'échantillons

nombre de variables.

F=4 for example in the above example.

⊗ langage Naturel (NLP)

ex: sentences

Sample 1: 'The cat is on the mat' 6 words

Sample 2: 'Time flies like a arrow' 5 words

⋮

Sample n: 'I am fine today' 4 words

Raw data

↳ "data cleaning"

mapping (w_i : idx)

↳ ① Tokenization → vocabulaire avec tous les mots

Sample 1:

↳ ['The', 'cat', 'is', 'on', 'the', 'mat']

↳ ② words → ids

Sample 1 → [1, 6, 3, 15, 63, 26]

↳ ③ padding

max sentence length = 10

Samples → [1, 6, 3, 15, 63, 26, 0, 0, 0, 0]

(4) Embedding

list_idx $\xrightarrow{\text{embedding}}$ Emb(list) $\in \mathbb{R}^{D}$.

text representation

(1) encode similarities between words meanings

(2) encode 'grammar'

→ INPUT X (TENSOR ...)

(N , T , D)

N: samples

T: ex: 10 words

D: embedding size

(X) other sequential data types

↳ 'Speech'

↳ 'Music'

RNNs → represent the notion of sequentiality by having parameter sharing

↳ a RNN shares the same weights across several time-steps.

↳ Apply these models to data with ≠ input sequence lengths.

IA/ Dynamical System

$$\boxed{s^{(t)} = f(s^{(t-1)}, \theta)}$$

parameter θ

dynamical system driven by an external signal $x^{(t)}$

$$\boxed{s^{(t)} = f(s^{(t-1)}, x^{(t)}, \theta)}$$

typical equation of a RNN

$$\boxed{h^{(t)} = f(h^{(t-1)}, x^{(t)}, \theta)}$$

Neural Network parameter

state of RNN

↳ hidden representation of the RNN

↳ output of the RNN hidden layer.

↳ $h^{(t)}$ = 'lossy' summary of the input data until timestep t $(x_1 \dots x(t))$

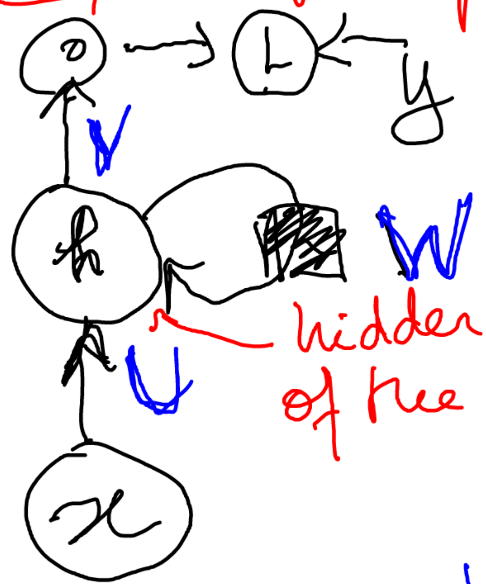
→ map an arbitrary 'long' sequence in a fixed_length vector.

→ unfolded dynamical system.

$$\boxed{h^{(t)} = g(x^{(t)}, x^{(t-1)}, x^{(t-2)}, \dots x^{(1)}, h^{0})}$$

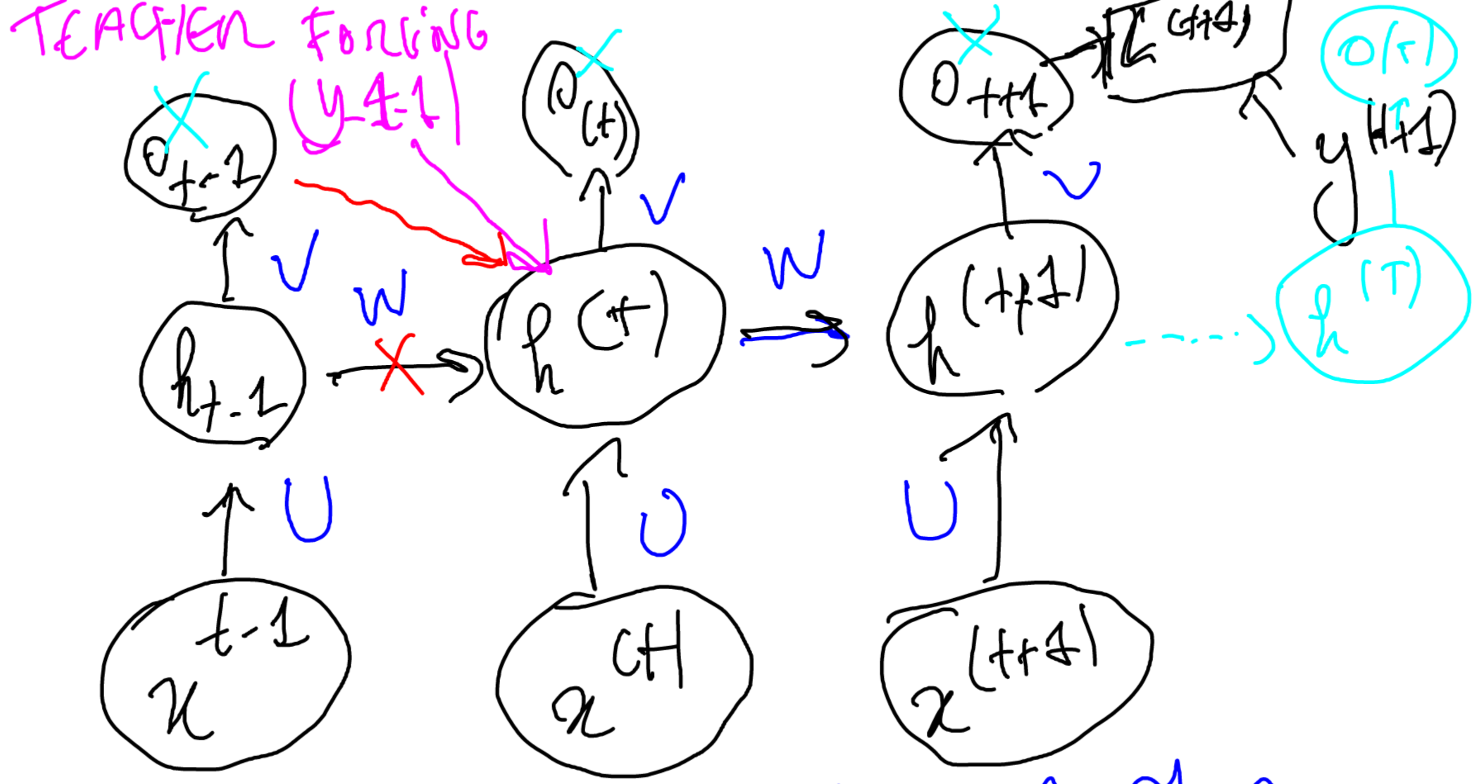
↳ Generally, we use the same transition function f with the same parameters θ at every timestep

IB) Graph of Recurrent Neural Networks



hidden layer of the RNN.

'compact computational graph for RNN'



unfolded computational graph of a RNN

Some ≠ types of RNNs

- RNN producing an output at each timestep & have recurrent connections between hidden units

1. produce an output at each timestep but have only recurrent connections only from the output at timestep at the hidden units of the timestep

3. RNN with recurrent connections between hidden units → 'reads' an entire sequence and outputs a single output $o^{(\tau)}$

forward propagation equations for the RNN

input-to-hidden connection

⊗ RNN layer : $\begin{cases} a^{(t)} = \boxed{U} x^{(t)} + W h^{(t-1)} + b \\ h^{(t)} = \tanh\left(a^{(t)}\right) \end{cases}$

hidden-to-hidden connection.

⊗ dense layer : $\left\{ \begin{array}{l} \underbrace{o^{(t)}}_{\text{logits}} = \boxed{V} \times h^{(t)} + c \\ \hat{y}^{(t)} = \text{softmax}\left(o^{(t)}\right) \end{array} \right.$

hidden-to-output connection

⊗ Associated Loss / Cost function

$$\mathcal{L}\left(\{x^{(1)} \ldots x^{(T)}\}, \{y^{(1)} \ldots y^{(T)}\}\right)$$

$$= \sum_{t \in \{1..T\}} \mathcal{L}^{(t)} = -\sum_{t \in \{1..T\}} \log p_{model}\left(y^{(t)} \mid \{x^{(1)} \ldots x^{(t)}\}\right)$$

C/ Teacher Forcing

From RNNs that have output to hidden recurrent connections

↳ can be trained with a technique called $\boxed{\text{Teacher Forcing}}$:

↳ replacing the output-to hidden connection by label-to-hidden connection

↳ $h^{(t)}$ depend not of $o^{(t-1)}$ but prediction of the RNN

of $y^{(t-1)}$ ground truth

↳ Advantages :

⊗ Improve performance by giving the ground-truth from previous timestep

⊗ Ease training of the Recurrent Neural Network by avoiding the 'Backpropagation Through Time' (BPTT) which compute gradients accross all timesteps

Maximum likelihood criterion

2 timesteps (x_1, x_2) Input data (y_1, y_2)

$$\log p(y_1, y_2 \mid x_1, x_2)$$

$$= \log p(y_2 \mid y_1, x_1, x_2) + \log p(y_1 \mid x_1, x_2)$$

↳ Cons

↳ discrepancy between what is done during training & what is done at inference

D) The Challenge of long-Term Dependencies

'long-term dependencies'

↳ Input data with long sequences.

⟹ IN THE TRAINING PROCESS,

pb of → vanishing gradients
↳ gradients très petits

↳ exploding gradients
↳ gradients très grands.

Intuition derrière

$h^{(t)} = W^T h^{(t-1)}$ ↳ parameter sharing

Same W for each timestep.

$h^{(t)} = (W^t)^T h^{(0)}$

$\begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_N \end{bmatrix}$

$W = U \Lambda U^T$ ← matrice de valeurs propres.

Let's assume that (to simplify) $W^t = U \Lambda^t U$

$$\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix}$$

diagonal matrix

W^t as Λ^t

$(\lambda_1 \dots \lambda_n)^t$

exploding gradients ← $> 1 : (\lambda_1 \dots \lambda_n)^t$ large values.

vanishing gradients ← $< 1 : (\lambda_1 \dots \lambda_n)^t$ small values

→ 'gradient clipping'

→ clip(min(∇, ε)) ← gradient clipped to a epsilon value if too large.

⟹ Recurrent Neural Networks with Gated Units

→ LSTM Long Short Term Memory Networks

II / LSTM (and other Gated Recurrent Neural Networks

LSTM: In practice, one of the RNN the most used to process sequential data.

→ Introduce a 'cell memory' (recurrent self-loop) where the gradient can flow for long durations (long sequences)

Internal recurrence

addition to the outer recurrence

$$h^{(t+1)} = f(h^{(t)}, x^{(t)})$$

Compute internal gates

→ forget gate $f_i^{(t)}$

→ external input gate $g_i^{(t)}$

→ output gate $q_i^{(t)}$

Recurrent equations in a LSTM

forget gate

$U^f x^{(t)}$

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right)$$

Internal state of the LSTM cell

$$\boxed{s_i^{(t)}} = \underbrace{f_i^{(t)}}_{\text{forget gate}} s_i^{(t-1)} + \underbrace{g_i^{(t)}}_{\text{External input gate}} \times \sigma\left(b_i + \sum_j U_{i,j}^s x_j^{(t)} + \sum_j W_{i,j}^s h_j^{(t-1)}\right)$$

External input gate

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right)$$

output gate $q_i^{(t)}$

$$\boxed{q_i^{(t)}} = \sigma\left(b_i^q + \sum_j U_{i,j}^q x_j^{(t)} + \sum_j W_{i,j}^q h_j^{(t-1)}\right)$$

output $h_i^{(t)}$ (hidden representation of the RNN) of the LSTM cell

$$\boxed{h_i^{(t)} = \tanh(s_i^{(t)}) \times q_i^{(t)}}$$

internal recurrent state

output gate

→ still a dynamical system

$$h^{(t)} \to f_\theta(h^{(t-1)}, x^{(t)})$$