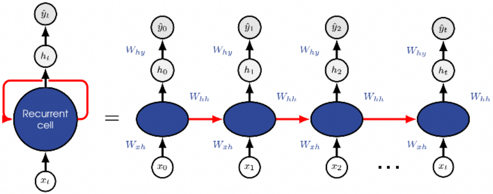


Recurrent Neural Networks

Vincent Barra
LIMOS, UMR 6158 CNRS, Université Clermont Auvergne

LIMOS LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

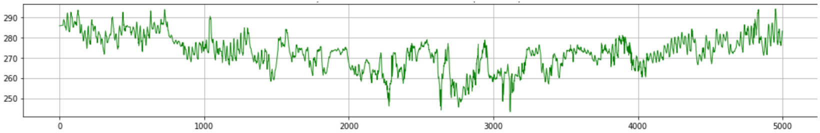
cnrs

UCA UNIVERSITÉ Clermont Auvergne

SEQUENTIAL DATA

Sequential data

- Timeseries data (temperature, pressure, stock market...)
- Speech / music
- Videos
- ...



Problem!

- Arbitrary length
- Huge number of parameter for a model ?

Properties

Need for memory

- Data in a sequence is not identically, independently distributed
- Need for a context, thus for memory

Paul is a small boy and is hungry. He goes to the restaurant and eats a lot.

Properties

Need for memory

- Data in a sequence is not identically, independently distributed
- Need for a context, thus for memory

Paul is a small boy and is hungry. He goes to the restaurant and eats a lot.

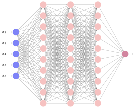
THE question

How to model sequential data, context and memory ?

Data representation

Neural nets need numerical values, not words

small $\cancel{\to}$

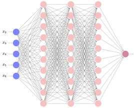


$\cancel{\to}$ boy

Data representation

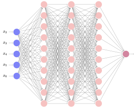
Neural nets need numerical values, not words

small $\cancel{\rightarrow}$



$\cancel{\rightarrow}$ boy

$\begin{pmatrix} 0.4 \\ \vdots \\ 0.1 \end{pmatrix} \rightarrow$

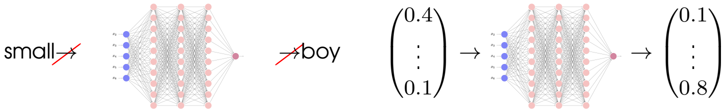


$\rightarrow \begin{pmatrix} 0.1 \\ \vdots \\ 0.8 \end{pmatrix}$

LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Data representation

Neural nets need numerical values, not words



small $\cancel{\to}$ … $\cancel{\to}$ boy

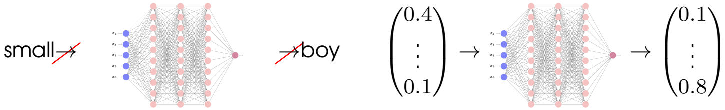
$$\begin{pmatrix} 0.4 \\ \vdots \\ 0.1 \end{pmatrix} \to \cdots \to \begin{pmatrix} 0.1 \\ \vdots \\ 0.8 \end{pmatrix}$$

Corpus

Paul, small, hungry, he, goes, restaurant, and, lot,...

DATA REPRESENTATION

Neural nets need numerical values, not words



small $\cancel{\rightarrow}$ $\cancel{\rightarrow}$ boy

$$\begin{pmatrix}0.4\\ \vdots \\ 0.1\end{pmatrix} \rightarrow \quad \rightarrow \begin{pmatrix}0.1\\ \vdots \\ 0.8\end{pmatrix}$$

Corpus

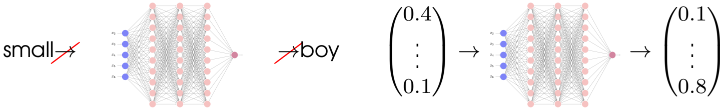
Paul, small, hungry, he, goes, restaurant, and, lot,...

Indexing

a→ 1
boy→ 2
...
small→ 16

Data representation

Neural nets need numerical values, not words



small ~~→~~

~~→~~boy

$$\begin{pmatrix}0.4\\ \vdots \\ 0.1\end{pmatrix} \rightarrow \quad \rightarrow \begin{pmatrix}0.1\\ \vdots \\ 0.8\end{pmatrix}$$

Corpus

Paul, small, hungry, he, goes, restaurant, and, lot,...

Indexing

a→ 1
boy→ 2
. . .
small→ 16

One-hot encoding

a =(1,0,...,0)
boy = (0,1,0,...,0)
. . .
small = (0,0,...,1)

DATA REPRESENTATION

Why not directly using the index as a descriptor ?

Example : distance between “a” and “small”

1. Indexes: $d^2(”a”,”small”) = (16-1)^2 = 225$
2. One hot encoding: $d^2(”a”,”small”)^2 = 2$

- Indexes : distance depends on the values of the index
- One hot encoding : whatever two different words, they have the same distance if they are different

Embeddings

Word2vec

Learns word embeddings by estimating the likelihood that a given word is surrounded by other words.
Bag of words, skip Gram.

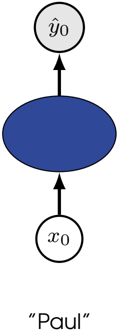
Dimensions

- One-hot vectors: high-dimensional and sparse
- word embeddings: low-dimensional and dense.

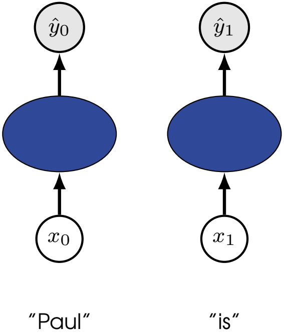
Generalization

- One-hot vectors: constrained by the corpus
- word embeddings: Generalization, capabilities.

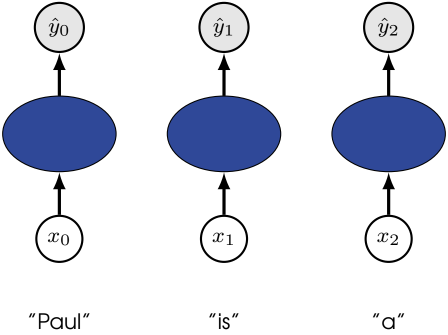
Processing individual data point



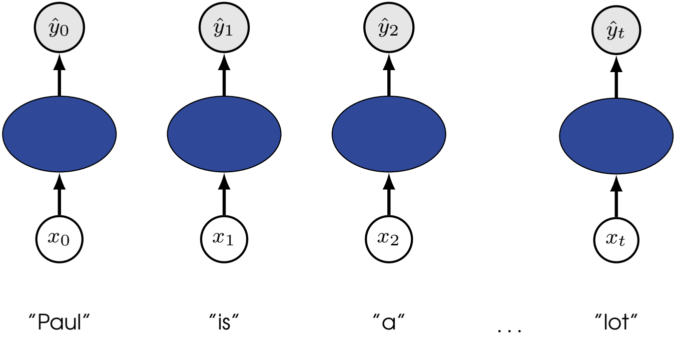
PROCESSING INDIVIDUAL DATA POINT



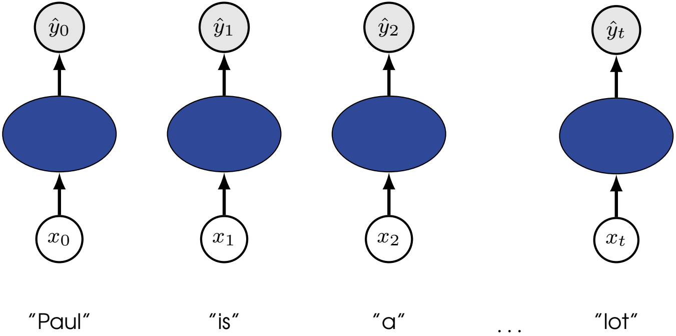
Processing individual data point



Processing individual data point

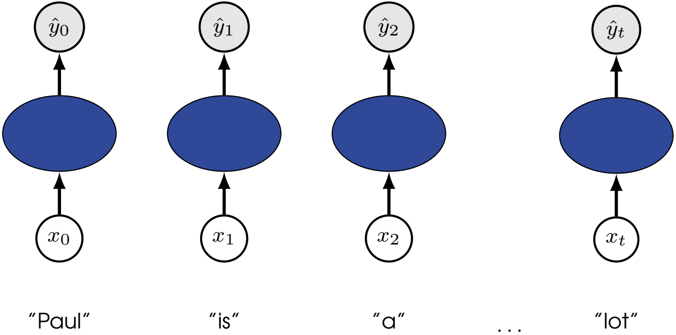


PROCESSING INDIVIDUAL DATA POINT



$$\hat{y}_t = f(x_t)$$

Processing individual data point

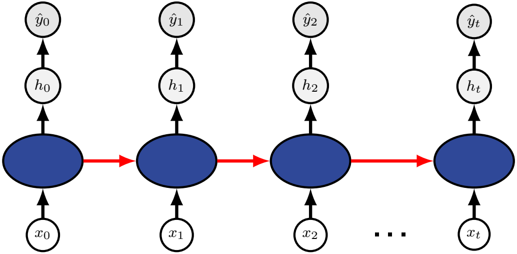


$$\hat{y}_t = f(x_t)$$

	Paul	is	a	small	....	He	goes	to	the	...
t	0	1	2	3	...	8	9	10	11	...

x_8 depends on x_0 $\Rightarrow$ with this model, no possible relation.

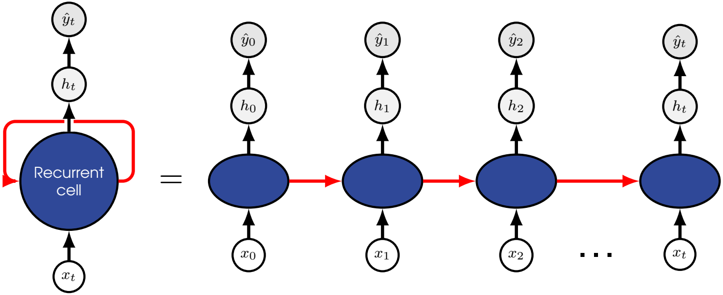
INTUITION: NEURONS WITH RECURRENCE



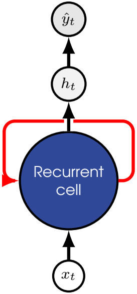
$$\hat{y}_t = f(x_t, h_{t-1})$$

- x_t : input
- $\hat{y}_t$: output
- h_{t-1}: past memory

Folded version



Recurrent Neural Networks



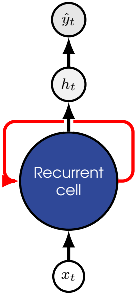
Apply a recurrence relation each time step to process a sequence

$$(\forall t \geq 0) \;\; h_t = f_W(x_t, h_{t-1})$$

- h_t: current cell state
- f_W: neural network with parameter matrix $\boldsymbol{W}$
- x_t: input
- h_{t-1}: old cell state (memory)

To keep memory, $\boldsymbol{W}$ is shared through time.

Recurrent Neural Networks



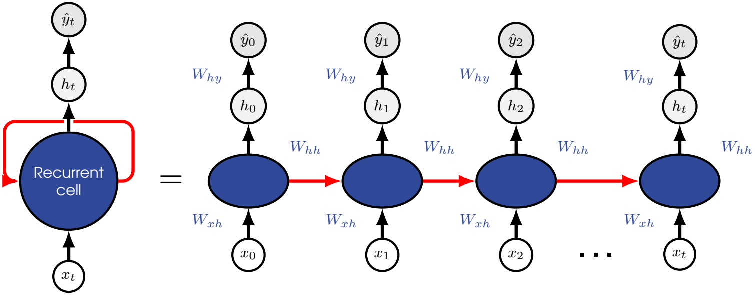
1. Update hidden state

$$h_t = tanh(W_{xh}^T x_t + W_{hh}^T h_{t-1})$$

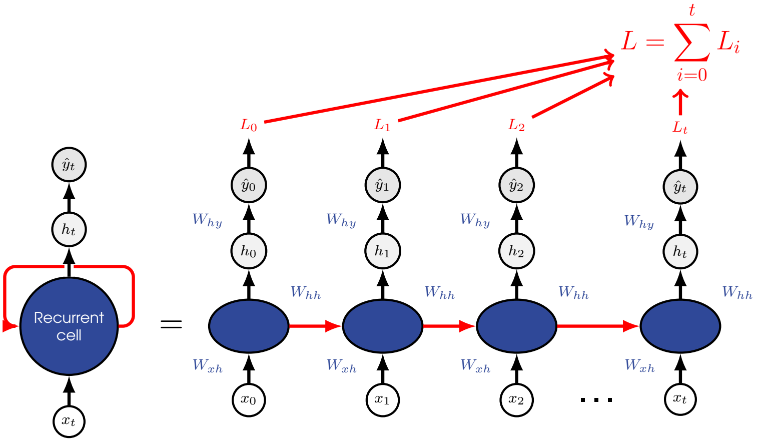
2. Compute output vector

$$\hat{y}_t = W_{hy}^T h_t$$

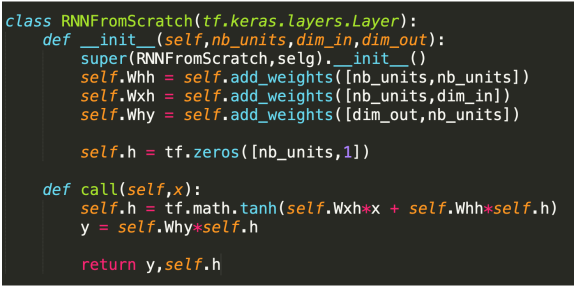
Folded version- forward pass



Folded version- forward pass



Keras implementation from scratch



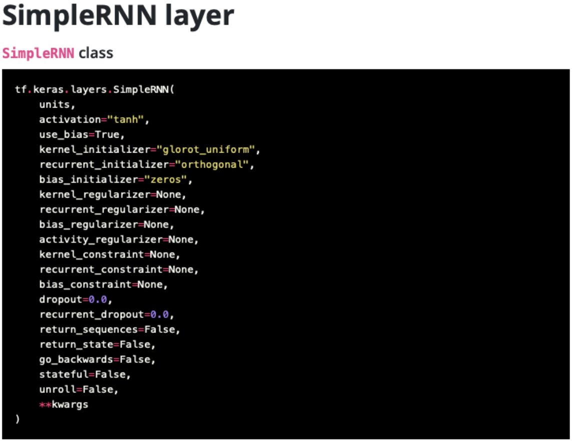
```
class RNNFromScratch(tf.keras.layers.Layer):
    def __init__(self,nb_units,dim_in,dim_out):
        super(RNNFromScratch,selg).__init__()
        self.Whh = self.add_weights([nb_units,nb_units])
        self.Wxh = self.add_weights([nb_units,dim_in])
        self.Why = self.add_weights([dim_out,nb_units])

        self.h = tf.zeros([nb_units,1])

    def call(self,x):
        self.h = tf.math.tanh(self.Wxh*x + self.Whh*self.h)
        y = self.Why*self.h

        return y,self.h
```

Keras implementation: SimpleRNN

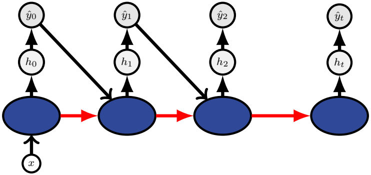


SimpleRNN layer

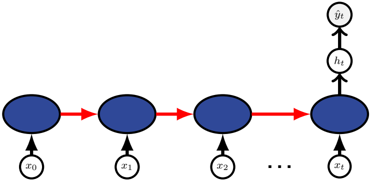
SimpleRNN class

```
tf.keras.layers.SimpleRNN(
    units,
    activation="tanh",
    use_bias=True,
    kernel_initializer="glorot_uniform",
    recurrent_initializer="orthogonal",
    bias_initializer="zeros",
    kernel_regularizer=None,
    recurrent_regularizer=None,
    bias_regularizer=None,
    activity_regularizer=None,
    kernel_constraint=None,
    recurrent_constraint=None,
    bias_constraint=None,
    dropout=0.0,
    recurrent_dropout=0.0,
    return_sequences=False,
    return_state=False,
    go_backwards=False,
    stateful=False,
    unroll=False,
    **kwargs
)
```

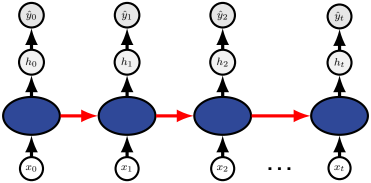
Some architectures



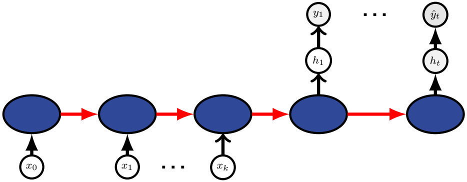
One to many
Captioning, music generation



Many to one
Sentiment analysis

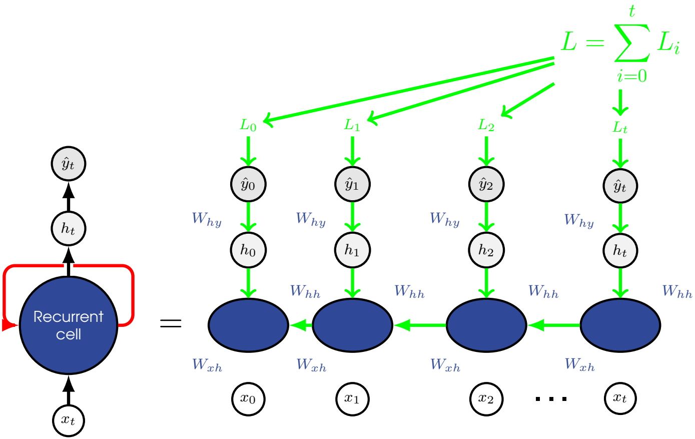


Many to Many
Video annotation



Many to many
Translation

Unfolded version- Backward pass



UNFOLDED VERSION- BACKWARD PASS

Computing the gradient w.r.t. h_0 involves

- many factors of $\boldsymbol{W}_{hh}$
- repeated gradient computation

Many high values

- Exploding gradients
- Gradient clipping

⇒ Bouncing and unstable optimization

In all cases, possibility to loose long-term dependencies.

LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

UNIVERSITÉ Clermont Auvergne

Unfolded version- Backward pass

Computing the gradient w.r.t. h_0 involves

- many factors of $\boldsymbol{W}_{hh}$
- repeated gradient computation

Many high values

- Exploding gradients
- Gradient clipping

$\Rightarrow$ Bouncing and unstable optimization

Many small values

- Vanishing gradients

$\Rightarrow$ No gradient at all

In all cases, possibility to loose long-term dependencies.

BACKPROPAGATION THROUGH TIME

BPTT

- Basically chain rule as in classical backpropagation
- a bit more tricky, since gradients survive over time

Implementation

Already implemented, in Keras, using the classical `train` method.

What's next ?

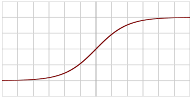
Limits

1.
   - x_1 goes t times through $tanh$.

   $\Rightarrow$ The influence of x_1 is much lesser than the one of x_t

   - W is shared: the weight associated to h_1 doesn't compensate this loss of memory

   $\Rightarrow$ RNN → short memory (the deeper the shorter)
2. Vanishing/exploding gradients



Limits

Some alternatives, improvments: LSTM, GRU... See next lecture !

LIMOS LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

cnrs

UCA UNIVERSITÉ Clermont Auvergne