MSE | MASTER OF SCIENCE IN ENGINEERING

Machine Learning

T-MachLe

13. Recurrent Neural Networks

Andres Perez Uribe
Jean Hennebert

Plan

13.1 From feed-forward to recurrent networks

13.2 Time-delay-Neural Networks (TDNN)

13.3 Backpropagation through time (BPTT)

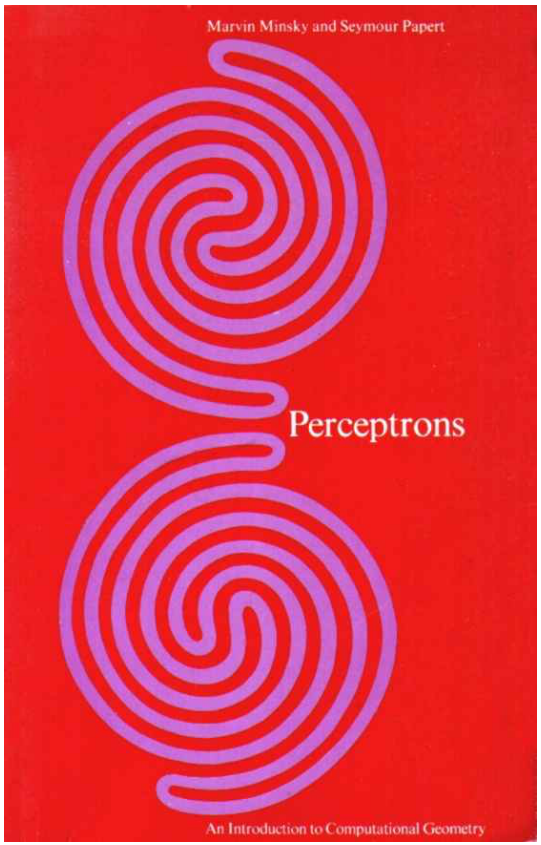
13.4 The vanishing gradients problem

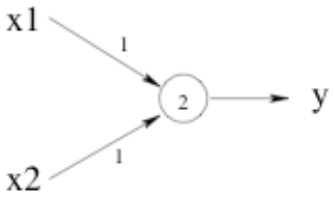
13.5 Long short-term Memory (LSTM) networks

13.6 Applications

Practical Work 13

Threshold logic



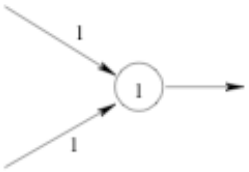


ET

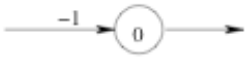
x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

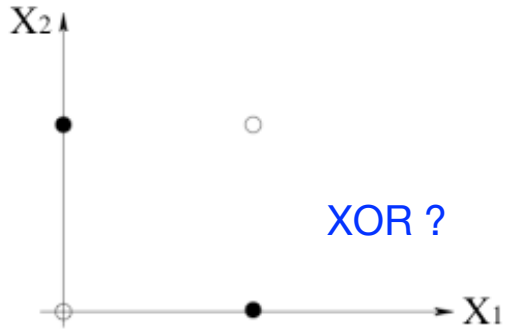
y { ○ : 0
● : 1



OU



NON

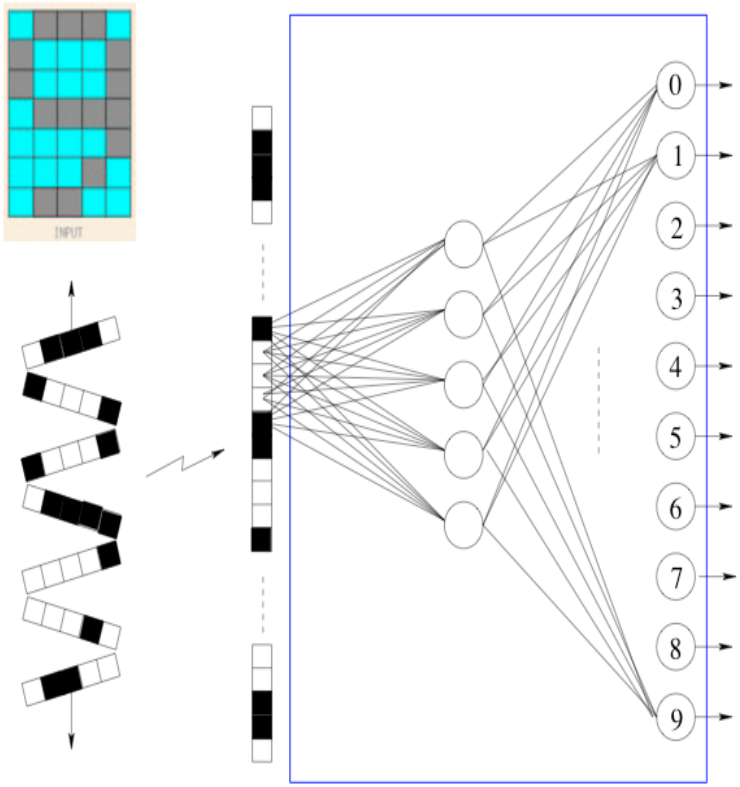


XOR ?

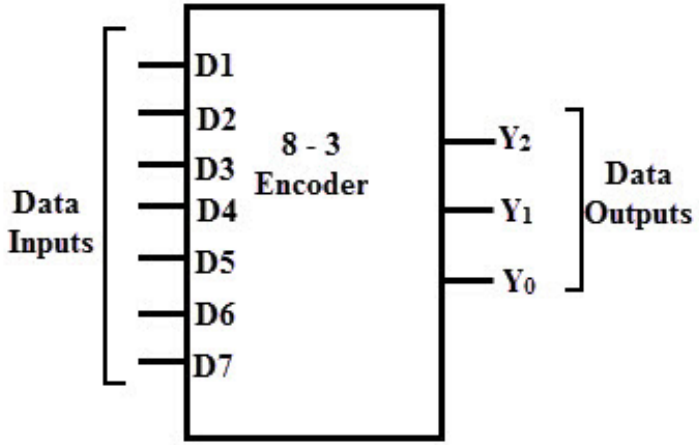
Minsky & Pappert, 1969

Combinational behavior of an MLP

Combinational refers to systems whose output is a function of the present value of the inputs only



MLP



Encoder

Taking care of sequences

Lots of information that we store in our brains is not random access, because they were learned as a sequence. Examples:

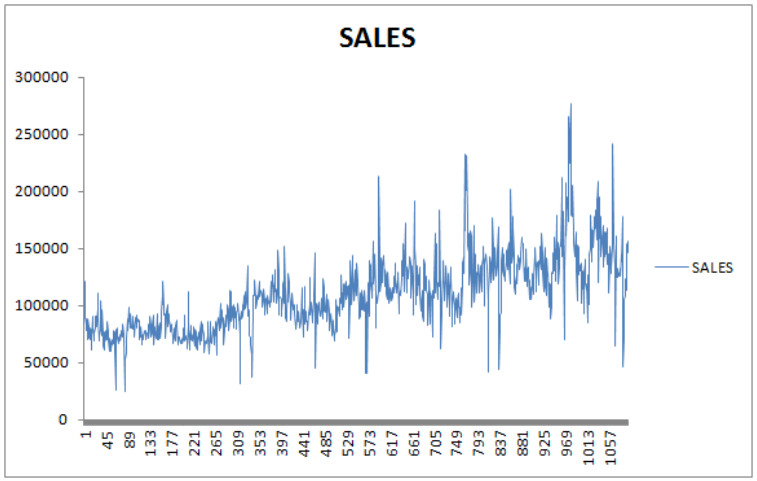
- Try to list the alphabet backwards
- Try to list the musical notes in an octave backwards
- Try to say your phone number backwards
- Try to sing the lyrics of a song starting the in the middle of a verse

How can we incorporate this into a Machine Learning algorithm ?

Taking care of sequences (2)

Lots of data is of temporal nature and generally does not change in an abrupt manner. Examples:

- The mean temperature of a day (e.g., it depends on seasonality, weather conditions, etc)
- Financial indicators
- Heart frequency



How can we deal with this sort of data using Neural Networks ?

Examples of sequences

DO, RE -> MI
RE, MI -> FA
MI, FA -> SOL
FA, SOL -> LA
SOL, LA -> SI
LA, SI -> DO

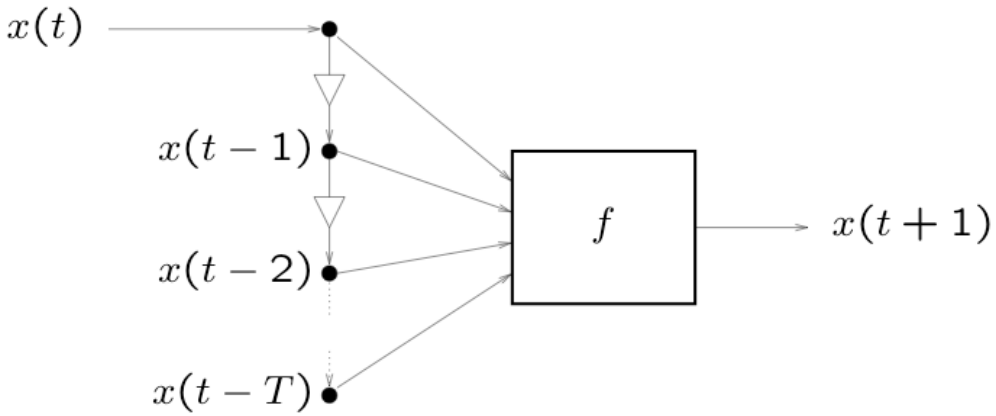
Songs:
DO,RE,MI -> DO
DO,DO,RE -> DO

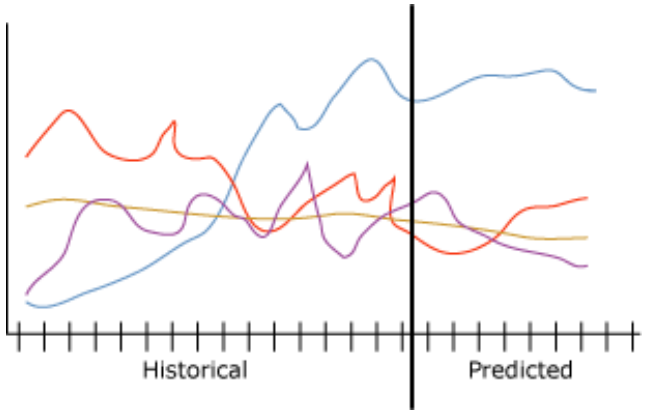
chatbot:
Comment tu vas ? -> bien
Comment tu t'appelle ? -> Pepper

CHF vs € (t, t-1,.. t-n) -> CHF vs € (t+1)

TDNN

Time series: A series of data points given in time order.
Problem: predict future values from historical ones.
Given {x(t), x(t-1), … x(t-n)} predict x(t+s), where s is the time horizon.

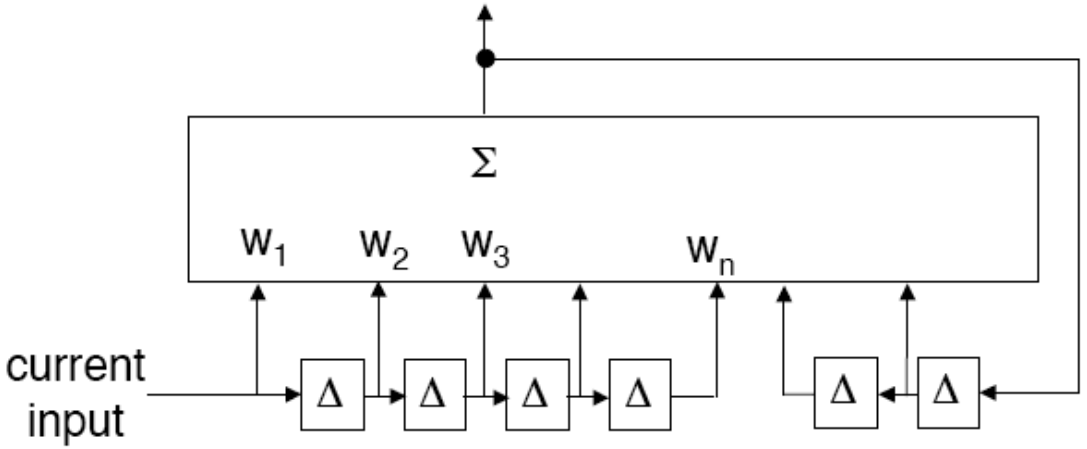




if f is an MLP we talk about Time-Delay Neural Networks

DTRNN

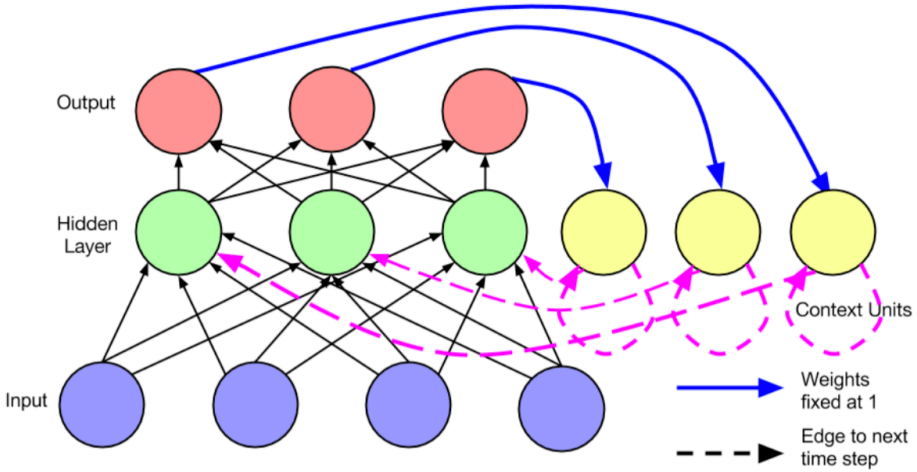
Discrete Time Recurrent Neural Network



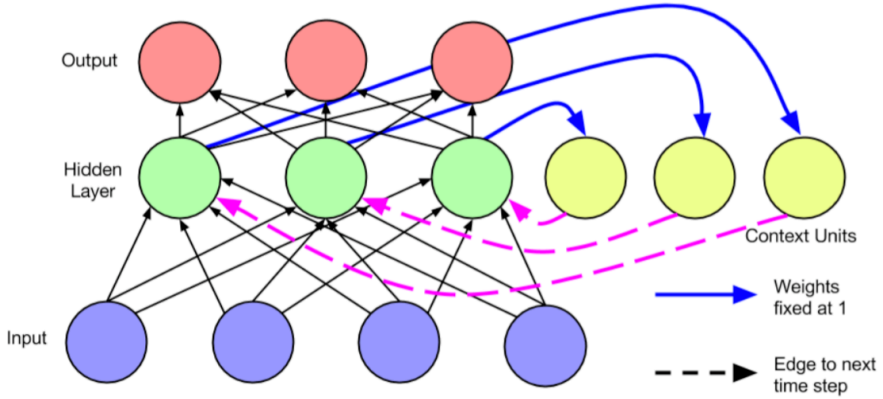
Memorize the previous state of the network and use it with time-delay inputs to compute the predicted value

Elman and Jordan Networks

Memorize previous output (Jordan) or hidden (Elman) values and use them in the following iteration. Trains as a feed-froward network using Backpropagation



Jordan Networks (1986)

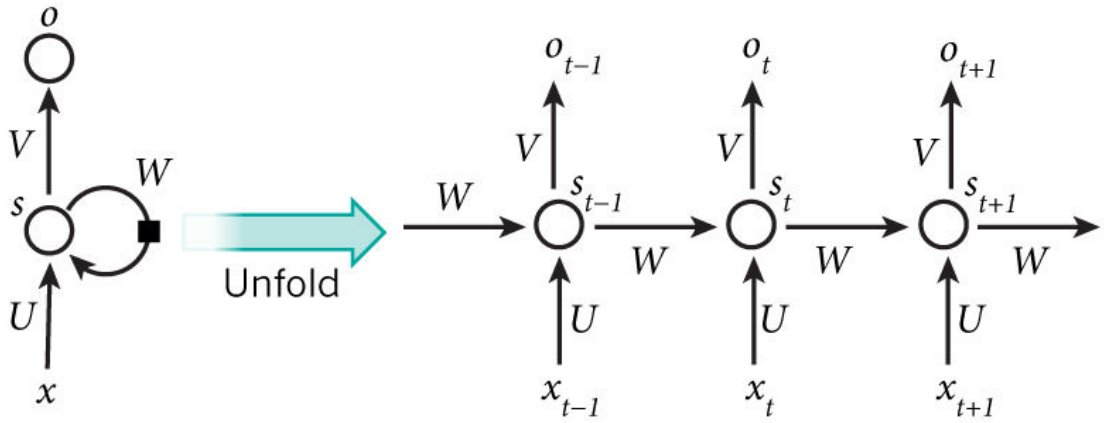


Elman Networks (1990)

Lipton et al., “A Critical Review of Recurrent Neural Networks for Sequence Learning.”, http://arxiv.org/abs/1506.00019

Recurrent Neural Networks

- The training is similar to that of a feed-forward network, but each epoch must run through the observations in sequential order.

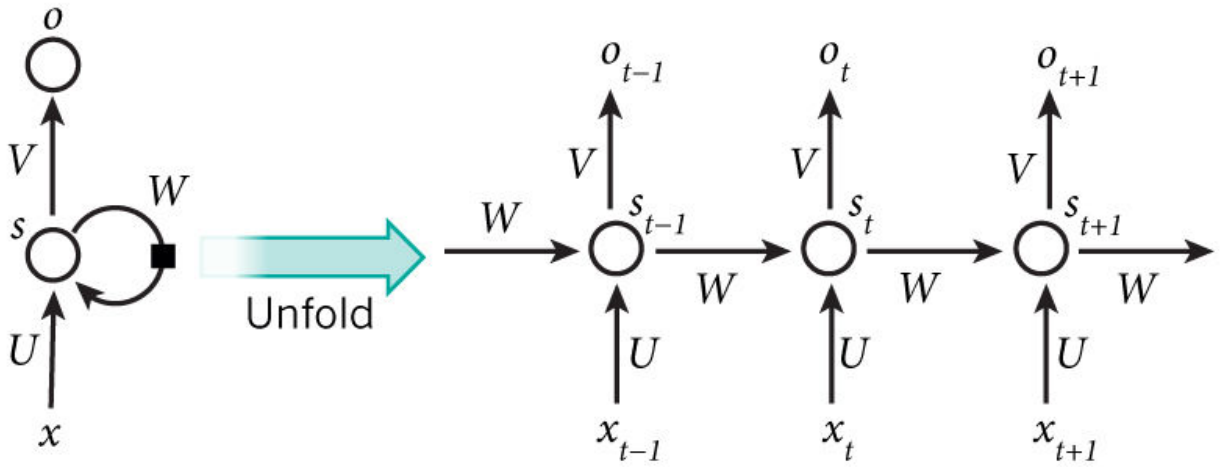


- Supposing that the network has been unfolded to a depth of k=3, each training pattern consists of [x(t-1), x(t), x(t+1) ; O(t+1)]

- Given an error function $E(O_{t+1},\hat{O}_{t+1})$, the objective is to find W and U by using gradient descent to minimize E.

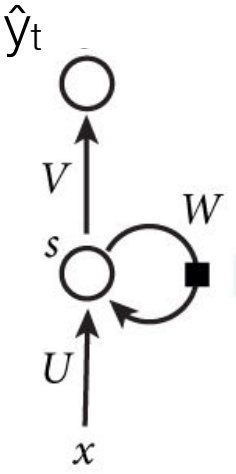
Weight sharing in time

The final learned weights U and W are the same when we unfold the computation performed by the recurrent unit in time:



Back propagation through time (BPTT)

Task: given x(t-n) … x(t-1),x(t),x(t+1) -> predict y(t+1),y(t+2)…y(t+m)



$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$\hat{y}_t = \text{softmax}(Vs_t)$$

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

y_t is the correct output at time t, and $\hat{y}_t$ is the predicted one. We want to predict a full sequence of y_t's. That full sequence is treated as one training example, thus we must sum up the errors.

Just like we sum up the errors, we also sum up the gradients at each time step for one training example:

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

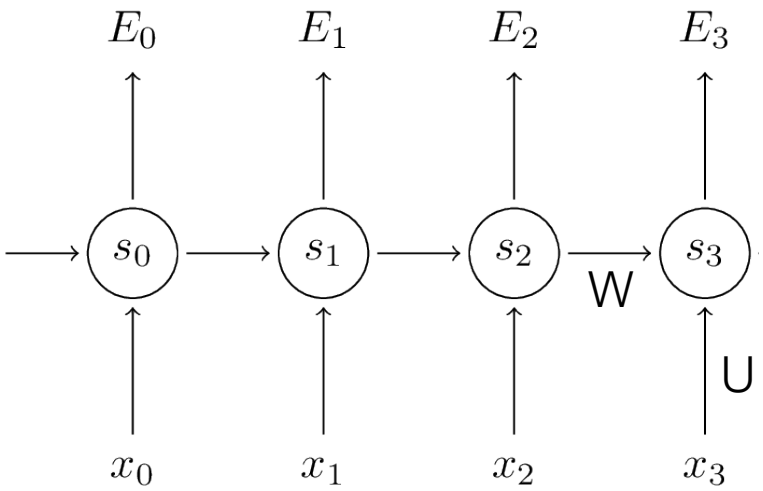
from http://www.wildml.com

BPTT (2)

Suppose a “simple” task: given x(t-n) … x(t-1),x(t),x(t+1) -> predict y(t+1)

….at a given moment: learn to predict y_3 based on x_3, x_2, x_1 and x_0

$E_3 = E(y_3, \hat{y}_3)$



$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

$$\hat{y}_t = \text{softmax}(V s_t)$$

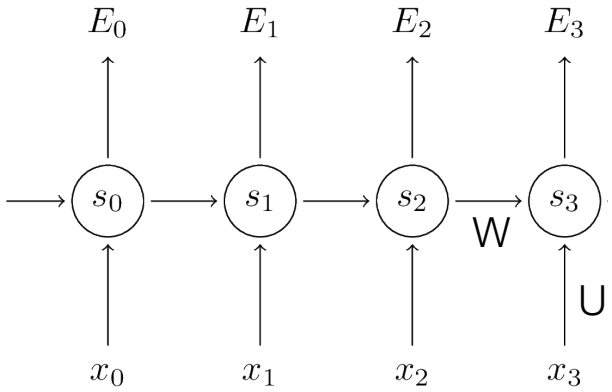
$$s_t = \tanh(U x_t + W s_{t-1})$$

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

from http://www.wildml.com

BPTT (3)

We want to compute: $$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial W}$$

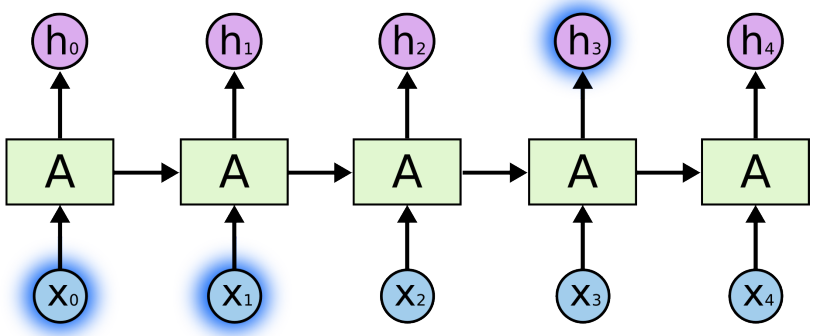


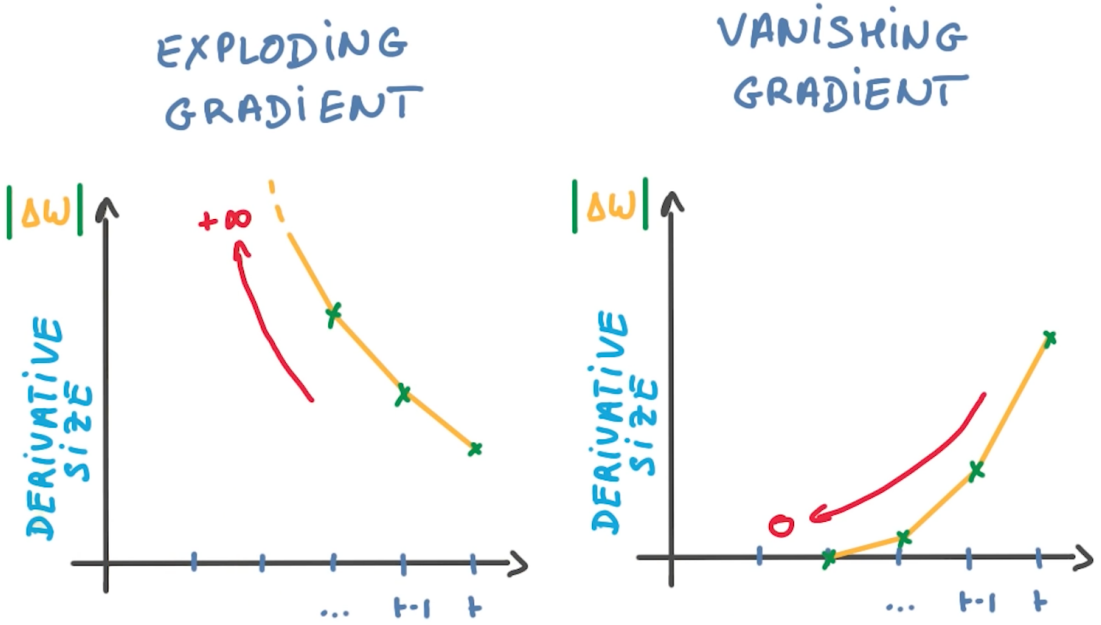
Now, note that s_3 is a function of s_2, which depends on W and s_1, and so on: $s_3 = \tanh(Ux_3 + Ws_2)$, $s_2 = \tanh(Ux_2 + Ws_1)$, etc…

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

Vanishing/exploding gradients

Gradient contributions from “far away” steps become zero, and the state at those steps doesn’t contribute to what you are learning: we end up not learning long-range dependencies.



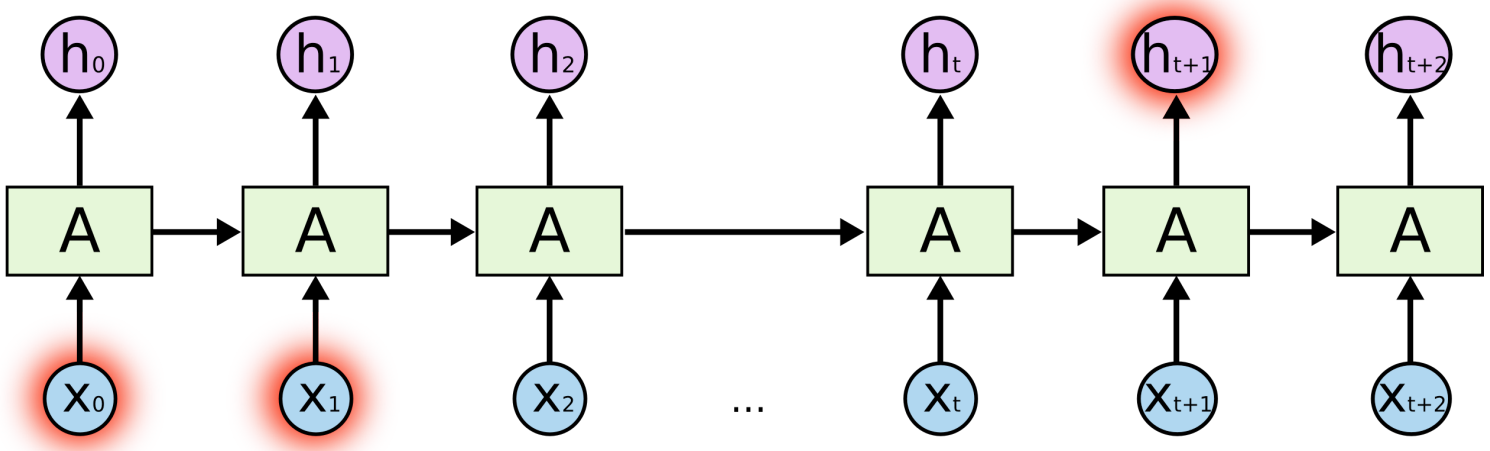


BPTT works for short-time scale dependencies between inputs and outputs

from http://www.wildml.com

Key problem

- Learning long-term dependencies is hard



DO -> RE
DO, RE -> MI
DO, RE -> ___ , ___ , ___ , _?_

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Towards modern RNNs

Juergen Schmidhuber, IDSIA

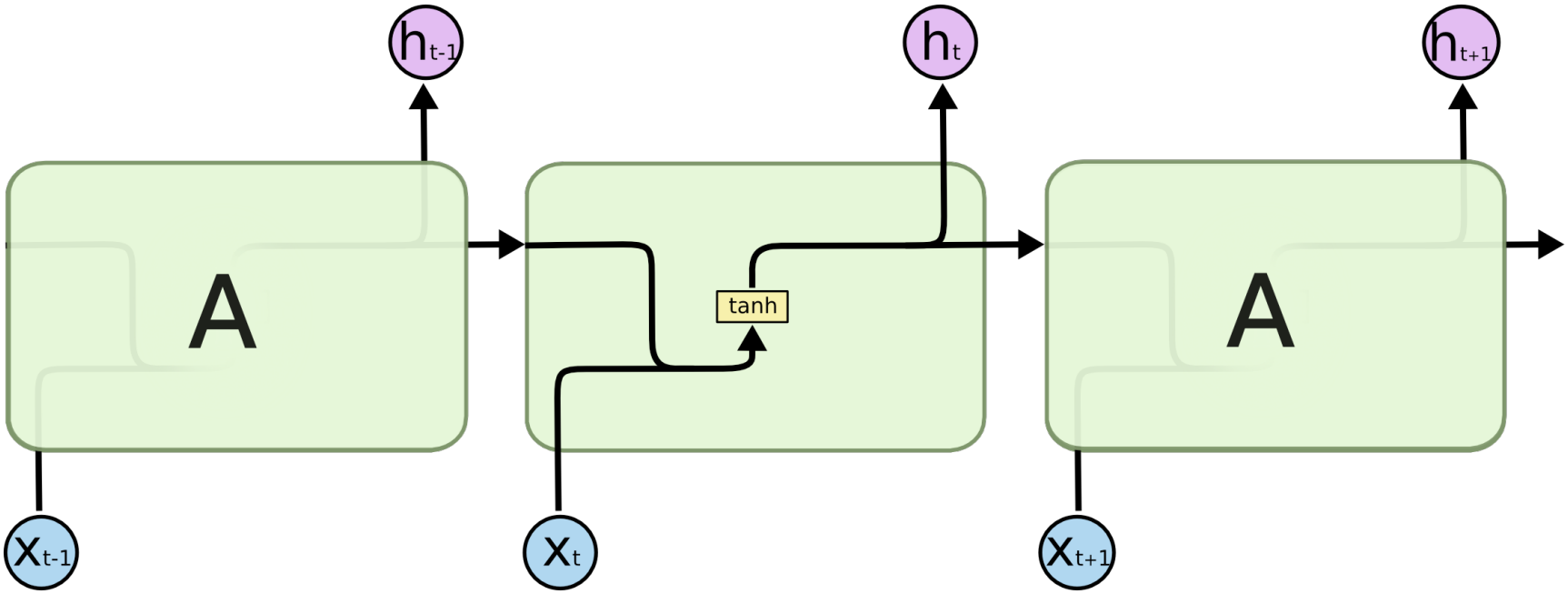
Avoiding vanishing gradients:

- Instead of multiplying the previous hidden state by a matrix to get the new state, add something to the old hidden state to get the new one.
- Gate all operations to store only what is useful and to retrieve only what can help predicting the target output.

Memory cells with a self-connected recurrent edge (Hochreiter & Schmidhuber, 1997)
Possibility of forgetting (Gers, Schmidhuber, Cummins, 1999)

Standard Recurrent Neural Network

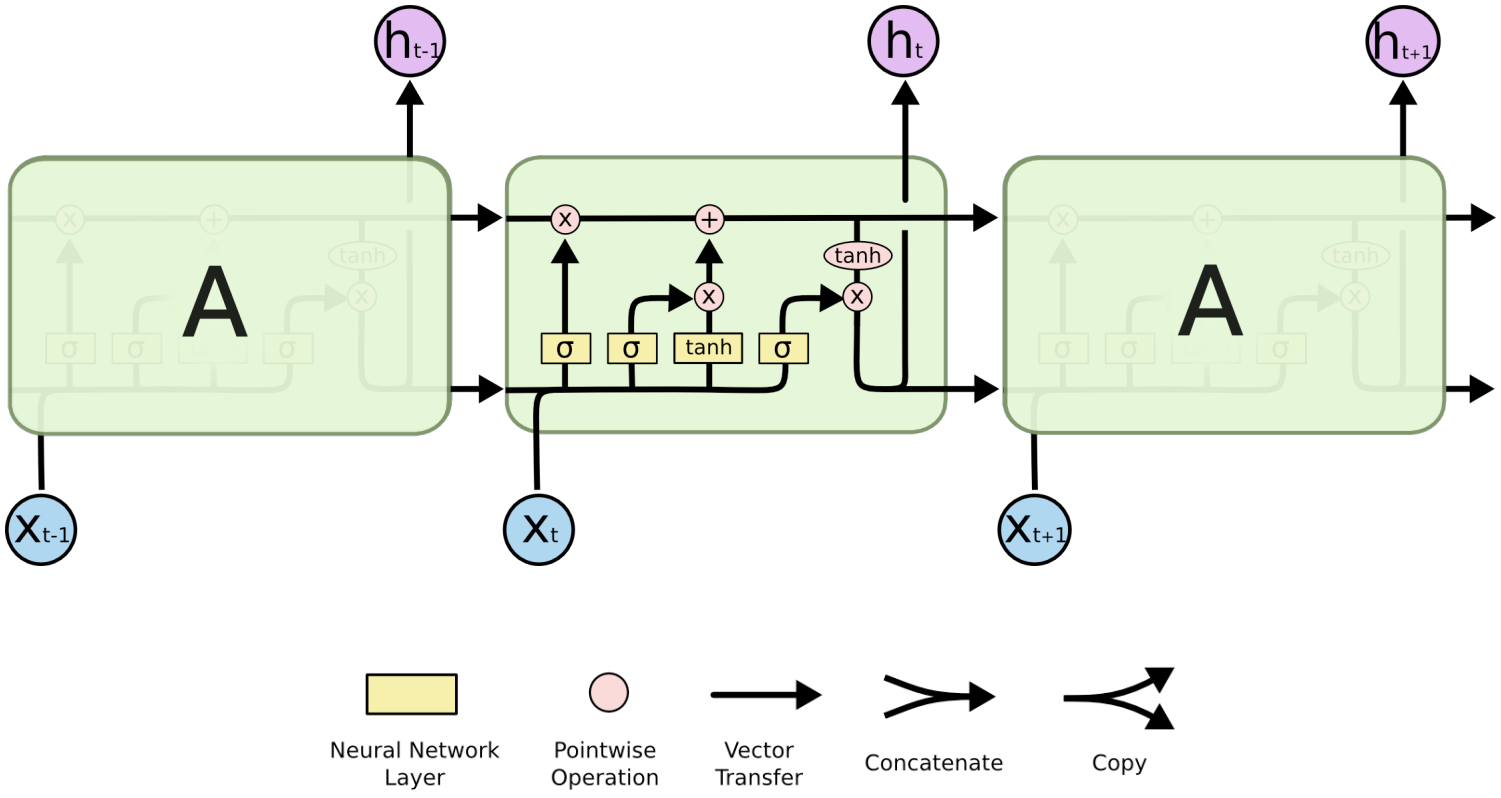
- All recurrent neural networks have the form of a chain of repeating modules of neural network. Here a single layer of tanh functions.



From Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

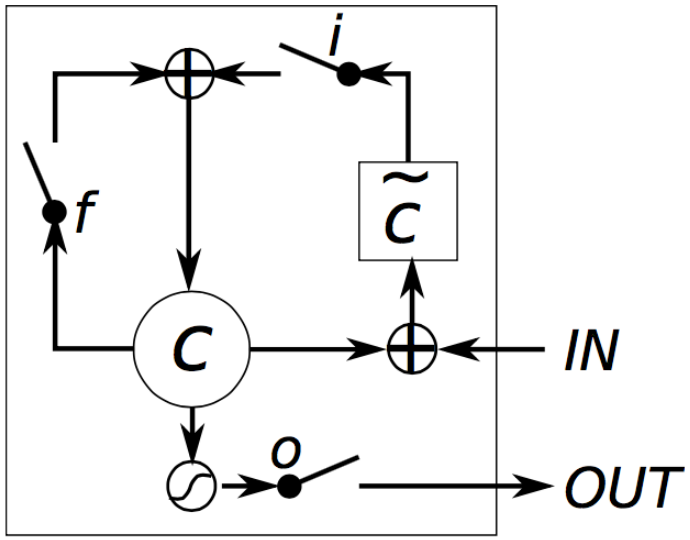
Long Short Term Memory Networks (LSTM)

- An LSTM is a RNN composed of four different layers



From Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

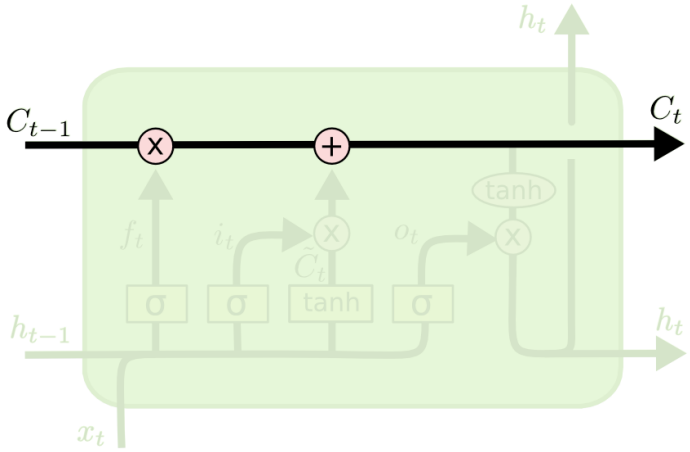
The LSTM building blocks

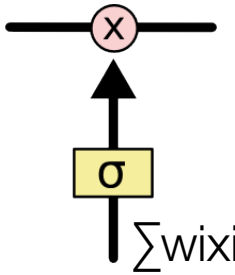


- There are three gates (i,f,o) controlled by “Perceptrons” weighing the inputs and the recurrent outputs
- Given an input IN, we compute a new state c’ that can:
  - be ignored (i = 0)
  - replace the previous state (f=0 and i=1)
  - be used to compute a new state C + c’ (f=1 and i=1)
- Given a state C, the network outputs OUT (o=1) or not (o=0)

LSTMs: Cell state or Memory

- The cell state is kind of like a conveyor belt.
- The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.





Gates are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,” while a value of one means “let everything through!”

From Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Language example

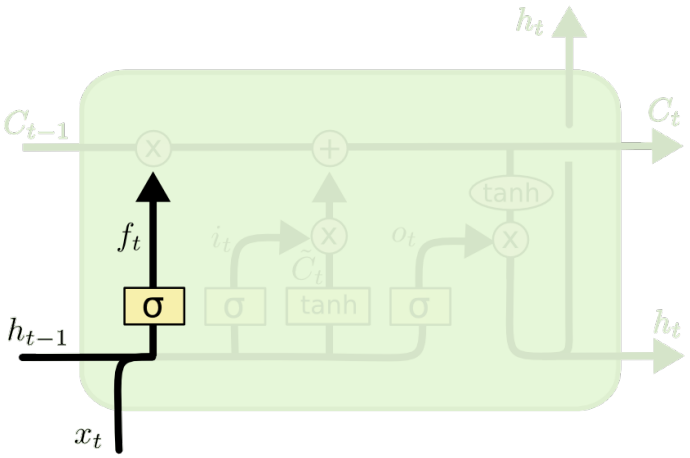
AdaLovelace was the daughter of LordByron. _____ met CharlesBabbage who was mathematician and inventor. _____ designed the first mechanical computer. She _____ interested in his invention and came up with the first computer program.

Objective: complete the sentences

What to memorize in the cell state (c) from the sequence of words (inputs) to decide what to output ?

LSTMs: Forget gate

- Should we continue to remember this “bit” of information or not ?
- A $f_t = 1$ represents “completely keep this” while a 0 represents “completely get rid of this.”



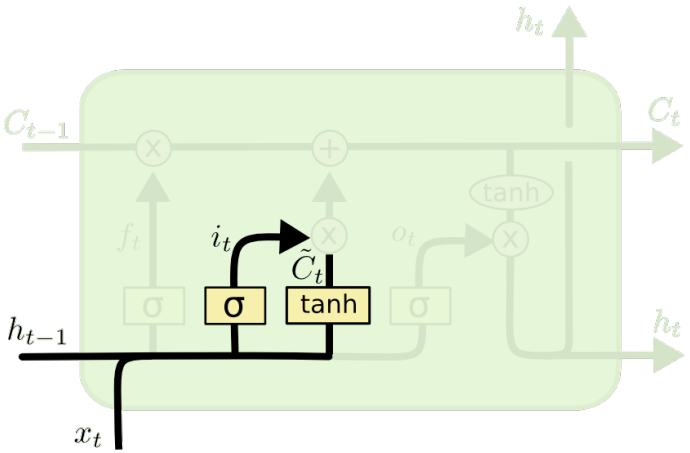
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Example: we want to predict the next word based on all the previous ones. In such a problem, the cell state might include the gender of the present subject, so that the correct pronouns (he or she) can be used. When we see a new subject in the input (x_t), we want to forget the gender of the old subject.

From Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs: Input gate

- Should we update this “bit” of information or not?
  - If so, with what?



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

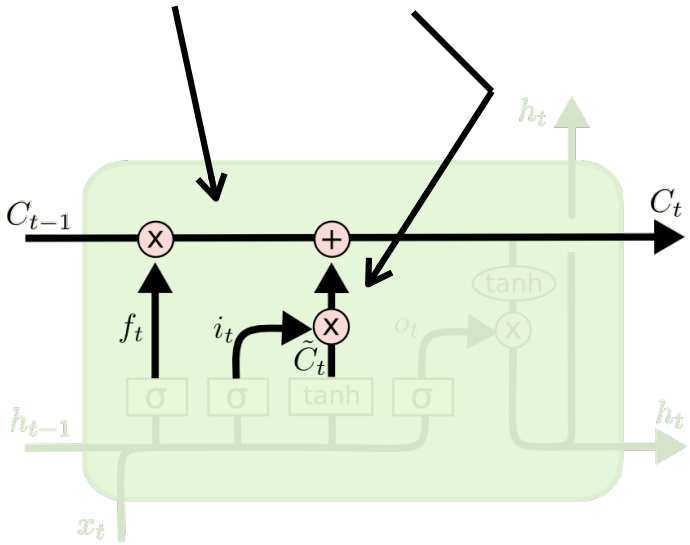
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Example: in the example of our language model, we want to add the gender of the new subject to the cell state, to replace the old one we’re forgetting.

From Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs: Memory update

- Forget that + memorize this



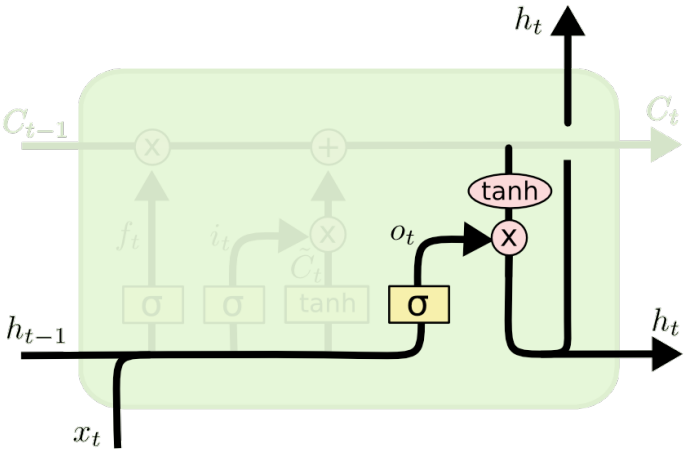
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Example: in the case of the language model, this is where we will actually drop the information about the old subject's gender and add the new one.

From Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs: Output gate

- What shall we output given the context ?
- The output is a function of the cell state (C_t). A tanh pushes the cell state values towards +1 or -1
- A « Perceptron » selects what to output based on the current input x_t and the previous output h_{t-1}, by multiplying its output o_t and tanh(c_t)



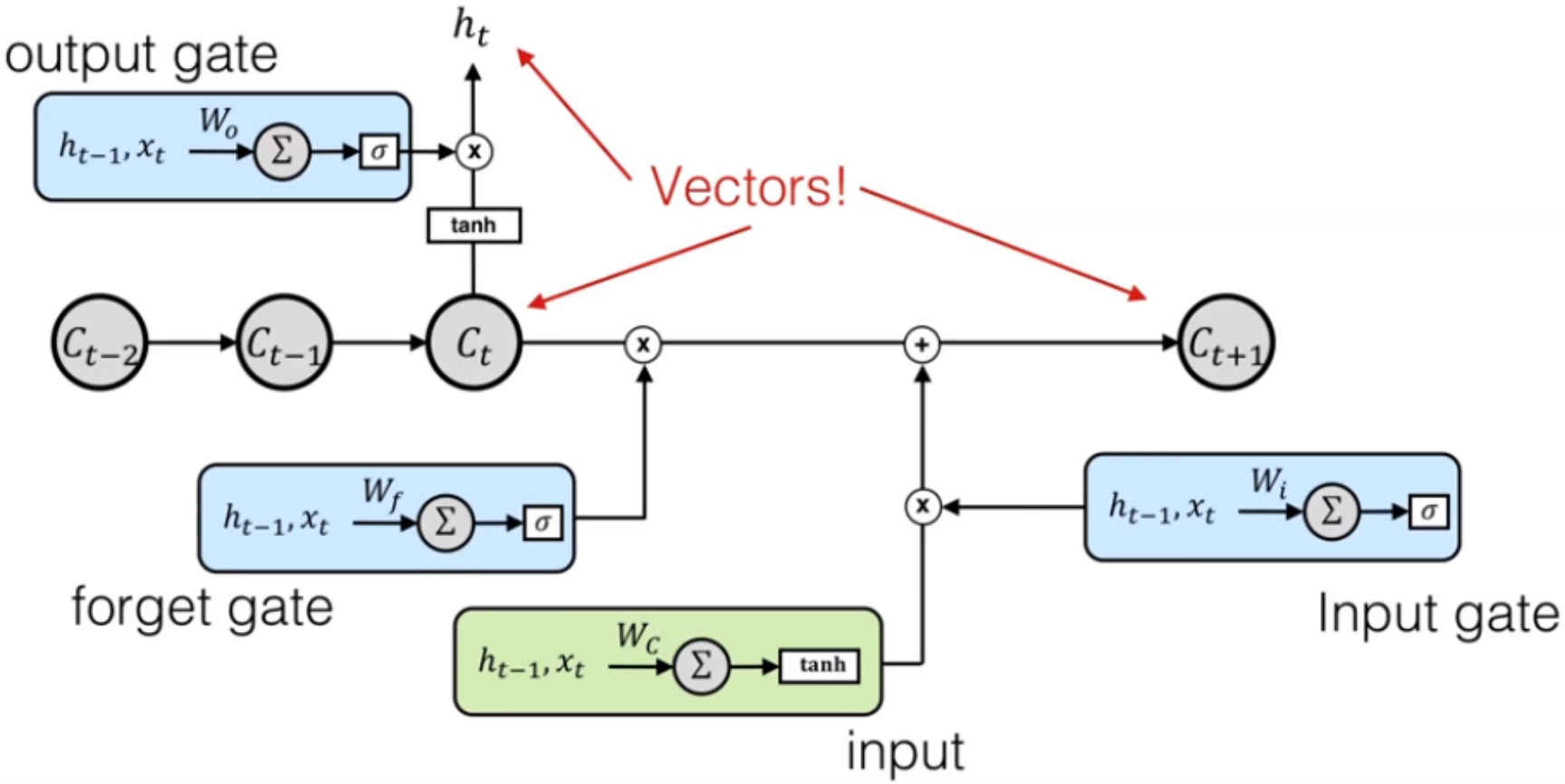
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

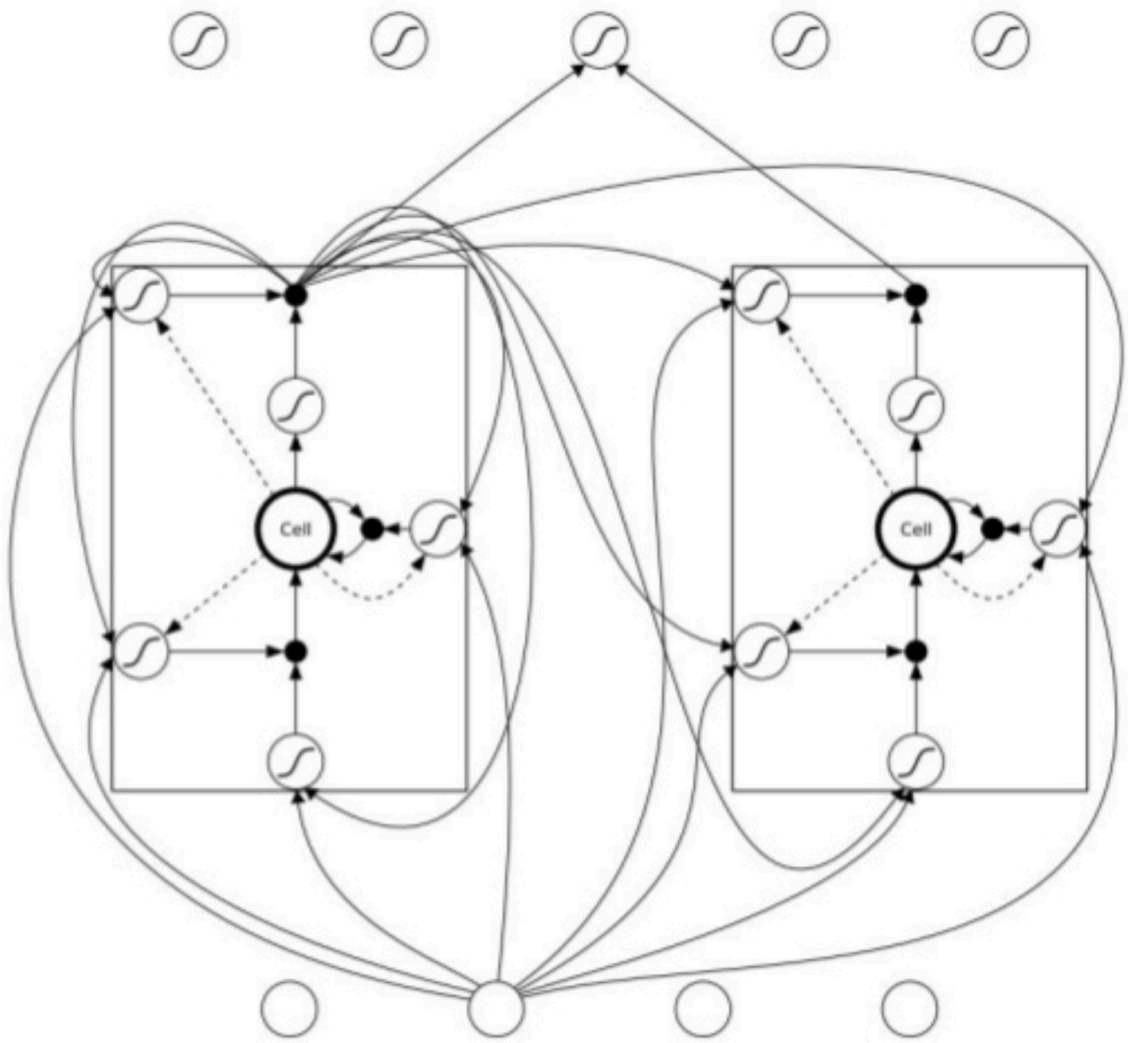
Example: in the case of the language model, given that we just saw a new subject, this cell might decide how to conjugate a verb, based on its knowledge of the subject, whether it is plural or singular.

From Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

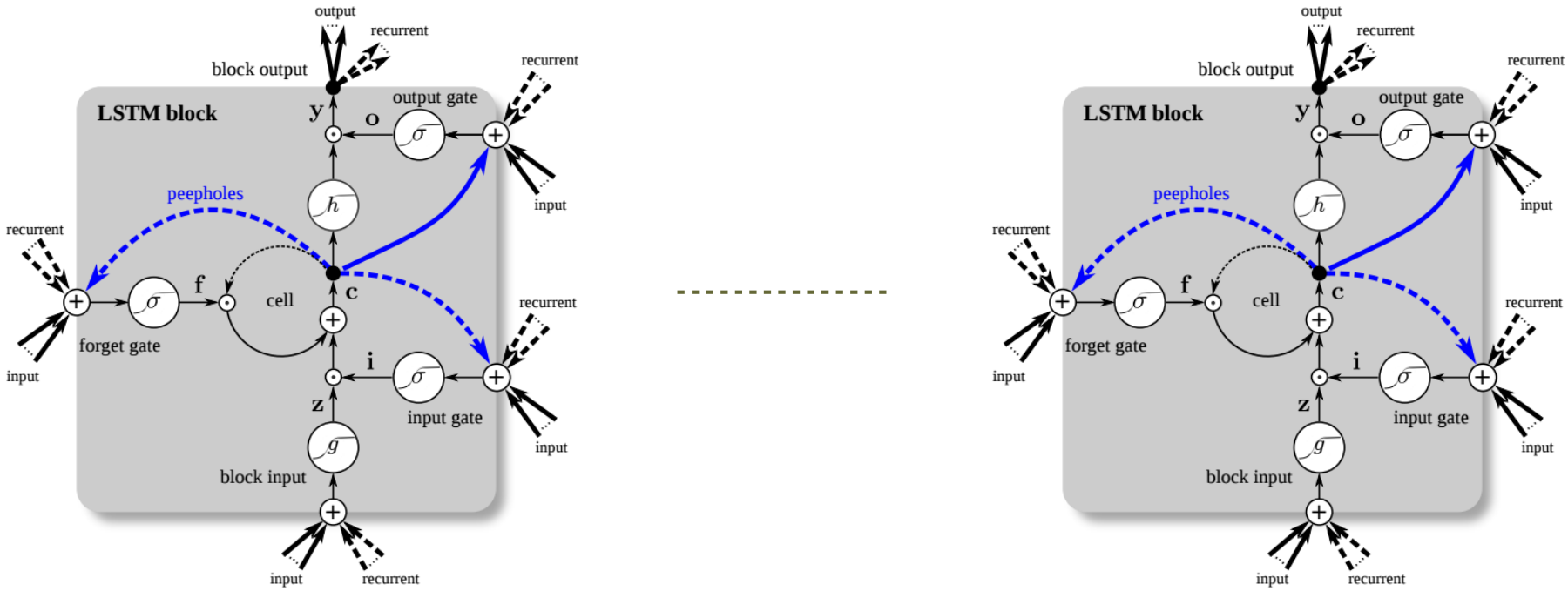
The LSTM building blocks



A two-unit LSTM network



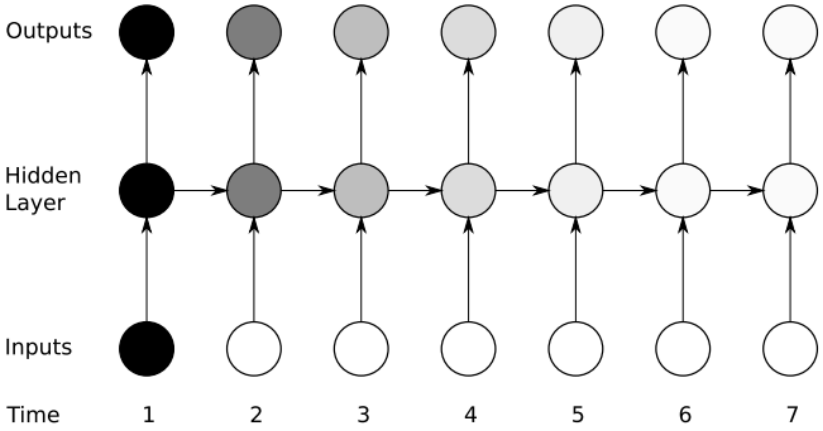
LSTM complexity



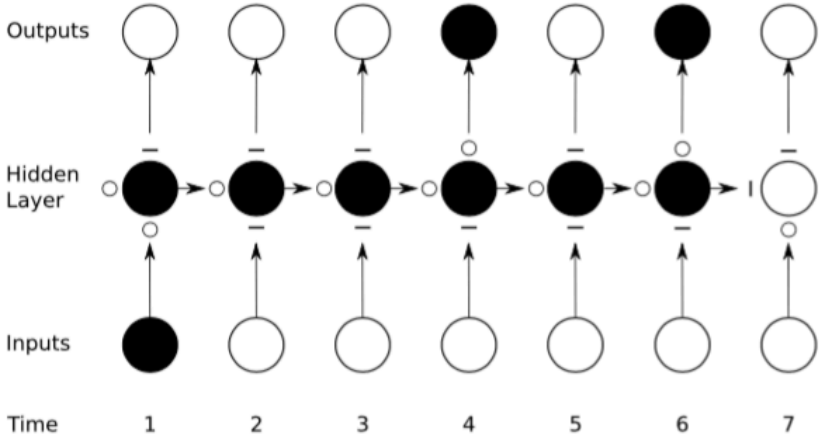
4 x (Ninputs + Nrecunits + bias) x Nrecunits

weights for 32 LSTM units & 2-dim inputs: 4 x (2 + 32 + 1) x 32 = 4480

LSTM vs RNNs



Input forgetting when looking for long-term dependencies in RNNs

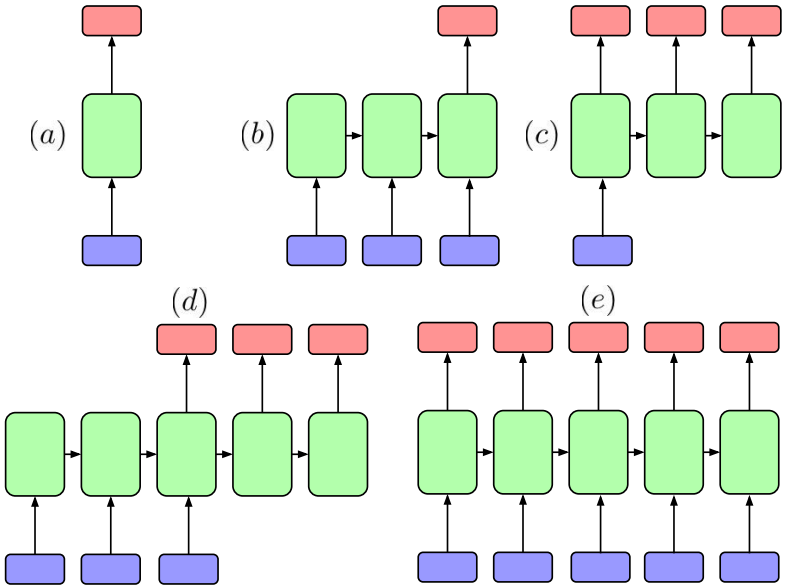


Memorized inputs for long-term dependencies in LSTMs

On the LSTM (right), the memory cell « remembers » the first input as long as the forget gate is open ('o') and the input gate is closed ('-'). The sensitivity of the output can be switched on and off by the output gate without affecting the rest of the cell.

from Graves, Alex, « Supervised sequence labelling », 2012

LSTM architecture & applications



a) Feed-forward network

b) Text and video classification: a sequence is mapped to one fixed length vector

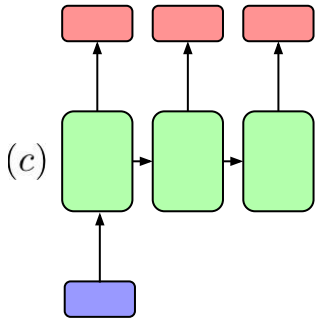
c) Image captioning: the input image is a single non-sequential data point.

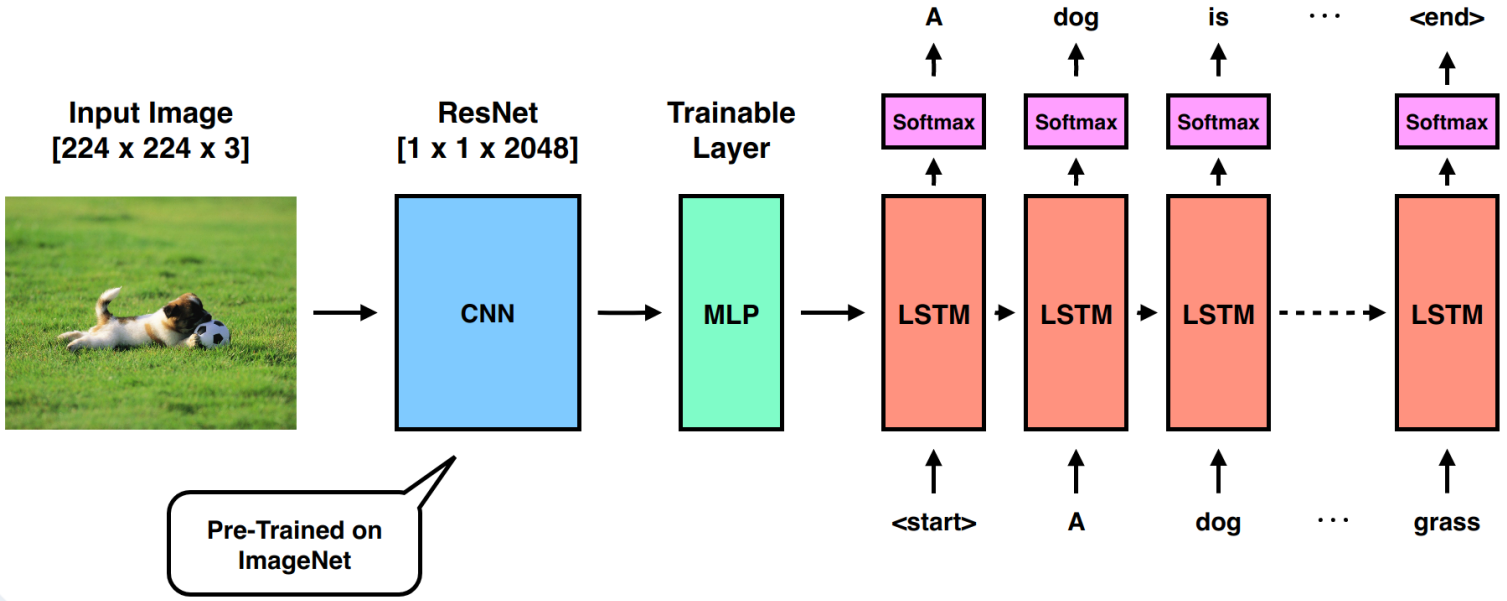
d) Natural language translation, a sequence-to-sequence task (they might have varying and different sizes)

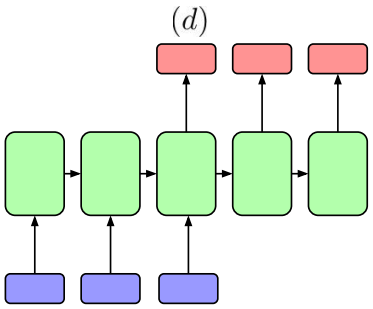
e) Learn a generative model for text, predicting at each step the following character.

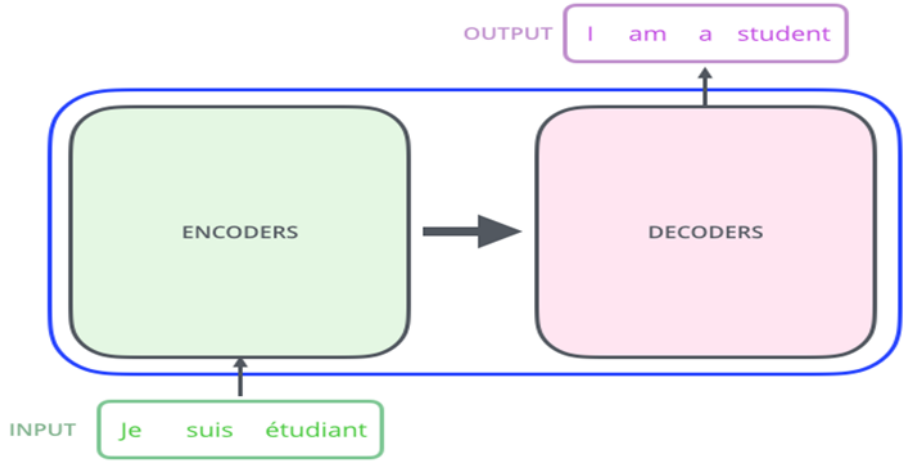
Lipton et al., “A Critical Review of Recurrent Neural Networks for Sequence Learning.”, http://arxiv.org/abs/1506.00019

Example applications

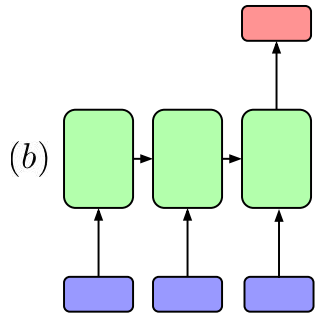




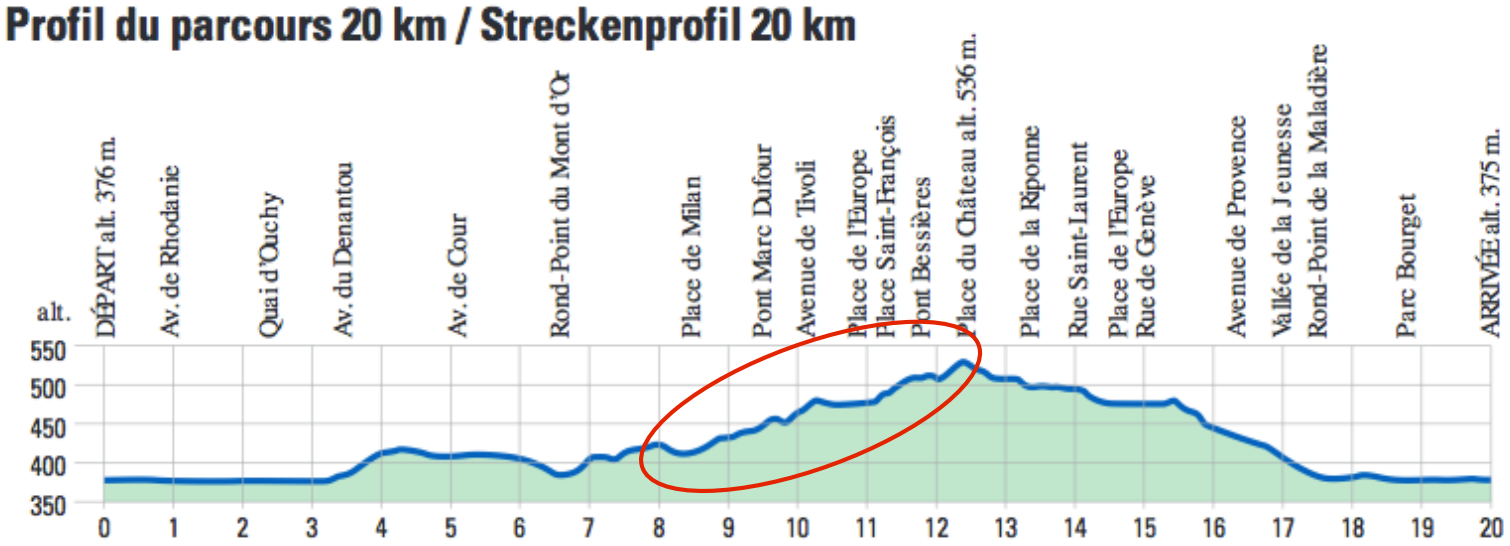




Runner speed prediction (case study)

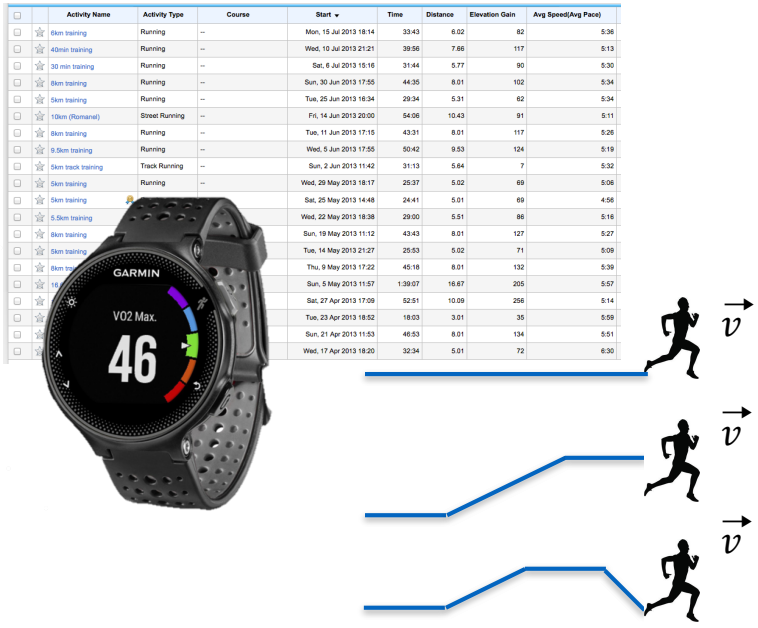


How to guide the beginner runner on uphill slopes ?

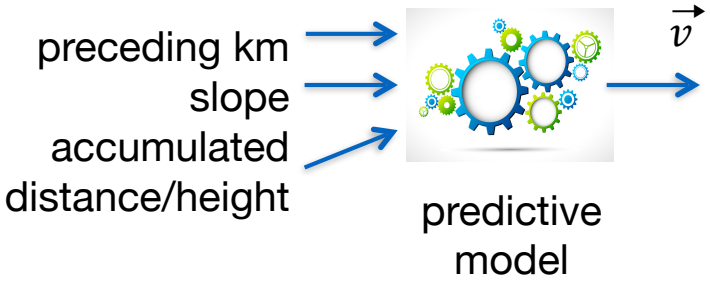


20km de Lausanne

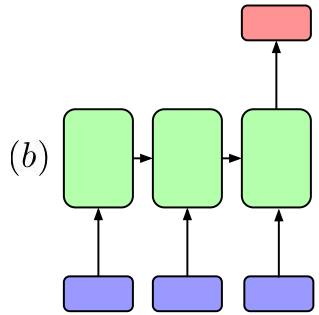
Runner speed prediction (2)



Runner training data

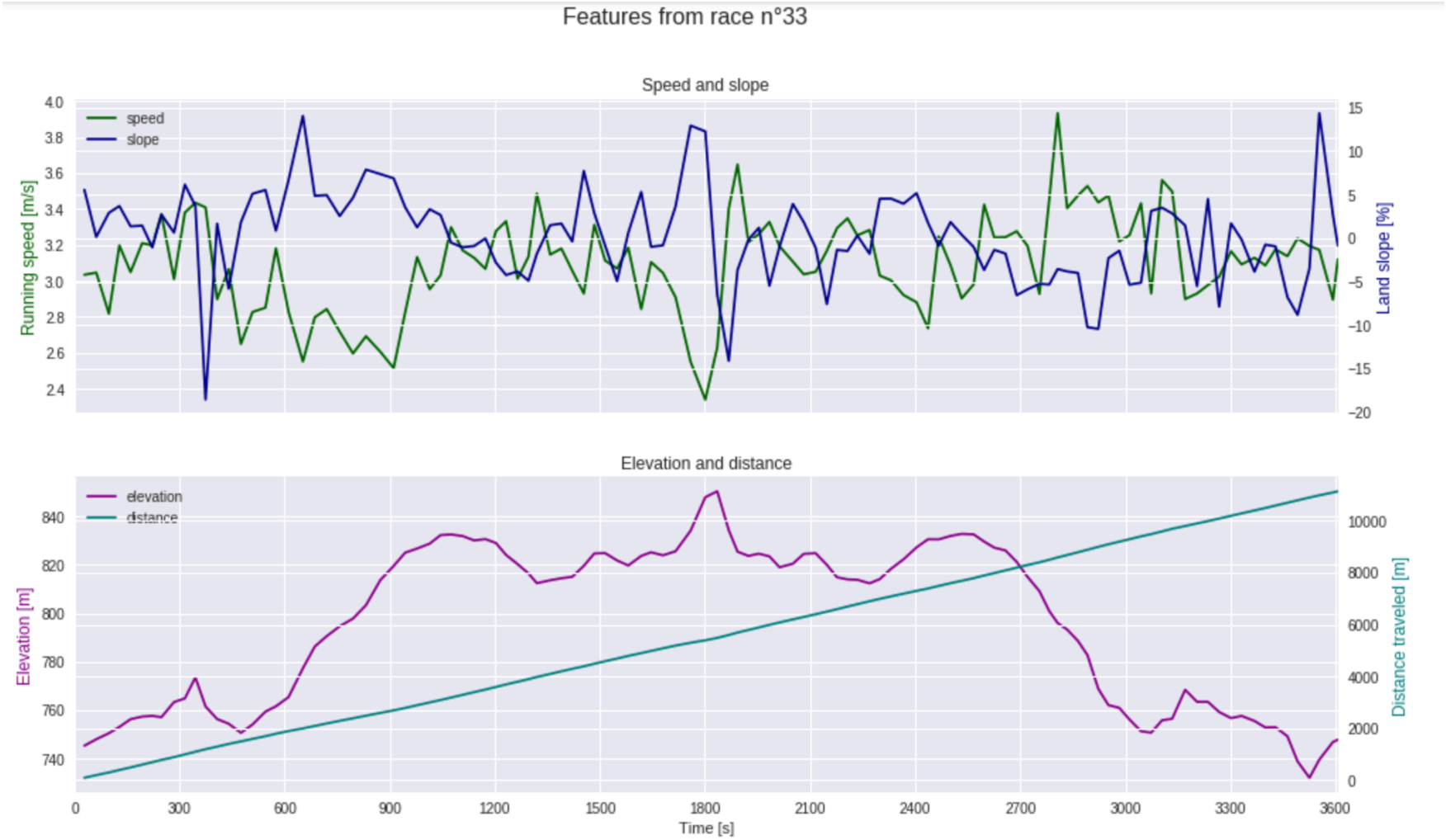


Runner speed prediction (3)

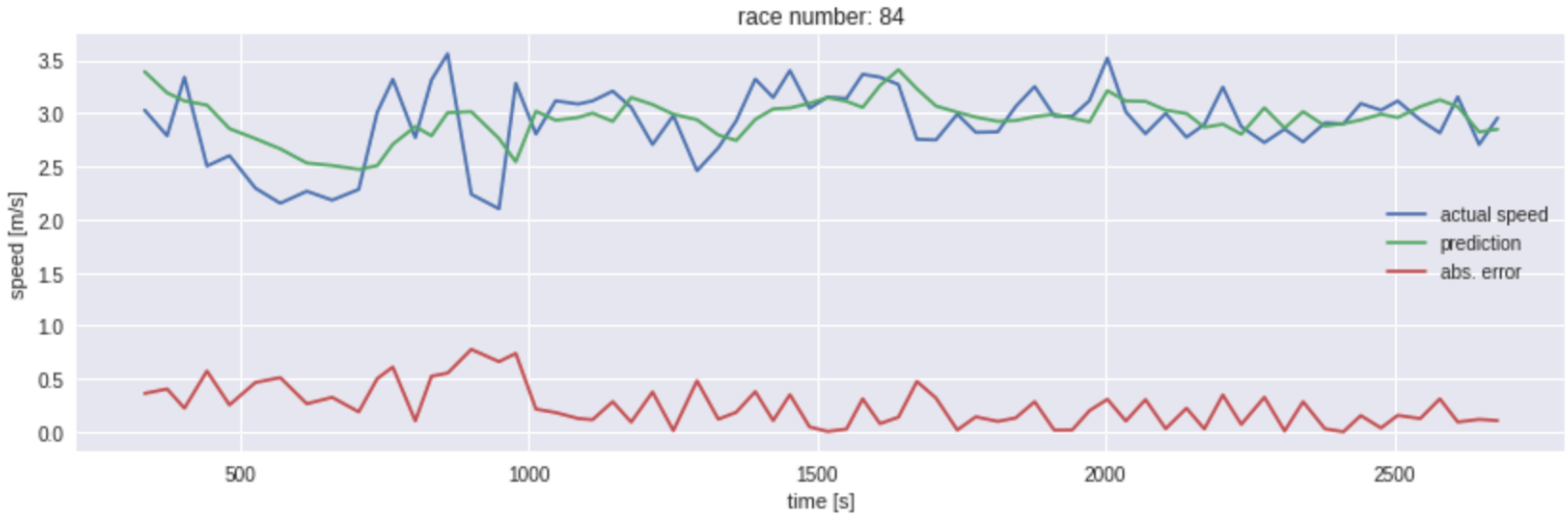


v(t-n)	….	v(t-2)	v(t-1)	v(t)	speed ?
t-n	….	t-2	t-1	t	next time
slope(t-n)	….	slope(t-2)	slope(t-1)	slope(t)	next slope

Typical training session



Speed prediction (example)



What is the race time prediction error ?