

Extra

Arthur Charpentier

UQAM

Actuarial Summer School 2019

RNN (recurrent neural nets)

RNN : Recurrent neural network

Class of neural networks where connections between nodes form a directed graph along a temporal sequence.

Recurrent neural networks are networks with loops, allowing information to persist.

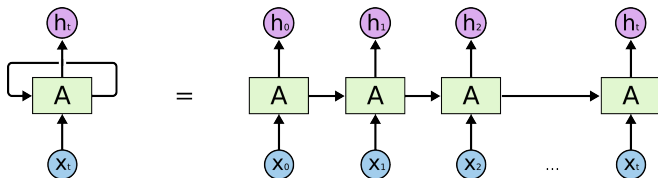
Classical neural net $y_i = m(\mathbf{x}_i)$

Recurrent neural net

$$y_t = m(\mathbf{x}_t, y_{t-1}) = m(\mathbf{x}_t, m(\mathbf{x}_{t-1}, y_{t-2})) = \dots$$

RNN (recurrent neural nets)

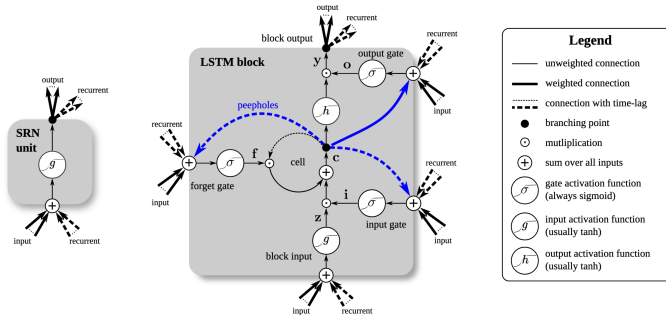
A is the neural net, h is the output (y) and x some covariates.



source <https://colah.github.io/>

See Sutskever (2017, [Training Recurrent Neural Networks](#))
From recurrent networks to LSTM

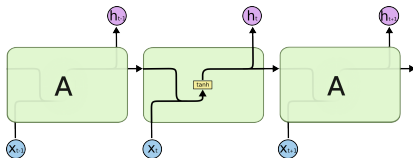
RNN (recurrent neural nets)



source Greff *et al.* (2017, [LSTM: A Search Space Odyssey](#))
see Hochreiter & Schmidhuber (1997, [Long Short-Term Memory](#))

RNN (recurrent neural nets)

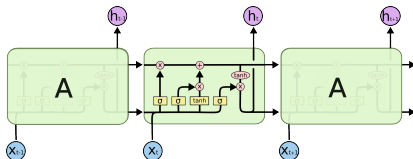
A classical RNN (with a single layer) would be



source <https://colah.github.io/>

"In theory, RNNs are absolutely capable of handling such 'long-term dependencies'. A human could carefully pick parameters for them to solve toy problems of this form. Sadly, in practice, RNNs don't seem to be able to learn them" see Benghio et al. (1994, [Learning long-term dependencies with gradient descent is difficult](#))

RNN (recurrent neural nets)



“RNNs can keep track of arbitrary long-term dependencies in the input sequences. The problem of “vanilla RNNs” is computational (or practical) in nature: when training a vanilla RNN using back-propagation, the gradients which are back-propagated can “vanish” (that is, they can tend to zero) “explode” (that is, they can tend to infinity), because of the computations involved in the process” (from [wikipedia](#))

RNN (recurrent neural nets)

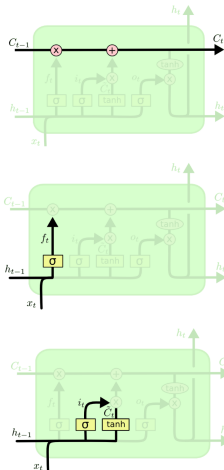
C is the long-term state

H is the short-term state

forget gate: $f_t = \text{sigmoid}(\mathbf{A}_f[h_{t-1}, x_t] + b_f)$

input gate: $i_t = \text{sigmoid}(\mathbf{A}_i[h_{t-1}, x_t] + b_i)$

new memory cell: $\tilde{c}_t = \tanh(\mathbf{A}_c[h_{t-1}, x_t] + b_c)$



RNN (recurrent neural nets)

final memory cell: $c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t$

output gate: $o_t = \text{sigmoid}(\mathbf{A}_o[h_{t-1}, x_t] + b_o)$

$h_t = o_t \cdot \tanh(c_t)$

