Sequence models

Attila Bagoly

Applications

- Speech recognition
  - Input: sequence of pressure values
  - Output: sequence of words
- Music generation
  - Input: 0
  - Output sequence of notes
- Sentiment classification
  - Input: sequence of words
  - Output: rating (1-5)
- Machine translation
  - Input: sequence of words
  - Output: sequence of words
- Video activity recognition, summarization, etc.:
  - Input: sequence of pictures
  - Output: labels, sequence of words, etc.
- DNA sequence analysis:
  - Input: AGCCCT…
  - Output: eg. sequence of A,G,C,T which corresponds to a protein

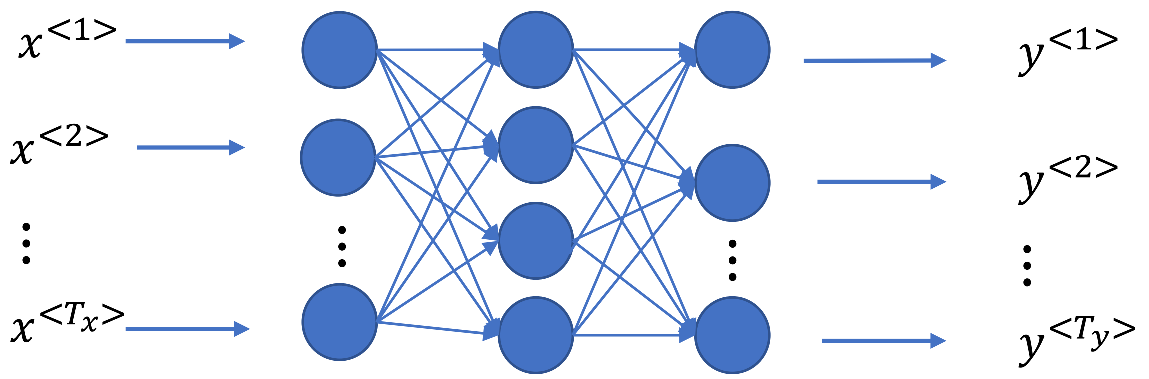
Representation

- **x**: Hello World
  $x^{<1>}$ $x^{<2>}$
- Or character level: $x^{<1>} = h,\ x^{<2>} = e,\ x^{<2>} = l$ …
- Vocabulary: list of words or characters:

$$\begin{bmatrix} a \\ b \\ \vdots \end{bmatrix} \text{ or } \begin{bmatrix} a \\ and \\ \vdots \end{bmatrix}$$

- We encode the characters or words with the position index in the vocabulary (after pre-processing)
- Input: one-hot encoded indexes or it's lower dimensional embedding
- Song: vocabulary of notes

Neural network

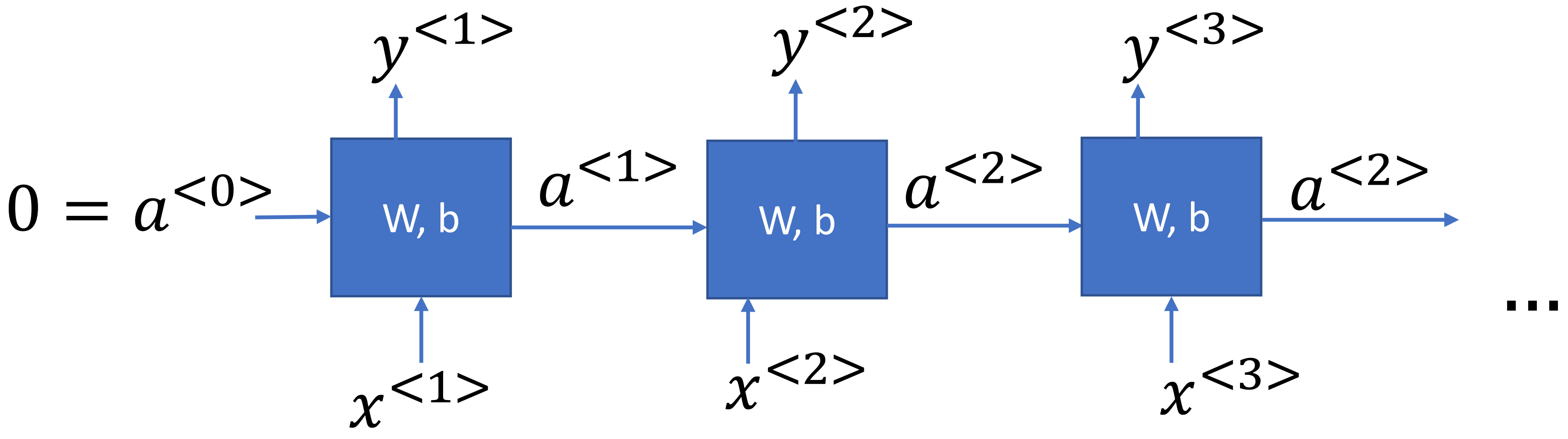


- Different input and output sizes:
  - Text sizes will vary example by example
  - Find the maximal text, pad with zeros the others (not too good)
- Bigger problem: doesn't share learned features across different positions of the text

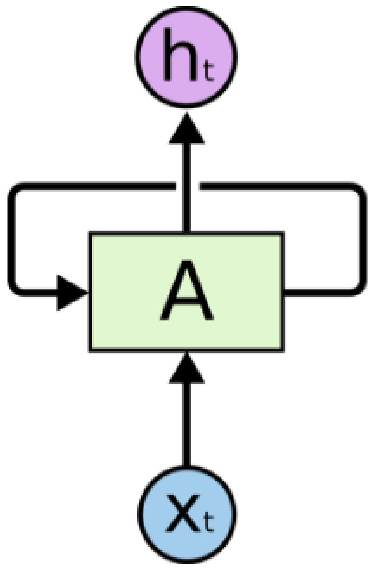
Apple is good for your health. Apple juice is cheap.

You learned in position 1 that apple is a fruit (found some weights). In position 7 you can't use the weights from position 1, you would have to learn the same weights again.

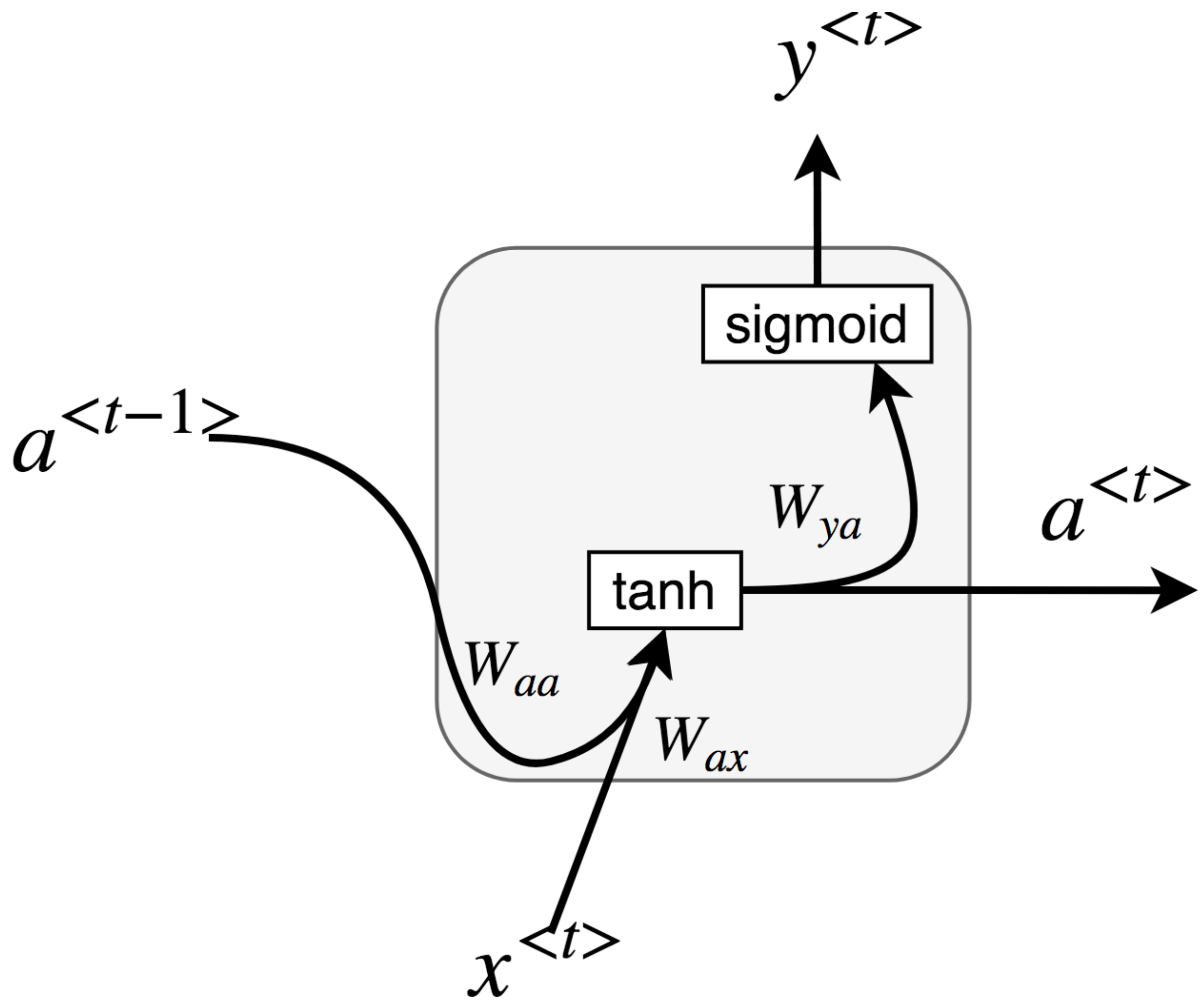
Recurrent neural networks (RNNs)



- $a^{<1>} = g(W_{aa}a^{<0>} + W_{ax}x^{<1>} + b_a)$
- $y^{<1>} = g\left(W_{ya}a^{<1>} + b_y\right)$
- $a^{<2>} = g(W_{aa}a^{<1>} + W_{ax}x^{<2>} + b_a)$
- $y^{<2>} = g\left(W_{ya}a^{<2>} + b_y\right)$
- …

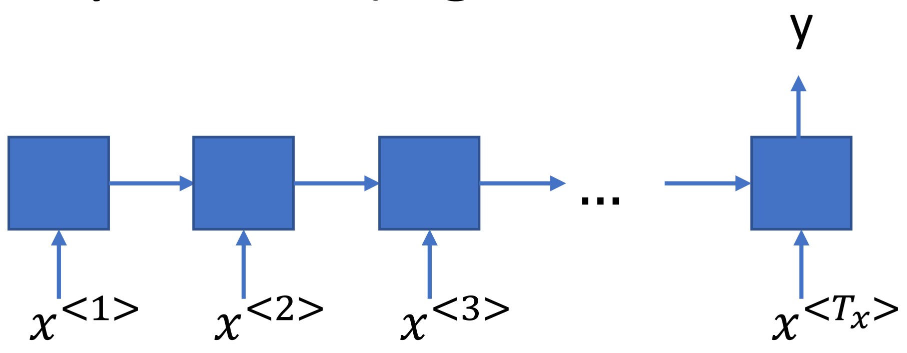


RNN cell

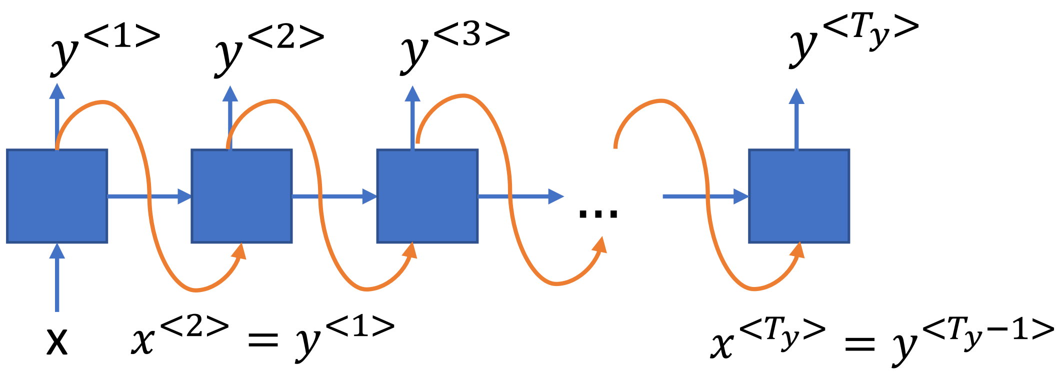


RNN architectures

- Many-to-one (e.g. sentiment classification)

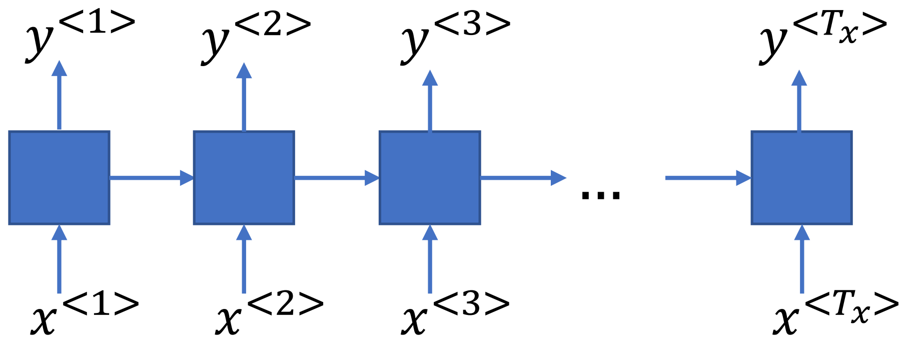


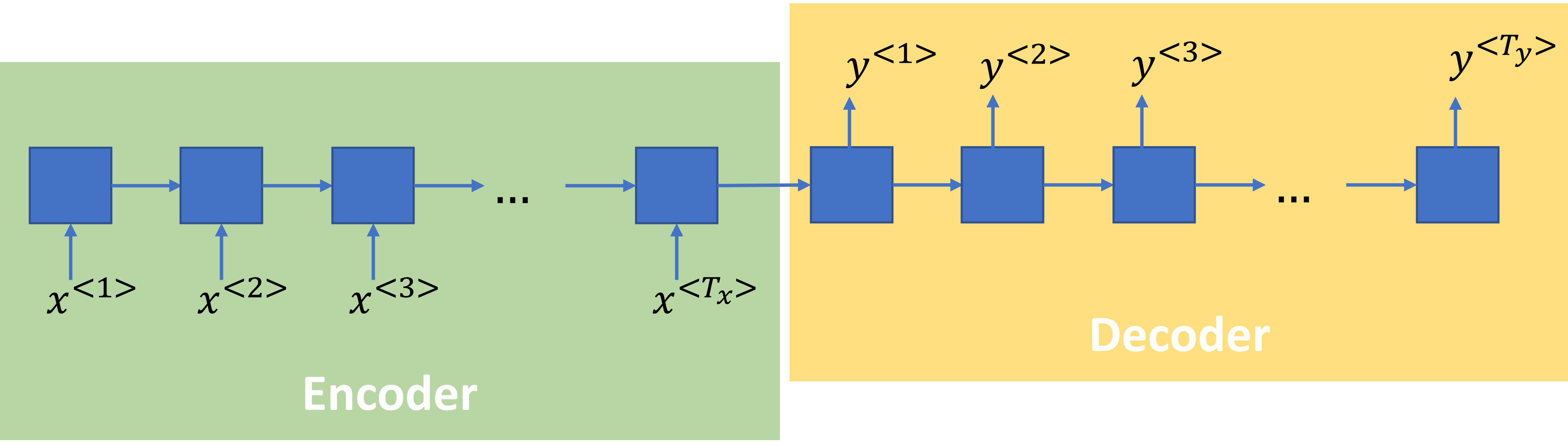
- One-to-many (e.g. music generation)



RNN architectures

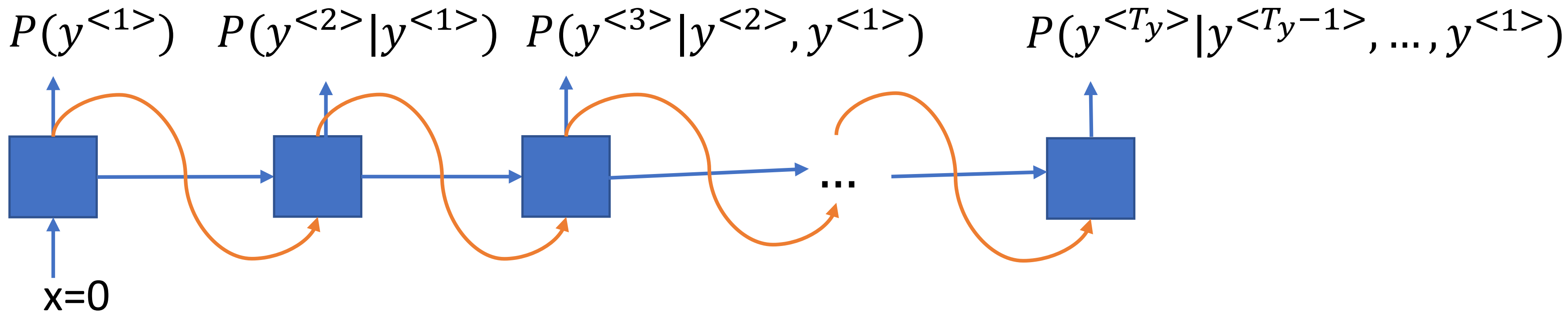
- Many-to-many: 2 case: $T_x = T_y$ or $T_x \neq T_y$





Langauge modelling

- Language model: P(sentences)
- Language model with RNN:


$P(y^{<1>})$
$P(y^{<2>}|y^{<1>})$
$P(y^{<3>}|y^{<2>},y^{<1>})$
$P(y^{<T_y>}|y^{<T_y-1>},\dots,y^{<1>})$
…
x=0

$$P(y^{<1>},y^{<2>},\dots,y^{<T_y>}) =$$

$$P(y^{<1>})\,P(y^{<2>}|\,y^{<1>})P(y^{<3>}|y^{<2>}y^{<1>})\dots P(y^{<T_y>}|y^{<T_y-1>},\dots)$$

Backpropagation

- Loss function:

$$L^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -\sum_k y_k^{<t>} \log \hat{y}_k^{<t>}$$

$$L = \sum_{t=1}^{T_y} L^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

- RNN: $a^{<t>} = g(W_{aa} a^{<t-1>} + W_{ax}\, x^{<t-1>} + b_a)$
- Let’s say we know: $\frac{\partial L}{\partial a^{<t>}} = \frac{\partial L}{\partial y^{<t>}} \frac{\partial y^{<t>}}{\partial a^{<t>}}$
- We need: $\frac{\partial L}{\partial W_{aa}}, \frac{\partial L}{\partial W_{ax}}, \frac{\partial L}{\partial b_a}$ $\Longleftarrow$ $\frac{\partial L^{<t>}}{\partial W_{aa}} = \frac{\partial L^{<t>}}{\partial a^{<t>}} \frac{\partial a^{<t>}}{\partial W_{aa}}$

$$\frac{\partial a^{<t>}}{\partial W_{aa}} = g'(a^{<t>}) \cdot \left(a^{<t-1>} + W_{aa} \frac{\partial L}{\partial a^{<t-1>}} \right) \Longrightarrow$$

Backprop trough time

Problems with RNN

- When we train we have to do backpropagation trough time
- Long sentences: vanishing gradient problem
- Vanishing gradient: g' small, $(g')^N \rightarrow 0$ (long sentence = big N)
- Similar to very deep networks
- ResNet: skip connection (linear information flow)

$$F(x) \rightarrow F(x) + x$$
$$F'(x)\frac{\partial L}{\partial x} \rightarrow F'(x)\frac{\partial L}{\partial x} + \frac{\partial L}{\partial x}$$

- NEW architectures instead of RNN with linear information flow

Gate recurrent unit (GRU)

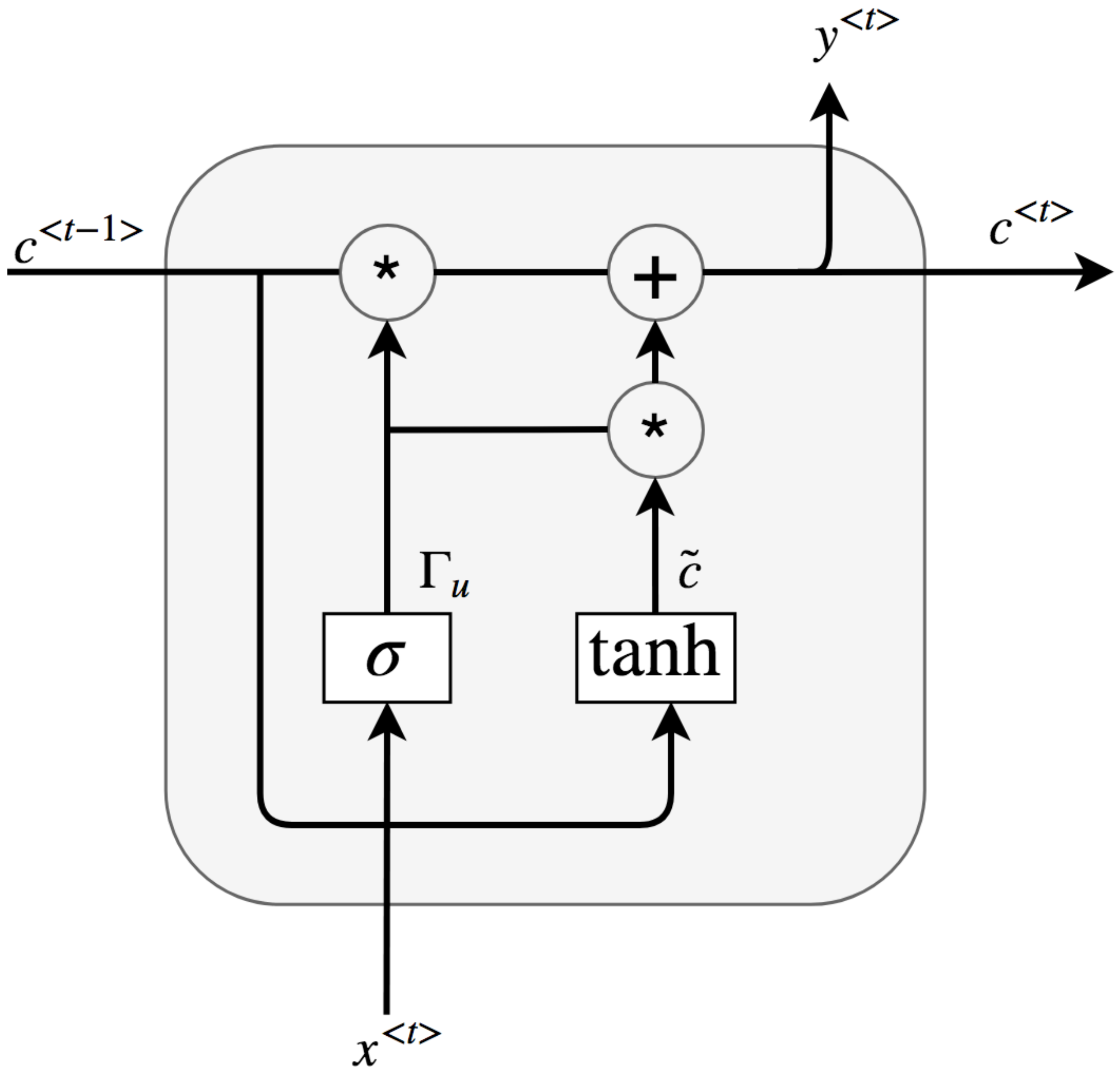
- New hidden variable: c = memory cell
- Candidate value: $\tilde{c} = g(W_{cc}c^{<t-1>} + W_{cx}x^{<t>} + b_c)$
- Update gate: $\Gamma_u = \sigma(W_{uc}c^{<t-1>} + W_{ux}x^{<t>} + b_u)$
- Update rule: $c^{<t>} = \Gamma_u \cdot \tilde{c} + (1 - \Gamma_u) \cdot c^{<t-1>}$
- Activation: simply $a^{<t>} = c^{<t>}$
- Example: c_0 = plural (0) / singular (1)

$\Gamma_u = 0$ $\Gamma_u = 1$ $\Gamma_u = 0$ $\Gamma_u = 0$ $\Gamma_u = 0$ $\Gamma_u = 1$

$c_0^{<1>} = 0$ $c_0^{<2>} = 1$ $c_0^{<3>} = 1$ $c_0^{<7>} = 1$ $c_0^{<8>} = 1$ $c_0^{<12>} = 0$

The cat, who belongs to my friend, was red. The other cats …

Gate recurrent unit (GRU)

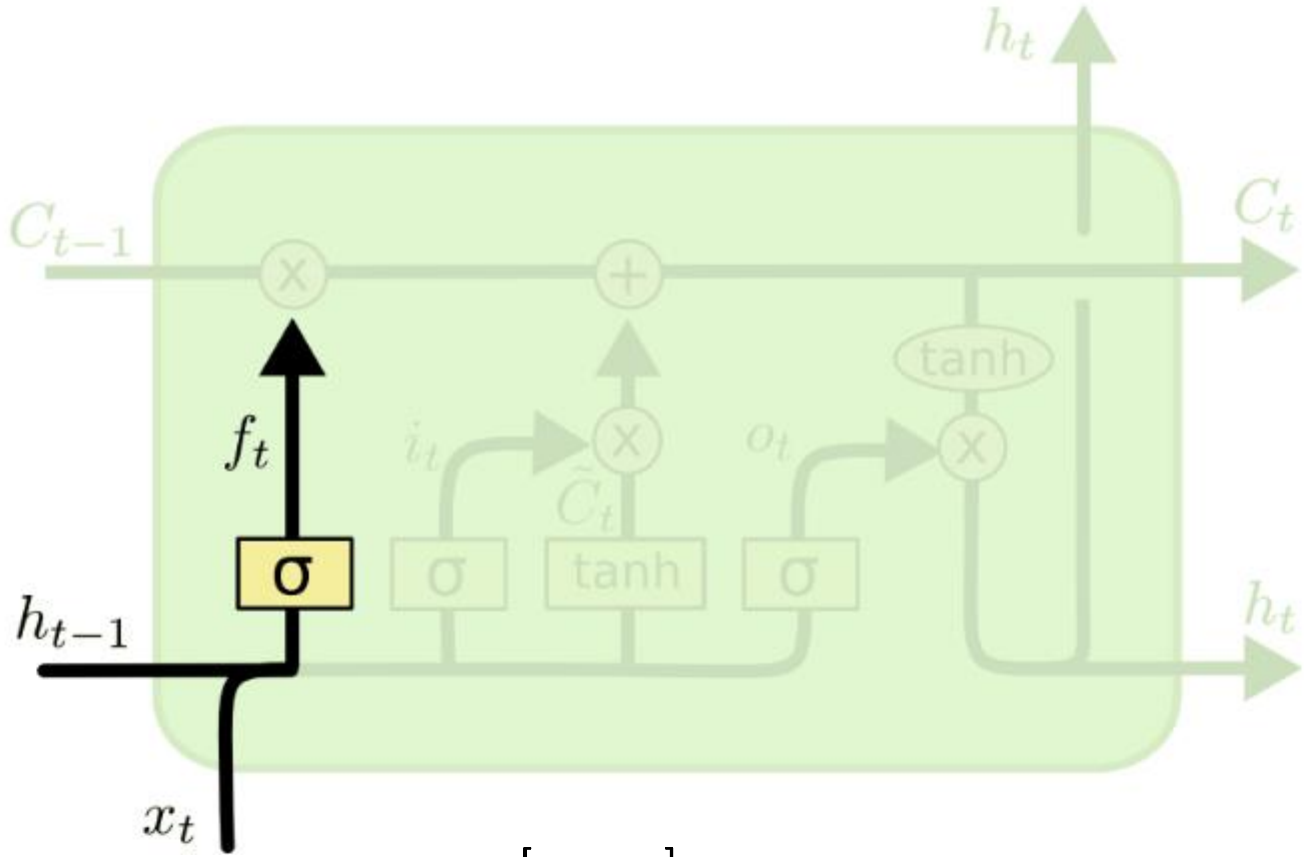


Long short-term memory (LSTM)

- GRU is a simpler model which is able to learn long-term dependencies
- More powerful model is LSTM
- Solves the vanishing problem very similarly: linear information flow trough time
- Candidate value: $\tilde{c} = g(W_{cc}c^{<t-1>} + W_{cx}x^{<t>} + b_c)$
- It uses more gates:
  - Update gate: $\Gamma_u = \sigma(W_{uc}c^{<t-1>} + W_{ux}x^{<t>} + b_u)$
  - Forget gate: $\Gamma_f = \sigma\left(W_{fc}c^{<t-1>} + W_{fx}x^{<t>} + b_f\right)$
  - Output gate: $\Gamma_o = \sigma(W_{oc}c^{<t-1>} + W_{ox}x^{<t>} + b_o)$
- Update rule: $c^{<t>} = \Gamma_u \cdot \tilde{c} + \Gamma_f \cdot c^{<t-1>}$
- Activation: $a^{<t>} = \Gamma_o \cdot g(c^{<t>})$

Long short-term memory (LSTM)

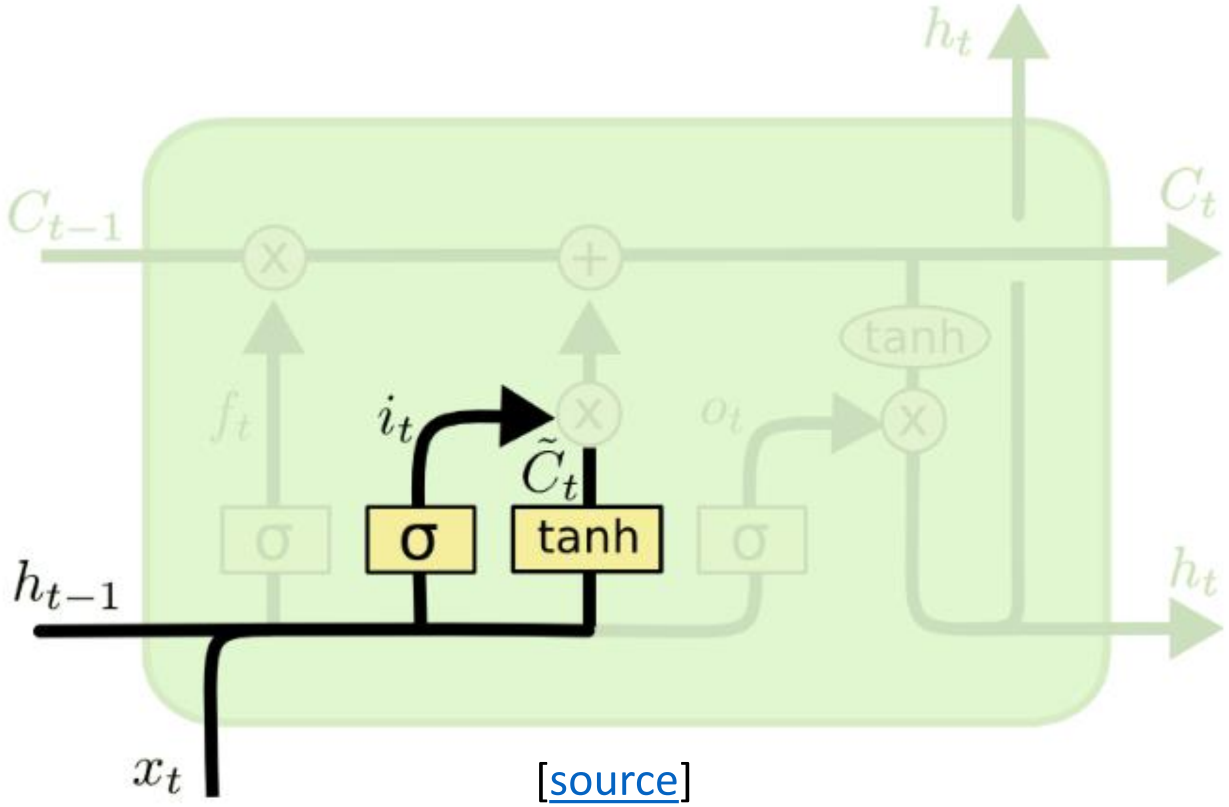
- Forget gate:
  - New parameters which control what information we forget



[source]

Long short-term memory (LSTM)

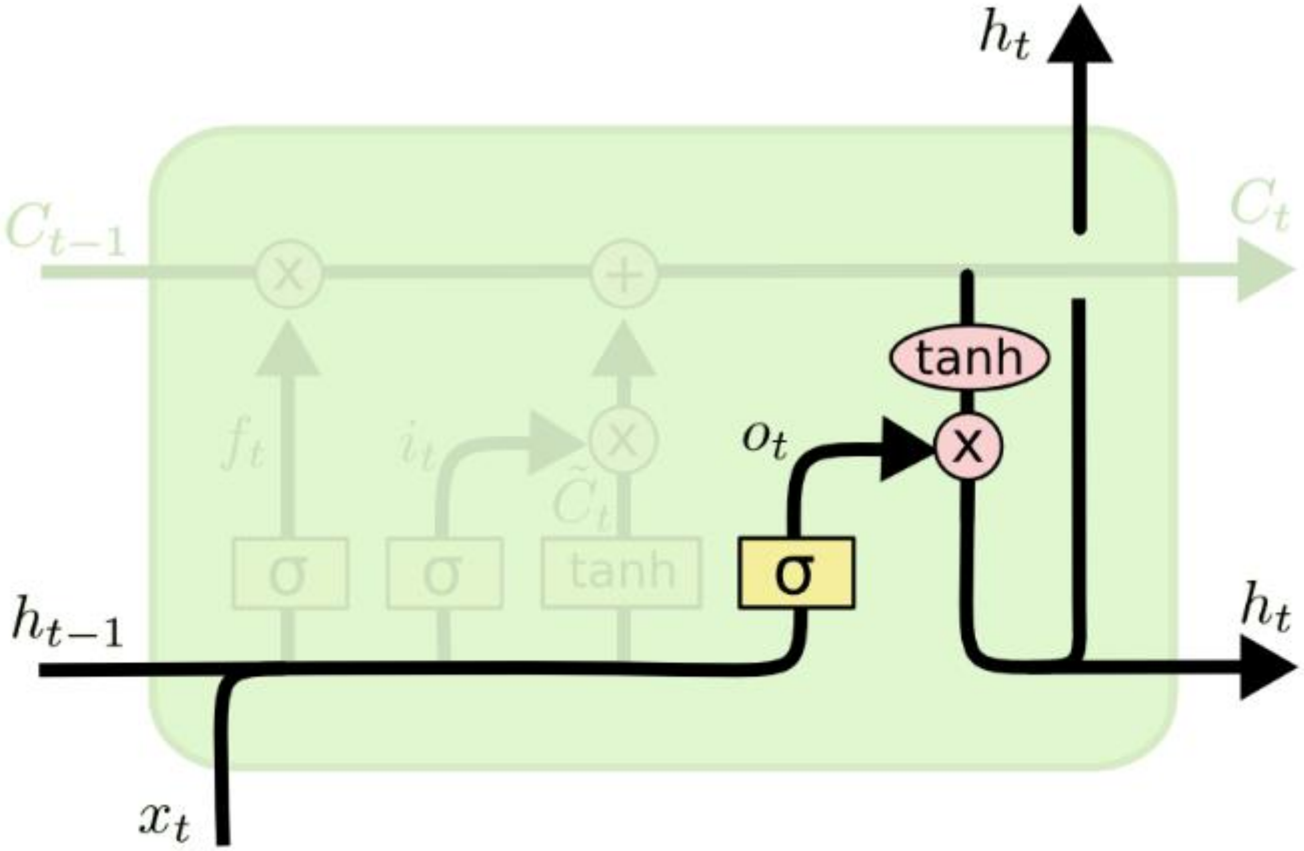
- We learn parameters for a candidate value: new c vector
- We also learn parameters for an update gate: how much information we want to store



[source]

Long short-term memory (LSTM)

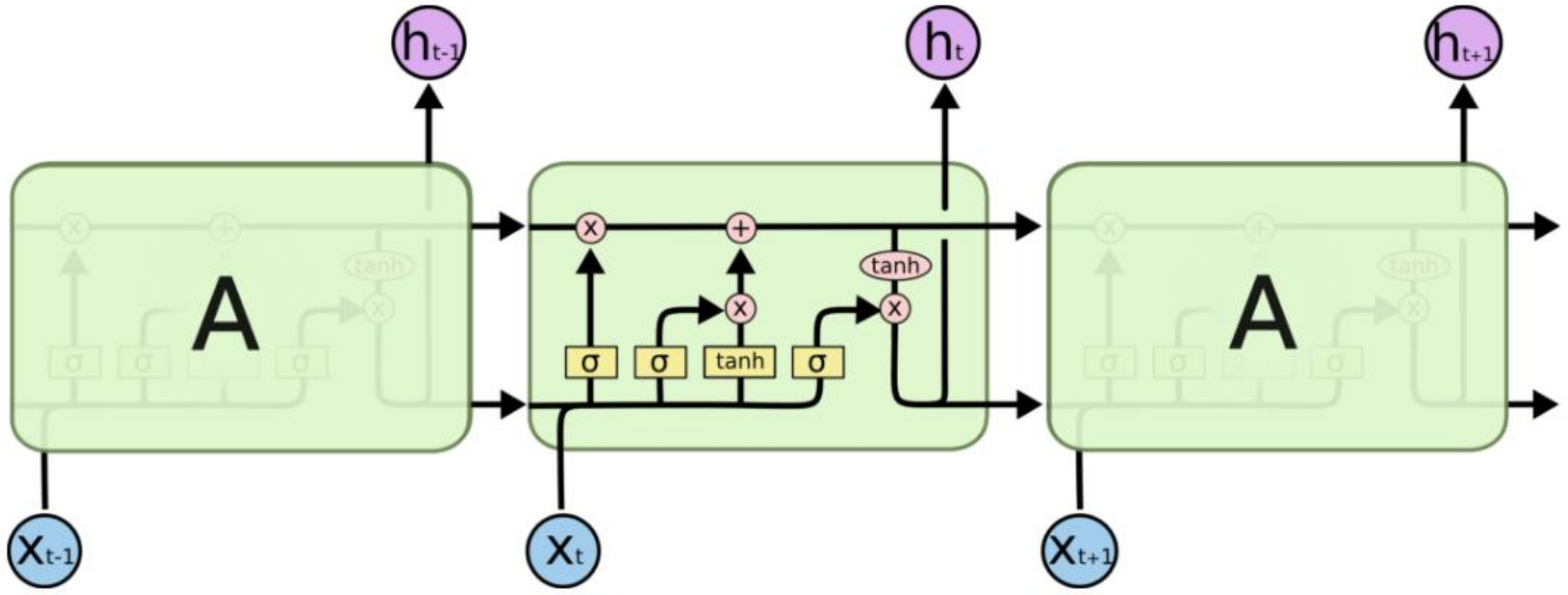
- Output gate: what information we want to the cell to output
- This gate also has learned parameters



[source]

LSTM

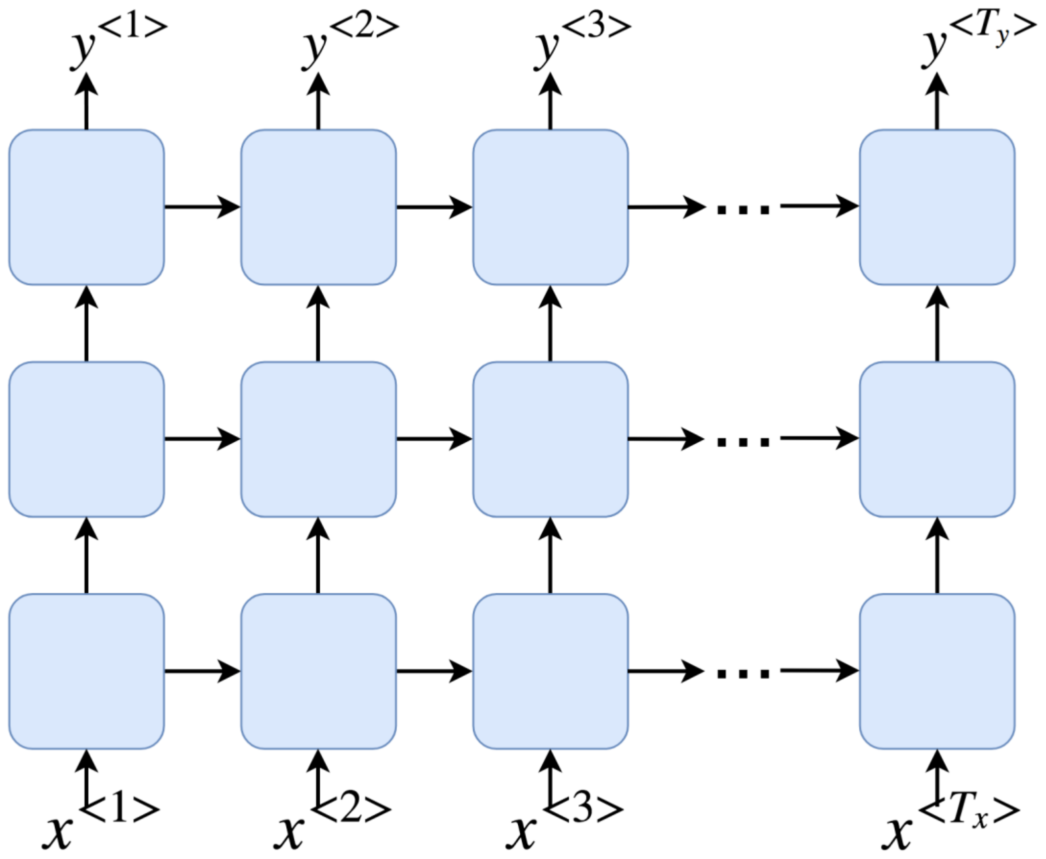
- The full LSTM cell



[source]

Deep RNNs

- We can stack RNN, GRU or LSTM cells in top of each other
- Typically a few cell (e.g. LSTM contains 4 “layers” => few layer is a deep network)



Demo notebooks

- Music generation:

https://github.com/qati/DeepLearningCourse/blob/master/demo_notebooks/lecture_09/piano.ipynb

- Tweet generation:

https://github.com/qati/DeepLearningCourse/blob/master/demo_notebooks/lecture_09/tweets.ipynb