Recurrent Neural Networks

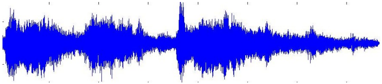
Machine Learning

A. Carlier

2024

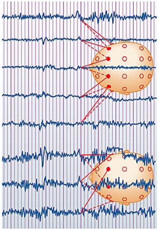
Outline

Sequential Data

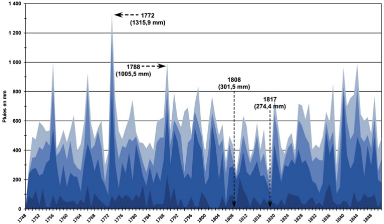


Audio

Video



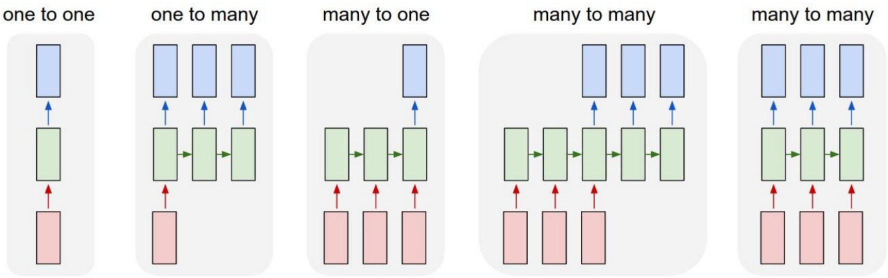
Medical Data



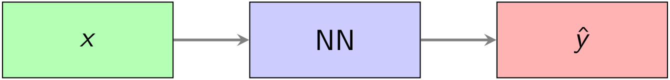
Physics-based data

But also... Textual data!

Sequential Problems

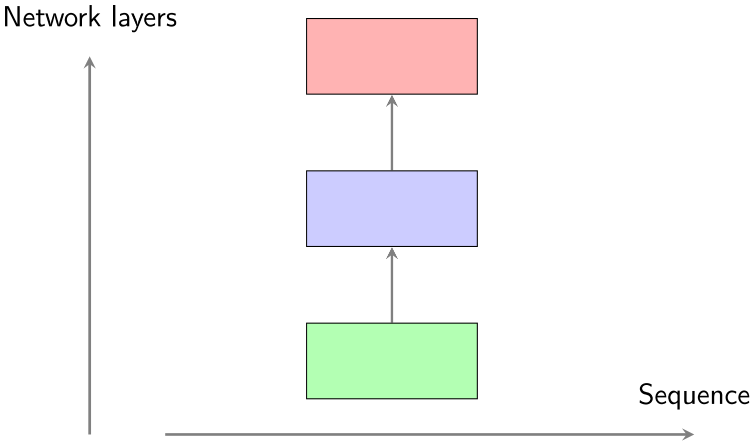


One to one



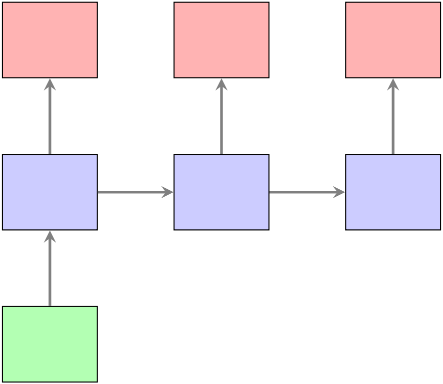
Examples : Multi-layer perceptrons, Convolutional neural networks, etc.

One to one

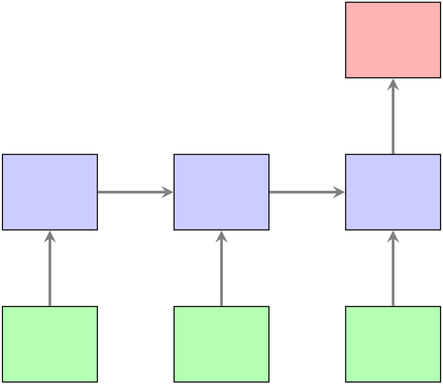


Another representation: the input is at the bottom, and the output is on top.

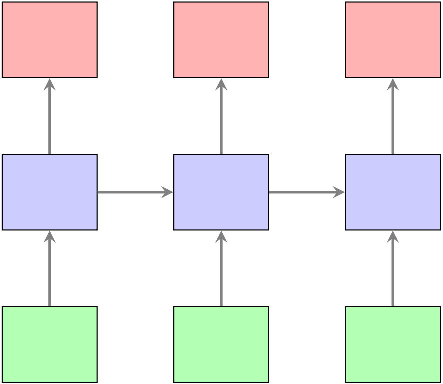
One to many



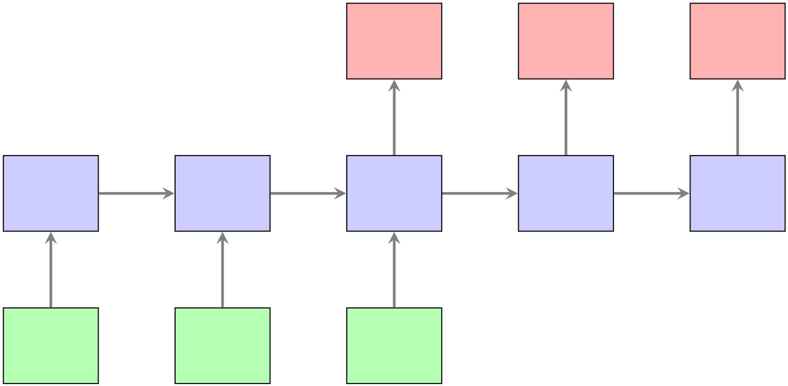
Many to one



Many to many



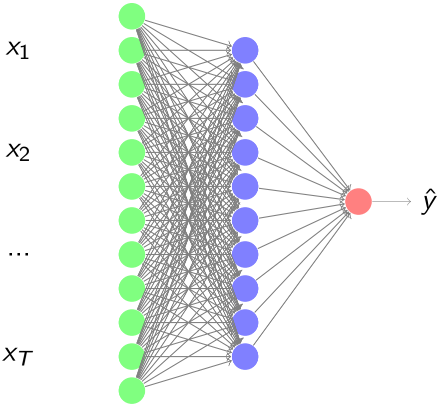
Many to many



Outline

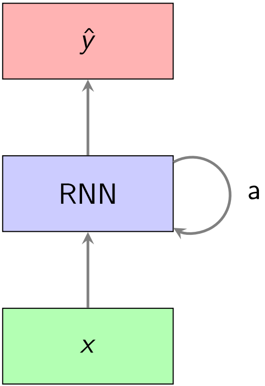
Sequential data

A standard multi-layer perceptron is not well suited to sequential data processing:

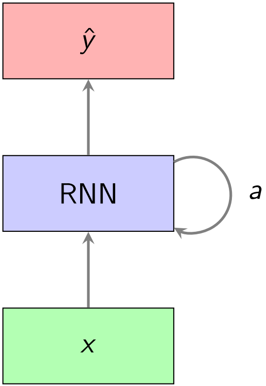


Sequences are of variable length and each data in the sequence is processed independently!

Recurrent neuron



Recurrent neuron

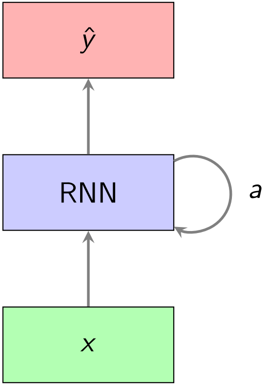


Let $(x^{<i>})_{i=1..T_x}$ be an input sequence

$$a^{<t>} = f_W(a^{<t-1>}, x^{<t>})$$

The same function f and the same parameters W are used for each sequence step.

Standard recurrent neuron

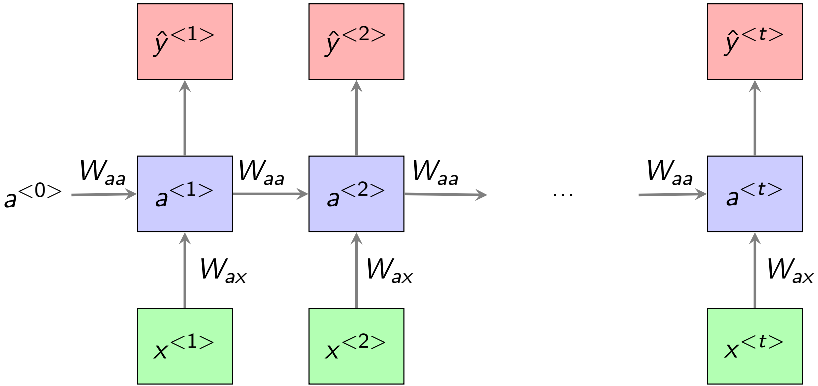


Let $(x^{<i>})_{i=1..T_x}$ be an input sequence

$$
\begin{aligned}
a^{<0>} &= \vec{0} \\
a^{<t>} &= \tanh(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \\
\hat{y}^{<t>} &= g(W_{ya}a^{<t>} + b_y)
\end{aligned}
$$

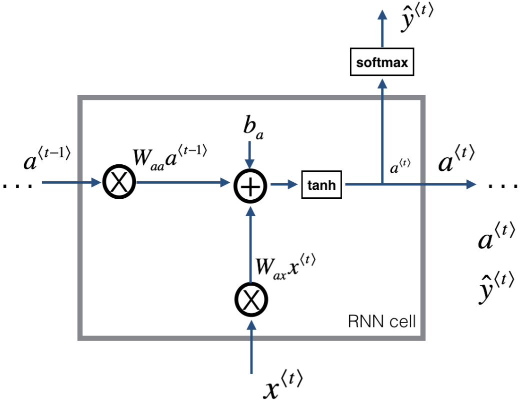
g represents the activation function of the output layer, which depends on the problem (typically `sigmoid`, `softmax`, or `linear`).

Recurrent network - developed representation



The same parameters are reused for each sequence step.

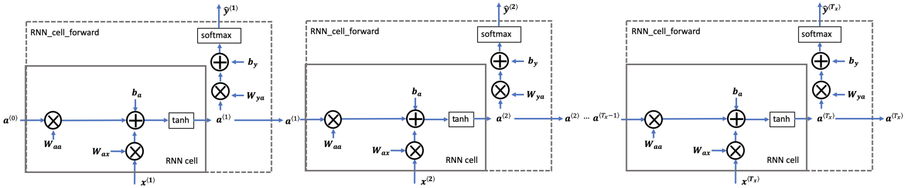
Recurrent neuron : *forward* pass



$$a^{\langle t \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b_a)$$

$$\hat{y}^{\langle t \rangle} = softmax(W_{ya}a^{\langle t \rangle} + b_y)$$

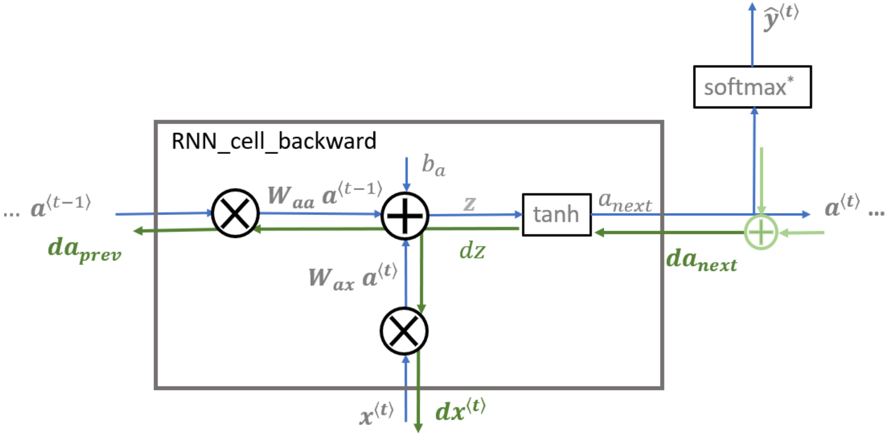
Recurrent neuron : *forward* pass



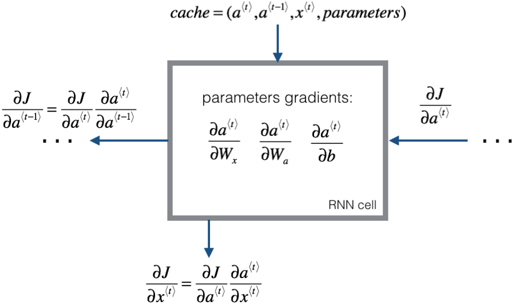
Predictions are made sequentially. Computations can not be parallelized efficiently in a recurrent network, which makes them rather slow.
On the other hand, the same parameters are reused for each sequence step which makes them parameter efficient and less prone to overfitting.

Recurrent neuron : *backward* pass

Backpropagation through time



Recurrent neuron : *backward* pass



$$a^{\langle t \rangle} = \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)$$

$$\frac{\partial \tanh(x)}{\partial x} = 1 - \tanh(x)^2$$

$$\frac{\partial a^{\langle t \rangle}}{\partial W_{ax}} = (1 - \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2) x^{\langle t \rangle T}$$

$$\frac{\partial a^{\langle t \rangle}}{\partial W_{aa}} = (1 - \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2) a^{\langle t-1 \rangle T}$$

$$\frac{\partial a^{\langle t \rangle}}{\partial b} = \sum_{batch} (1 - \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2)$$

$$\frac{\partial a^{\langle t \rangle}}{\partial x^{\langle t \rangle}} = W_{ax}{}^T . (1 - \tanh(W_{ax} x^{\langle t \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2)$$

$$\frac{\partial a^{\langle t \rangle}}{\partial a^{\langle t-1 \rangle}} = W_{aa}{}^T . (1 - \tanh(W_{ax} x^{\langle t-1 \rangle} + W_{aa} a^{\langle t-1 \rangle} + b)^2)$$

Recurrent neuron : *backward* pass

The gradient of the objective function with respect to the parameters includes the following term:

$$\prod_{t=1}^{T-1} \frac{\partial a^{<t+1>}}{\partial a^{<t>}}$$

This term can cause **vanishing and exploding gradients**!

Gradient clipping

In order to prevent exploding gradients, gradient clipping is often used:

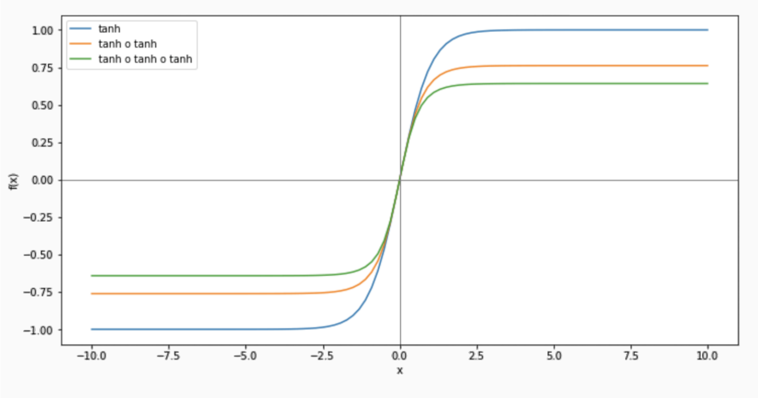
If $||g|| > c$, then

$$g \leftarrow c \frac{g}{||g||}$$

In `Keras` for example, one can instantiate an optimizer using the *clipnorm* attribute:

```
opt = SGD(lr=0.01, momentum=0.9, clipnorm=1.0)
```

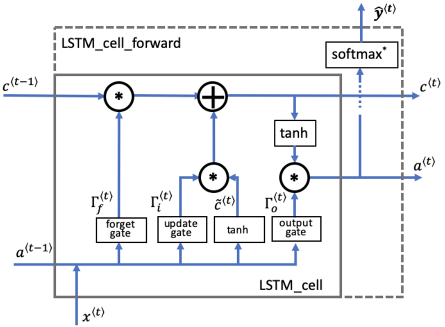
Long-term dependencies



Using `tanh` as an activation function can cause issues in long sequences: tanh(tanh(... x)...) tends towards 0!

Vanishing gradients

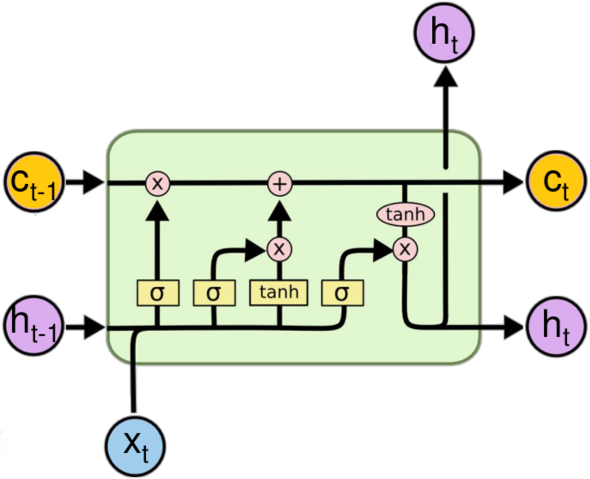
In 1997, Hochreiter and Schmidhuber proposed a new recurrent cell that enables long-term dependency learning and mitigates vanishing gradient problems: the `LSTM` (Long Short-Term Memory).



Outline

Long Short-Term Memory

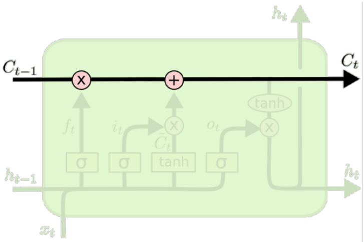
Reduction of the dissipation problem with a **gating mechanism** and a **memory cell**.



Long Short-Term Memory

A key component of the `LSTM` is its memory cell:

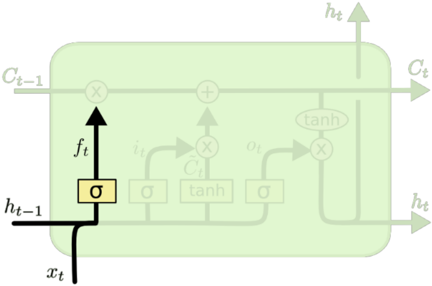
- Few operations alter it.
- It lets the information flow.



Long Short-Term Memory

Forget gate:

$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$



Long Short-Term Memory

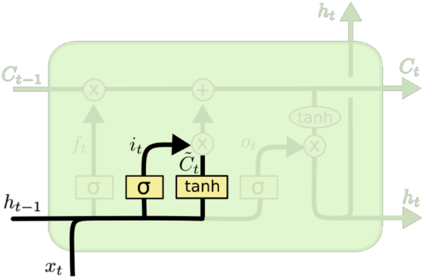
Input gate:

$$i_t = \sigma(U_i x_t + W_i h_{t-1} + b_i)$$

Input cell:

$$\tilde{C}_t = \tanh(U_g x_t + W_g h_{t-1} + b_g)$$

The input gate i_t controls which information $\tilde{C}_t$ enters the memory cell.



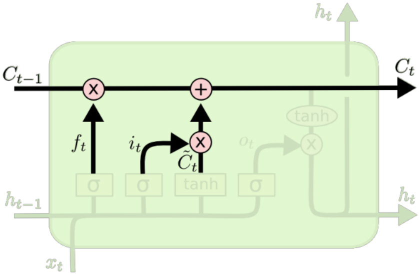
Long Short-Term Memory

Memory cell update:

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

where $*$ is the element-wise product.

The memory cell forgets information using f_t, and integrates new information using i_t.



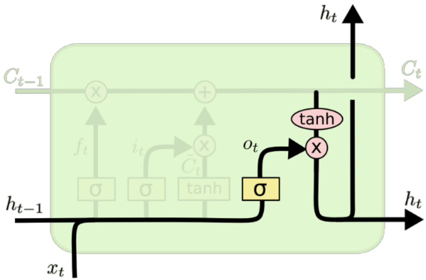
Long Short-Term Memory

Output gate:

$$o_t = \sigma(U_o x_t + W_o h_{t-1} + b_o)$$

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

The output gate controls what comes out of the memory cell.



Long Short-Term Memory

Reduction of the dissipation problem with a **gating mechanism** and a **memory cell**.

$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f)$$

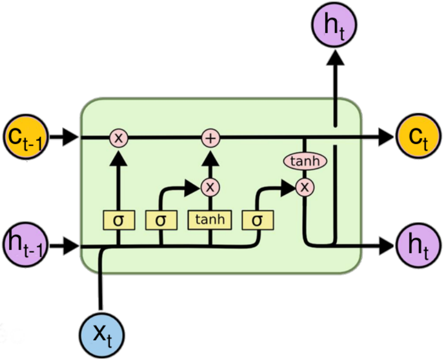
$$i_t = \sigma(U_i x_t + W_i h_{t-1} + b_i)$$

$$\tilde{C}_t = \tanh(U_g x_t + W_g h_{t-1} + b_g)$$

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

$$o_t = \sigma(U_o x_t + W_o h_{t-1} + b_o)$$

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

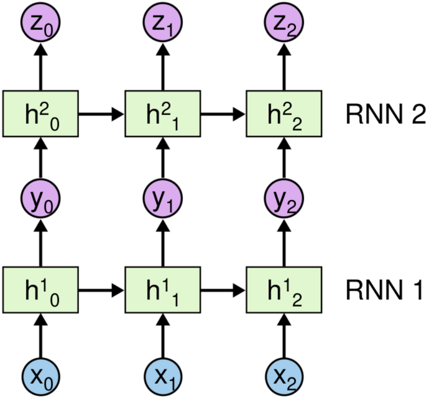


Outline

Recurrent neural networks

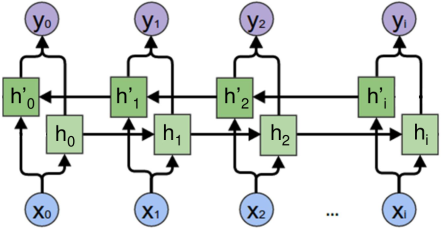
Deep Recurrent Neural Networks can be built by composing recurrent layers:

- Each layer can be a standard RNN, a LSTM, a GRU, etc.
- The first layer output sequence serves as input to the second layer, etc.



Bidirectional Networks

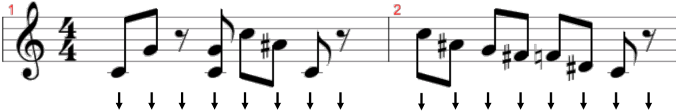
- A second RNN reads the input sequence backwards.
- This allows using information from both the past and the future.
- Both RNN have a different set of parameters.



Example: classification of musical genre

Goal: recognizing the musical genre from a music score

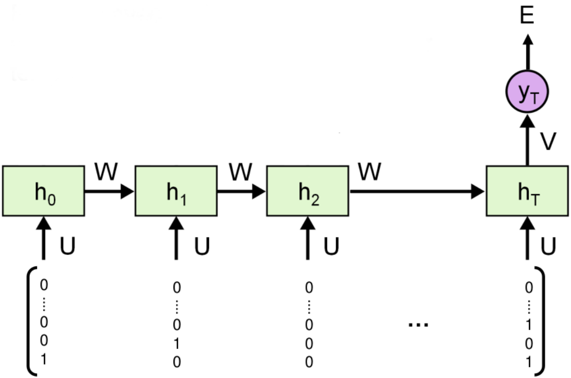
Input data:



$$
\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&1&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&1&1&0&1&0&1&0&0&1&1&0&1&0
\end{pmatrix}
$$

Example: classification of musical genre

Many-to-one problem:



$$E = \text{cross\_entropy}(y_T, c)$$

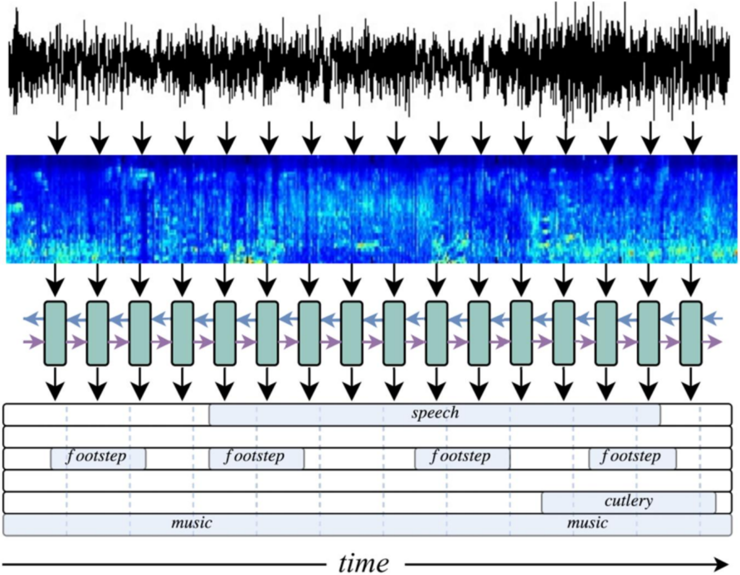
$$y_T = \text{Softmax}(V h_T + d)$$

$$h_t = \tanh(U x_t + W h_{t-1} + b)$$

x

Example : acoustic event detection

Many-to-many problem:



Outline

Sequential Data

A sentence is a sequence:

- of words :

  Luke , I am your father .

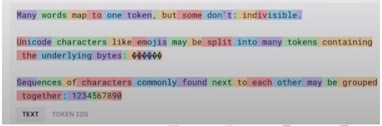
- of syllabs :

  Lu ke , I am your fa ther .

- or characters :

  L u k e , I a m y o u ...

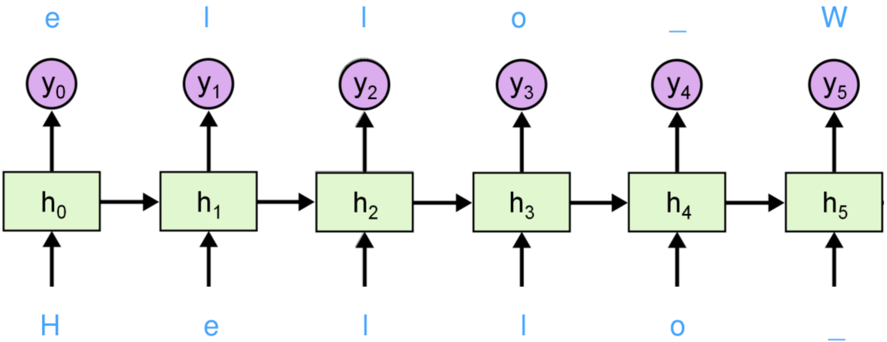
These are called *tokens*. (In practice, an optimal *tokenization* is learned from the corpus).



Language Model

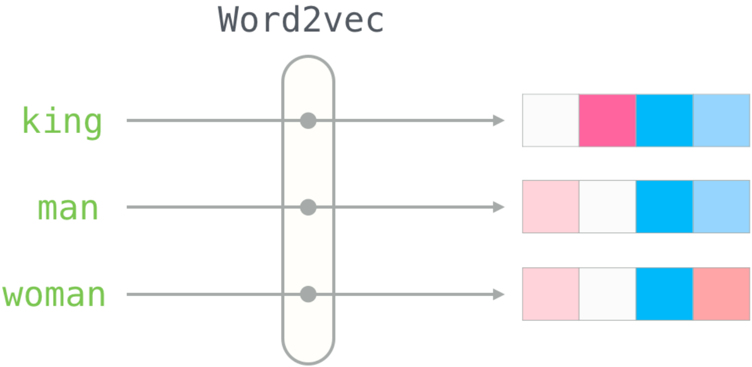
Goal: predict the next token in a sequence.

- Training dataset: tokenized text
- Input : sequence of tokens $x^{<1>}, \ldots, x^{<t>}$
- Target (label): $x^{<2>}, \ldots, x^{<t+1>}$
- Loss function: cross-entropy averaged over the sequence.



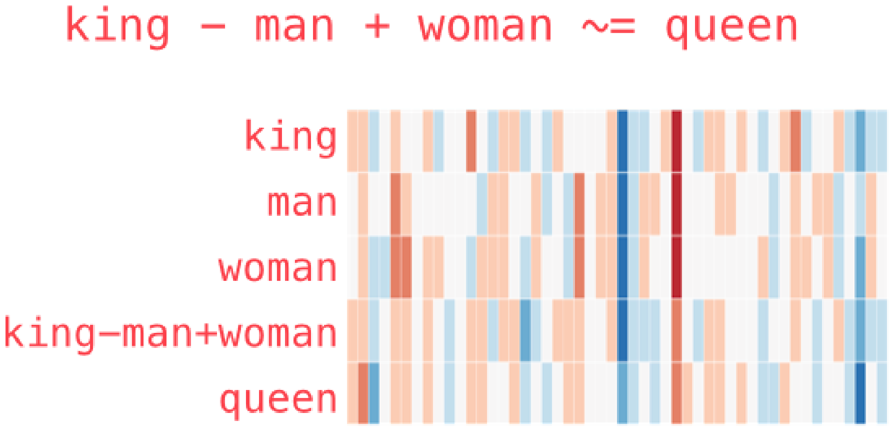
Word (token) embedding

How can we represent tokens numerically?



Word (token) embedding

Embeddings are numerical representations of tokens that convey a semantic meaning:



Word (token) embedding

An embedding layer is essentially a look-up table and can be learned from a training dataset.

Text generation using LSTM

At initialization:
"usb9xkrd9ruaiasdsaqj'4lmjwyd61se.lcn6jey0pbco40ab'65<8um324 nqdhm<ufwty*/w5bt'nm.zq«2rqm-a2'2mstu315wtNwdqNafqh"

After one epoch:
"to will an apple for a N shares of the practeded to working rudle and a dow listed that scill extressed holding a"

After 70 epochs:
"president economic spokesman executive for securities was support to put used the sharelike the acquired who "

Text Generation using Transformers

GPT-3 uses:

- 100k tokens
- an embedding dimension of 12788
- $\approx$ 100 layers
- $\approx$ 175 billions parameters