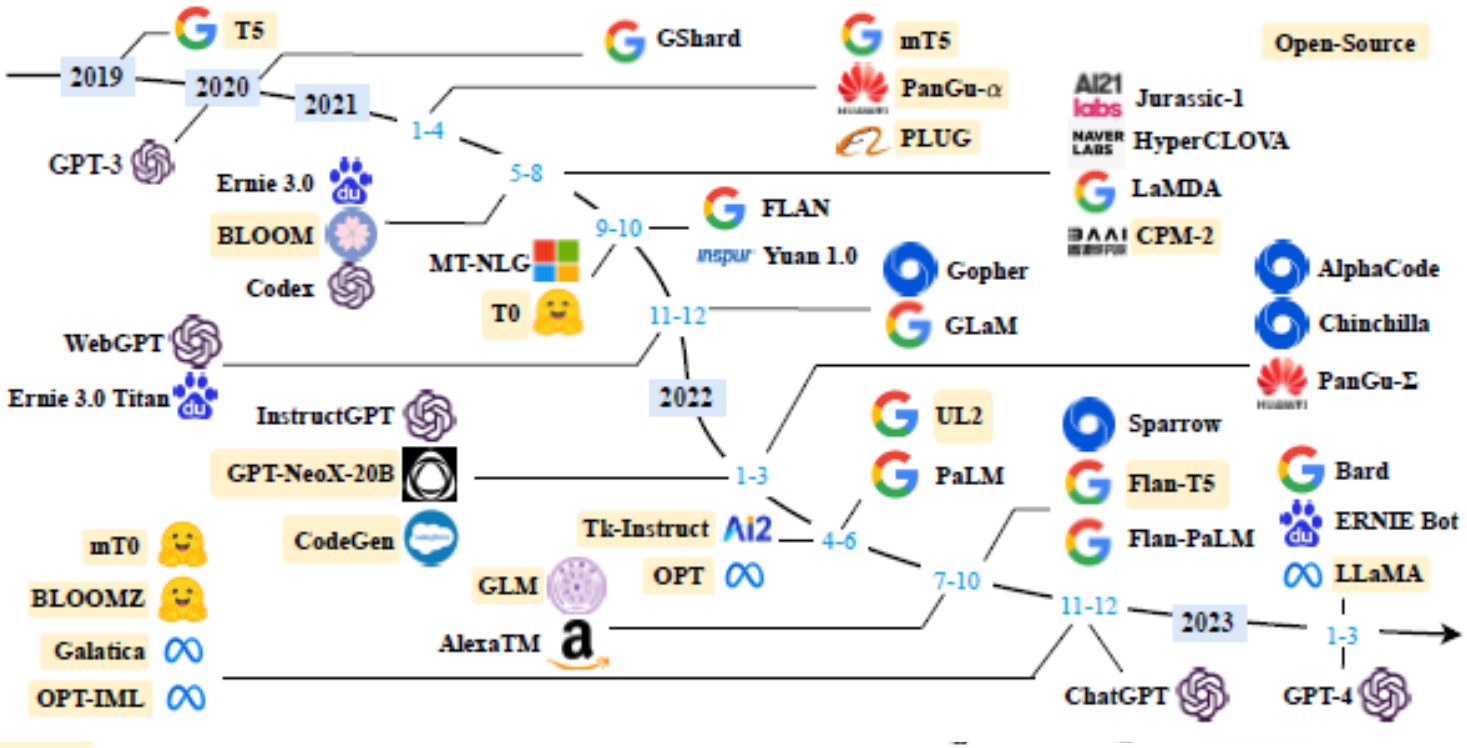
Into LLMs

CUNEF and DataLab ICMAT-CSIC

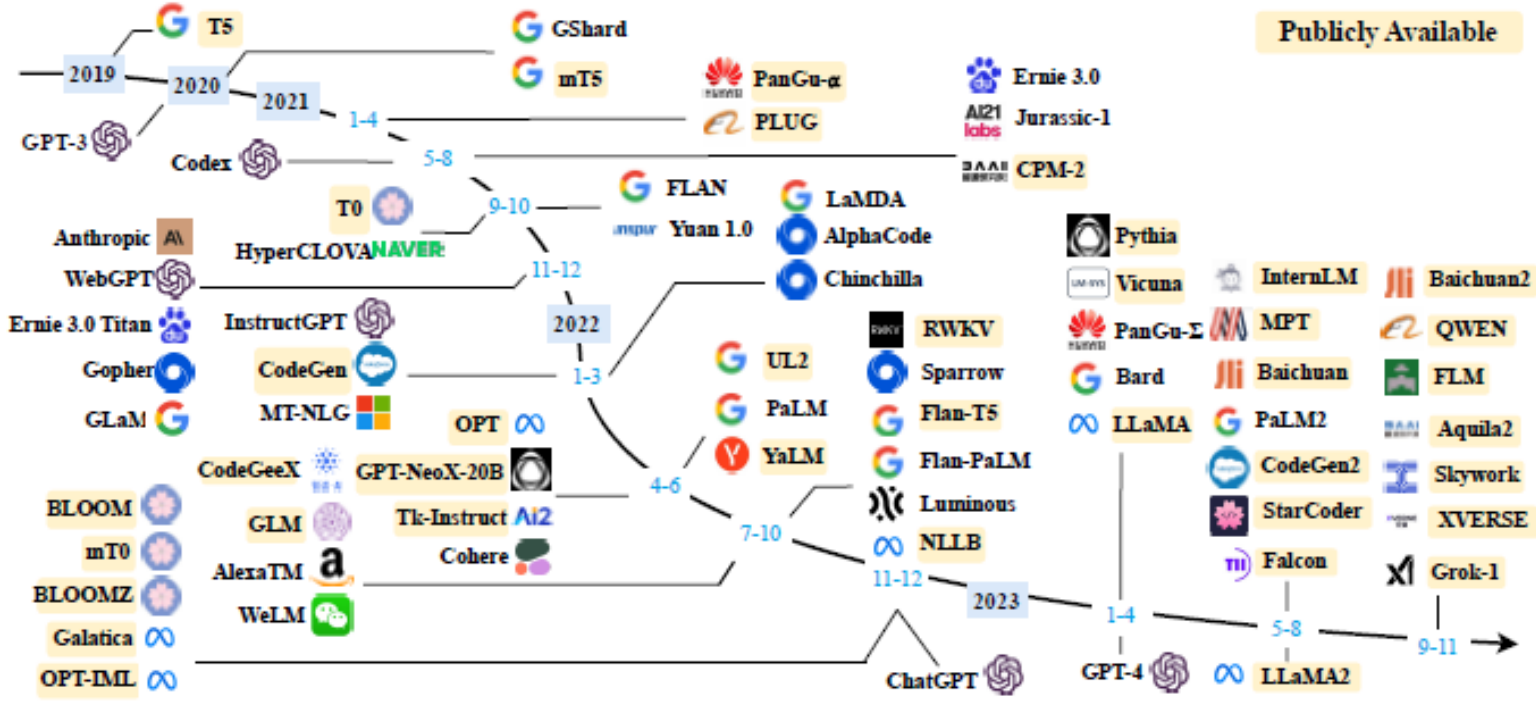
The LLM sprint



A survey of large language models https://arxiv.org/abs/2303.18223
Eight things to know about LLMs https://arxiv.org/abs/2304.00612

Images from sources quoted at various slides

The LLM sprint



A survey of large language models https://arxiv.org/abs/2303.18223
Eight things to know about LLMs https://arxiv.org/abs/2304.00612

Images from sources quoted at various slides

The AI/LLM hype

Policy

- The EU AI Act ‘last minute changes’
- Biden’s executive order on safe/secure/trustworthy AI: An RMF for generative AI
- IBM Research last Friday at Moncloa Palace
- …

Science

- Transformers can do Bayesian inference
- Transformers as statisticians…
- Learning the language of time series…
- …
- Towards evaluating the robustness of LLMs…

A sample example

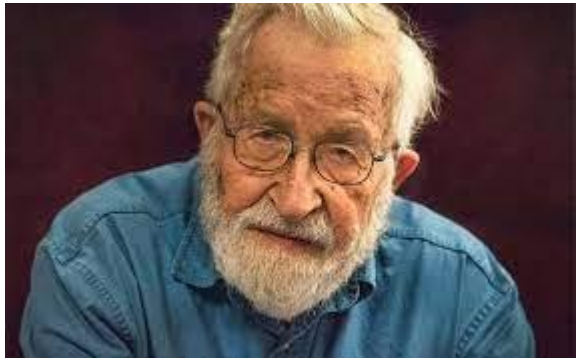
From Gallego, DRI (2022)
Code there

Text input	Prediction	True label
HOW MANY MOVIES ARE THERE GOING TO BE IN WHICH AGAINST ALL ODDS, A RAGTAG TEAM BEATS THE BIG GUYS WITH ALL THE MONEY?!!!!!!!! There's nothing new in ''The Big Green.'' If anything, you want them to lose. Steve Guttenberg used to have such a good resume (''The Boys from Brazil,'' ''Police Academy,'' ''Cocoon''). Why, OH WHY, did he have to do these sorts of movies during the 1990s and beyond?! So, just avoid this movie. There are plenty of good movies out there, so there's no reason to waste your time and money on this junk. Obviously, the ''green'' on their minds was money, because there's no creativity here. At least in recent years, Disney has produced some clever movies with Pixar.	Negative	Negative
When I first heard that the subject matter for Checking Out was a self orchestrated suicide party, my first thought was how morbid, tasteless and then a comedy on top of that. I was skeptical. But I was dead wrong. I totally loved it. The cast, the funny one liners and especially the surprise ending. Suicide is a delicate issue, but it was handled very well. Comical yes, but tender where it needed to be. Checking Out also deals with other common issues that I believe a lot of families can relate with and it does with tact and humor. I highly recommend Checking Out. A MUST SEE. I look forward to its release to the public.	Positive	Positive

Table 2 Results over the IMDb test set

Model	Test accuracy
LSTM	81.99%
Simple transformer	87.49%
RoBERTa	94.67%

A paradigm change in NLP



- From pre-deep learning
  - Language as a set of elements and rules to be combined
  - Context independent grammars
  - Closer to artificial languages (programming) than to natural ones
- To statistically based
  - Language as probabilities of word sequences
  - Computing frequencies of words, n-grams,…
  - Closer to natural language
  - Combined with deep nns (Transformers) SOTA
  - Almost a commodity (like vision)

Into LLMs. Objectives

- Short term. Provide a forum to facilitate understanding on latest developments in LLMs.
- Medium term. Explore possibility of creating a working group for less covered relevant areas in LLMs, Bayesian issues in relation to LLMs and security issues in LLMs.

Into LLMs. Schedule

- April 9th at 11.30 @ICMAT. David R Insua. DL, RNNs
- April 16th at 11.30 @CUNEF . Roi Naveiro. Transformers
- April 23rd at 11.30 @ICMAT. Carlos G Meixide. GPT
- May 7th at 11.30. @CUNEF. Victor Gallego. RLHF
- May 8th at 11.30. @ICMAT. David R Insua. LLMs, Bayes and Security

Into LLMs. Logistics

Zoom (recorded at ICMAT YouTube channel)

https://us02web.zoom.us/j/83182511916?pwd=bTdSWFY1OTJ6VWx1UTVmbFhYZktLUT09

Mailing list (if not contact roi.naveiro AT cunef.es)

llms-cunef-icmat-rg2024@googlegroups.com

Materials (slides, videos,…)

https://llms-cunef-icmat-rg2024.github.io/

For further info (roi or david.rios AT icmat.es)

Into LLMs
1. Intro RNNs: LSTMs and GRUs

Intro RNNs. LSTMs and GRUs. Key ideas

- LLMs are deep neural nets, hence a brief recall of dnns
- LLMs are recurrent neural nets, hence a brief recall of rnns (lstm, gru)
- dnns process numbers, hence a brief recall of means to convert text into vectors (embeddings)
- The evolution of NLP
- Hints on Bayesian nns

Intro RNNs. Background.

- Current Advances in Neural Networks https://www.annualreviews.org/doi/pdf/10.1146/annurev-statistics-040220-112019 Gallego, DRI
- Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling (https://arxiv.org/abs/1412.3555 Chung, Gulcehre, Cho, Bengio
- Chollet, Allaire (2018)
- Goodfellow, Bengio, Courville (2017)

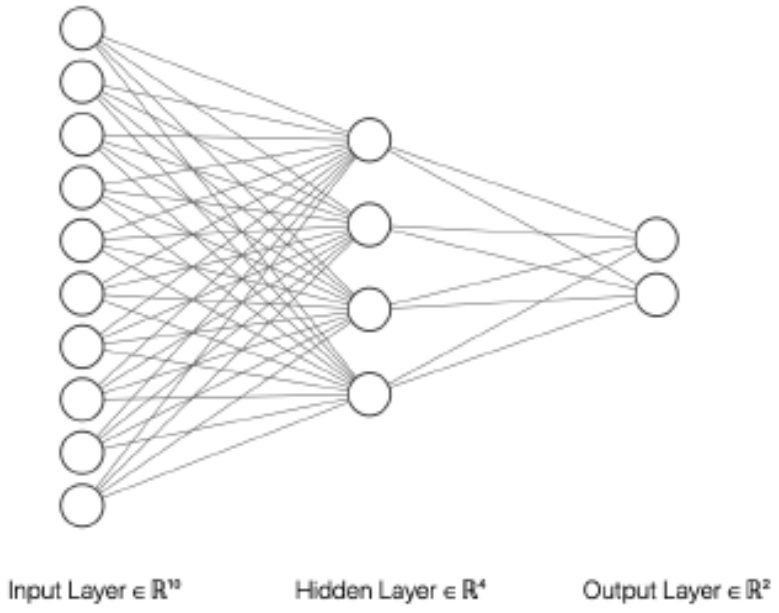
Videos

Summary of materials (slides, video, R labs) Chapter 7 in https://datalab-icmat.github.io/courses_stats.html#Introduction_to_Machine_Learning

https://www.youtube.com/watch?v=6niqTuYFZLQ from 8:55

From NNs to deep NNs

Shallow neural nets



$$y = \sum_{j=1}^{m} \beta_j \psi(x'\gamma_j) + \epsilon$$

$$\epsilon \sim N(0, \sigma^2),$$

$$\psi(\eta) = \exp(\eta)/(1 + \exp(\eta))$$

Linear in beta's, nonlinear in gamma's

Motivation. Cybenko's theorem

Any continuous function in r-dimensional cube may be approximated by models of type $\sum_{j=1}^{m} \beta_j \psi(x'\gamma_j)$ when the activation function is sigmoidal

(as m goes to infty)

Training

Given training data, maximise log-likelihood

$$\min_{\beta,\gamma} f(\beta,\gamma) = \sum_{i=1}^{n} f_i(\beta,\gamma) = \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{m} \beta_j \psi(x_i' \gamma_j) \right)^2$$

Gradient descent

Backpropagation to estimate gradient

Backprop. The gradient flow

Algorithm 18.1 BACKPROPAGATION

1 Given a pair x, y, perform a "feedforward pass," computing the activations $a_\ell^{(k)}$ at each of the layers $L_2, L_3, \ldots, L_K$; i.e. compute $f(x; \mathcal{W})$ at x using the current $\mathcal{W}$, saving each of the intermediary quantities along the way.

2 For each output unit ℓ in layer L_K, compute

$$\begin{aligned}\delta_\ell^{(K)} &= \frac{\partial L[y, f(x, \mathcal{W})]}{\partial z_\ell^{(K)}} \\ &= \frac{\partial L[y, f(x; \mathcal{W})]}{\partial a_\ell^{(K)}} \dot{g}^{(K)}(z_\ell^{(K)}),\end{aligned} \tag{18.10}$$

where $\dot{g}$ denotes the derivative of $g(z)$ wrt z. For example for $L(y, f) = \frac{1}{2}\|y - f\|_2^2$, (18.10) becomes $-(y_\ell - f_\ell) \cdot \dot{g}^{(K)}(z_\ell^{(K)})$.

3 For layers $k = K-1, K-2, \ldots, 2$, and for each node ℓ in layer k, set

$$\delta_\ell^{(k)} = \left(\sum_{j=1}^{p_{k+1}} w_{j\ell}^{(k)} \delta_j^{(k+1)} \right) \dot{g}^{(k)}(z_\ell^{(k)}). \tag{18.11}$$

4 The partial derivatives are given by

$$\frac{\partial L[y, f(x; \mathcal{W})]}{\partial w_{\ell j}^{(k)}} = a_j^{(k)} \delta_\ell^{(k+1)}. \tag{18.12}$$

From CASI (18)

Training with regularisation

$$\min_{\beta,\gamma} f(\beta,\gamma) = \sum_{i=1}^{n} f_i(\beta,\gamma) = \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{m} \beta_j \psi(x_i' \gamma_j) \right)^2$$

$$\min g(\beta,\gamma) = f(\beta,\gamma) + h(\beta,\gamma),$$

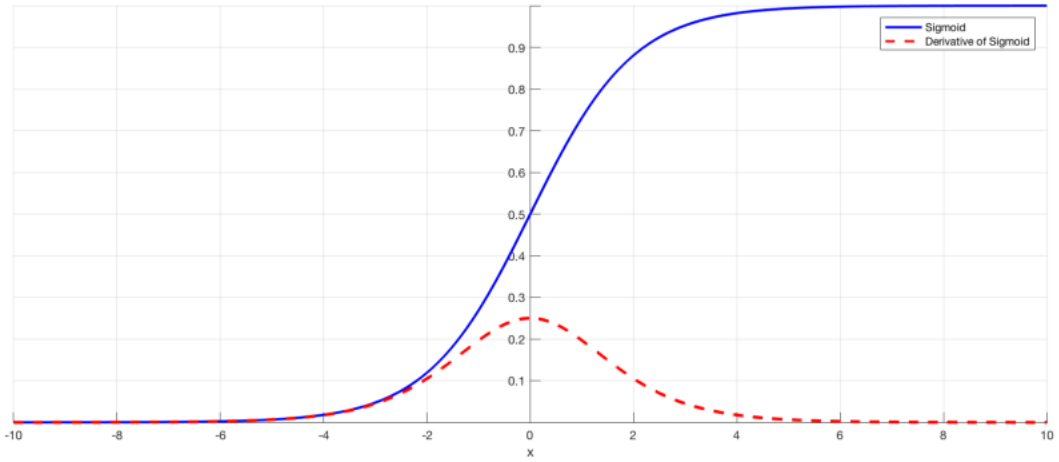
Weight decay $h(\beta,\gamma) = \lambda_1 \sum \beta_i^2 + \lambda_2 \sum \sum \gamma_{ji}^2$ Ridge

Vanishing gradient problem

- When using sigmoid activation functions, as the derivative is in (0,1), if we pile up several layers derivatives rapidly vanish, eventually, blocking training,…

$$\psi(z)' = \psi(z)(1 - \psi(z))$$

$$\psi(z) = \frac{1}{1+e^{-z}}$$



Regularisation II. Early stopping

- Training is iterative
- Preserve validation set. After each iteration, compute validation error.
- Typically, validation error reduces and then grows (due to overfitting)
- Stop before this happens-→ Early stopping
- May reduce network complexity

- Dropout

Bayesian analysis of shallow neural nets (fixed arch) (Muller, DRI, 98)

$$y = \sum_{j=1}^{m} \beta_j \psi(x'\gamma_j) + \epsilon$$

$$\epsilon \sim N(0, \sigma^2),$$

$$\psi(\eta) = \exp(\eta)/(1+\exp(\eta))$$

$$\beta_i \sim N(\mu_\beta, \sigma_\beta^2) \text{ and } \gamma_i \sim N(\mu_\gamma, S_\gamma^2)$$

$$\mu_\beta \sim N(a_\beta, A_\beta),\ \mu_\gamma \sim N(a_\gamma, A_\gamma),\ \sigma_\beta^{-2} \sim \quad Gamma(c_b/2, c_b C_b/2)$$

$$S_\gamma^{-1} \sim Wish(c_\gamma, (c_\gamma C_\gamma)^{-1}) \text{ and } \sigma^{-2} \sim Gamma(s/2, sS/2)$$

Bayesian analysis of shallow neural nets (fixed arch)

Start with arbitrary (β, γ, ν).
while *not convergence* **do**
  Given current (γ, ν), draw β from $p(\beta|\gamma, \nu, y)$ (a multivariate normal).
  for $j = 1, ..., m$, *marginalizing in* β *and given* ν **do**
    Generate a candidate $\tilde{\gamma}_j \sim g_j(\gamma_j)$.
    Compute $a(\gamma_j, \tilde{\gamma}_j) = \min\left(1, \frac{p(D|\tilde{\gamma},\nu)}{p(D|\gamma,\nu)}\right)$ with $\tilde{\gamma} = (\gamma_1, \gamma_2, \ldots, \tilde{\gamma}_i, \ldots, \gamma_m)$.
    With probability $a(\gamma_j, \tilde{\gamma}_j)$ replace γ_j by $\tilde{\gamma}_j$. If not, preserve γ_j.
  end
  Given β and γ, replace ν based on their posterior conditionals:
  $p(\mu_\beta|\beta, \sigma_\beta)$ is normal; $p(\mu_\gamma|\gamma, S_\gamma)$, multivariate normal; $p(\sigma_\beta^{-2}|\beta, \mu_\beta)$, Gamma; $p(S_\gamma^{-1}|\gamma, \mu_\gamma)$, Wishart; $p(\sigma^{-2}|\beta, \gamma, y)$, Gamma.
end

Classification

Use a multinomial likelihood

$$p(y|x,\beta,\gamma) = Mult(n=1, p_1(x,\beta,\gamma), \ldots, p_K(x,\beta,\gamma)),$$

Use softmax to compute class probabilities

$$p_k = \frac{\exp\{\beta_k \psi(x'\gamma_k)\}}{\exp\left\{\sum_{k=1}^{K} \beta_k \psi(x'\gamma_k)\right\}}$$

Other

Non-linear autoregression (Menchero, Montes, Muller, DRI)

$$y = \sum_{j=1}^{m} \beta_j \psi(x'\gamma_j) + \epsilon$$

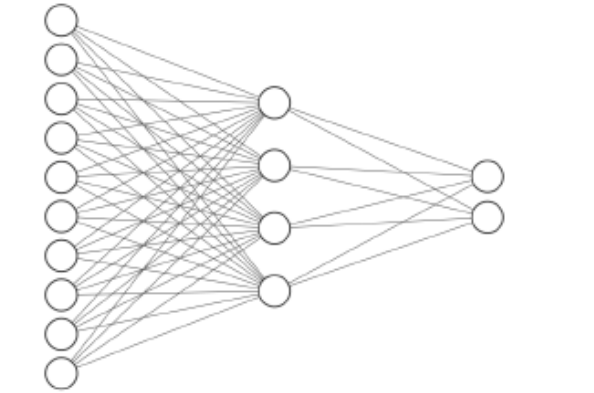
$$\epsilon \sim N(0, \sigma^2),$$

$$\psi(\eta) = \exp(\eta)/(1 + \exp(\eta))$$

Nonparametric regression (DRI, Muller)

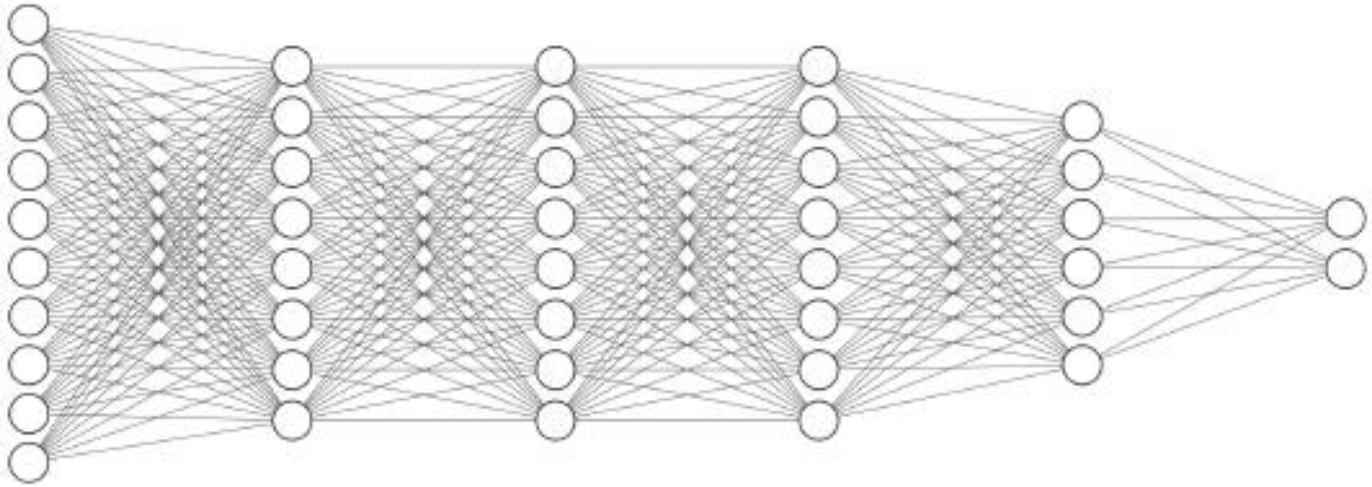
Gaussian processes

To deep nns



$$y = \sum_{j=1}^{m} \beta_j \psi(x'\gamma_j) + \epsilon$$
$$\epsilon \sim N(0, \sigma^2),$$
$$\psi(\eta) = \exp(\eta)/(1 + \exp(\eta))$$

(Shallow) Neural nets



$$\{f_0, f_1, ..., f_{L-1}\}$$

$$z_{l+1} = f_l(z_l, \gamma_l).$$

$$y = \sum_{j=1}^{m_L} \beta_j z_{L,j} + \epsilon$$
$$\epsilon \sim N(0, \sigma^2),$$

Deep neural nets

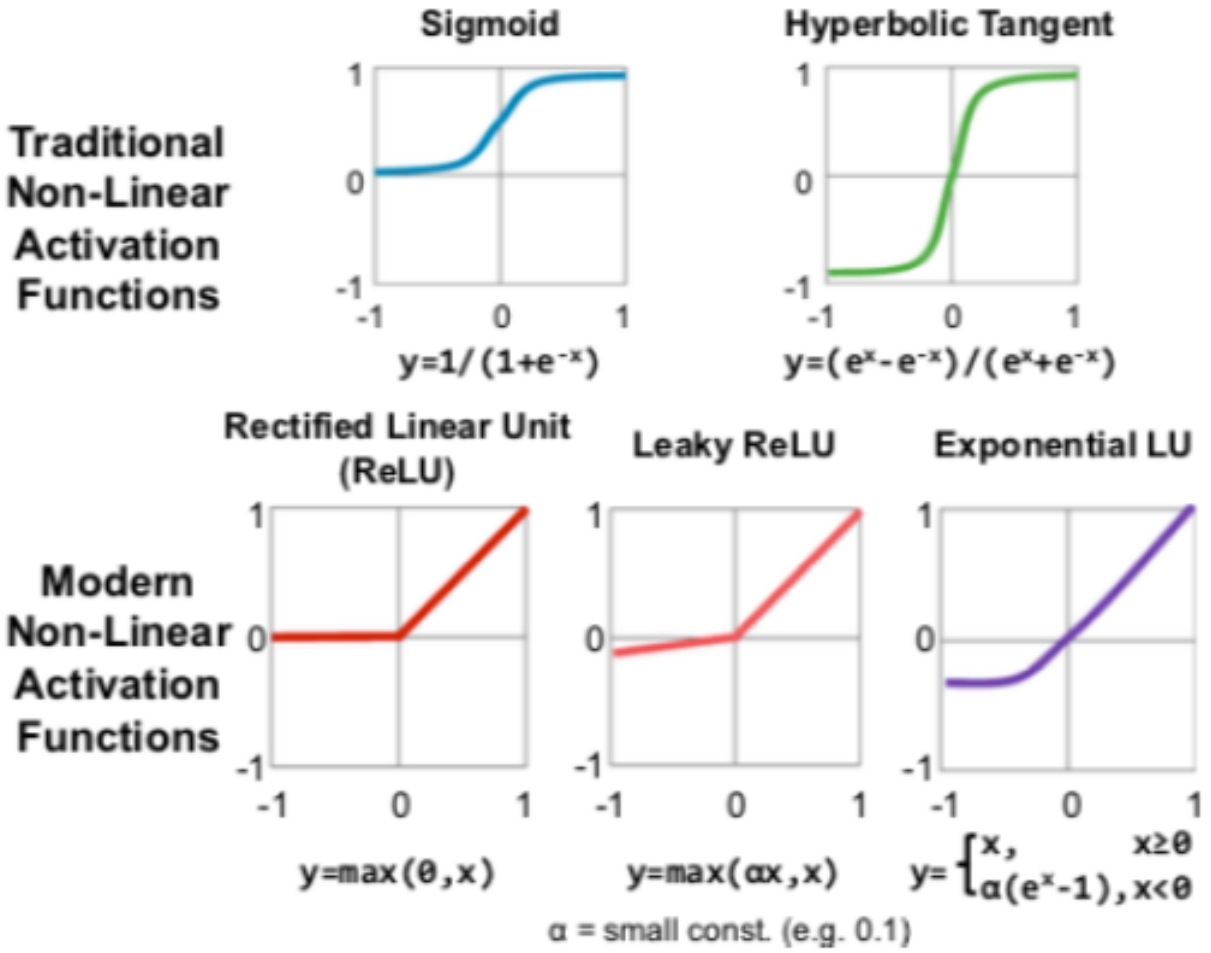
Motivation. New universal approximation theorems

Arbitrary-width. Remember Cybenko's (and others)

Arbitrary-depth.

Any (Lebesgue) integrable function can be arbitrarily approximated by RELU fully-connected neural network by as its depth goes to infinity (and its width is bounded by max(n+1,m) n dimension of origin, m dimension of image)

The evolution in activation functions



Training. Same idea!!!

Given training data, minimise -- log-likelihood + regulariser

$$\min g(\beta, \gamma) = f(\beta, \gamma) + h(\beta, \gamma),$$

Gradient descent

Backpropagation to estimate gradient

Problems

Evaluating the objective function. Depends on n

$$\sum_{i=1}^{n} f_i(\beta, \gamma)$$

Evaluating the gradient. Depends on n

$$\sum_{i=1}^{n} \nabla f_i(\beta, \gamma)$$

Each gradient sub-term $\nabla f_i(\beta, \gamma)$ over a large number of parameters and over a long backwards recurrence

Complexity is O(w) but w is getting pretty big in Deep networks

From gradient descent…

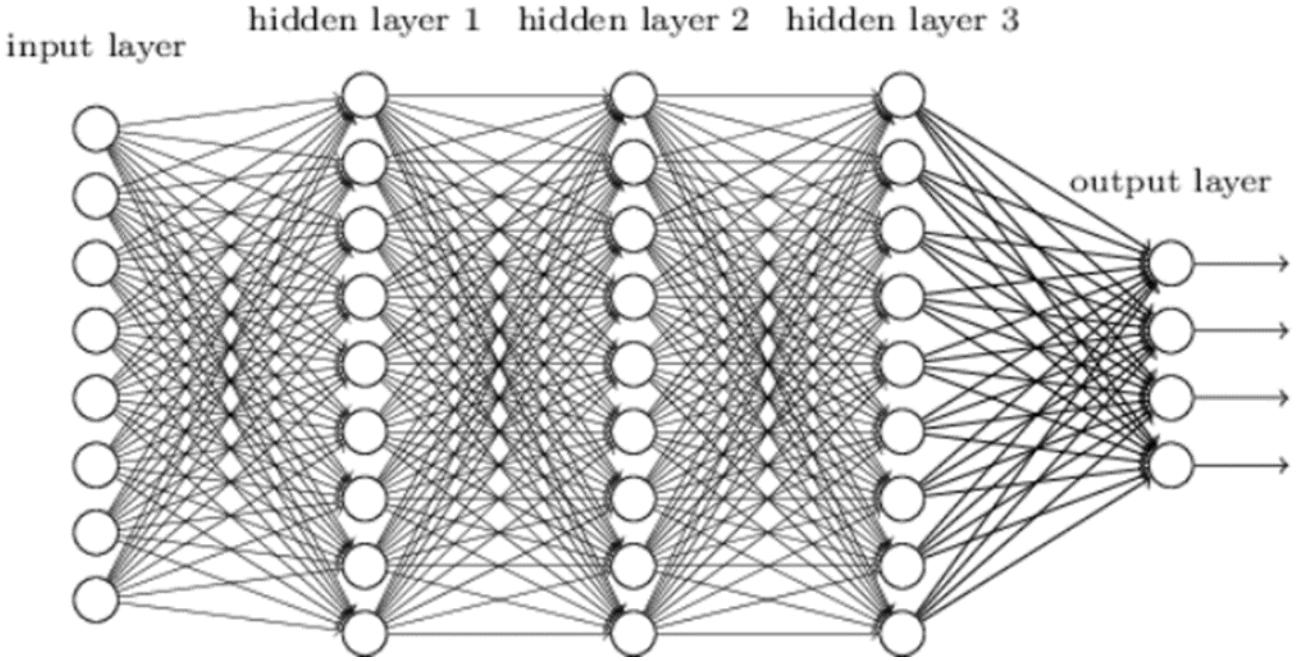
Training goes through minimising

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x},\mathrm{y}\sim\hat{p}_{\mathrm{data}}} L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

$$L(\boldsymbol{x}, y, \boldsymbol{\theta}) = -\log p(y \mid \boldsymbol{x}; \boldsymbol{\theta})$$

Requires gradient

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$



Might not even fit in memory, very slow anyway

… to stochastic gradient descent

(Randomly) sample a minibatch of size m'.

Approximate gradient

$$\boldsymbol{g} = \frac{1}{m'} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m'} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

Update via gradient descent

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \boldsymbol{g}$$

Stochastic gradient descent

Require: Learning rate ϵ_k.

Require: Initial parameter $\boldsymbol{\theta}$

while stopping criterion not met **do**

Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.

Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$

Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \hat{\boldsymbol{g}}$

end while

Use backprop at this stage

SGD. Robbins Monro conditions (1954!!!!)

If the learning parameters are chosen so that $\sum_{k=1}^{\infty} \epsilon_k = \infty \quad \sum_{k=1}^{\infty} \epsilon_k^2 < \infty$

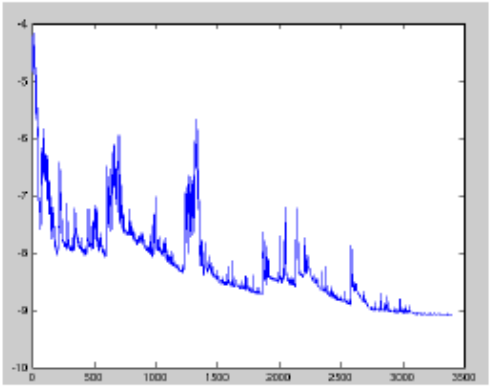
and the gradient estimator is unbiased

then SDG converges a.s. (to a local optimum)

NB: The minibatch size can even be 1 (some authors reserve SDG name to size 1 batches) , ie randomly sample just one instance and proceed

NB2: The batch now fits in memory!!!

NB3: May aid in escaping local minima!!!



SGD variants

Check Ruder's paper https://arxiv.org/abs/1609.04747

https://keras.io/api/optimizers/

Adam tends to be favoured

Trascends neural nets: very large scale optimization $\min J(\theta)$

Bayesian approaches in DL lagging….

Transfer learning

Adversarial machine learning

Explainability/Interpretability

NNs for sequence processing

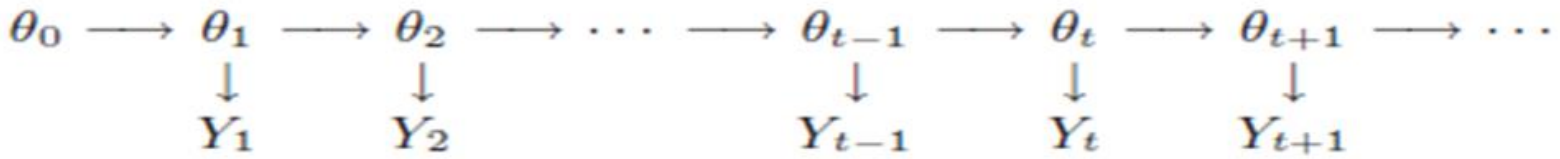
Motivation

- Fully connected NNs can approximate any function….
- But training can be super slow and may require lots of data
- In some domains, lots gained through specific archictectures

- In NLP, recurrent neural nets (RNNs), and successors
- More generally in sequence processing, RNNs, and successors

Time series (sequence) analysis

- Traditional models in time domain: **AR**IMA, exponential smoothing
- Models in frequency domain: Spectral analysis
- State space models: Kalman filter, Hidden Markov models, dynamic linear models (plus non linear and non gaussian extensions)



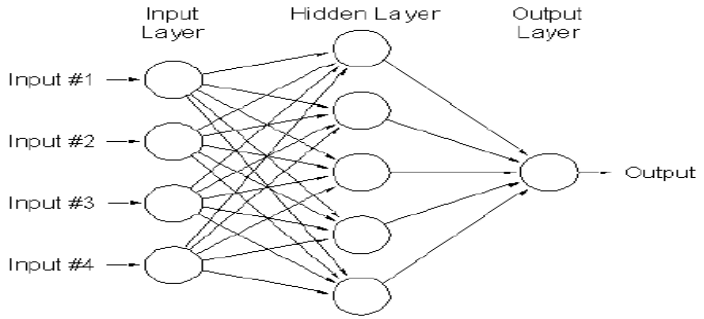
Check Prado; Ferreira, West (2021) for a comprehensive review

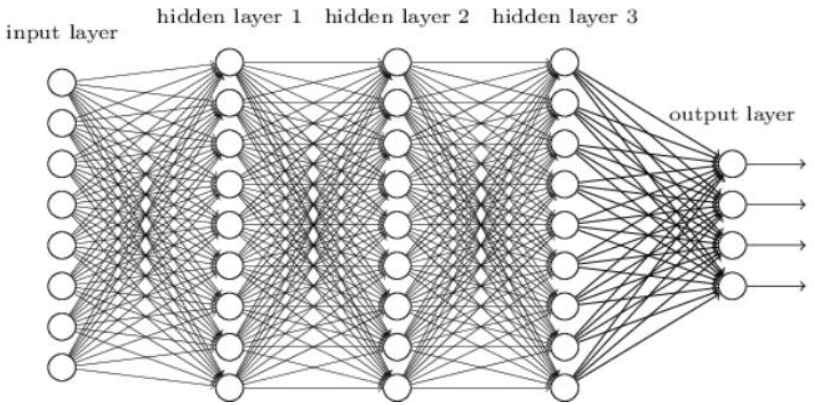
NNs for time series. Nonlinear autoregressions…

Shallow NN

$$y_j = \sum_{i=1}^{m} \beta_i \psi(x_k \omega_i) + \varepsilon_j$$

$$\min_{\beta, w} \sum_{k=1}^{n} \left(y_k - \sum_{i=1}^{m} \beta_i \psi(x_k \omega_i) \right)^2$$





- Input. Some entries, prior time series observations
- Output, value of time series to be forecast (one step ahead, two steps ahead,….)

Fully connected… theoretically OK, but lot to be gained from special structure

RNNs. Key ideas

- Specialized for sequential data (numbers, words, letters,….)
- Can process sequences much longer than those achievable with fully connected NNs
- More amenable to parallelisation (transformers)
- Each neuron has one (or more) ‘internal memory’ (hidden) to store info about previous entries
- Trained with variants of standard algos: backprop through time
- Created in 80’s and late 90’s, yet their recent successes (SIRI, etc…) made them fashionable.
- Latest wave: Transformers (attention is all you need), LLM, Chat-GPT,… ‘the’ architecture
- Myriad applications
  - Speech recognition
  - Language modeling and text generation
  - Automatic translation
  - Image description generation
  - Summaries, Q&A
  - Image, Video,….
  - Math proofs, Stats,…
  - Molecular description, drug Discovery,….

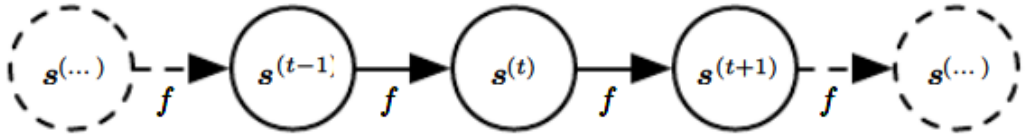
Core concept
Recurrence and computational graphs

Recurrence and unfolded computational graphs

- Dynamical system. Recurrence unfolded

$$s^{(t)} = f(s^{(t-1)}; \theta)$$

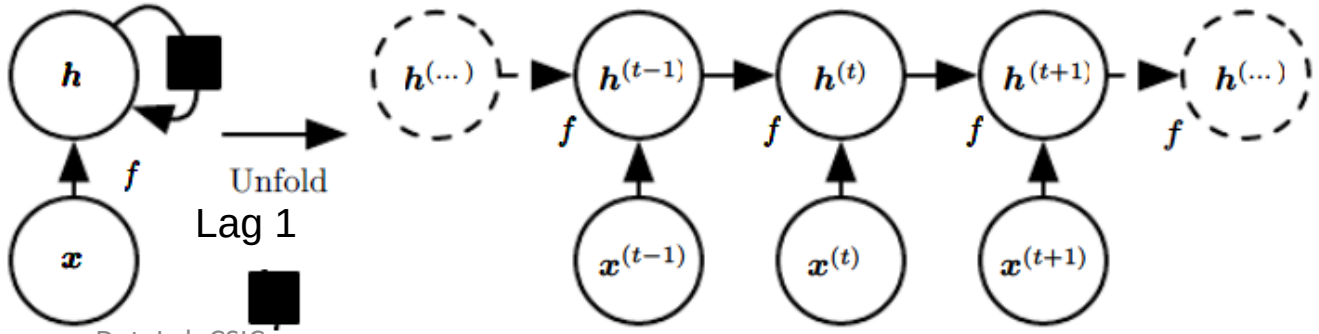
$$\begin{aligned} s^{(3)} &= f(s^{(2)}; \theta) \\ &= f(f(s^{(1)}; \theta); \theta) \end{aligned}$$



- Every recurrent function as recurrent NN h (hidden) state

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta)$$

$$\begin{aligned} h^{(t)} &= g^{(t)}(x^{(t)}, x^{(t-1)}, x^{(t-2)}, \dots, x^{(2)}, x^{(1)}) \\ &= f(h^{(t-1)}, x^{(t)}; \theta) \end{aligned}$$



Core concept
Working with text data

Goodfellow et al 12, Chollet and Allaire 6

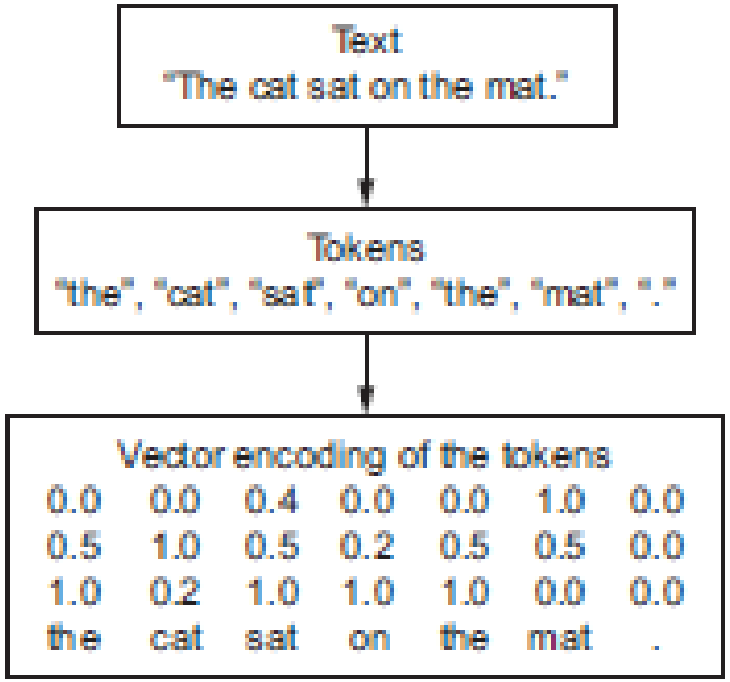
For intuitive intro
https://colah.github.io/posts/2014-07-NLP-RNNs-Representations/

Text

Sequence of characters

Sequence of words

Sequence of n-grams



Tokenize and vectorize

- Segment text into words, transform each word into a vector
- Segment text into characters, transform each character into a vector
- Extracts n-grams of words or characters, transform each n-gram into a vector

One hot encoding

Unique integer to each word

Word, vector of zeroes, except a 1 for the word

Sparse and high dimensional

Bag of words

Word embeddings

Dense word vectors

Low-dimensional, floating point vectors

Learned from data

- Jointly with the main task. Embedding layer
- Precomputed or pretrained embeddings. Word2vec, Glove,…

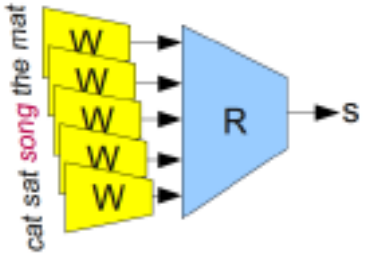
Word embedding. Example

Parametrised function $W : \text{words} \to \mathbb{R}^n$

Train to predict if a 5gram is valid

$$R(W(\text{“cat”}), W(\text{“sat”}), W(\text{“on”}), W(\text{“the”}), W(\text{“mat”})) = 1$$

$$R(W(\text{“cat”}), W(\text{“sat”}), W(\text{“song”}), W(\text{“the”}), W(\text{“mat”})) = 0$$



BERT trains by 300 bn tokens to predict the next word

RNNs

RNNs

- Emerge to process sequences, specially with different length inputs
- Add feedback connections

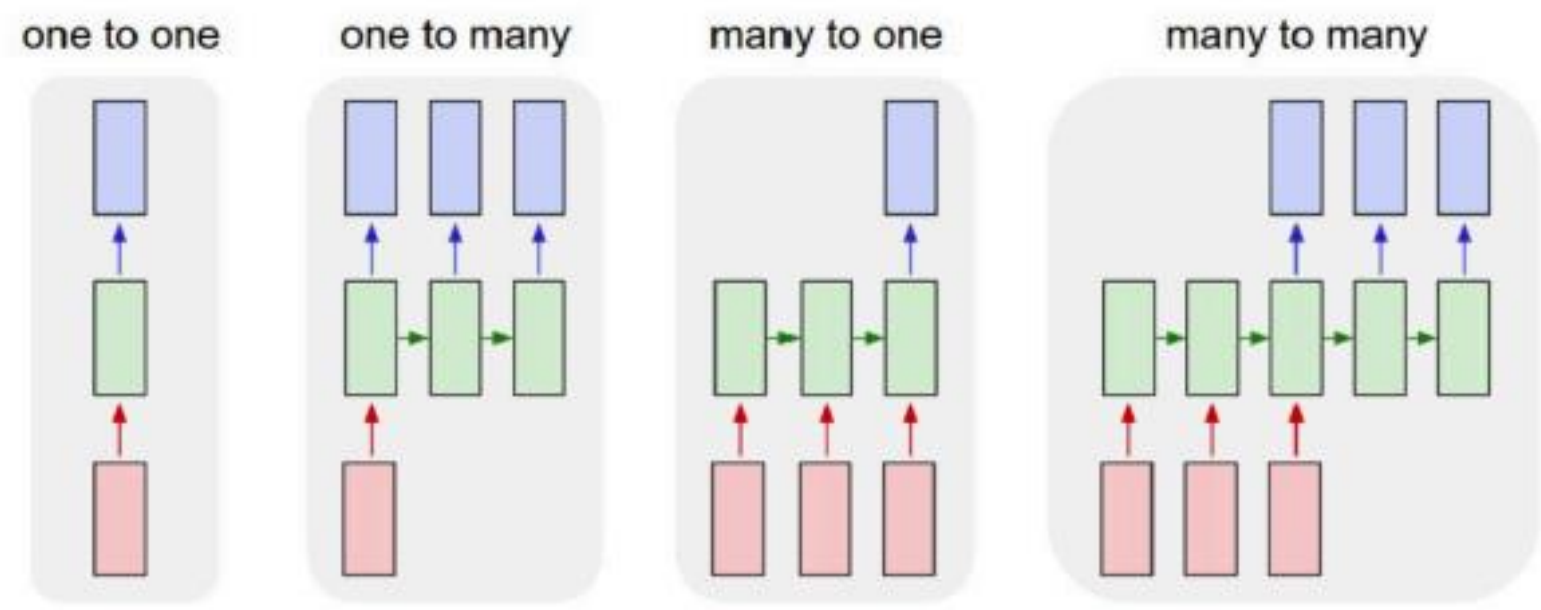


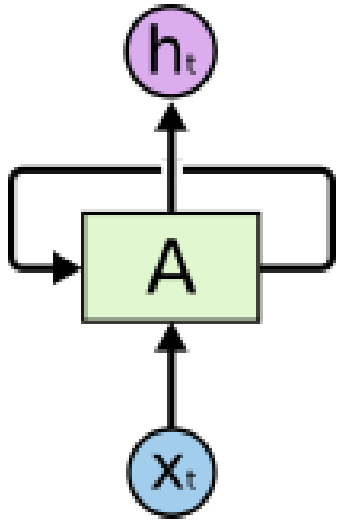
Image-Object

Image-Caption

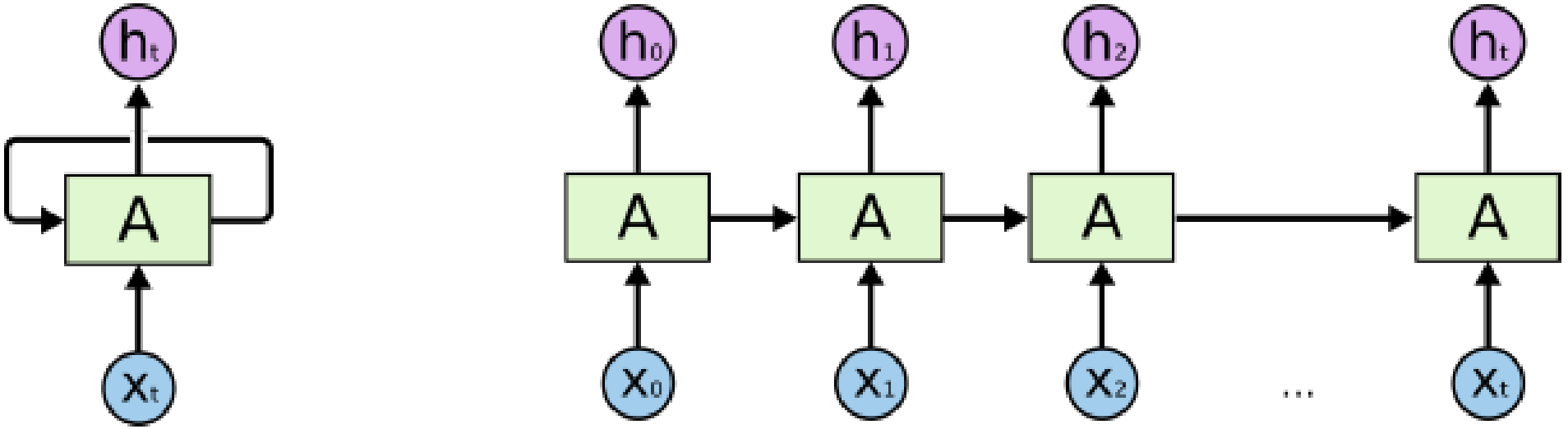
Text-Sentiment

Machine translation

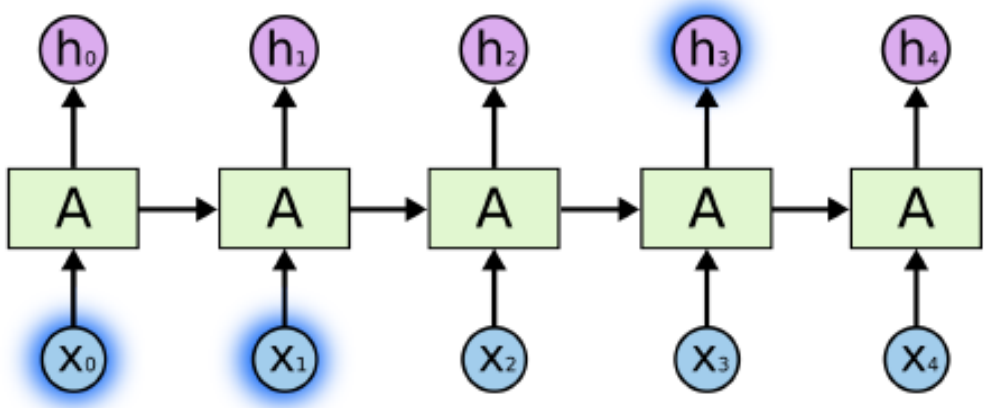
RNNs



RNNs

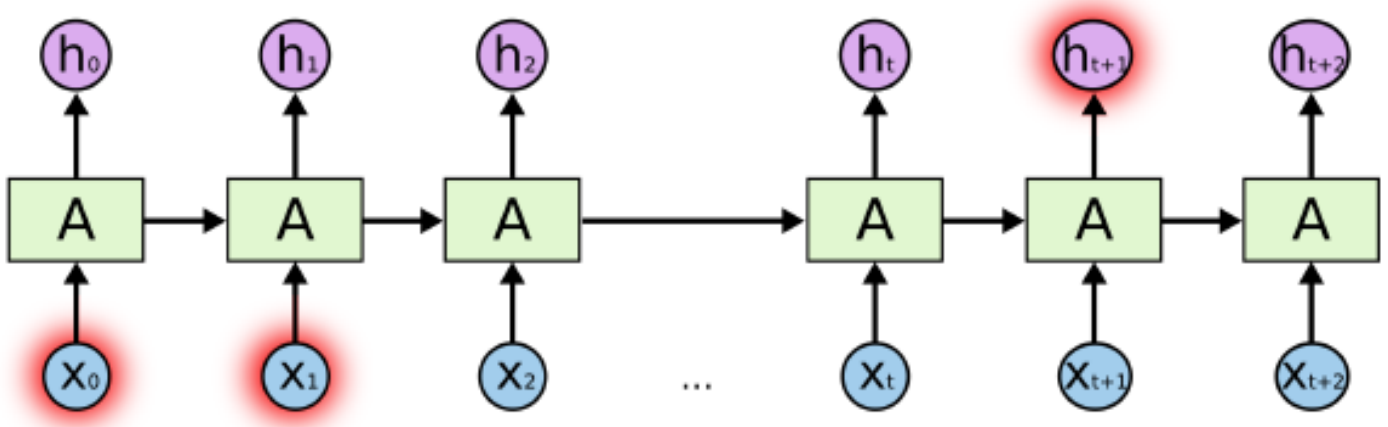


RNNs: Short-term vs Long-term dependencies



The clouds are in the …..

RNNs: Short-term vs Long-term dependencies



I grew up in France…. I speak fluent ….

RNNs. Elman's model

Network updates internal state *h* updated at each step

$$h_t = f_W(h_{t-1}, x_t)$$

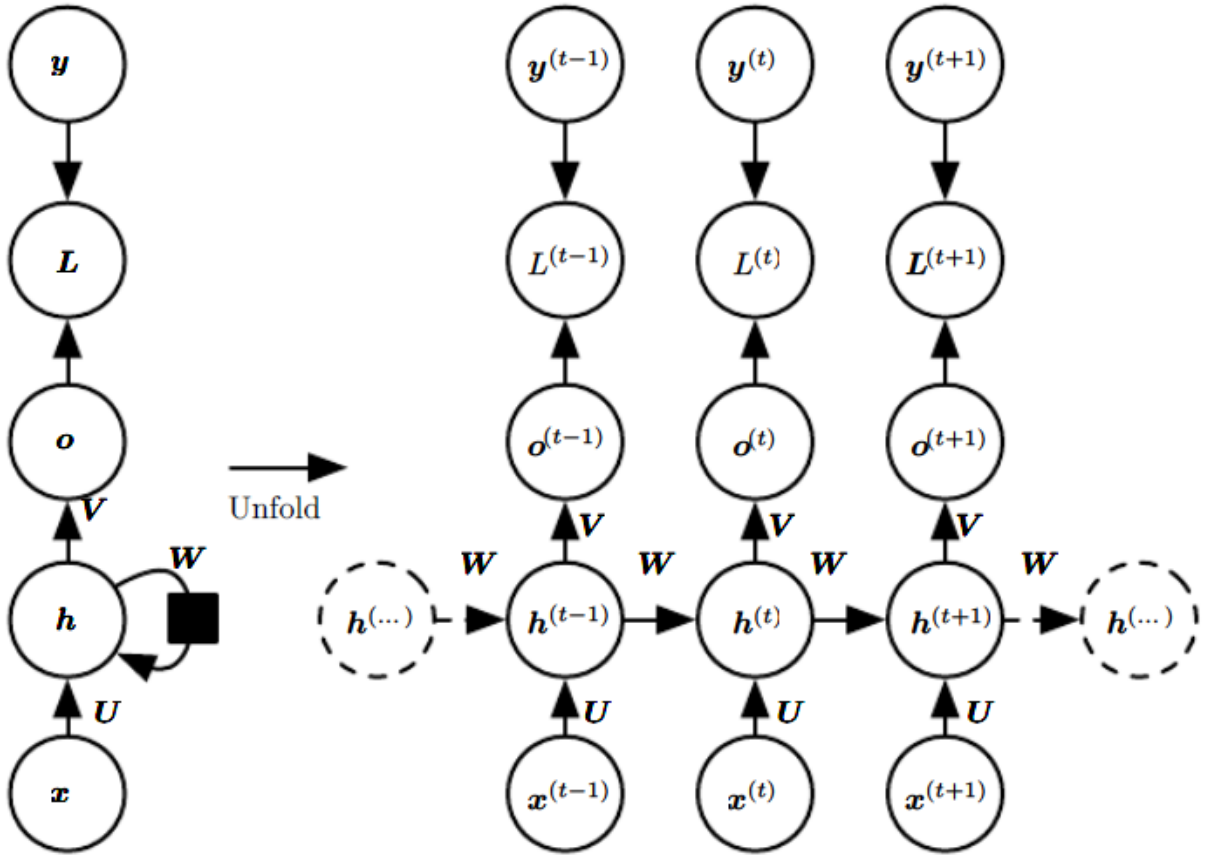
e.g.

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$y_t = W_{hy}h_t$$

Weights reused at each time:

- learn patterns independently of position
- reduction of number of parameters

RNN. One output per step, recurrence between hidden nodes



Parameters U, W, V

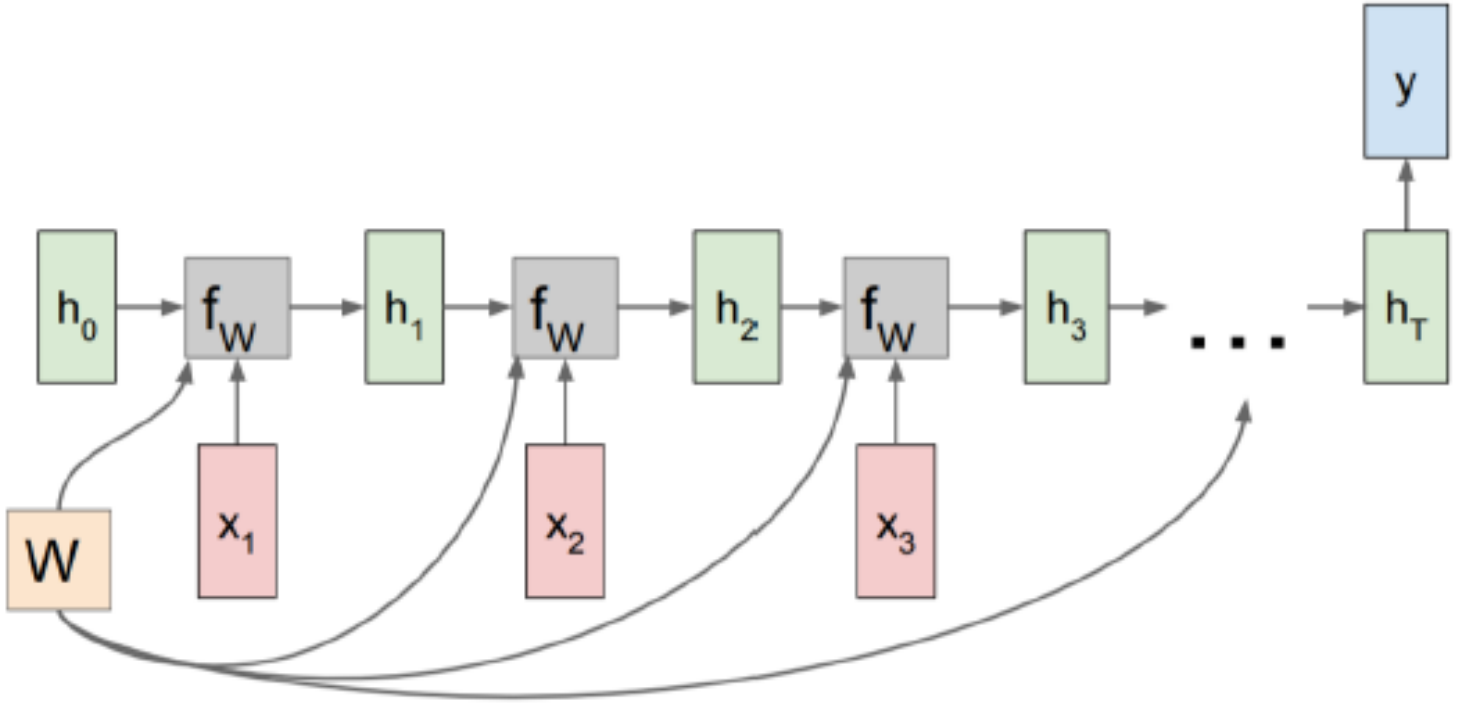
For example,

$$\begin{aligned} \boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\ \boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\ \boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\ \hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)}) \end{aligned}$$

$$\begin{aligned} & L\left(\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, \dots, \boldsymbol{y}^{(\tau)}\}\right) \\ &= \sum_t L^{(t)} \\ &= -\sum_t \log p_{\text{model}}\left(y^{(t)} \mid \{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(t)}\}\right) \end{aligned}$$

RNN: many to one example

Assigning sentiment (-,+) to a tweet



Training: Backprop through time

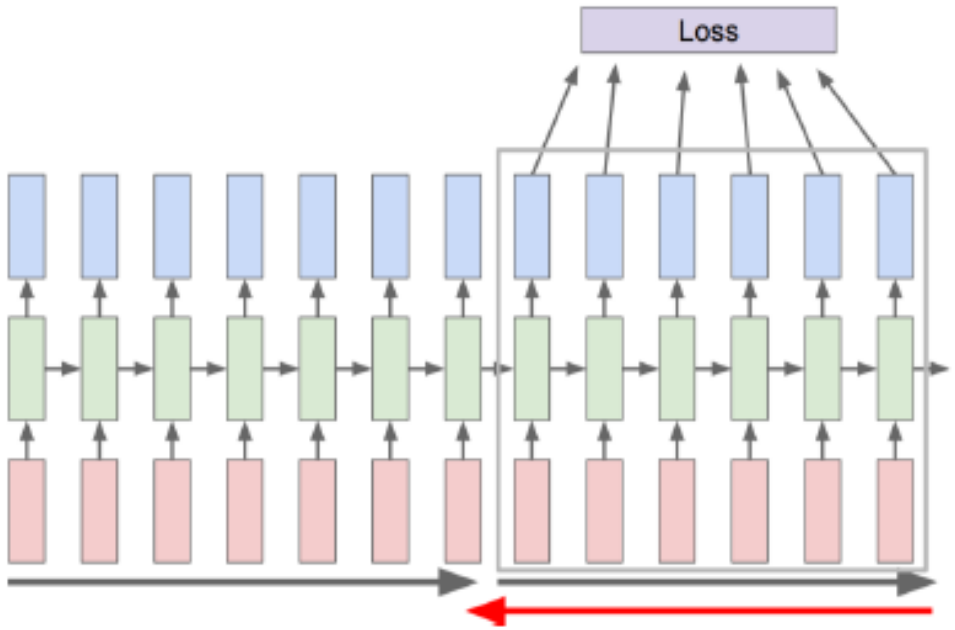
Unfolding the (computational) graph

Applying backprop

Limiting steps back for stability:

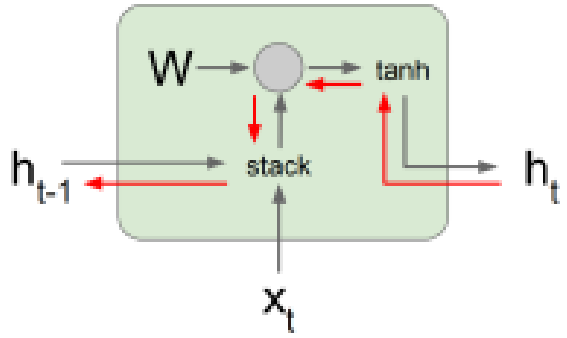
Truncated backprop

SGD or Adam or …



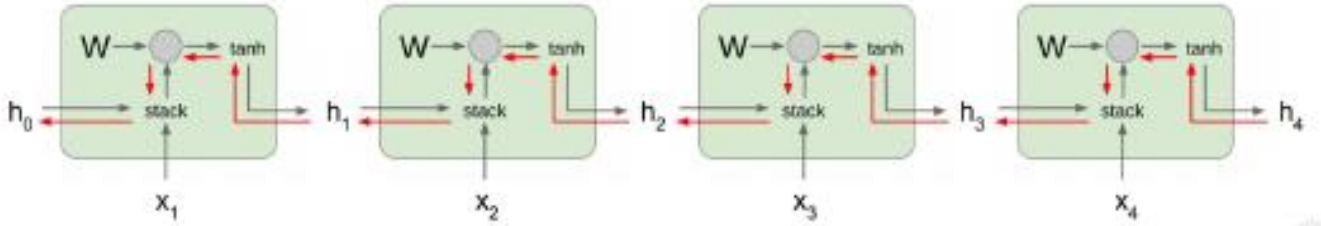
Problem with Elman's model….

Backprop one step back



$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

Over time



Repeated multiplications by W.

- If biggest eigenvalue>1, gradient explosion (gradient clipping)
- If biggest eigenvalue<1, gradient vanishing (LSTM, GRU)

$$\boldsymbol{W} = \boldsymbol{V}\mathrm{diag}(\boldsymbol{\lambda})\boldsymbol{V}^{-1}$$

$$\boldsymbol{W}^t = \left(\boldsymbol{V}\mathrm{diag}(\boldsymbol{\lambda})\boldsymbol{V}^{-1}\right)^t = \boldsymbol{V}\mathrm{diag}(\boldsymbol{\lambda})^t\boldsymbol{V}^{-1}$$

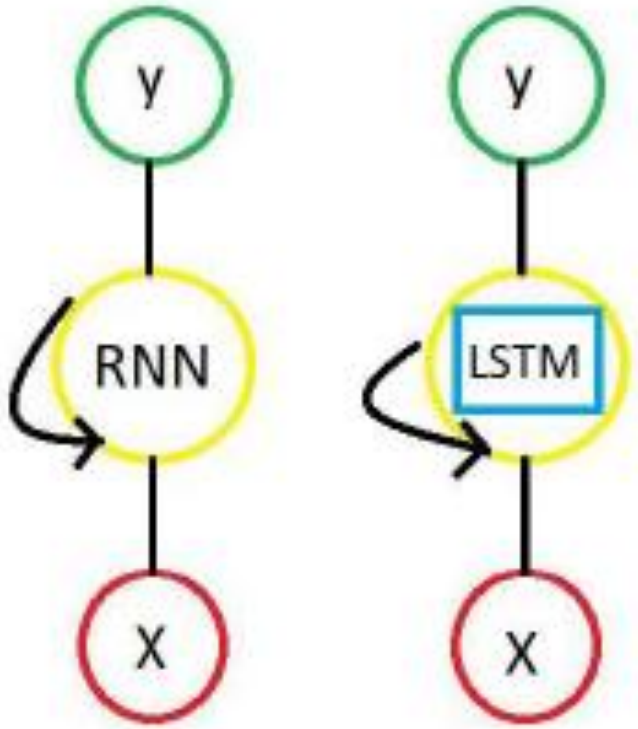
Long Short-term memory (LSTM) NNs

Hochreichter, Schmidhuber

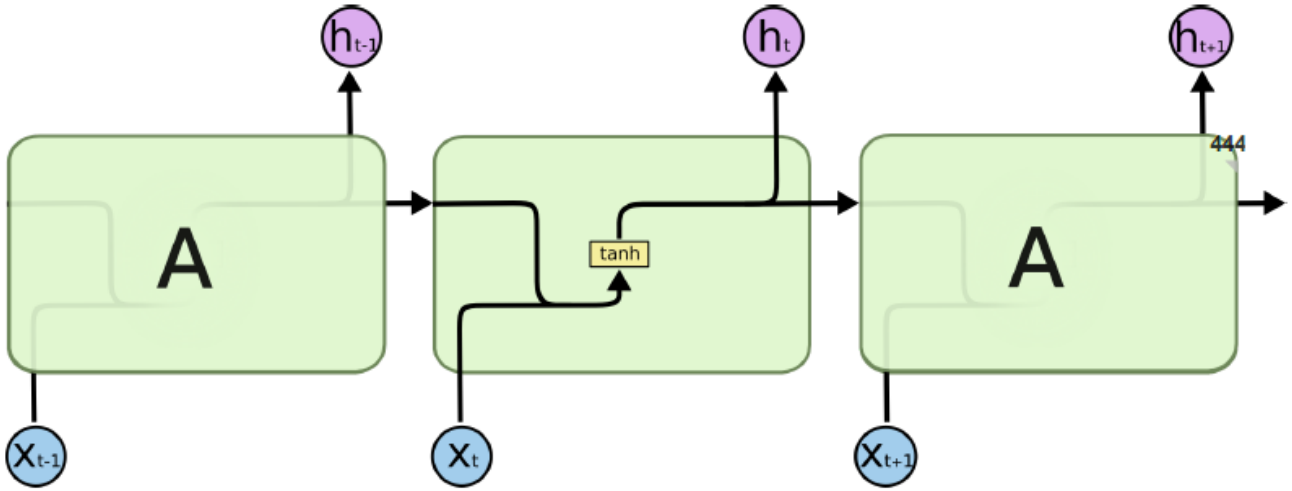
LSTM

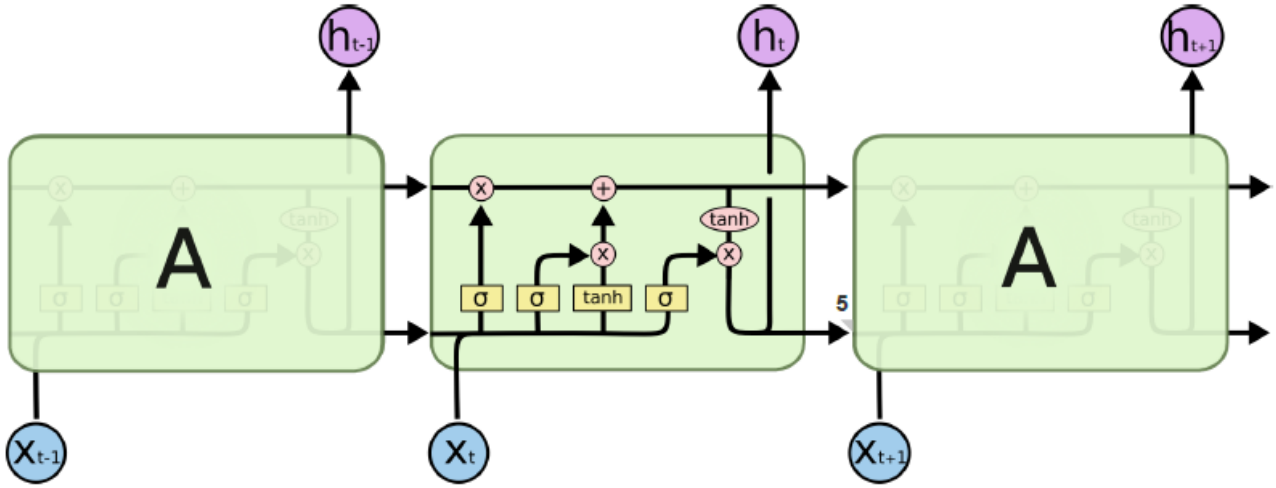
- Introduced by Hochreiter y Schmidhuber in 1997 but only used (a lot!!!) in last decade for NLP
- Hidden cells substituted by LSTM cells mitigating

vanishing and explosion



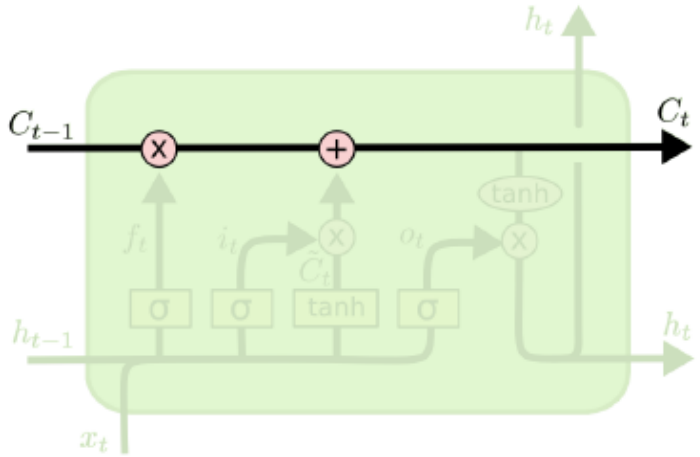
From RNNs to LSTMs



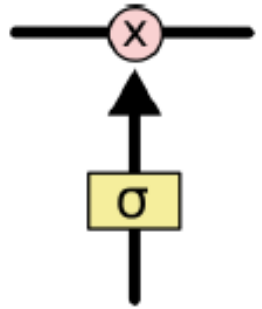


Basic ingredients

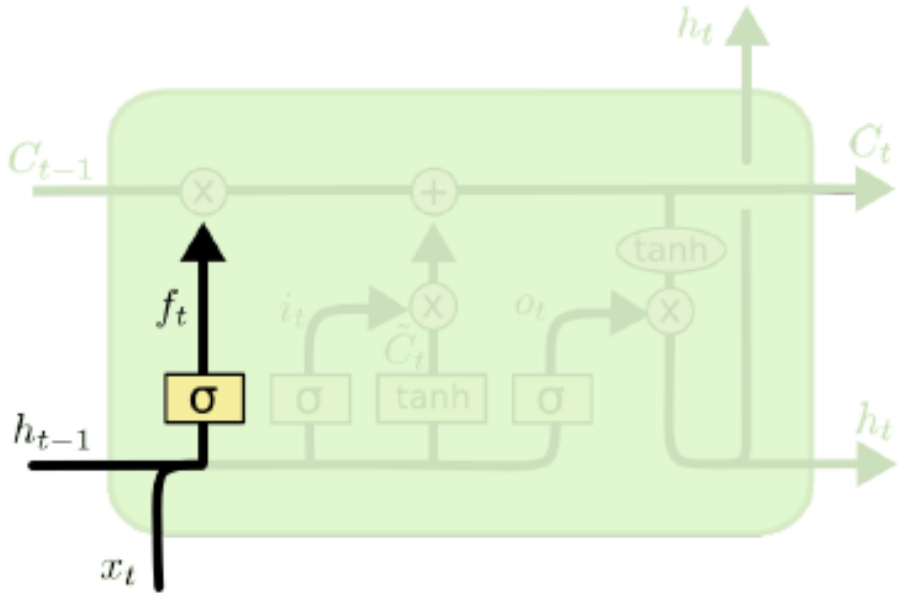
Memory Cell state (another hidden state)



Gate

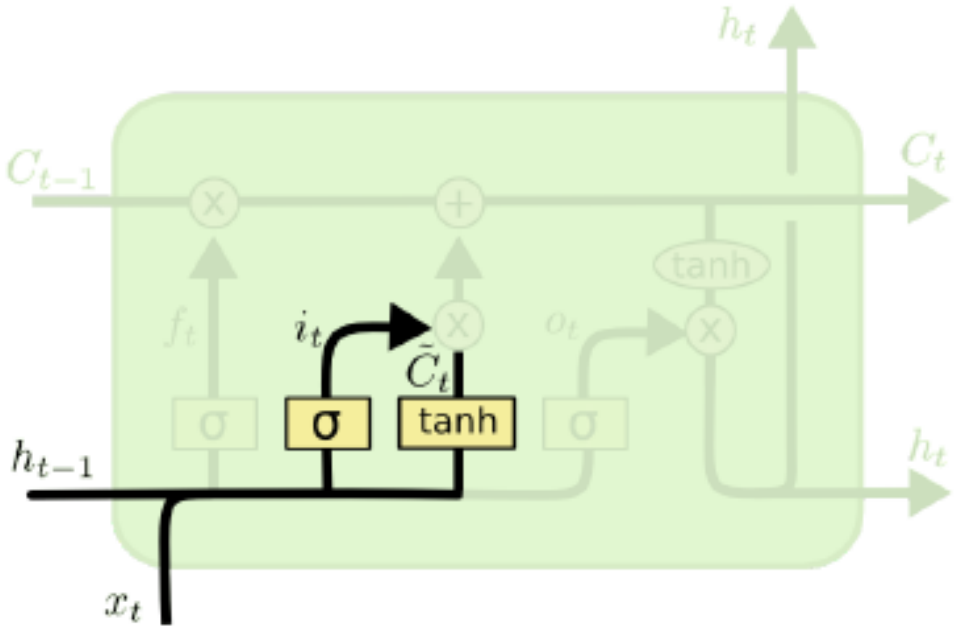


I) Info to be forgotten. Forget gate



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

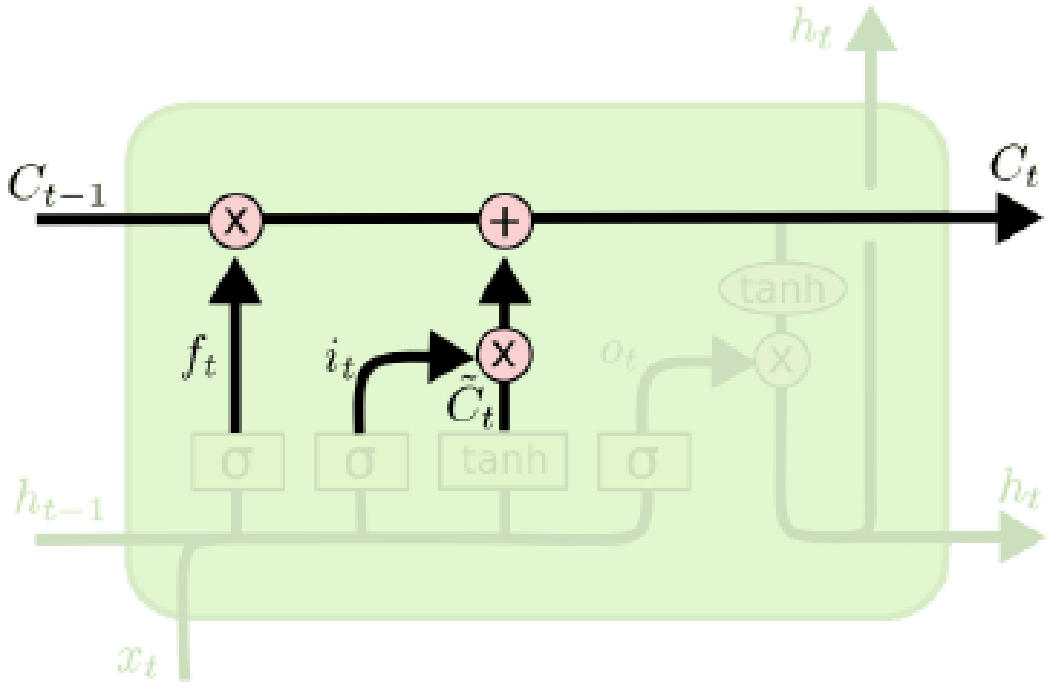
ii) Info to be stored in cell state



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

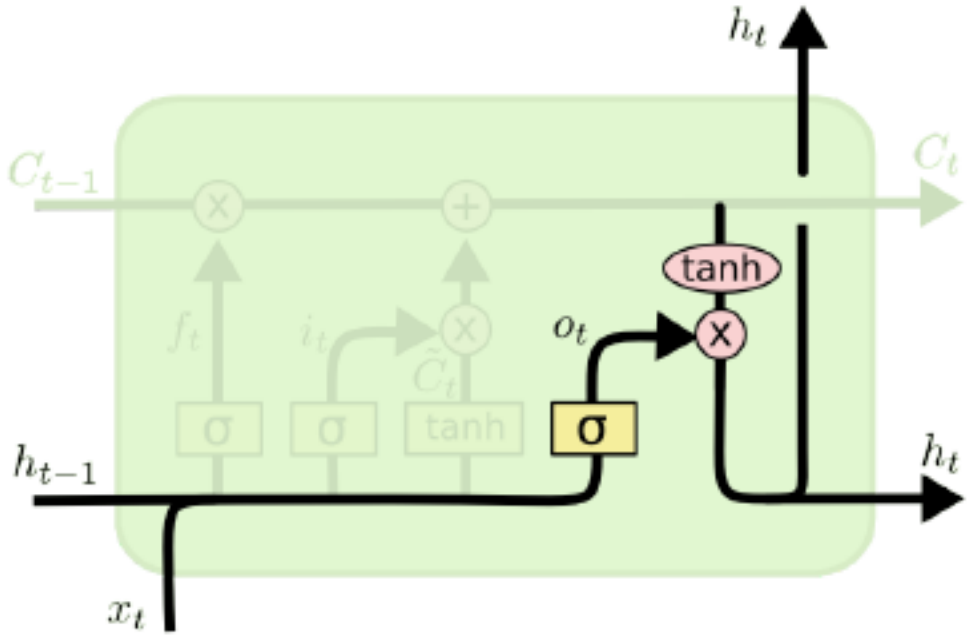
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

iii) Update cell state



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

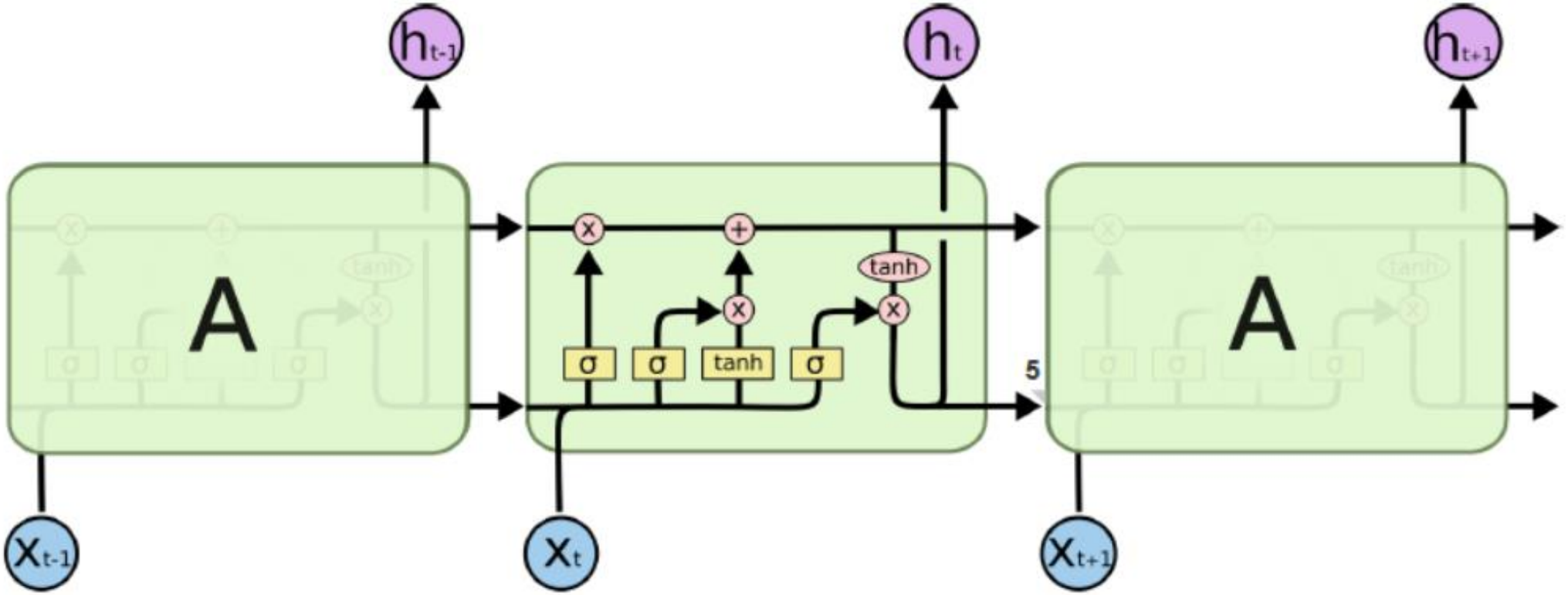
iv) Decide output (amount of content exposure)



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

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

LSTM Global scheme



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

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

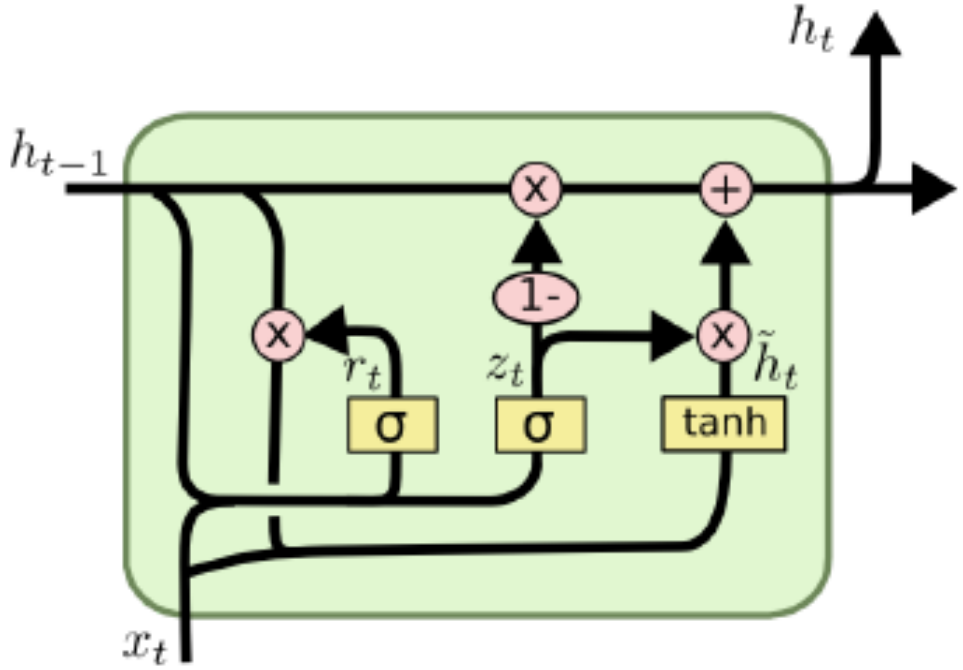
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

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

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

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

Gate recurrent unit. GRU



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Lab in R at github site

RNNs in Keras. From Lab

```
model <- keras_model_sequential()

model %>%
  layer_embedding(input_dim = vocab_size, output_dim = 4) %>%
  layer_lstm(4, return_sequences = TRUE, go_backwards=TRUE) %>%
  layer_global_average_pooling_1d() %>%
  layer_dense(units = 4, activation = "relu") %>%
  layer_dense(units = 1, activation = "sigmoid")

model %>% compile(  optimizer = 'adam',  loss = 'binary_crossentropy',  metrics = list('accuracy'))

history <- model %>% fit(  partial_x_train,  partial_y_train,  epochs = 25,  batch_size = 512,  validation_data = list(x_val,
y_val))
```

Classical RNNs in Keras (R)

layer_simple_rnn(….)
layer_lstm(…)
layer_gru(…)
bidirectional(layer_lstm(…))
layer_conv_1d(…)

OHE text_tokenizer
Layer_embedding(input_dim,output_dim)
Word2vec, GloVe

April 16th 11:30

Transformers and LLMs

Attention is all you need

(with Roi Naveiro @CUNEF site)

https://us02web.zoom.us/j/83182511916?pwd=bTdSWFY1OTJ6VWx1UTVmbFhYZktLUT09

Transformers

Videos

Transformers for poets

- https://www.youtube.com/watch?v=SZorAJ4I-sA Basic intro
- https://www.youtube.com/watch?v=_UVfwBqcnbM Detailed non-tech intro

Some technical details

- https://www.youtube.com/watch?v=S27pHKBEp30 Contextual intro

Attention is all you need. The seminal paper

https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Its annotated version

https://nlp.seas.harvard.edu/annotated-transformer/

Formal algos for transformers

https://arxiv.org/pdf/2207.09238.pdf

Software oriented explanations

https://osanseviero.github.io/hackerllama/blog/posts/random_transformer/

https://peterbloem.nl/blog/transformers