Neural Language Modelling

PA154 Language Modeling (10.1)

Pavel Rychlý

pary@fi.muni.cz

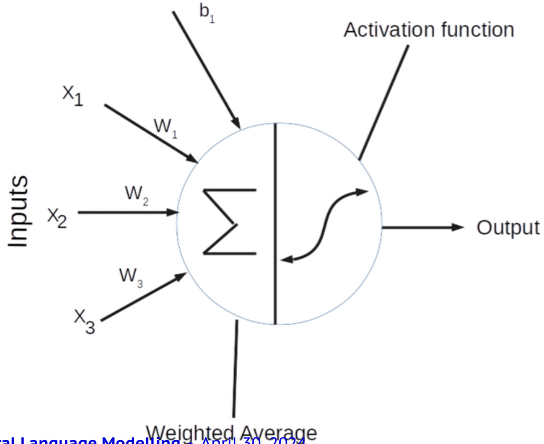
April 30, 2024

Deep Learning

- deep neural networks
- many layers
- trained on big data
- using advanced hardware: GPU, TPU
- supervised, semi-supervised or unsupervised

Neuron

- basic element of neural networks
- many inputs (numbers), weights (numbers)
- activation (transfer) function (threshold)
- one output: $y = \sigma(\sum_{j=0}^{m} w_j x_j + b)$



Neuron on an embedding

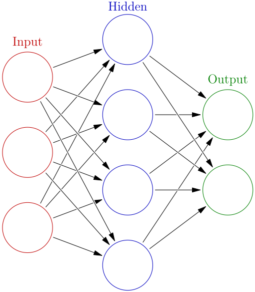
- input: embedding (vector of nubmers)
- output: $y = \sigma(\sum_{j=0}^{m} w_j x_j + b)$
- linear classifier
- hyperplane cutting the vector space
- selects one feature

Neural Networks

- Input/Hidden/Output layer
- Input/output = vector of numbers
- hidden layer = matrix of parameters (numbers)

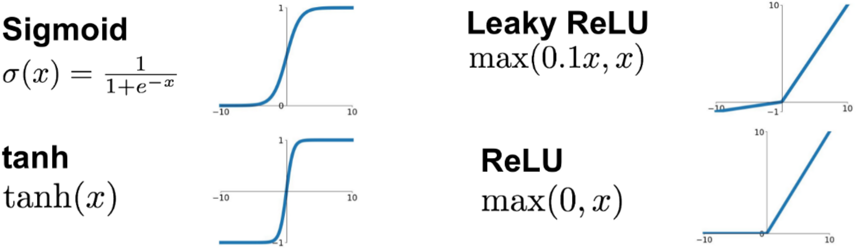
$$y_k = \sigma\left(\sum_{j=0}^{m} w_{kj} x_j\right)$$

$$Y = \sigma(WX^T)$$



Activation Functions

- crucial component of NN
- non-linear function
- many layers without non-linear activation functions are equivalent to single layer (linear combination of inputs)



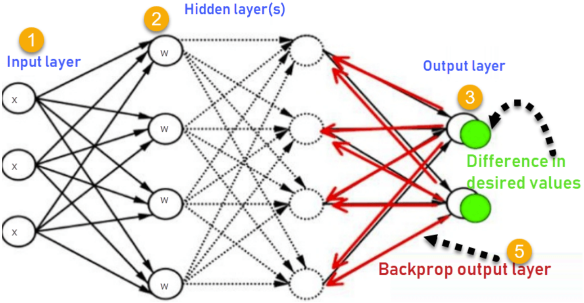
https://en.wikipedia.org/wiki/Activation_function

One-hot representation

- words/classes: `[0 0 0 1 0 0 0 0]`
- for each word/class one input
- one input activated (1), others deactivated (0)
- whole input vector could be large = size of vocabulary ($\approx$30k)
- sequence of words requires sequence of one-hot vectors
- first layer transforms words into word embeddings
- usually not represented explicitly as vectors, using one single number
- output:
  - one-hot vector during training = expecting one word/class
  - probability distribution during usage
  - cannot be represented using single number

Training Neural Networks

- supervised training
- example: input + result
- difference between output and expected result (loss function)
- adjusts weights according to a learning rule
- backpropagation (feedforward neural networks)
- gradient of the loss function, stochastic gradient descent (SGD)



Training Language Models

- core function of language models: predict the following word:

$$P(x_5|x_1, x_2, x_3, x_4)$$

- input: context = x_1, x_2, x_3, x_4
- output: probability distribution of the following word
- training:
  - input: x_1, x_2, x_3, x_4
  - output: x_5 (one-hot vector)
- training data:
  - get any text (corpus)
  - extract all n-grams

Why are NNs better than statistics

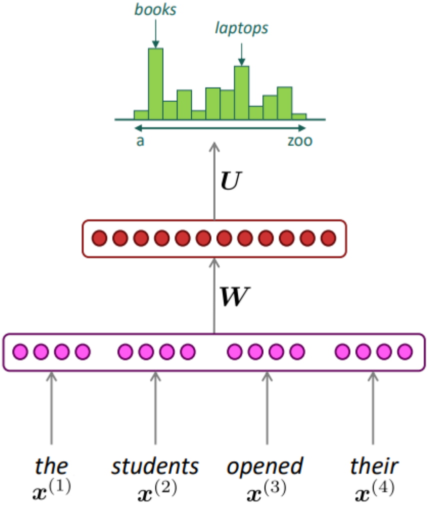
- continues space representation
  - items (words/tags/classes) are not atomic
  - no sparsity problem
  - don't need to store all observed n-grams
  - vectors handles relations
  - many realations, not explicit, unknown
- NN can represent any function (if deep enough)
  - structure of the function is not pre-defined

Why are NNs used only last 10 years

- big training data
- powerful hardware
  - Moore's law: memory size, processor's speed doubles every few years
  - matrix processing using GPU, TPU
- better learning strategies, NN optimizatons
  - Adam, AdaFactor optimizers
  - dropout
  - attention
- ready to use libraries/framewoks, datasets

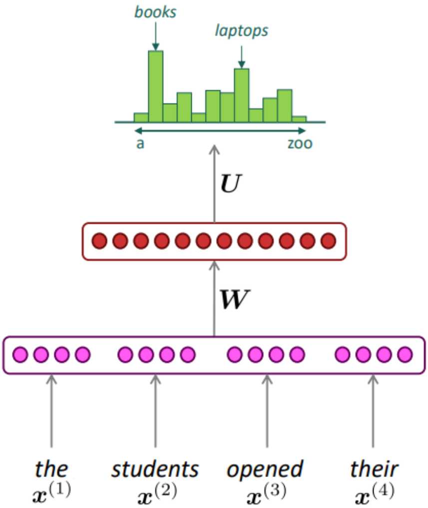
A fixed-window neural Language Model

- v = vocabulary size
- d = embedding size
- h = hidden layer size
- Input: IDs of words (sparse representation of one-hot vector)
- E: embeddings ($v \times d$)
- W: hidden layer ($4d \times h$)
- U: output layer ($h \times v$)



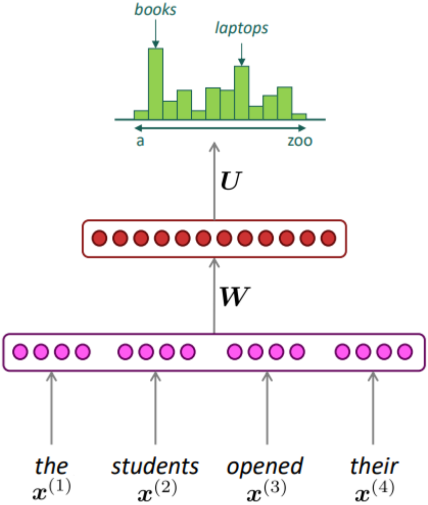
A fixed-window neural Language Model

- E: embeddings ($v \times d$)
- W: hidden layer ($4d \times h$)
- U: output layer ($h \times v$)
- $A = E(X)$
- $B = f(WA)$
- $Z = softmax(UB)$



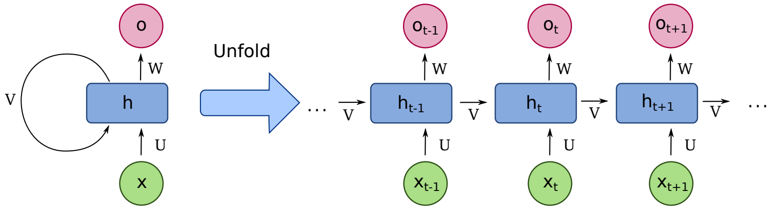
A fixed-window neural Language Model

- fixed window is too small
- enlarging window enlarges W
- window can never be large enough!
- x_1 and x_2 are multiplied by completely different weights in W.
- No symmetry in how the inputs are processed.



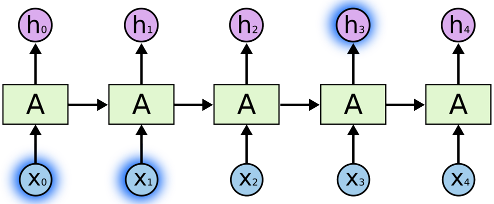
Recurrent Neural Network (RNN)

- dealing with long inputs
- feedforward NN + internal state (memory)
- finite impulse RNN: unroll to strictly feedforward NN
- infinite impulse RNN: directed cyclic graph
- additional storage managed by NN: gated state/memory
- backpropagation through time

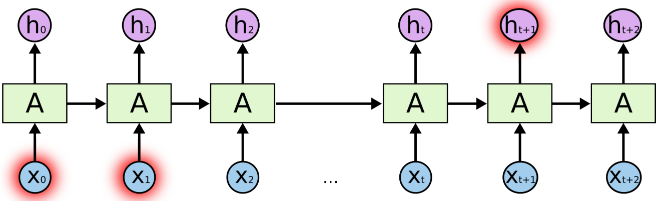


Problem of Long-Term Dependencies

- *the clouds are in the* ***sky*** *.*

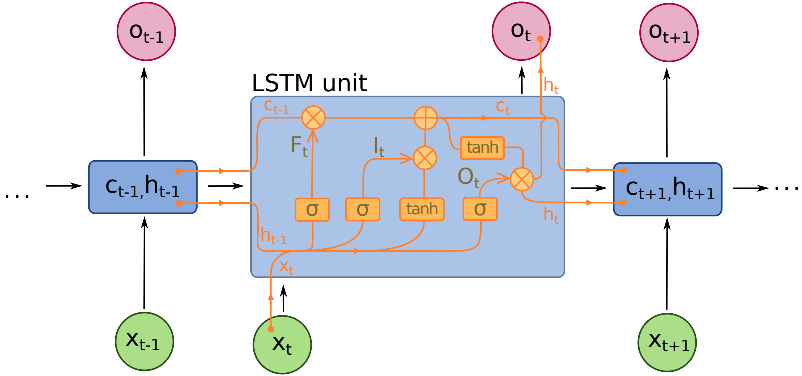


- *I grew up in France... I speak fluent* ***French*** *.*

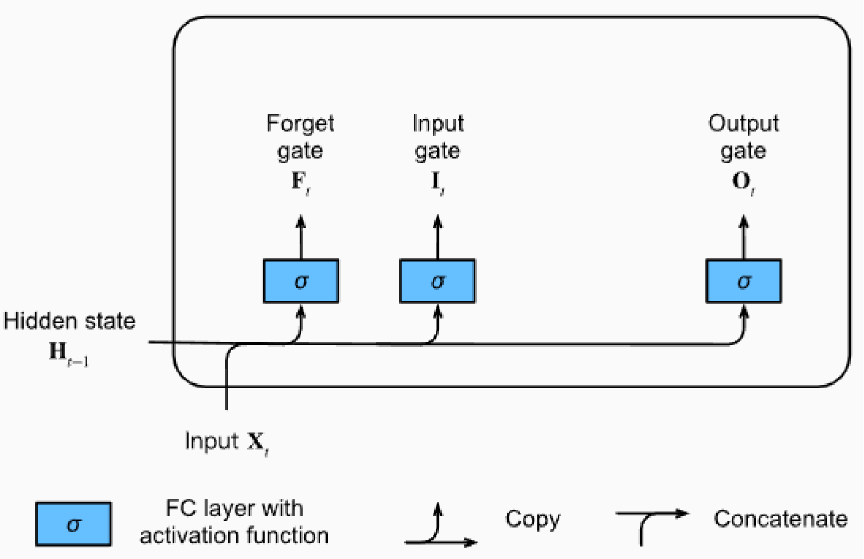


Long short-term memory (LSTM)

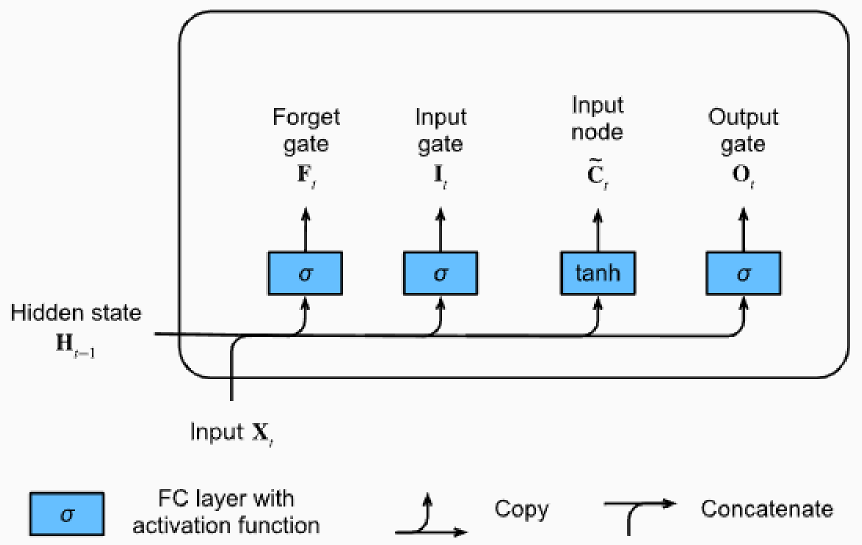
- LSTM unit: cell, input gate, output gate and forget gate
- cell = memeory
- gates regulate the flow of information into and out of the cell



LSTM – Gates

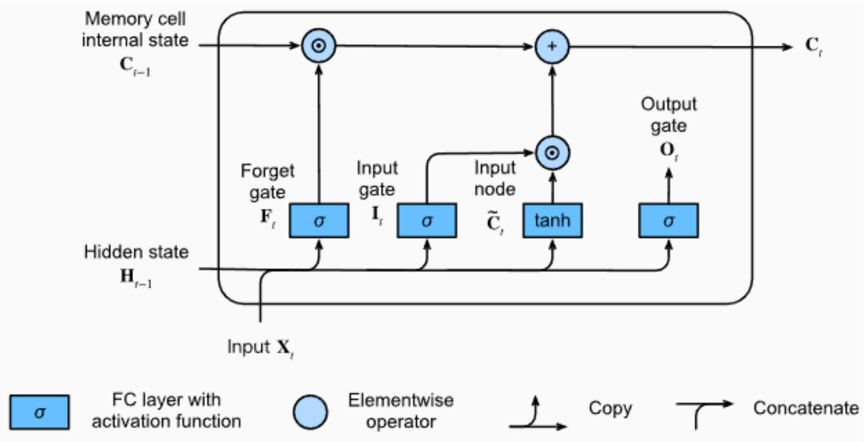

Forget gate $\mathbf{F}_t$
Input gate $\mathbf{I}_t$
Output gate $\mathbf{O}_t$
σ
Hidden state $\mathbf{H}_{t-1}$
Input $\mathbf{X}_t$
FC layer with activation function
Copy
Concatenate

LSTM – Input



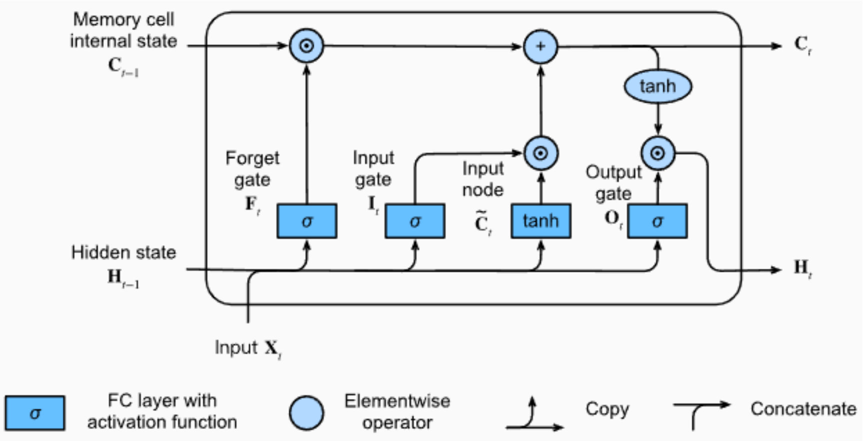
Forget gate
$\mathbf{F}_t$
Input gate
$\mathbf{I}_t$
Input node
$\tilde{\mathbf{C}}_t$
Output gate
$\mathbf{O}_t$
σ
σ
tanh
σ
Hidden state
$\mathbf{H}_{t-1}$
Input $\mathbf{X}_t$
σ
FC layer with activation function
Copy
Concatenate

LSTM – Memory



Memory cell
internal state
$\mathbf{C}_{t-1}$
$\mathbf{C}_t$
Output
gate
$\mathbf{O}_t$
Forget
gate
$\mathbf{F}_t$
Input
gate
$\mathbf{I}_t$
Input
node
$\tilde{\mathbf{C}}_t$
σ
tanh
Hidden state
$\mathbf{H}_{t-1}$
Input $\mathbf{X}_t$
FC layer with
activation function
Elementwise
operator
Copy
Concatenate

LSTM – Hidden state

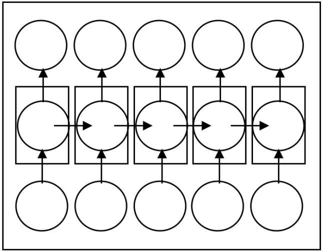


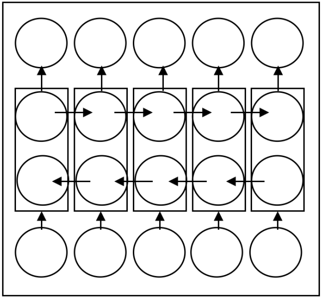
GRU, BRNN, ...

- Gated recurrent unit (GRU)
- fewer parameters than LSTM
- memory = output

- Bi-directional RNN
- two hidden layers of opposite directions to the same output

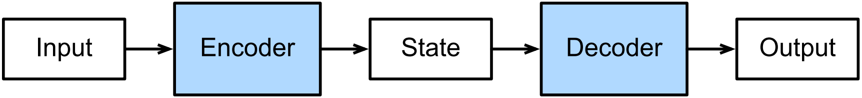
- hierarchical, multilayer





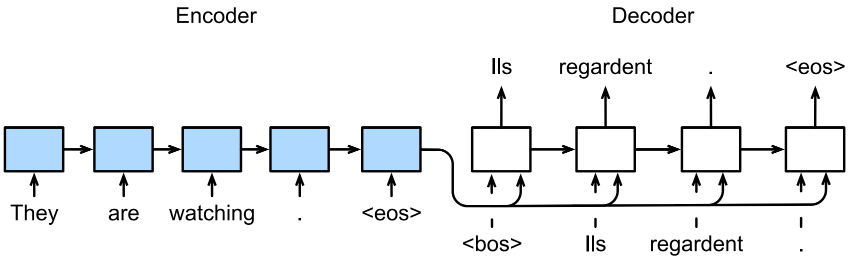
Encoder-Decoder

- variable input/output size, not 1-1 mapping
- two components
- Encoder: variable-length sequence $\rightarrow$ fixed size state
- Decoder: fixed size state $\rightarrow$ variable-length sequence



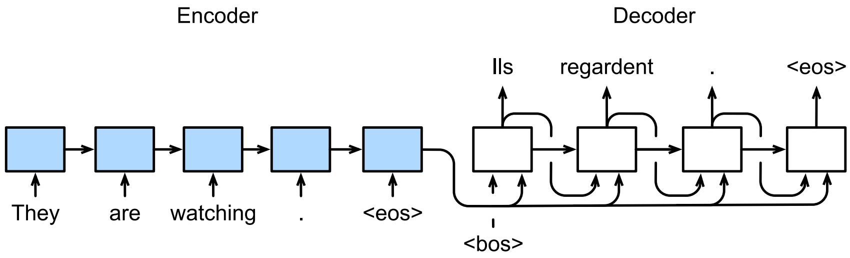
Sequence to Sequence

- Learning
  - Encoder: Input sequence $\rightarrow$ state
  - Decoder: state + output sequence $\rightarrow$ output sequence



Sequence to Sequence

- Using
  - Encoder: Input sequence $\rightarrow$ state
  - Decoder: state + sentence delimiter $\rightarrow$ output



Multi-layer encoder/decoder