Faculty of Computer Science, Dalhousie University *6-Nov-2024*

CSCI 4152/6509 — Natural Language Processing

Lecture 17: Neural Networks and NLP

Location: Carleton Tupper Building Theatre C Instructor: Vlado Keselj
Time: 16:05 – 17:25

Previous Lecture

- Message calculation: 4 cases
- Inference tasks using message passing
  1. Marginalization with one variable
  2. Marginalization with multiple variables
  3. Conditioning with one variable
  4. Conditioning with multiple variables
  5. Completion in general
- Product-sum algorithm example 1
  - Conditioning with one variable in the “burglar-earthquake” example
- Product-sum algorithm example 2
  - Completion in the HMM example with POS Tagging

17 Neural Network Models

Slide notes:

Neural Networks and Deep Learning

- Neural Network and Deep Learning models attracted a lot of attention lately, especially in the NLP area
- They have shown great or promissing results in the areas such as:
  - word embedding (semantic word embedding in vector space)
  - language modelling
  - machine translation
  - speech recognition
  - other: classification, sequence tagging, question answering, etc.
- Hype mixed with tangible results, but they have clearly become important part of NLP

Slide notes:

Popularity of Deep Learning Models for NLP

- Artificial Neural Networks research, 1958 perceptron
- Backpropagation training 1986
- Neural Networks used since then but no significant success in NLP
- Important milestone: AlexNet winning ImageNet competition on Sep 30, 2012
- word2vec 2013, Mikolov et al. at Google
- Development of larger models since then

Slide notes:

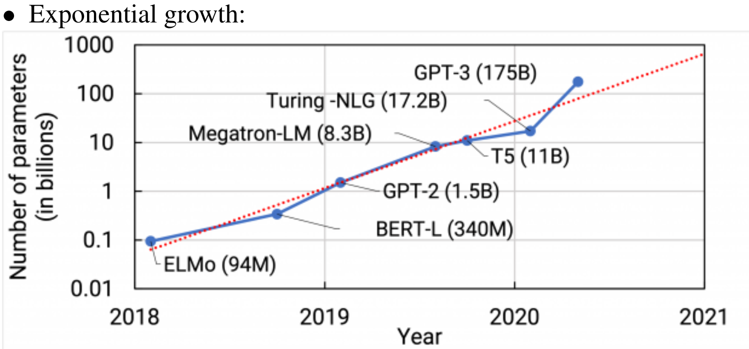
Large Deep Learning Models

- ELMo (Embedding from Language Model) 2018 by Allen Institute for Artificial Intelligence and University of Washington, 94mil parameters
- BERT (Bidirectional Encoder Representations from Transformers) 2018 by Google, 340mil par.
- GPT-2 by OpenAI in 2019, 1.5bil. param.
- Megatron-LM bu NVIDIA, 8.3bil. param.
- Turing-NLG by Microsoft, 17.2bil. param.
- GPT-3 in 2020 by OpenAI, 175bil. param.
- Exponential growth in number of parameters

Slide notes:

Deep Learning Language Model Sizes

- Exponential growth:



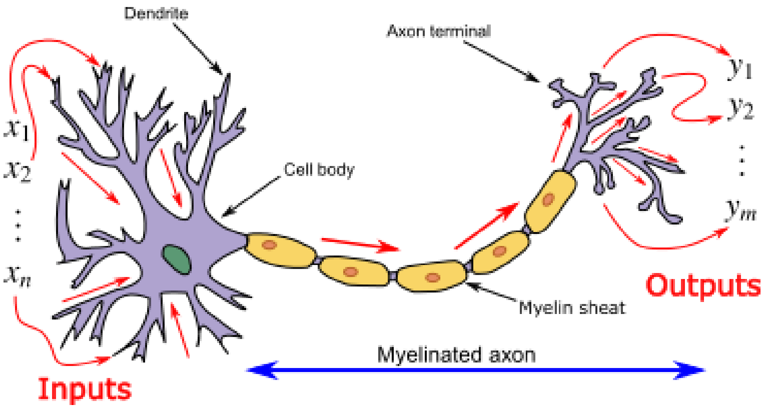
Slide notes:

Deep Learning Language Models

- These are pre-trained language models
- Used to generate text given a start
- With additional training, have potential to solve a range of NLP tasks
- Models are trained on very large text collected from Internet typically
  - E.g., GPT-3 is trained on 499 billion tokens
  - Wikipedia included with only 3 billion tokens
- Models train to simply predict next word, given previous words

Slide notes:

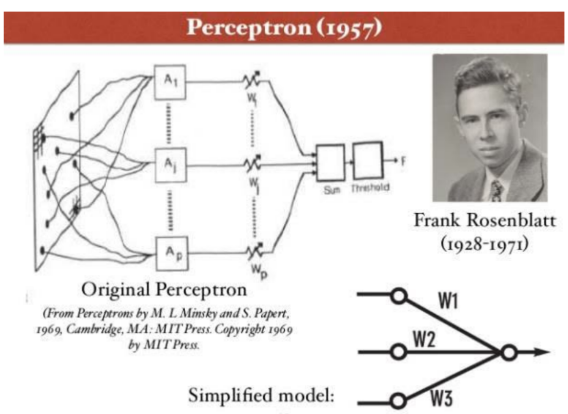
Biological Neuron



By Egm4313.s12 (Prof. Loc Vu-Quoc) - Own work, CC BY-SA 4.0,
`https://commons.wikimedia.org/w/index.php?curid=72816083`

Slide notes:

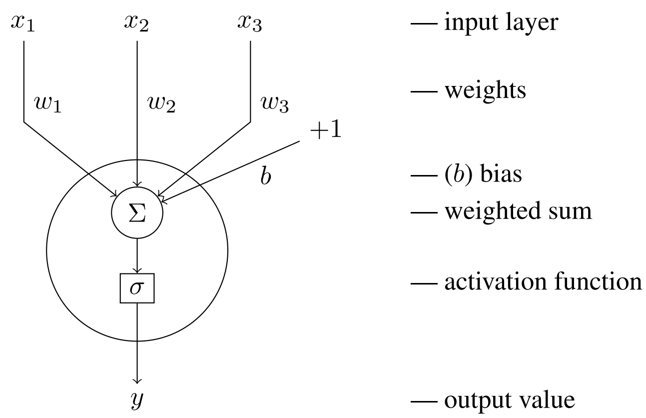
Traditional Perceptron (Artificial Neuron)



`https://www.simplilearn.com/what-is-perceptron-tutorial`

Slide notes:

Computation in Artificial Neuron (Perceptron)



$$y = \sigma(b + \sum_i x_i w_i) = \sigma(b + x_1 w_1 + x_2 w_2 + x_3 w_3)$$

Slide notes:

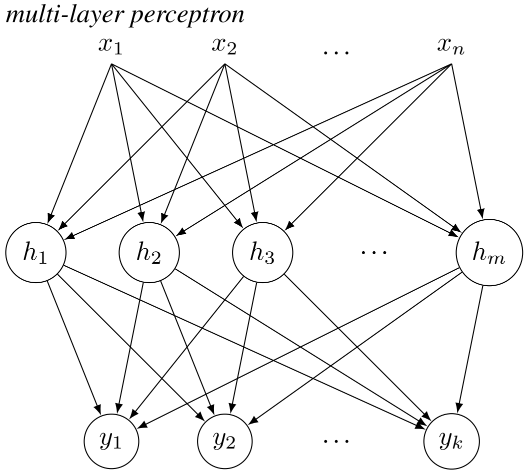
Perceptron Properties

- Biological neurons would imply activation function (non-linear transform) to be step function, or at least monotonically non-decreasing
- Could use identity function or linear function, but not a good idea
- If used as classifier ($y \geq 0$ or $y < 0$), similar to Naïve Bayes, SVM (Support Vector Machines), and logistic regression
  - linear separability
- Connected to make Neural Networks (brain analogy)

Slide notes:

Feedforward Neural Network

also called *multi-layer perceptron*



Slide notes:

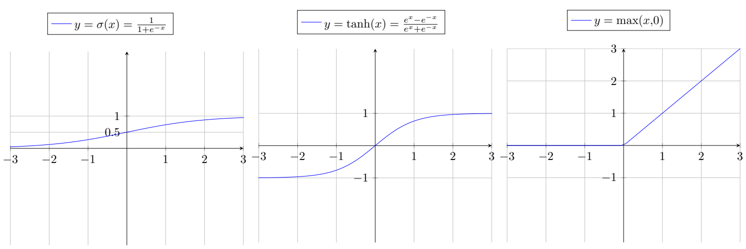
Activation Function

- must be non-linear
  - otherwise, the whole neural network would collapse into one neuron
- should be monotonically non-decreasing
- useful to be differentiable and relatively simple for speed of training
- Best known activation functions: sigmoid, tanh, ReLU (Rectified Linear Unit)

Slide notes:

Common Activation Functions

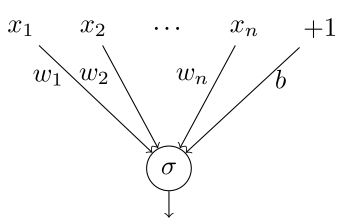
Sigmoid tanh ReLU



Slide notes:

Binary Classification with One Layer

- same as binary logistic regression

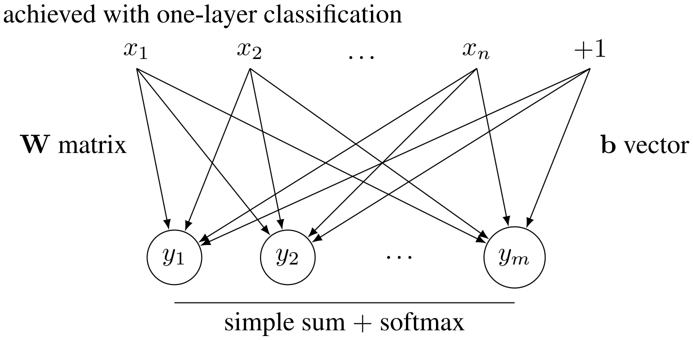


$$y = \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + \boldsymbol{b})$$

Slide notes:

Multinomial Logistic Regression

- achieved with one-layer classification



$$\mathbf{y} = \text{softmax}(\mathbf{W}\mathbf{x} + \mathbf{b})$$

Slide notes:

Softmax Function

- Softmax transforms numbers into positive domain using e^x; i.e., $\exp(x)$, function, and normalizing numbers into a probability distribution

$$\text{softmax}(\mathbf{x}) = [\frac{\exp(x_1)}{\sum_{i=1}^n \exp(x_i)}, \frac{\exp(x_2)}{\sum_{i=1}^n \exp(x_i)}, \ldots \frac{\exp(x_n)}{\sum_{i=1}^n \exp(x_i)}]$$

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^n \exp(x_j)}$$

- Example from Jurafsky and Martin:

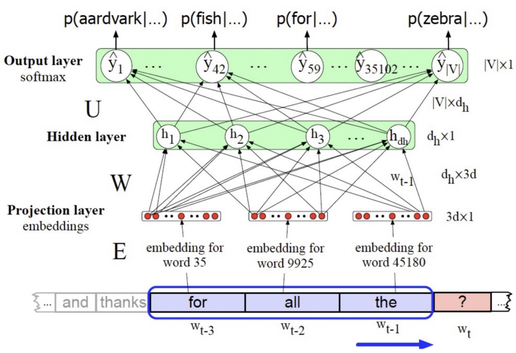
$$\mathbf{x} = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1]$$

$$\text{softmax}(x) = [0.055, 0.09, 0.006, 0.099, 0.74, 0.01]$$

Neural Language Model

Slide notes:

Neural Language Model

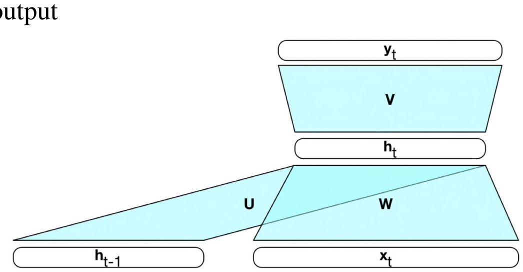


(Jurafsky and Martin)
The model has limited history, similarly to n-gram model

Slide notes:

Recurrent Neural Networks (RNN)

- Simple recurrent neural network presented as a feedforward network (Jurafsky and Martin, Figure 9.3)
- RNN is trained as a Language model by providing the next word as output



Slide notes:

RNN Unrolled in Time

– RNN unrolled in time; more clear view of training (Jurafsky and Martin, Figure 9.5)