Neural Networks

Intro

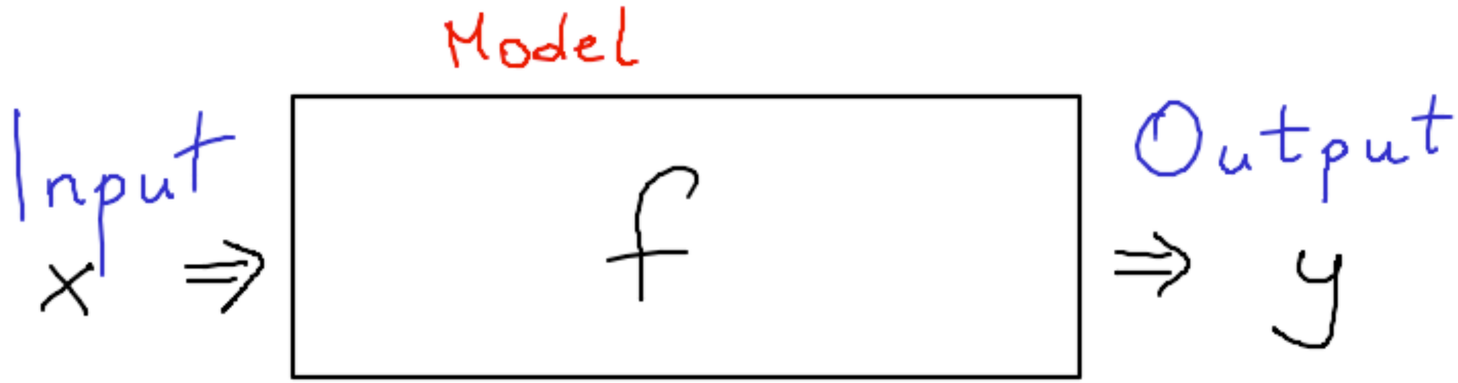
Rafał Nowak Feb 28, 2023

Input data

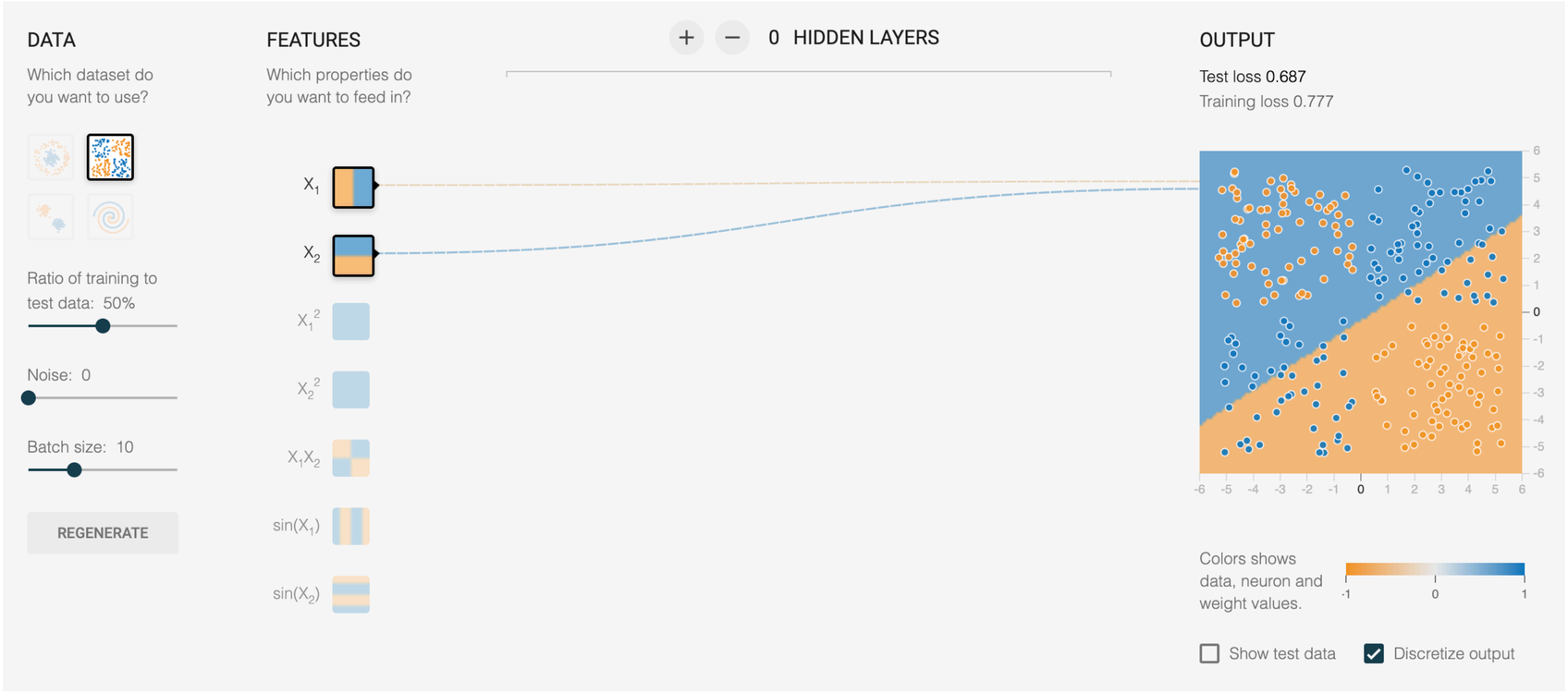
- structured
  - tabular (json, csv, …)
- unstructured
  - images, video, text (natural language)

Computation graph

- neural network = parametrized, non-linear function

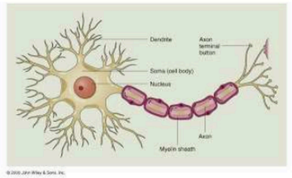


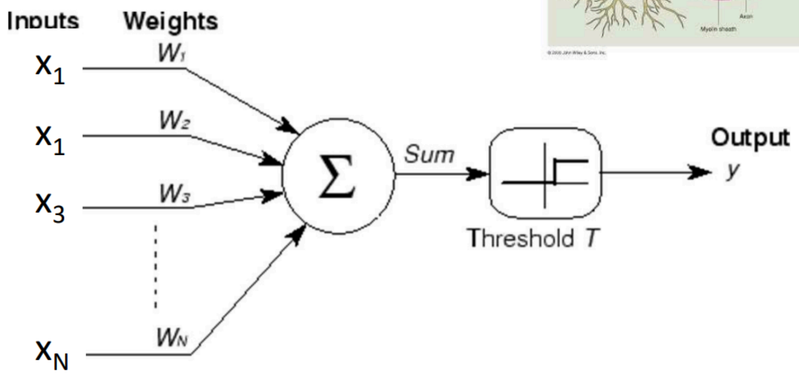
Linear model



source: Playground Tensorflow

Recap: the perceptron



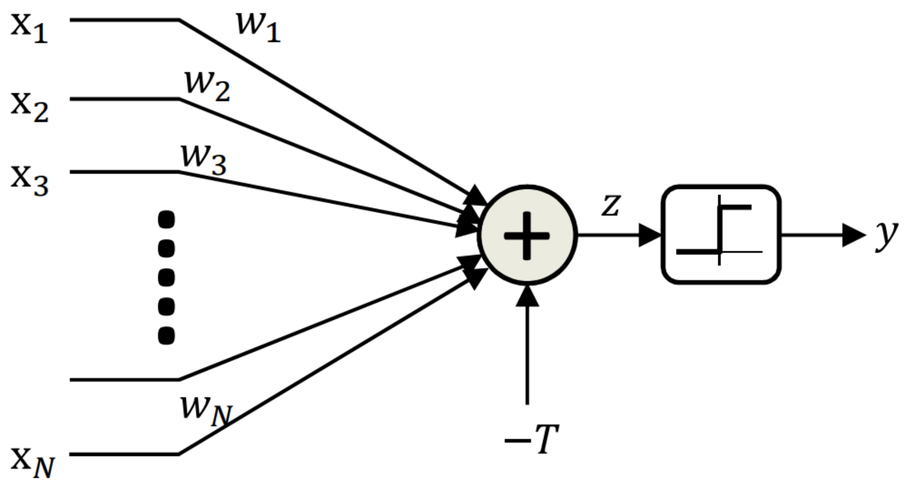


$$y = \begin{cases} 1 \; if \; \sum_i w_i x_i \geq T \\ 0 \; else \end{cases}$$

- A threshold unit
  - "Fires" if the weighted sum of inputs exceeds a threshold
  - Electrical engineers will call this a *threshold gate*
    - A basic unit of Boolean circuits

https://www.cs.cmu.edu/~bhiksha/courses/deeplearning/Spring.2019/www/slides.spring19/lec2.universal.pdf

A better figure

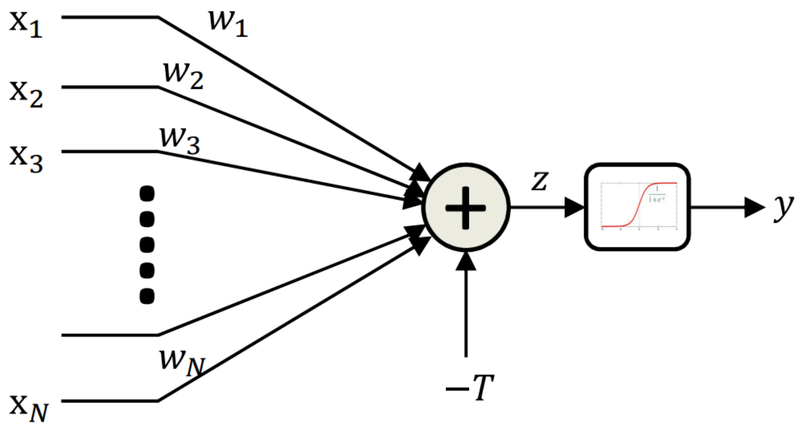


$$z = \sum_i w_i x_i - T$$

$$y = \begin{cases} 1 \; if \; z \geq 0 \\ 0 \; else \end{cases}$$

- A threshold unit
  - "Fires" if the weighted sum of inputs and the "bias" T is positive

The “soft” perceptron (logistic)



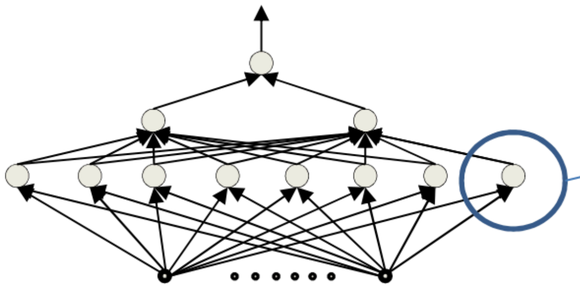
$$z = \sum_i w_i x_i - T$$

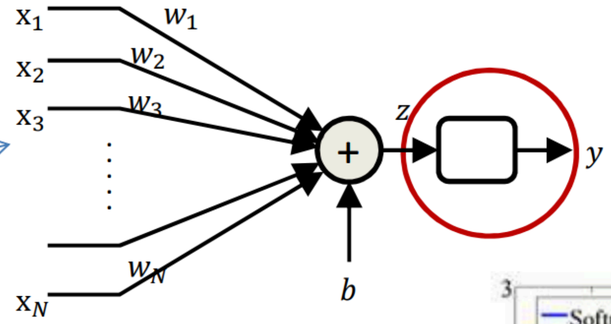
$$y = \frac{1}{1 + exp(-z)}$$

- A “squashing” function instead of a threshold at the output
  - The sigmoid “activation” replaces the threshold
    - **Activation:** The function that acts on the weighted combination of inputs (and threshold)

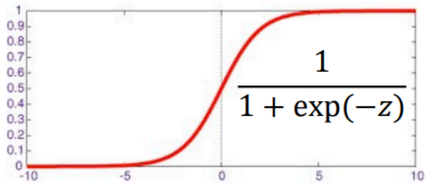
https://www.cs.cmu.edu/~bhiksha/courses/deeplearning/Spring.2019/www/slides.spring19/lec2.universal.pdf

Other “activations”

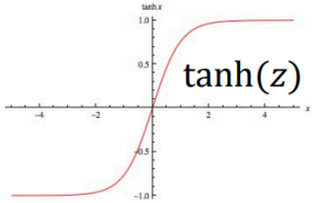


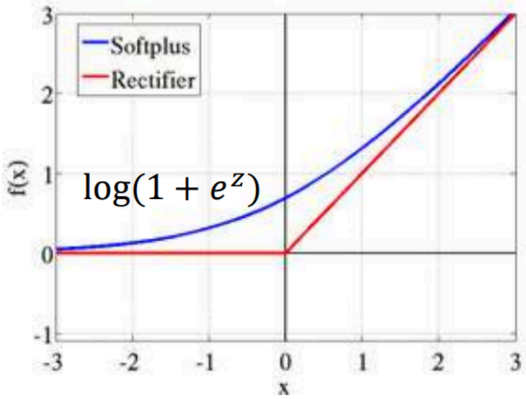


sigmoid



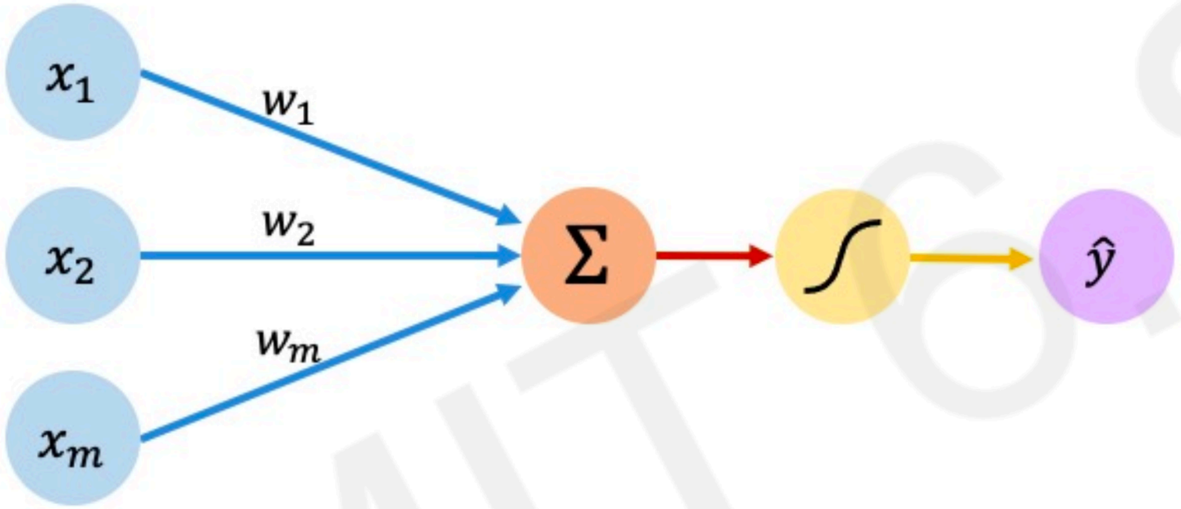
tanh





https://www.cs.cmu.edu/~bhiksha/courses/deeplearning/Spring.2019/www/slides.spring19/lec2.universal.pdf

The Perceptron: Forward Propagation



Inputs Weights Sum Non-Linearity Output

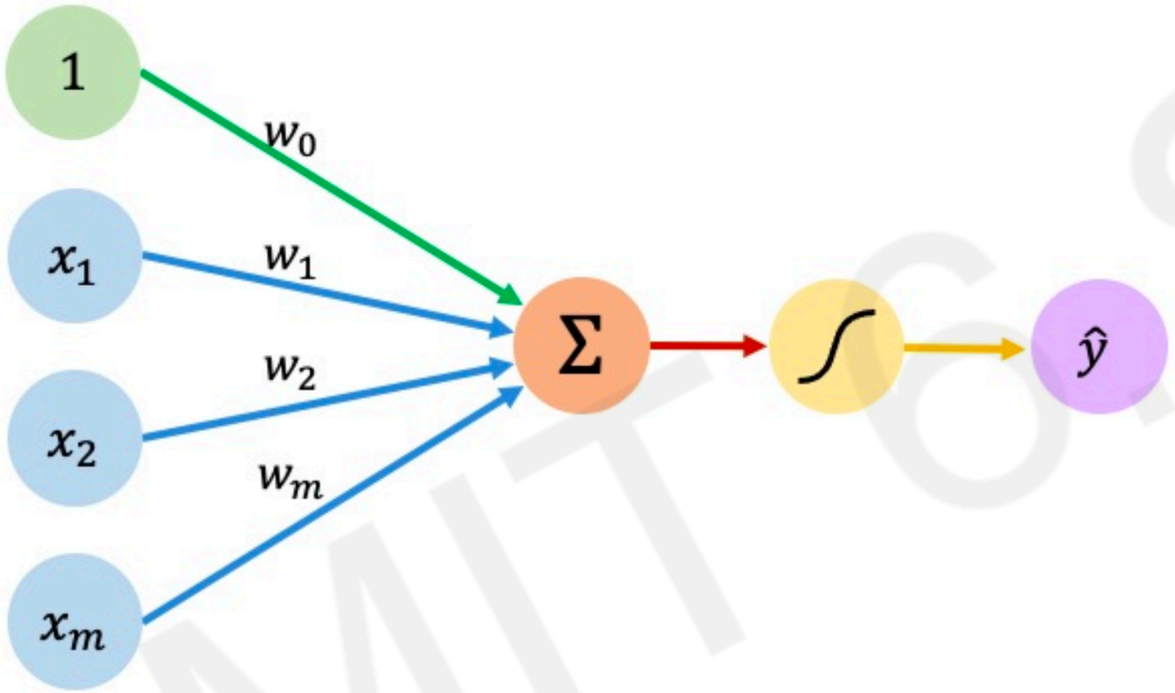
Output

Linear combination of inputs

$$\hat{y} = g\left(\sum_{i=1}^{m} x_i \, w_i\right)$$

Non-linear activation function

The Perceptron: Forward Propagation



Output

Linear combination of inputs

$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i \, w_i\right)$$

Non-linear activation function

Bias

Perceptron as Boolean Function

https://www.cs.cmu.edu/~bhiksha/courses/deeplearning/Spring.2019/www/slides.spring19/lec2.universal.pdf

Exercise:

- model AND, OR and NOT gate with single perceptron

Multi layered perceptron (MLP)

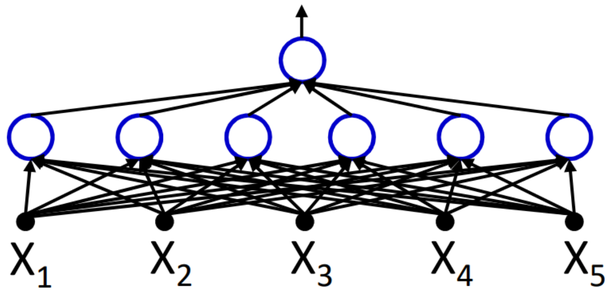
- MLPs are universal Boolean functions, i.e. can compute any Boolean function
- Single hidden layer is enough

Truth Table

X_1	X_2	X_3	X_4	X_5	Y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

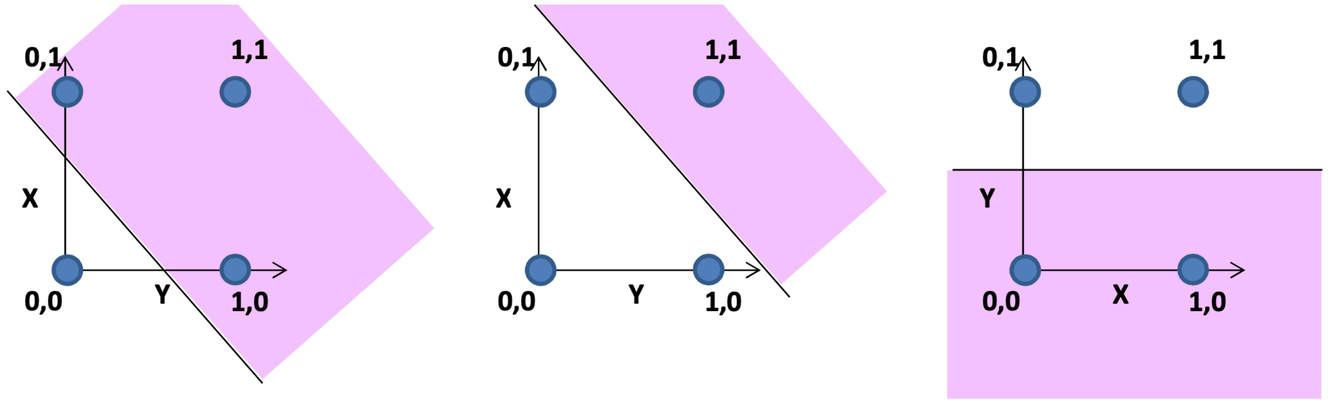
Truth table shows *all* input combinations for which output is 1

$$Y = \bar{X}_1\bar{X}_2X_3X_4\bar{X}_5 + \bar{X}_1X_2\bar{X}_3X_4X_5 + \bar{X}_1X_2X_3\bar{X}_4\bar{X}_5 + X_1\bar{X}_2\bar{X}_3\bar{X}_4X_5 + X_1\bar{X}_2X_3X_4X_5 + X_1X_2\bar{X}_3\bar{X}_4X_5$$



Boolean functions with a “real” perceptron

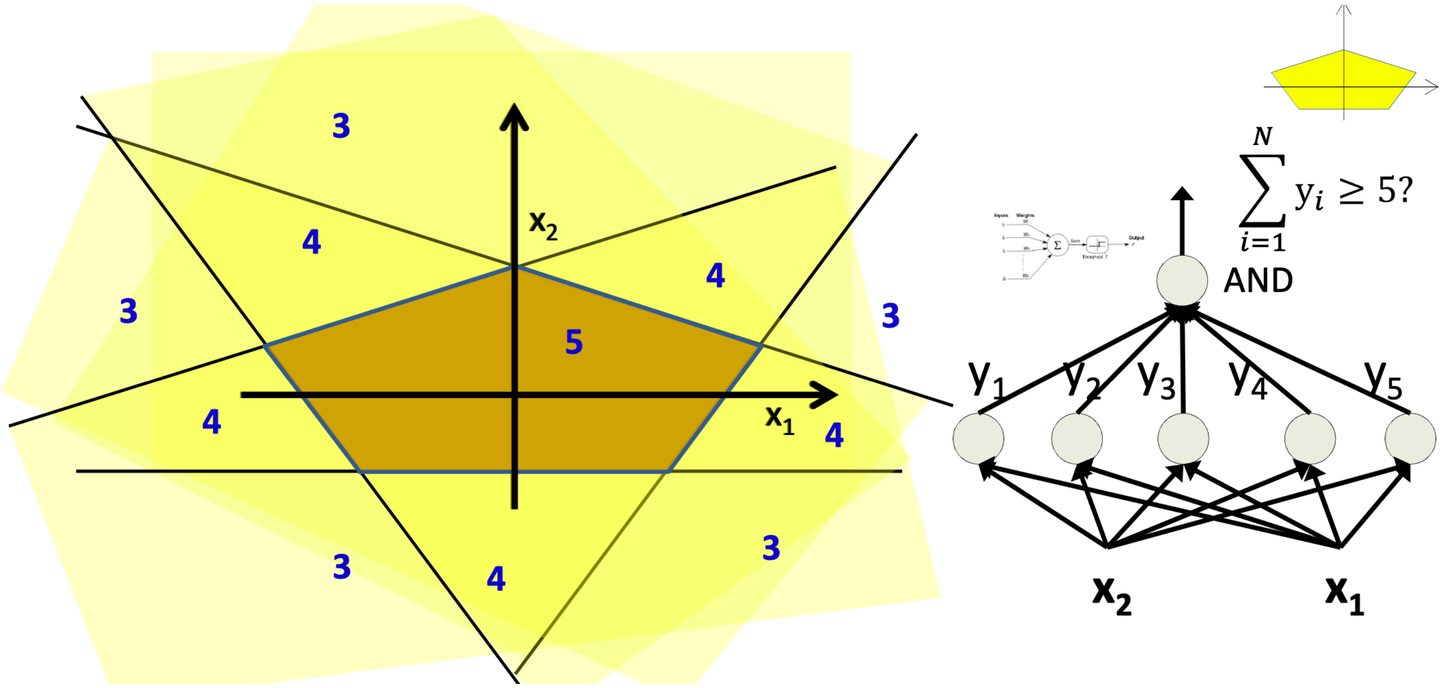
- OR, AND



- We cannot model XOR

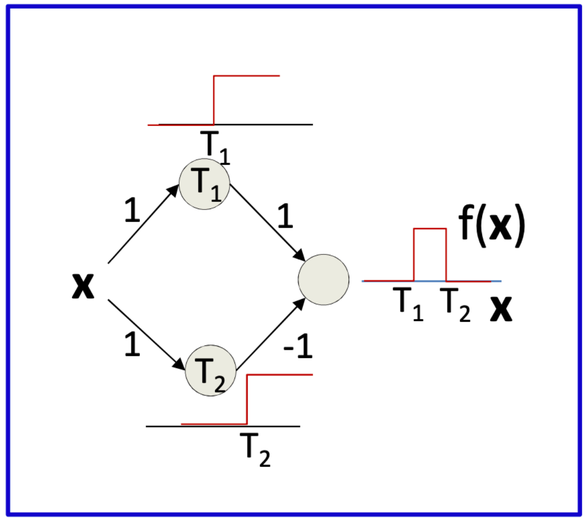
More complex shape

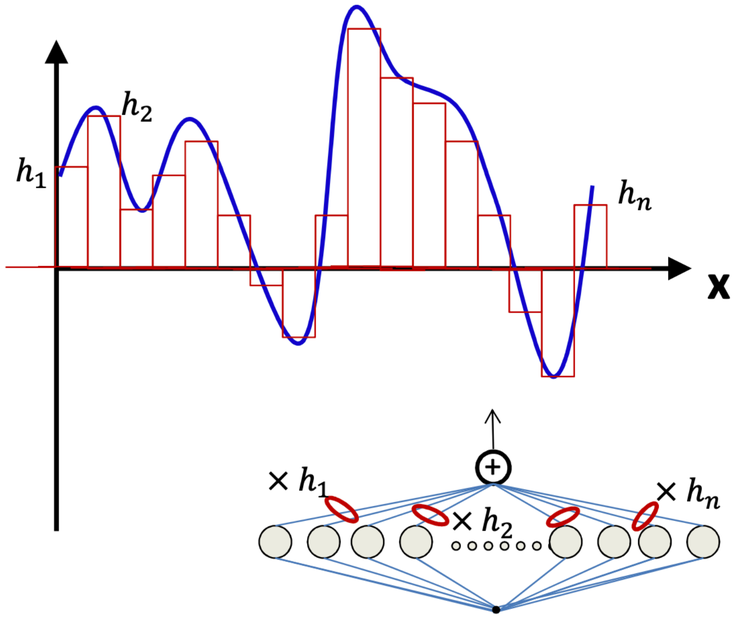
- One hidden layer is enough



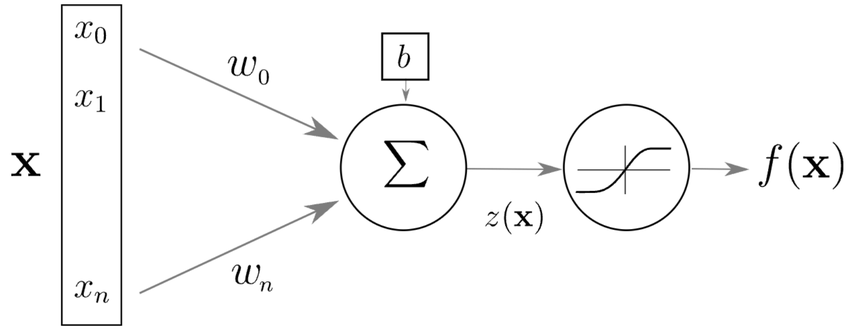
Universal Approximation Theorem

see demo
http://neuralnetworksanddeeplearning.com/chap4.html





Artificial Neuron



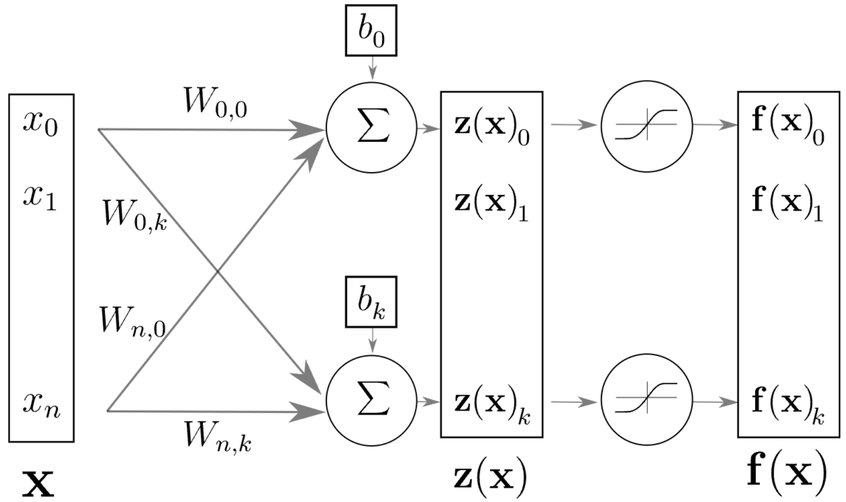
$$\mathrm{z}(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathrm{b}$$

$$\mathrm{f}(\mathbf{x}) = \mathrm{g}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathrm{b})$$

- $\mathbf{x}, \mathrm{f}(\mathbf{x})$ input and output
- $\mathrm{z}(\mathbf{x})$ pre-activation
- $\mathbf{w}, \mathrm{b}$ weights and bias
- g activation function

source: m2dsupsdlclass.github.io

Layer of Neurons

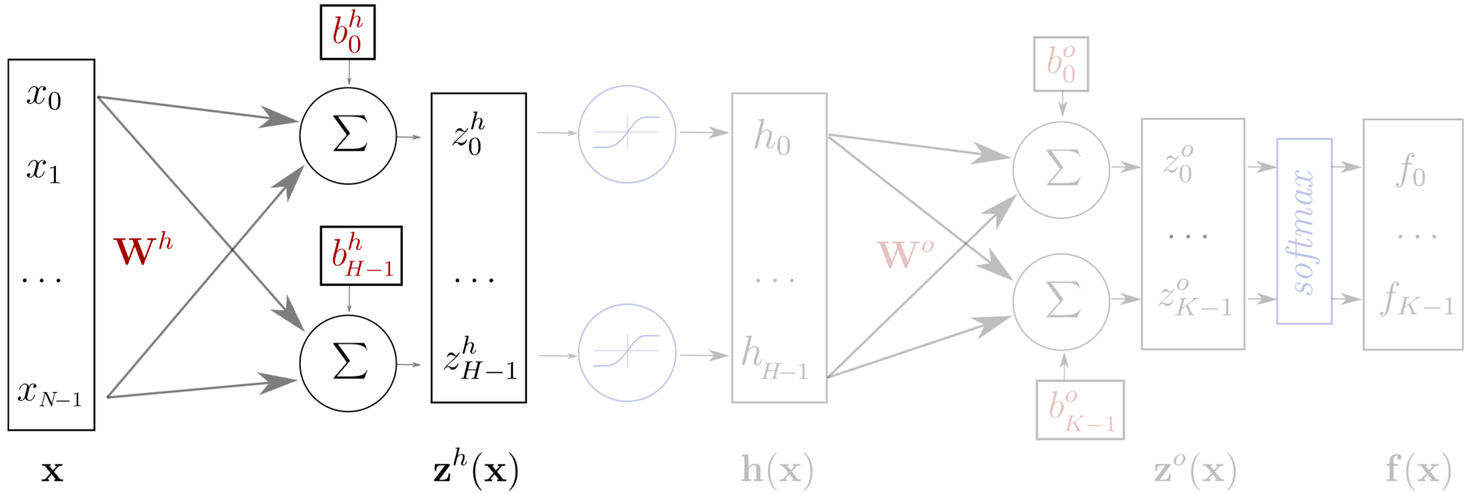


$$\mathbf{f(x)} = \mathrm{g}(\mathbf{z(x)}) = \mathrm{g}(\mathbf{Wx} + \mathbf{b})$$

- $\mathbf{W}, \mathbf{b}$ now matrix and vector

source: m2dsupsdlclass.github.io

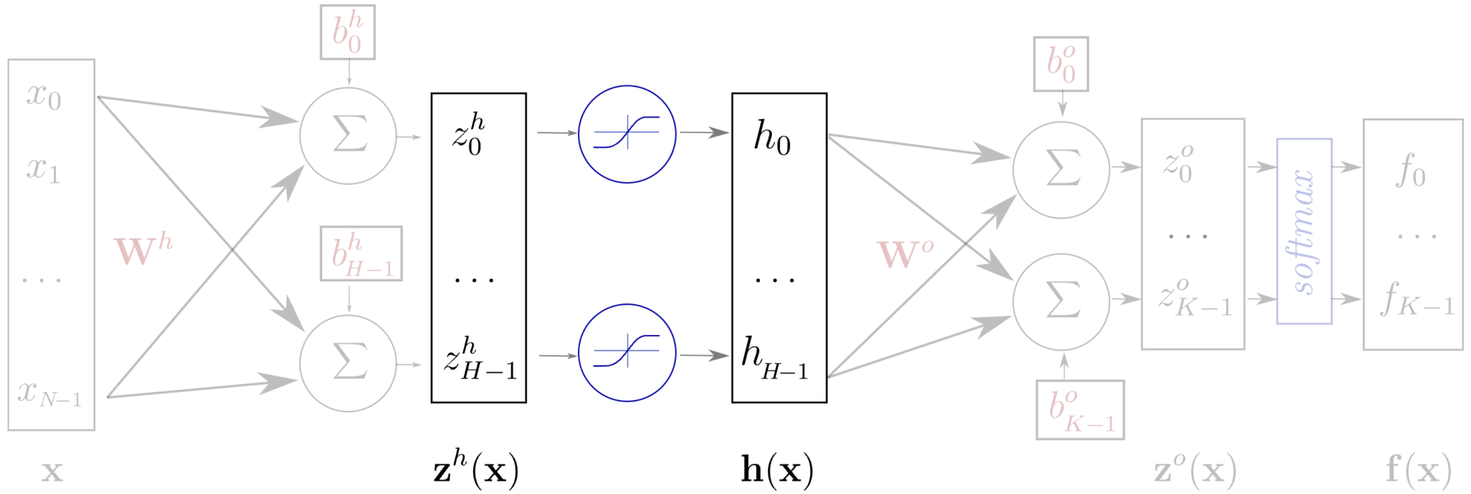
One Hidden Layer Network



- $\mathbf{z}^h(\mathbf{x}) = \mathbf{W}^h\mathbf{x} + \mathbf{b}^h$
- $\mathbf{h}(\mathbf{x}) = g(\mathbf{z}^h(\mathbf{x})) = g(\mathbf{W}^h\mathbf{x} + \mathbf{b}^h)$
- $\mathbf{z}^o(\mathbf{x}) = \mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o$
- $\mathbf{f}(\mathbf{x}) = \mathrm{softmax}(\mathbf{z}^o) = \mathrm{softmax}(\mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o)$

source: m2dsupsdlclass.github.io

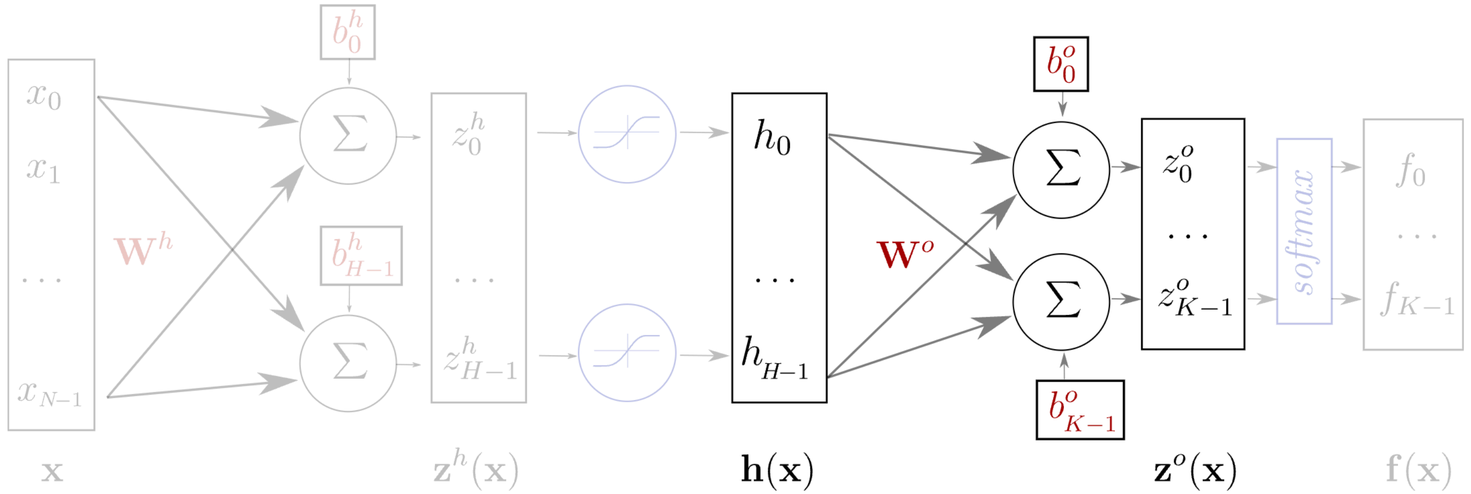
One Hidden Layer Network



- $\mathbf{z}^h(\mathbf{x}) = \mathbf{W}^h\mathbf{x} + \mathbf{b}^h$
- $\mathbf{h}(\mathbf{x}) = g(\mathbf{z}^h(\mathbf{x})) = g(\mathbf{W}^h\mathbf{x} + \mathbf{b}^h)$
- $\mathbf{z}^o(\mathbf{x}) = \mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o$
- $\mathbf{f}(\mathbf{x}) = \mathrm{softmax}(\mathbf{z}^o) = \mathrm{softmax}(\mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o)$

source: m2dsupsdlclass.github.io

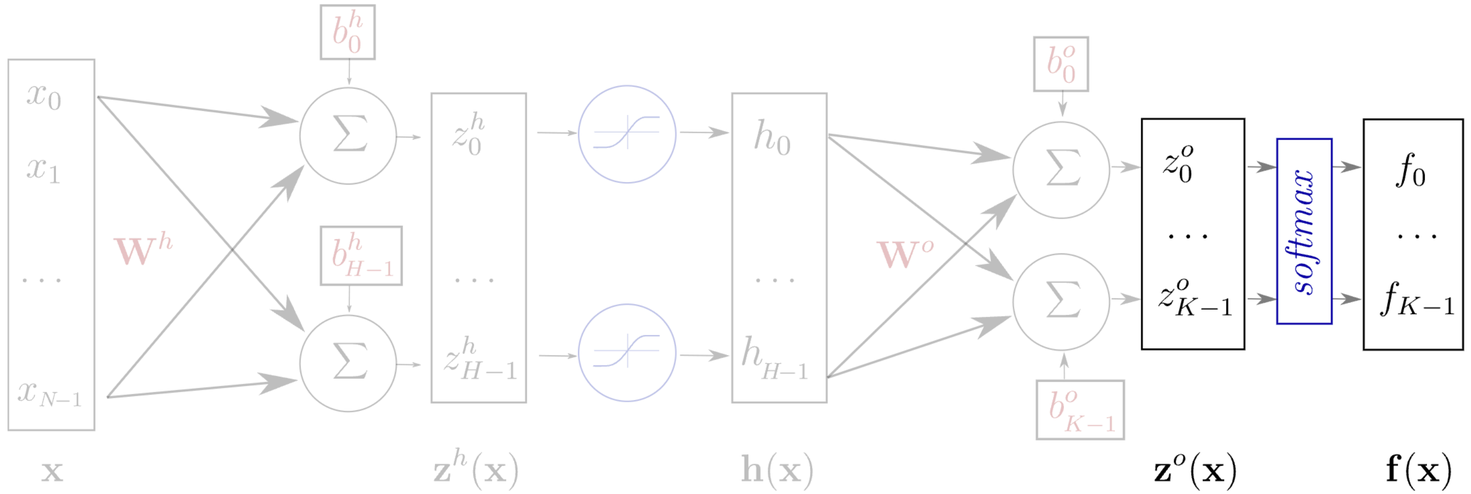
One Hidden Layer Network



- $\mathbf{z}^h(\mathbf{x}) = \mathbf{W}^h\mathbf{x} + \mathbf{b}^h$
- $\mathbf{h}(\mathbf{x}) = g(\mathbf{z}^h(\mathbf{x})) = g(\mathbf{W}^h\mathbf{x} + \mathbf{b}^h)$
- $\mathbf{z}^o(\mathbf{x}) = \mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o$
- $\mathbf{f}(\mathbf{x}) = \mathrm{softmax}(\mathbf{z}^o) = \mathrm{softmax}(\mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o)$

source: m2dsupsdlclass.github.io

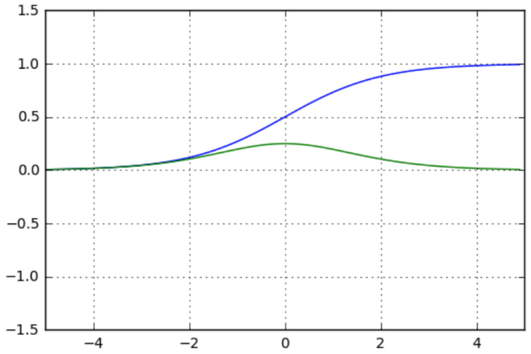
One Hidden Layer Network



- $\mathbf{z}^h(\mathbf{x}) = \mathbf{W}^h\mathbf{x} + \mathbf{b}^h$
- $\mathbf{h}(\mathbf{x}) = g(\mathbf{z}^h(\mathbf{x})) = g(\mathbf{W}^h\mathbf{x} + \mathbf{b}^h)$
- $\mathbf{z}^o(\mathbf{x}) = \mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o$
- $\mathbf{f}(\mathbf{x}) = \mathrm{softmax}(\mathbf{z}^o) = \mathrm{softmax}(\mathbf{W}^o\mathbf{h}(\mathbf{x}) + \mathbf{b}^o)$

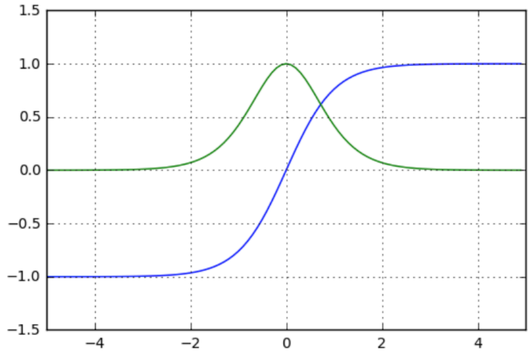
source: m2dsupsdlclass.github.io

Element-wise activation functions



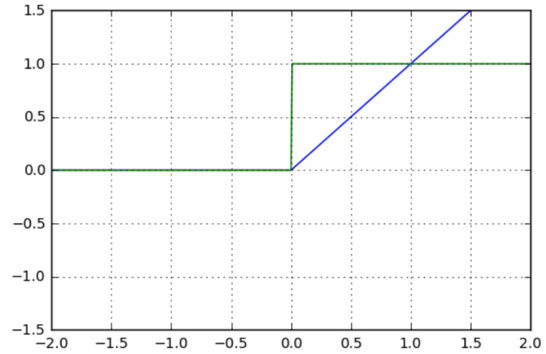
$$\text{sigm}(x) = \frac{1}{1+e^{-x}}$$

$$\text{sigm}'(x) = \text{sigm}(x)(1-\text{sigm}(x))$$



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

$$\tanh'(x) = 1-\tanh(x)^2$$



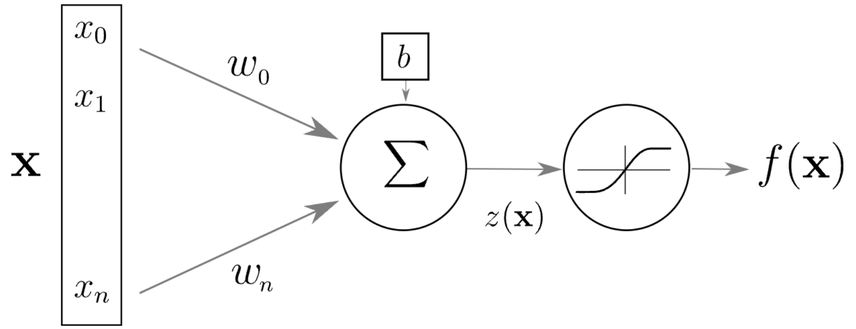
$$\text{relu}(x) = \max(0, x)$$

$$\text{relu}'(x) = 1_{x>0}$$

- blue: activation function
- green: derivative

source: m2dsupsdlclass.github.io

Artificial Neuron



$$\mathrm{z}(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathrm{b}$$

$$\mathrm{f}(\mathbf{x}) = \mathrm{g}(\mathbf{w}^{\mathrm{T}}\mathbf{x} + \mathrm{b})$$

- $\mathbf{x}, \mathrm{f}(\mathbf{x})$ input and output
- $\mathrm{z}(\mathbf{x})$ pre-activation
- $\mathbf{w}, \mathrm{b}$ weights and bias
- g activation function

source: m2dsupsdlclass.github.io

Libraries and Frameworks

PYTORCH

Microsoft CNTK

Caffe2

dmlc mxnet

gensim

spaCy

theano