Fully Connected Neural Network (FCNN)

Stian Stensli, Trygve Vang, Henrik Kronborg

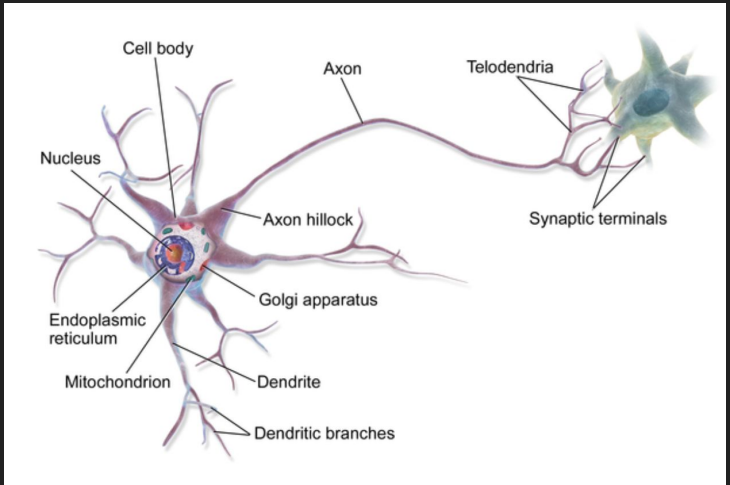
Topics

- Neuron
- Perceptron
- FCNN
- Learning

Neurons

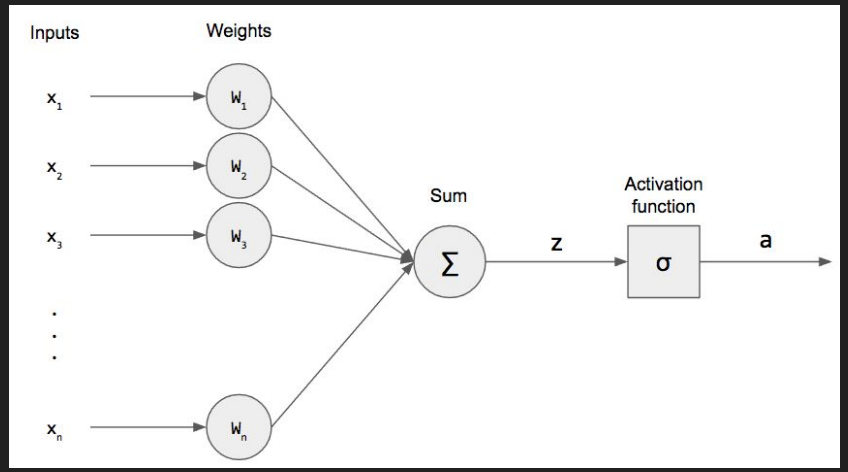
The nodes of a neural network

Researched since 1943
- Theory of how the brain might work
- Managed to learn simple boolean expressions



Perceptron

- Consists of:
  - Inputs: x_n
  - Weights: w_n
  - Bias: b
  - A sum: Z
  - An activation function: σ
  - Output: a



- Input/output mapping by an activation function
- Weights and bias are learnable parameters

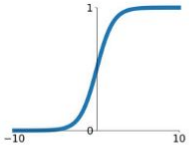
Activation functions

- Converts input signals (Z) to an output signal
  - X_n: Output from node n
  - W_n: Weight n
- Output is fed to the next layer

$$Z = \sum_{i=0}^{n} w_i x_i + b$$

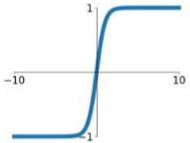
Sigmoid

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



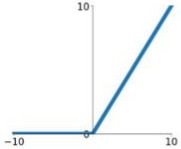
tanh

$\tanh(x)$



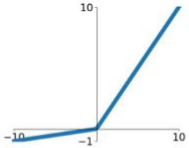
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

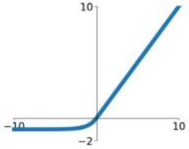


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

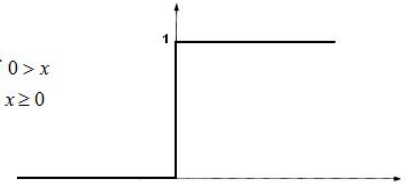
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



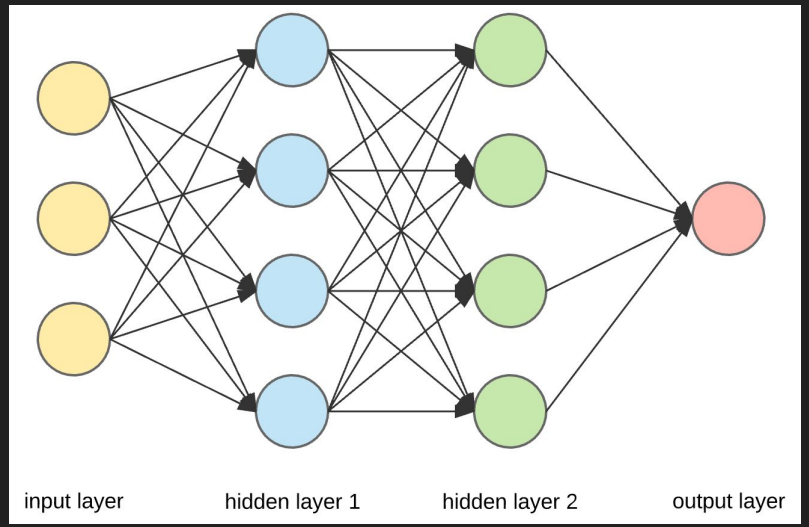
Unit step (threshold)

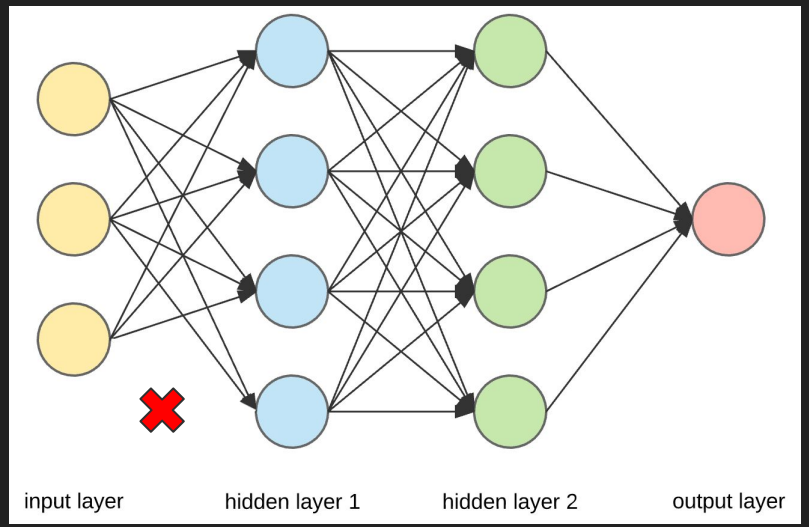
$$f(x) = \begin{cases} 0 \text{ if } 0 > x \\ 1 \text{ if } x \geq 0 \end{cases}$$



Fully Connected Neural Network

- Each perceptron is connected to **all** perceptrons in the previous and next layer
- Very basic network
- One or more fully connected layers are often used in neural networks



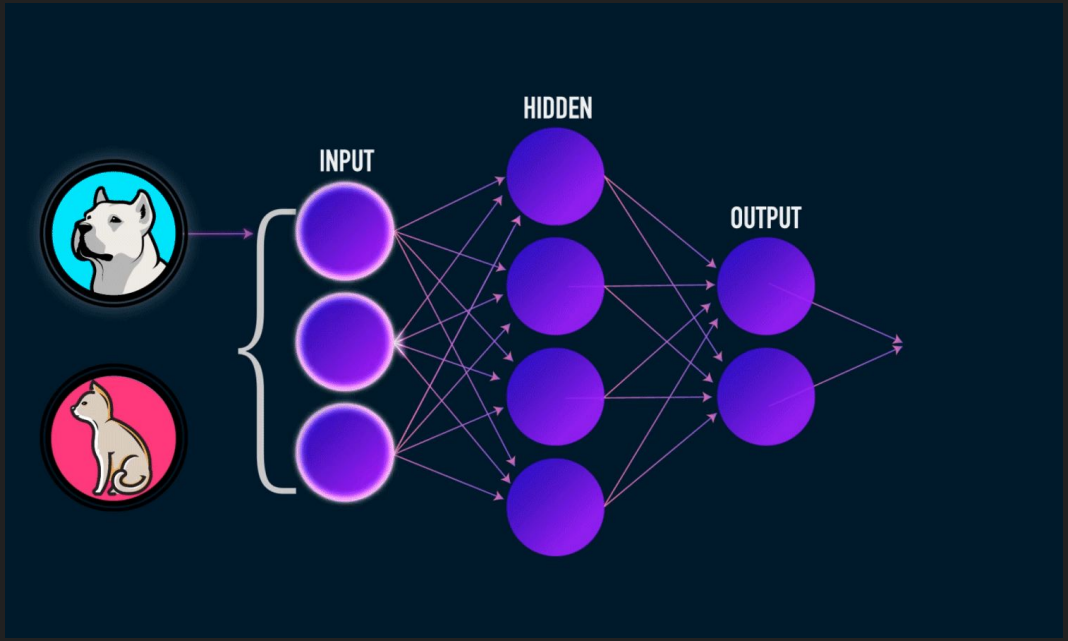


Learning

1. Data set is split into **training** data, **test** data (and sometimes **validation** data)
2. Foreach Epoch
    - 2.1. Foreach element in **training** data
        - 2.1.1. Send element as input to the network
        - 2.1.2. Calculate error with cost function
        - 2.1.3. Propagate error backwards to adjust weights
    - 2.2. If use of **validation** data, calculate accuracy
3. Calculate accuracy using **test** data

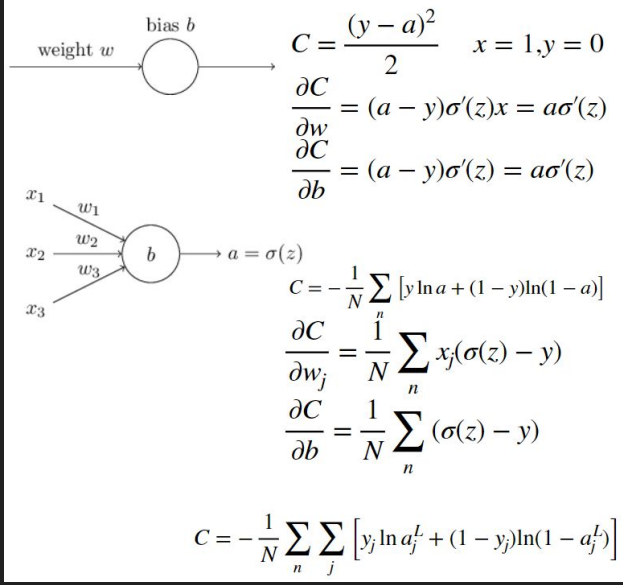
Feed forward

- The output from the previous layer determines the result of the current layer



Cost function

- Many cost functions
  - Mean Squared Error (MSE)
  - Cross Entropy
- Calculate error based on expected output
- Used to find how much weights should be adjusted



$$C = \frac{(y-a)^2}{2} \qquad x = 1, y = 0$$

$$\frac{\partial C}{\partial w} = (a-y)\sigma'(z)x = a\sigma'(z)$$

$$\frac{\partial C}{\partial b} = (a-y)\sigma'(z) = a\sigma'(z)$$

$$C = -\frac{1}{N}\sum_n \left[y \ln a + (1-y)\ln(1-a)\right]$$

$$\frac{\partial C}{\partial w_j} = \frac{1}{N}\sum_n x_j(\sigma(z) - y)$$

$$\frac{\partial C}{\partial b} = \frac{1}{N}\sum_n (\sigma(z) - y)$$

$$C = -\frac{1}{N}\sum_n \sum_j \left[y_j \ln a_j^L + (1-y_j)\ln(1-a_j^L)\right]$$

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \widehat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

Backpropagation/Gradient descent

- Propagate error from cost from output to input
- **Learning:** weight updating