Basics of Artificial Neural Networks (ANN)

Intro

- Intro
- Properties
- Application

Intro

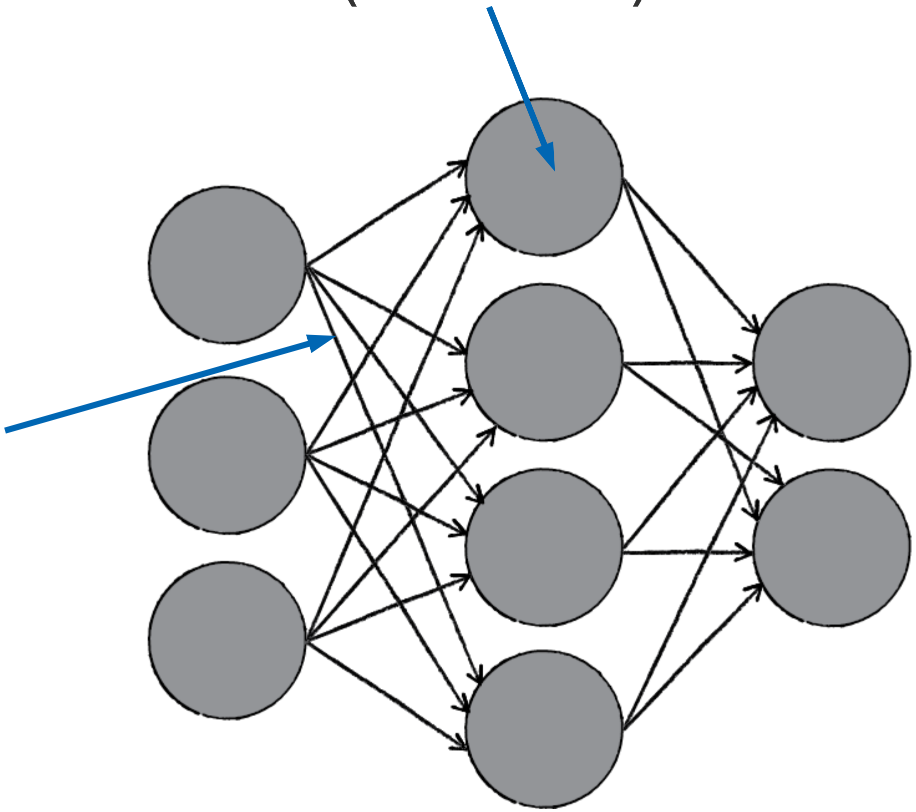
- Intro
- Properties
- Application

Intro/Definition

ANN:

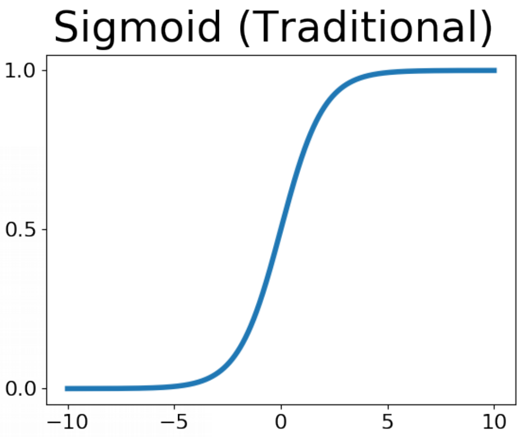
Network of small Computation Units (Neurons)

- Inspired by Brain
- Computation Model
- Not a Simulation
- Connections: Synapses

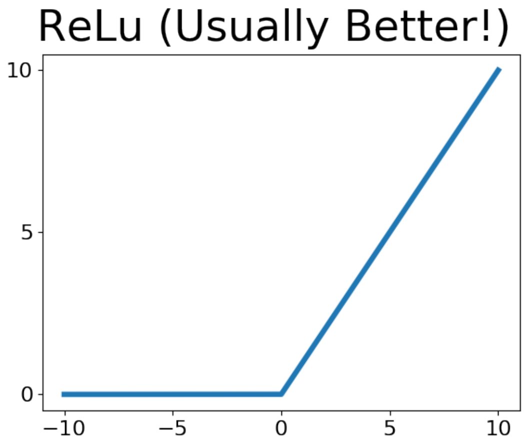


Intro/Single Perceptron

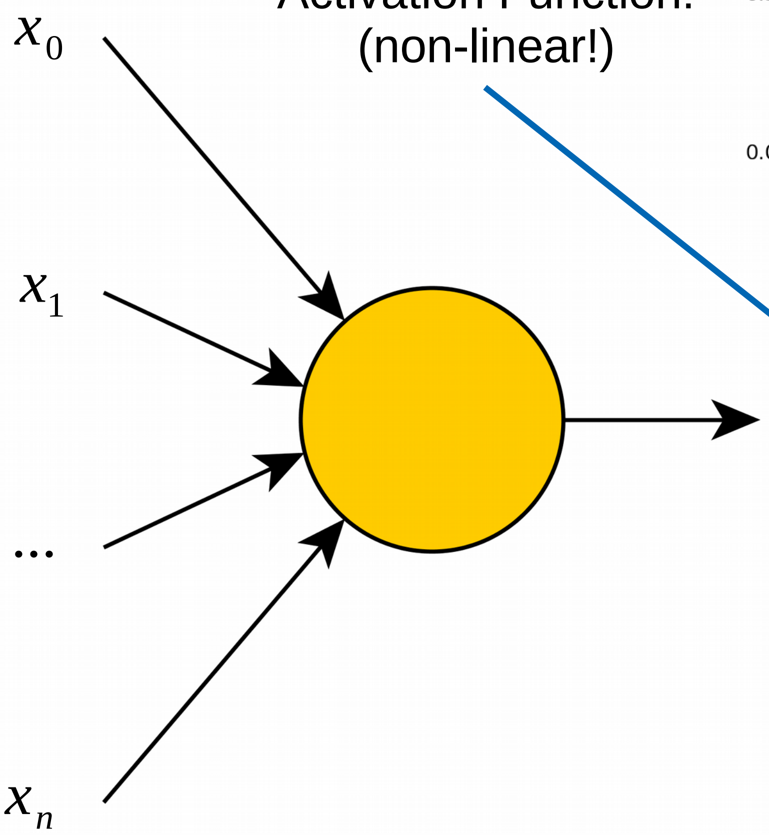
Sigmoid (Traditional)



ReLu (Usually Better!)



Activation Function:
(non-linear!)



$$y = act(b + w_0 \cdot x_0 + w_1 \cdot x_1 + \ldots + w_n \cdot x_n)$$

Bias

Weights
(Weighted Sum: Logit)

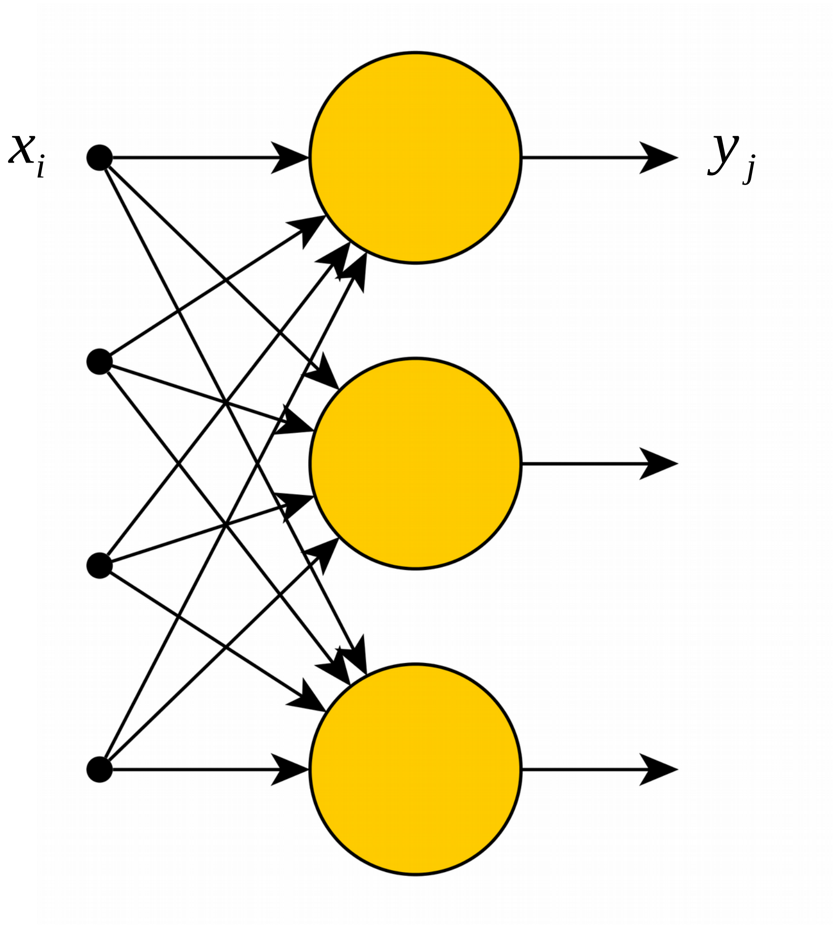
Intro/Layers

- 1 Output/Perceptron
- ➔ Use multiple
- Organized in Layers
- Weighted sum

$$y_j = act\left(b_i + \sum_i w_{ij} \cdot x_i\right)$$

- ➔ Matrix multiplication:

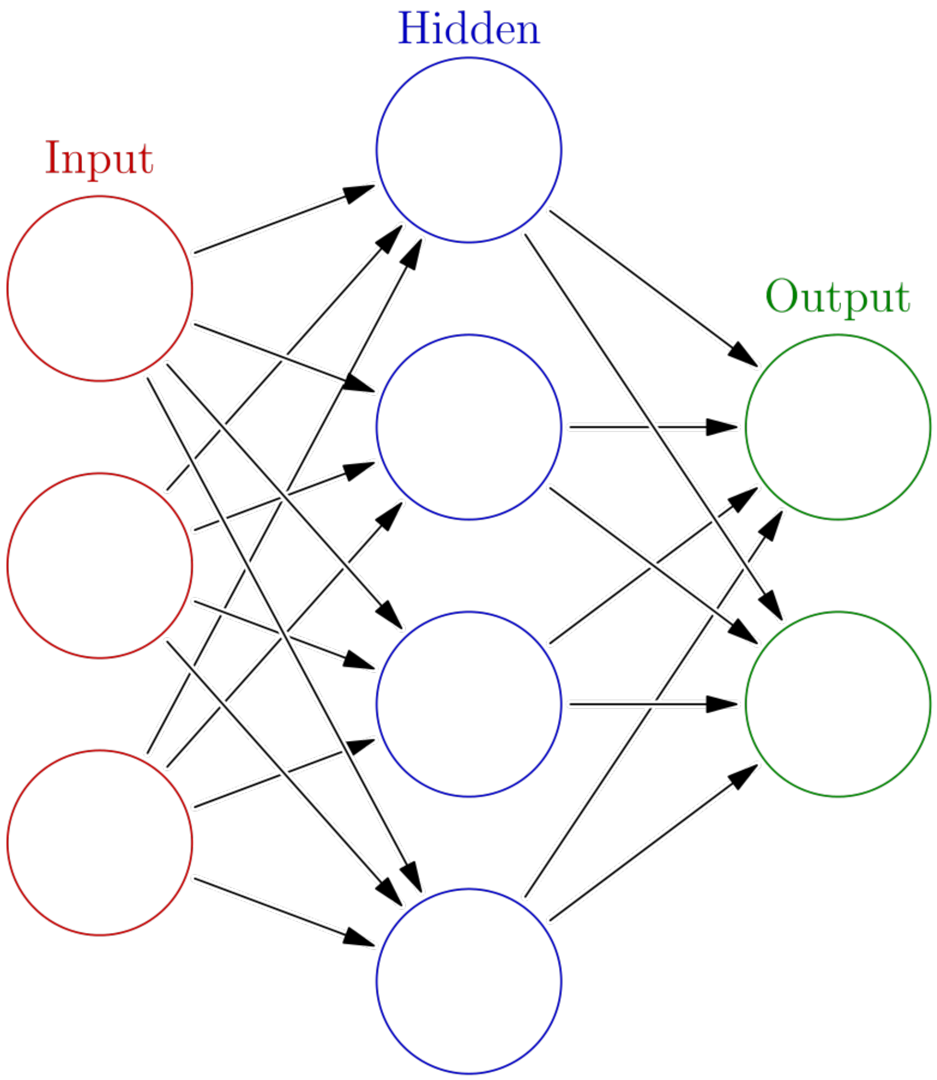
$$\vec{y} = act\left(\vec{b} + W^T \cdot \vec{x}\right)$$



Intro/Multi-Layer Perceptron

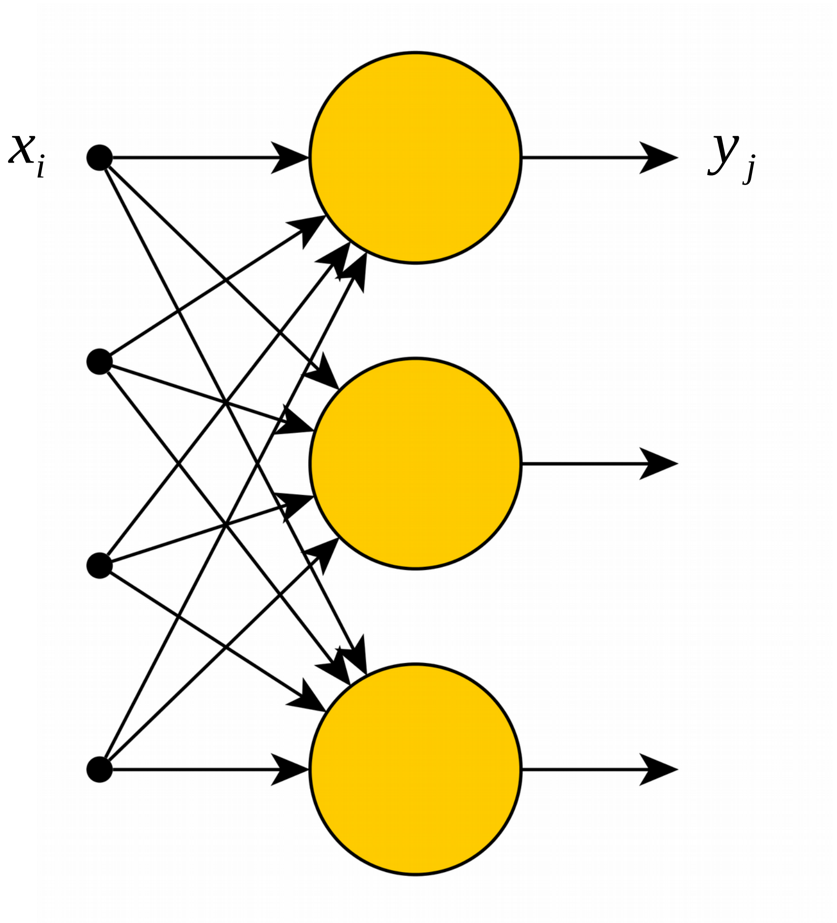
- Layers Stackable

➔ Deep NN



Intro/Softmax Layer

$$y_j = \frac{e^{b_j + w_j \cdot x_j}}{\sum_i e^{b_i + w_i \cdot x_i}}$$



- Sums up to 1
- Probability-Like
- Classification

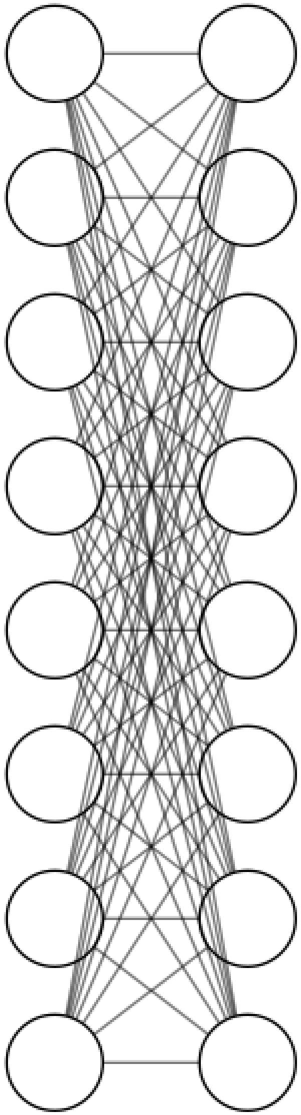
Intro/Convolutional Layer

- Fully Connected Layer
- 10^3 Neurons Each

➔ 10^6 Synapses

➔ Reduce Synapes to „Neighborhood“

- Works well for Images or Sequences

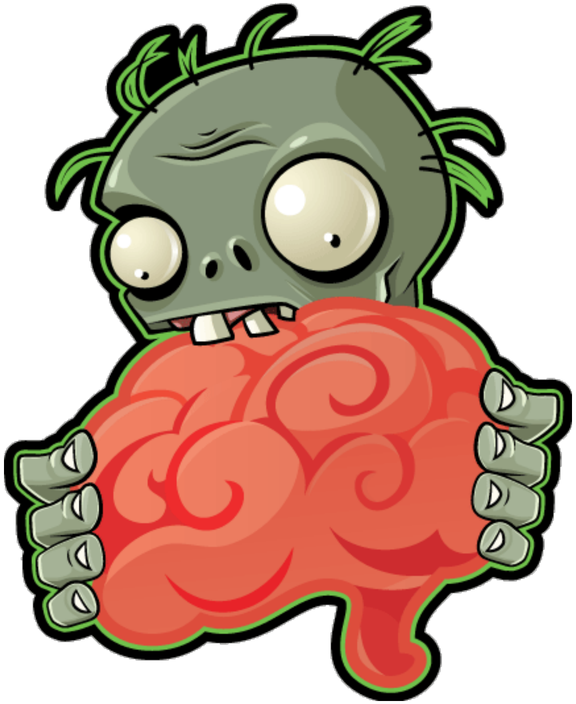


Properties

- Intro
- Properties
- Application

Properties

- Why are ANN so powerful/popular?
- They‘re like a human brain?
  - Not really… but…



- Model any (mathematical) Function!
- Some ANN Touring complete
  (model any computer program)
- They are trainable
  (/fittable/optimizable/programmable)
- Think of a computer program that writes itself
  - albeit not perfectly...

Properties/Boolean Logic

$$a \vee b = sig\left(10\left(a+b\right)-5\right)$$

$$sig\left(10\left(0+0\right)-5\right) \approx 0 ✓$$

$$sig\left(10\left(0+1\right)-5\right) \approx 1 ✓$$

$$sig\left(10\left(1+0\right)-5\right) \approx 1 ✓$$

$$sig\left(10\left(1+1\right)-5\right) \approx 1 ✓$$

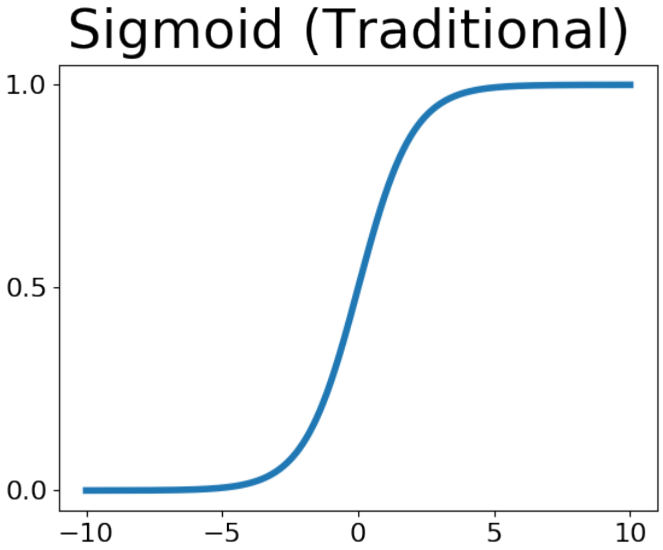
$$\neg a = sig\left(5-10 \cdot a\right)$$

$$sig\left(5-10 \cdot 0\right) \approx 1 ✓$$

$$sig\left(5-10 \cdot 1\right) \approx 0 ✓$$

$$a \wedge b = \neg\left(\neg a \vee \neg b\right)$$

$$\vdots$$

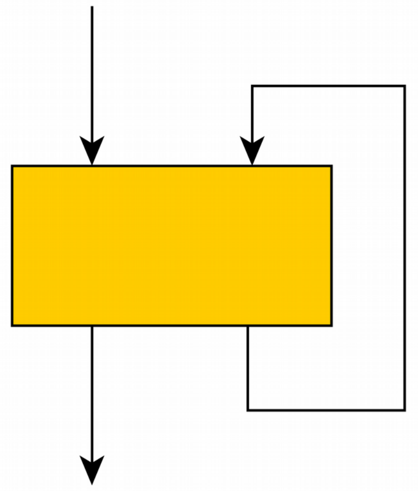


Properties/Memory

- We‘ve got boolean logic!
- Whats missing for Touring Completeness?
  - Memory!

➔ Recurrent Neural Networks RNN

- Some outputs fed back as inputs
- Special (forgetful) cells
  - LSTM (Long Short-Term Memory)
  - GRU, ...



Properties/Trainability

- Outputs of ANN can be trained
- Requires loss/error function, e.g. mean squared error
- ANN well derivable → fast optimization
  - Computed automaticall by Tensorflow, PyTorch, …
- (Stochastic) Gradient Descent (SGD) as Opimizer
  - better than complicated Methods
  - Process: Feed-Forward & Back-Propagation

$$\vec{y}=\vec{x}^{L_n}$$

$$\vec{x}^{L_i}=act\left(\vec{b}^{L_i}+\left(W^{L_i}\right)^T\cdot\vec{x}^{L_{i-1}}\right)$$

$$\frac{\partial\,\vec{x}^{L_i}}{\partial\,W^{L_j}}=act'\left(\vec{b}^{L_i}+\left(W^{L_i}\right)^T\cdot\vec{x}^{L_{i-1}}\right)\cdot\left(W^{L_i}\right)^T\cdot\frac{\vec{x}^{L_{i-1}}}{\partial\,W^{L_j}}\qquad(j<i)$$

That‘s it! - Questions?

Sources

- Definition ANN [Slide 4]: http://natureofcode.com/book/chapter-10-neural-networks/
- MLP Layers [Slide 7]: https://en.wikipedia.org/wiki/Artificial_neural_network
- Zombie Brain [Slide 11]: http://plantsvszombies.wikia.com/wiki/File:HDZombieAndBrain.png
- Remaining images created with yEd https://www.yworks.com/products/yed