Implementing Artificial Neural Networks (ANNs) with TensorFlow

Session 2: Multilayer Perceptron

University of Osnabrück
Institute of Cognitive Science

WS 2017 / 18

Lukas Braun, lbraun@uos.de

Old homework

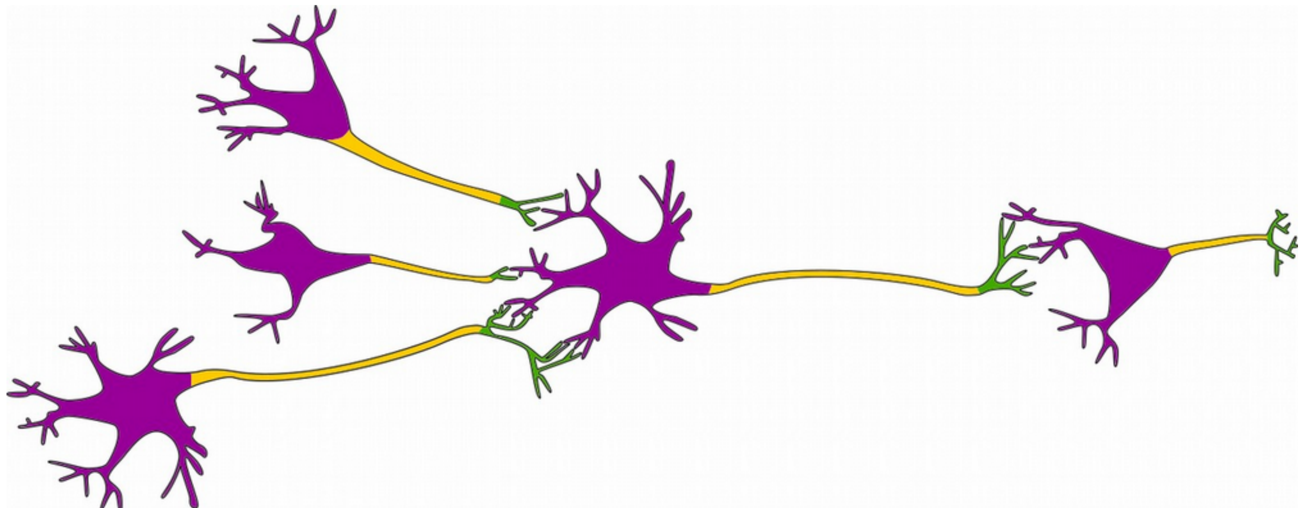
- Sign up into a homework group until 23:59 today (Monday 30th October)
- Any issues with installing TensorFlow?

New homework

- Solve homework 1 until Saturday 4th November and upload your solution into the public “Homework Submissions” / “01 Backpropagation and Gradient Descent” folder
- Name your files <group_id>_<task_name> i.e. 12_backpropagation-and-gradiend-descent
- Upload both, the original **ipython notebook** and the **HTML export**
- Download another group’s homework and insert your rating into the public spreadsheet until Monday November 6th at 23:59

Artificial Neural Networks

The Promise



Model

Speech
Text
Images
Videos
Labels
Entities
Words
Audio
Features

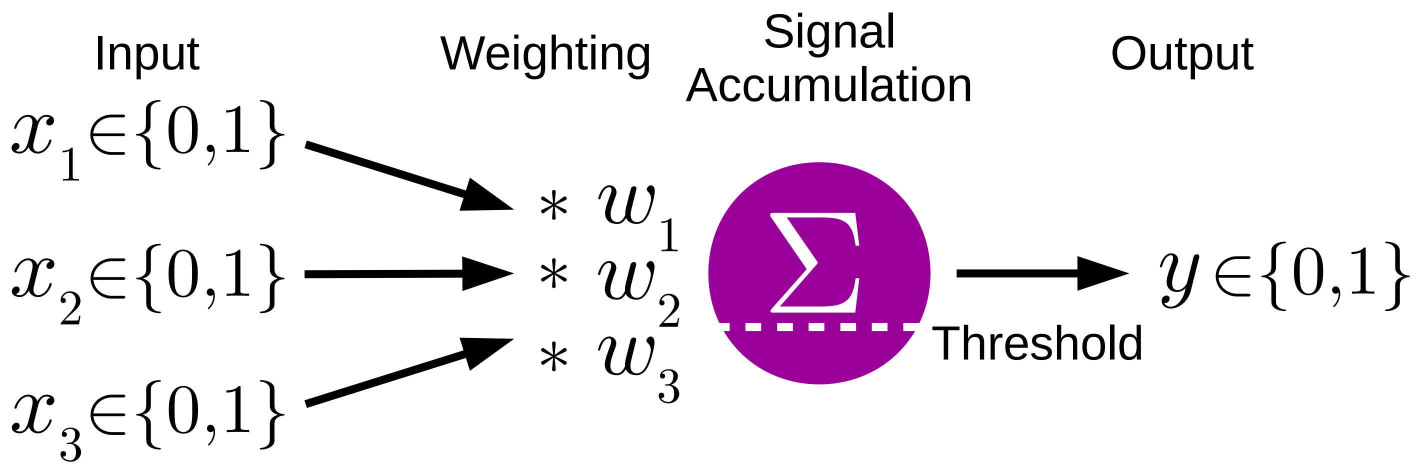
Input

Simple and reconfigurable building blocks

Output

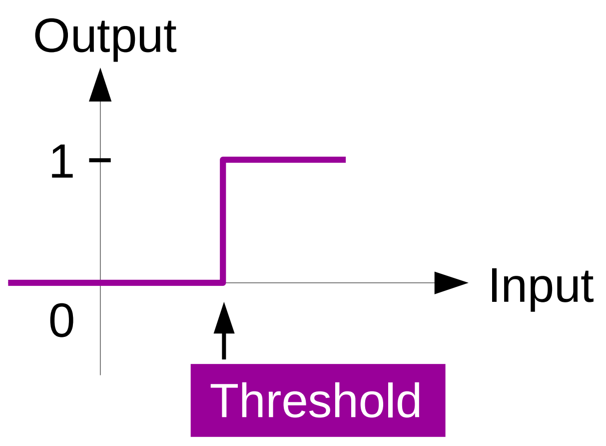
Speech
Text
Images
Videos
Labels
Entities
Words
Audio
Features

Perceptron



$$y = \sigma\left(\sum_i x_i w_i\right)$$

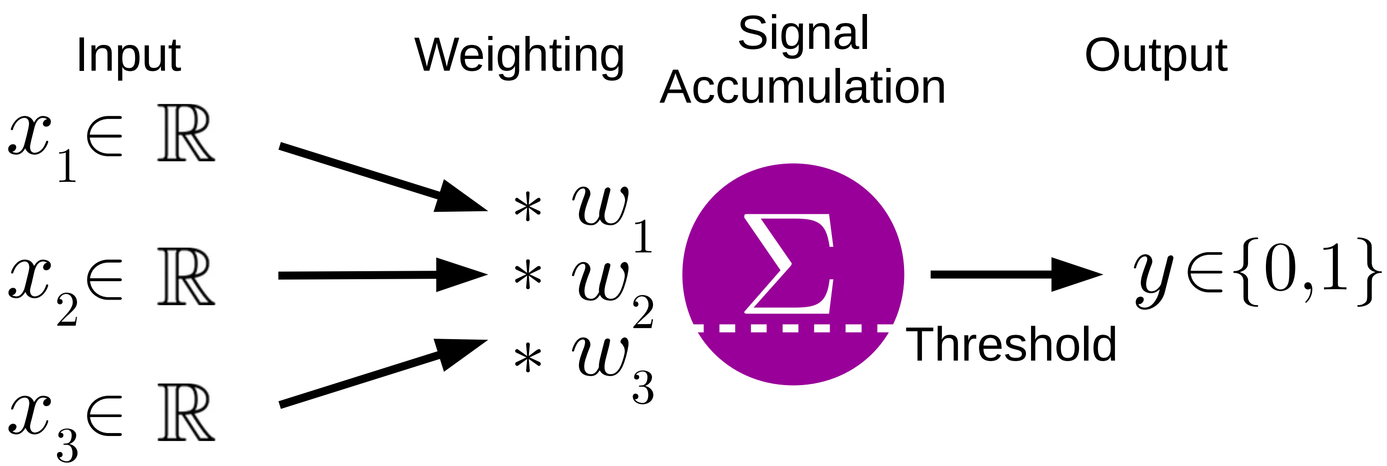
Step function



Improving the Model

Real valued inputs

Allow for non binary inputs



More differentiated inputs contain more information

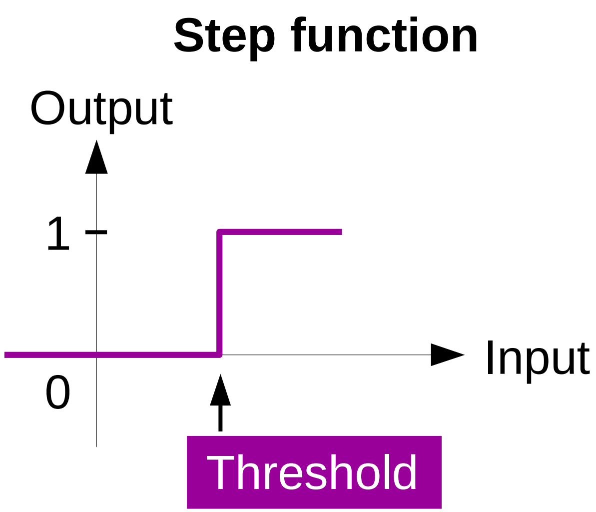
Bias

Make the activation function independent of parameters

Currently, the activation function is a function of the neuron's drive d and the threshold t

$$d = \sum_i x_i w_i$$

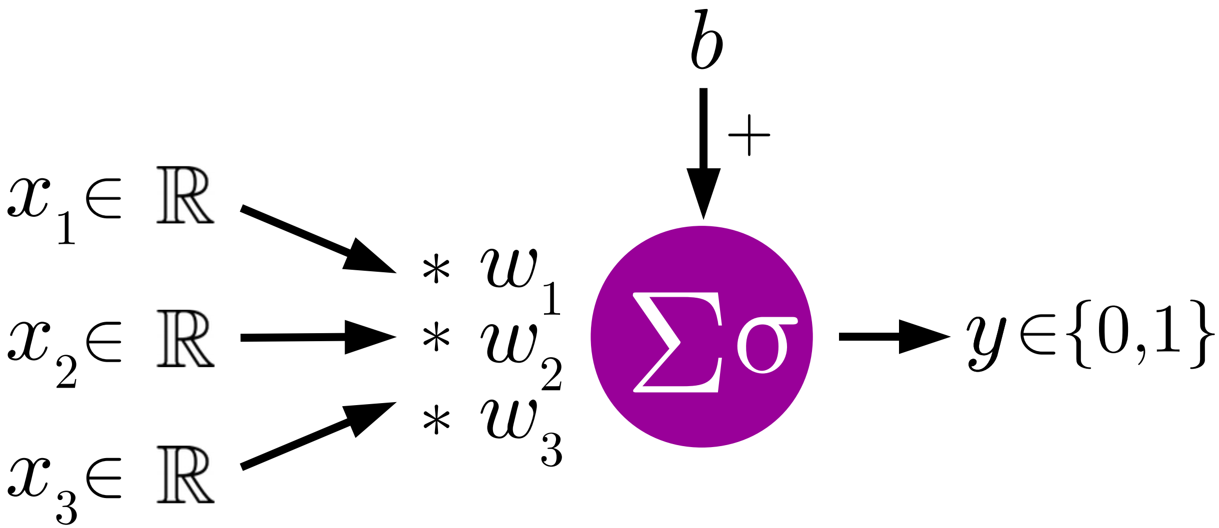
$$\sigma(d;t) = \begin{cases} 1, \text{if } d > t \\ 0, \text{otherwise} \end{cases}$$



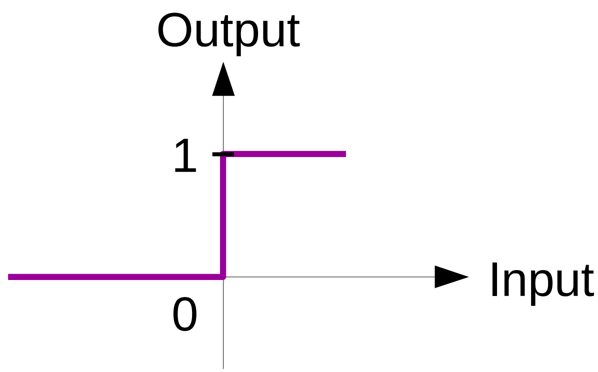
Bias

Instead of shifting the threshold value of the step function: shift the neuron's drive

$$d = \sum_i x_i w_i + b$$

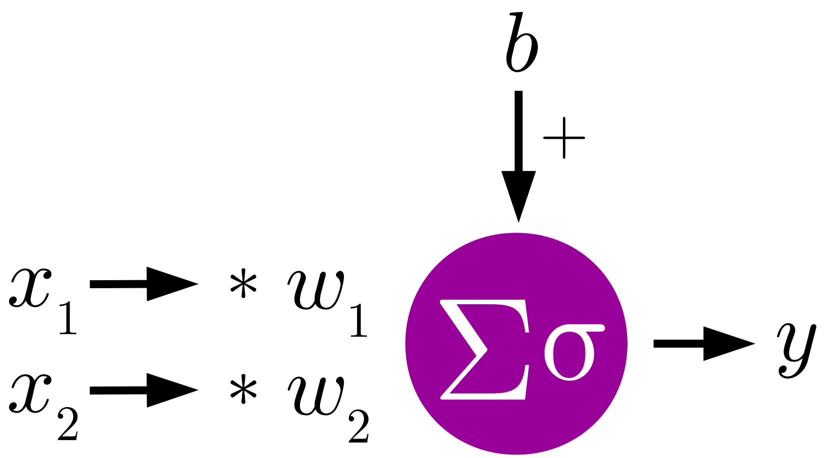


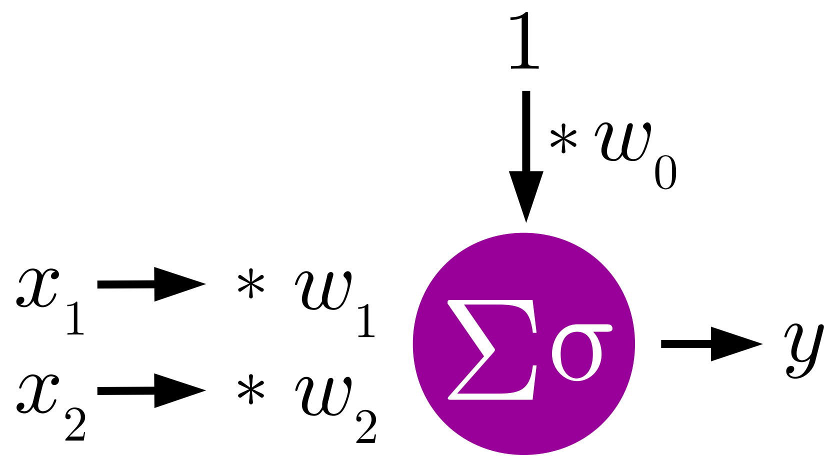
$$\sigma(d) = \begin{cases} 1, \text{if } d > t \\ 0, \text{otherwise} \end{cases}$$



Bias substitution

It is possible to substitute the bias by a weighted constant



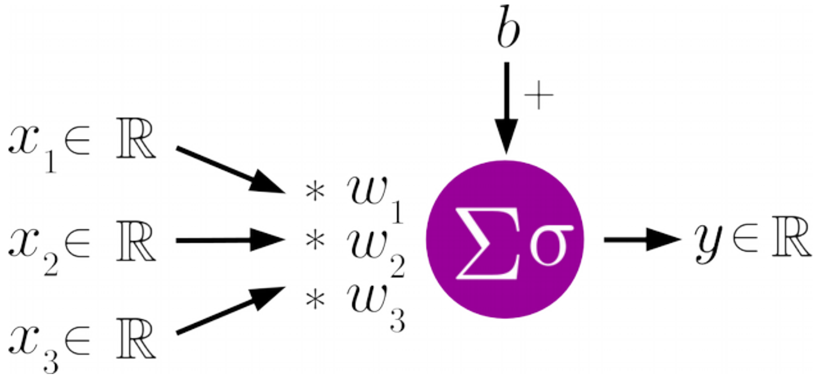


$$\sigma\left(\sum_{i=1}^{n} x_i w_i + b\right)$$

$$\sigma\left(\sum_{i=0}^{n} x_i w_i\right), \text{ with } x_0 = 1$$

Real valued outputs

Use an activation function which produces real valued outputs

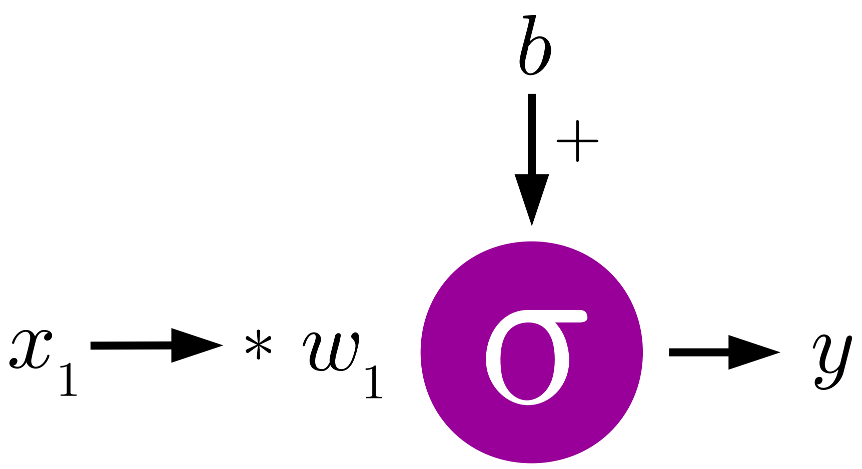


More differentiated outputs can be used to model (un-)certainty

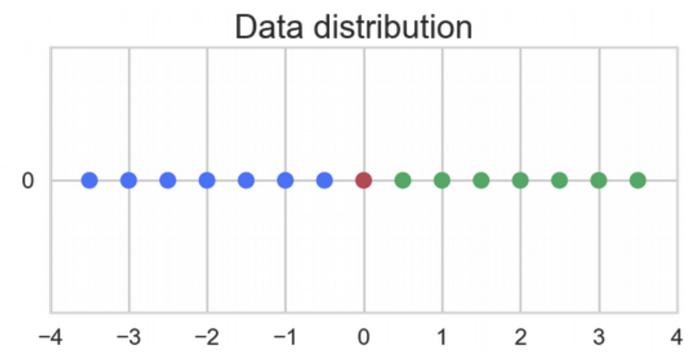
Transformations

Example

Model maps a single input to a single output



Random data consisting of two groups



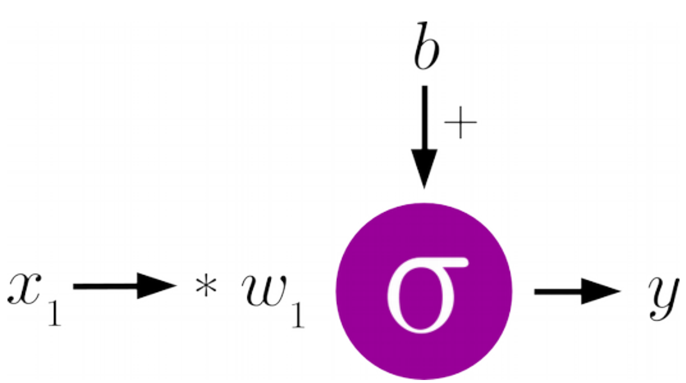
The following slides do not model the step by step transformation performed by the model, but rather show the effect of the individual transformations on a linear data distribution!

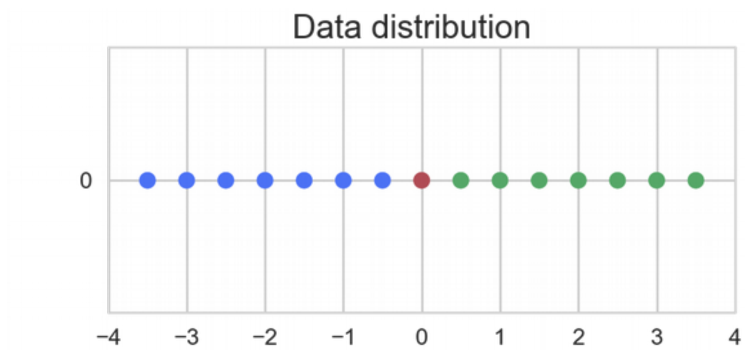
Effect of weight

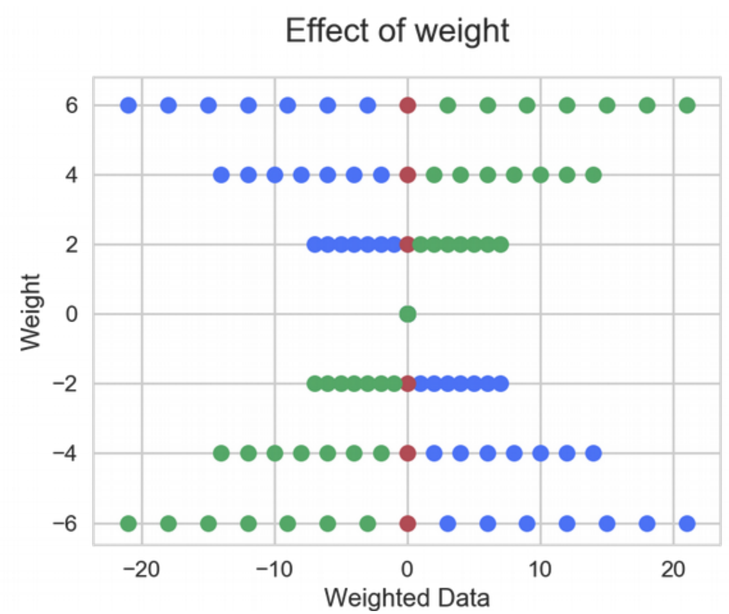
Weighting maps data onto a new range

Negative weights swap the order of the distribution

Relative differences are preserved



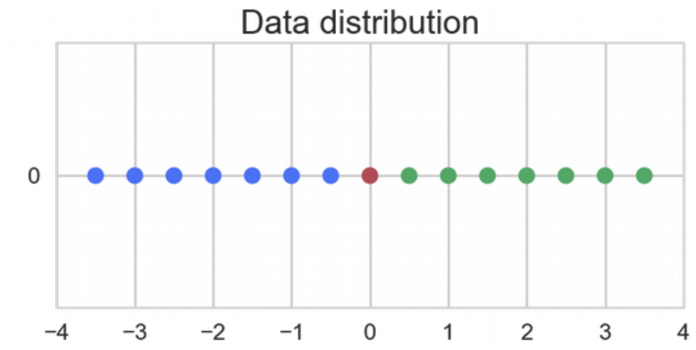


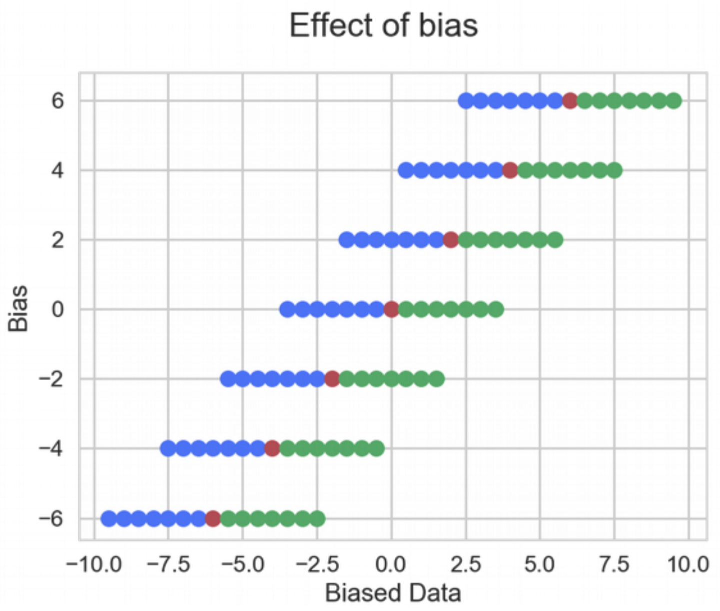


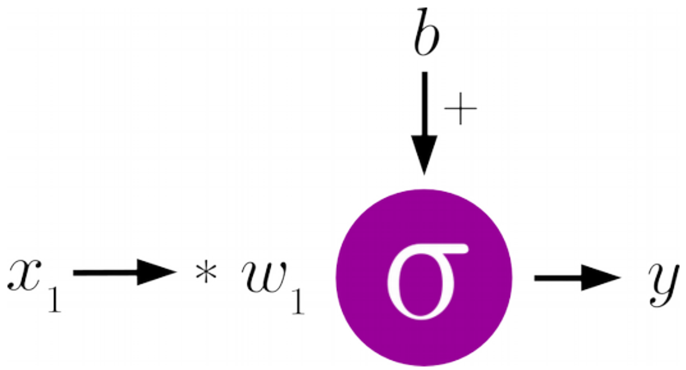
Effect of bias

Bias shifts data along the x axis

Relative differences are preserved



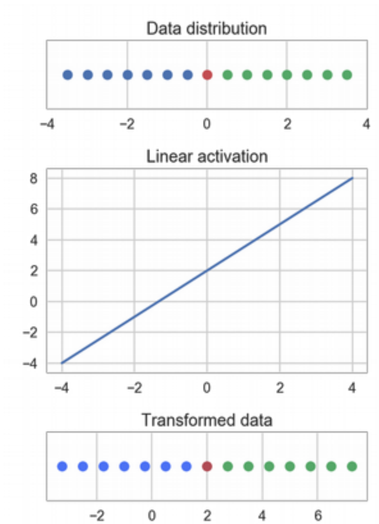


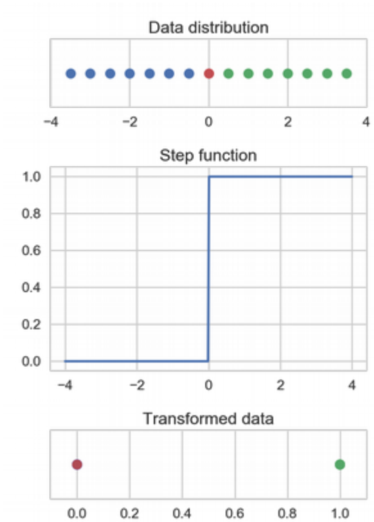


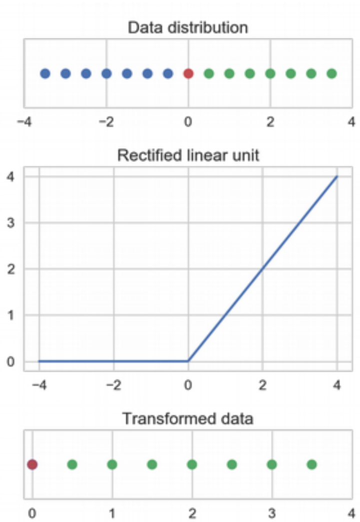
Effect of activation function

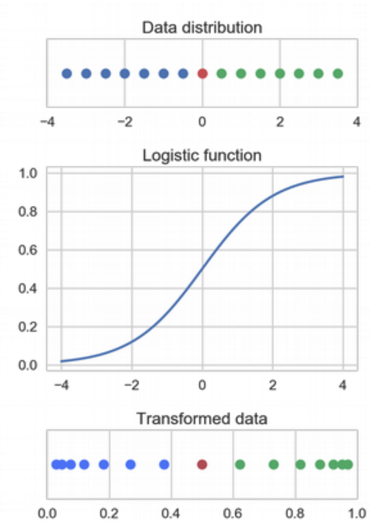
Linear activation

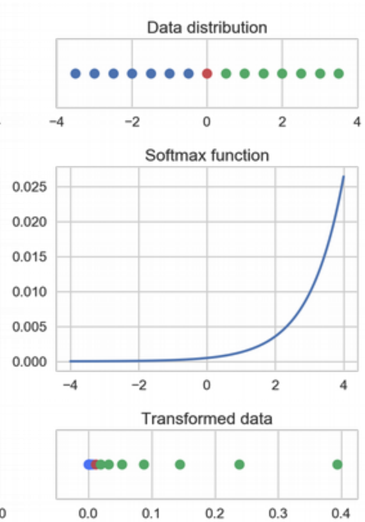
Non-linear activation functions











$$y = mx + b$$

$$y = \begin{cases} 1, \text{if } x > 0 \\ 0, \text{otherwise} \end{cases}$$

$$y = \begin{cases} x, \text{if } x > 0 \\ 0, \text{otherwise} \end{cases}$$

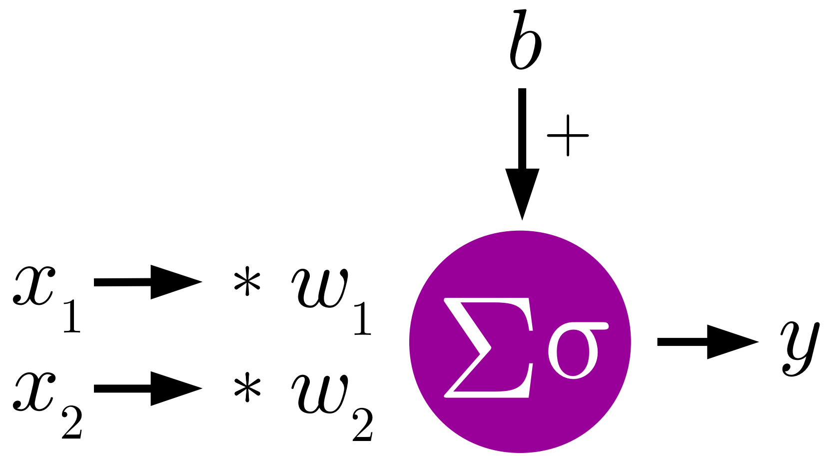
$$y = \frac{1}{1 + exp(-x)}$$

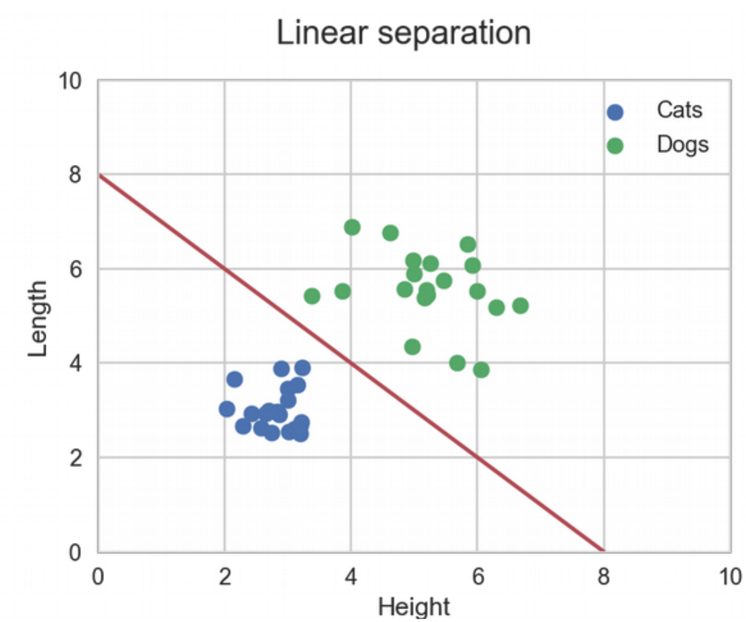
$$y_i = \frac{exp(x_i)}{\sum_n^N exp(x_n)}$$

Geometric interpretation

A single perceptron with step function as activation function is a linear, binary classifier

Inputs on one side of the linear separation are mapped to 0, the ones on the other side to 1





Linear and Nonlinear Transformations

Linearity

A transformation f is linear if and only if, the transformation is additive

$$f(a+b) = f(a) + f(b)$$

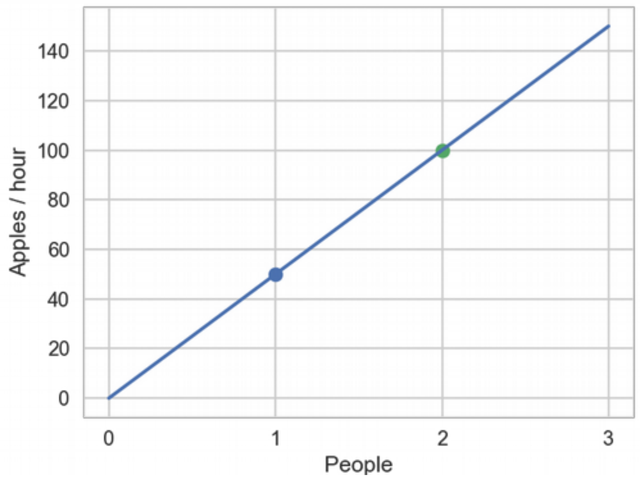
and, homogeneous

$$f(\alpha a) = \alpha f(a), \text{ for all } \alpha$$

Examples

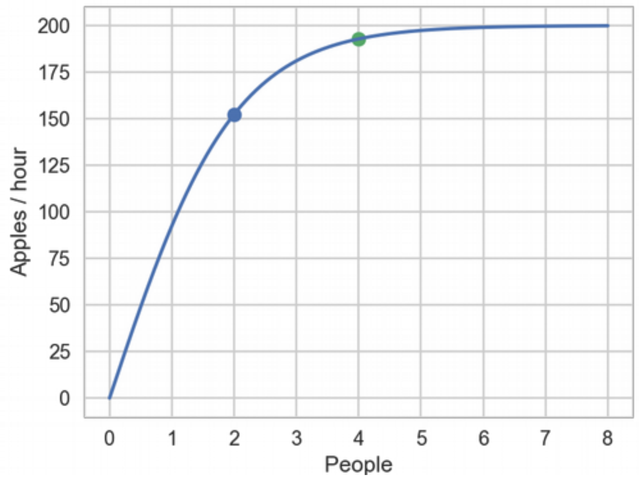
Linear transformation

Additive: You collect apples for two hours or you and your friend collect apples for one our each



Non-linear transformation

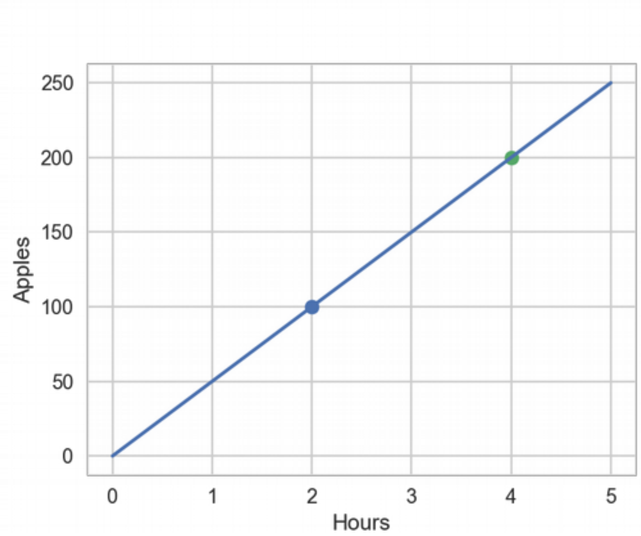
Efficiency in collecting apples first increases with more people helping, but then slowly saturates



Examples

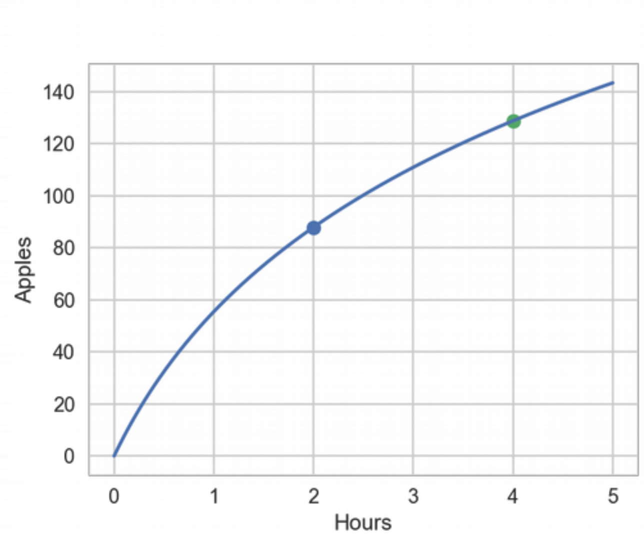
Linear transformation

Homogeneity: You collect twice as many apples, if you work twice as long



Non-linear transformation

You get slower and less productive the longer you collect apples



Combinations of linear transformations

Any successive combination of linear transformations can be represented by a single linear transform

Shift Shift Shift

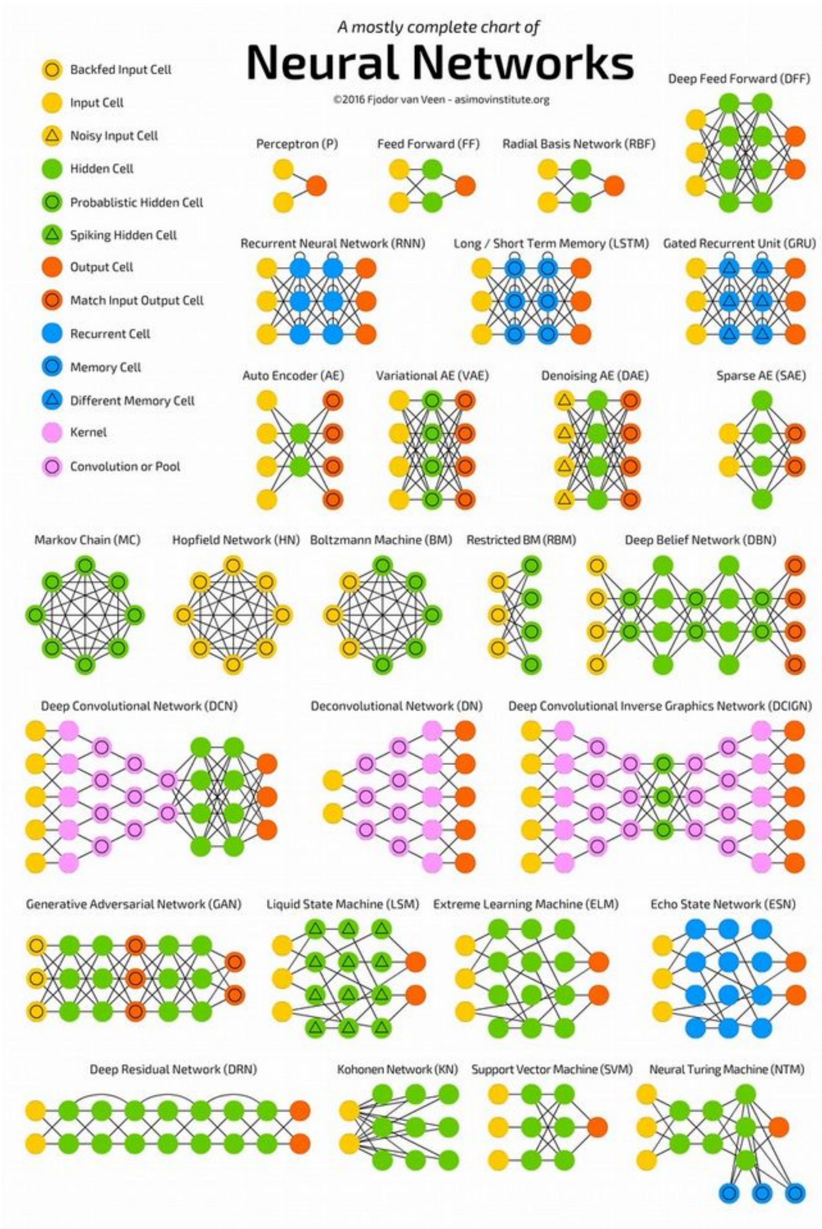
$$y = (((x^T W_1 + b_1) w_2 + b_2) \ldots) w_n + b_n$$

New range New range New range

Deep networks with linear activation functions, can be reduced to identical single-layer networks

Deep Feed-Forward Neural Networks

Types of neural networks



[1]

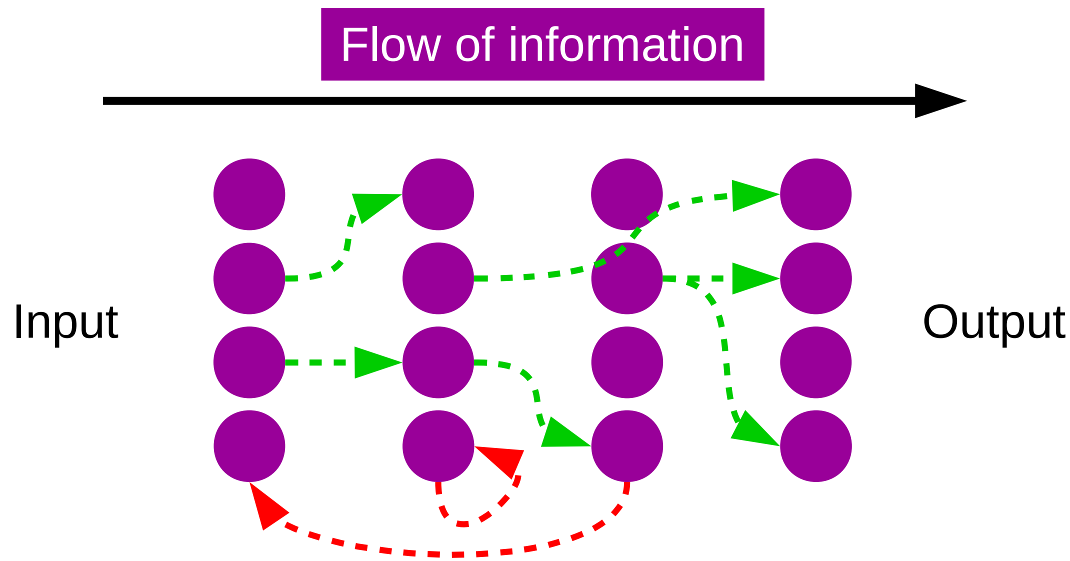
Connect multiple artificial neurons to a network

Find an overview and the related original papers at the Neural Network Zoo

[1] *asimovinstitute.org*

Feed-forward neural networks

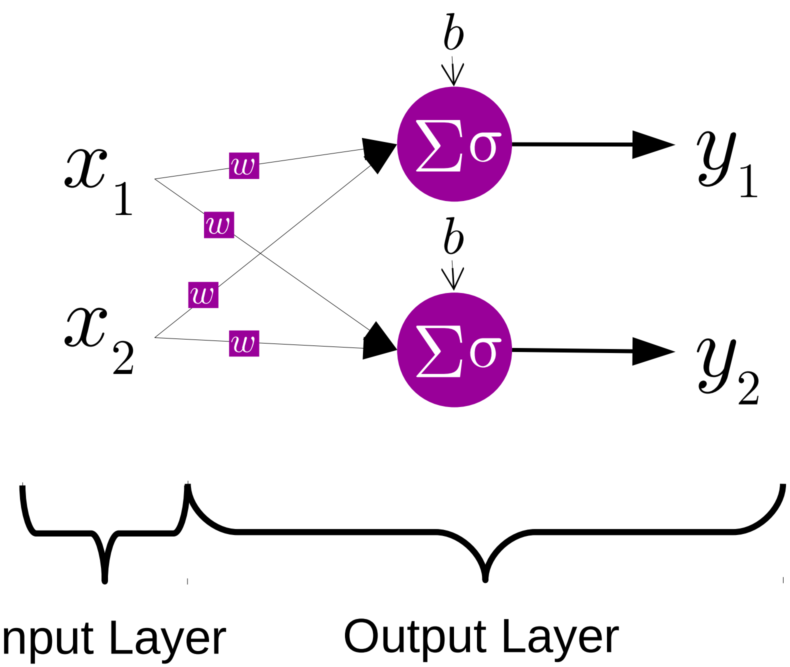
Information flows forwards from the input layer to the output layer

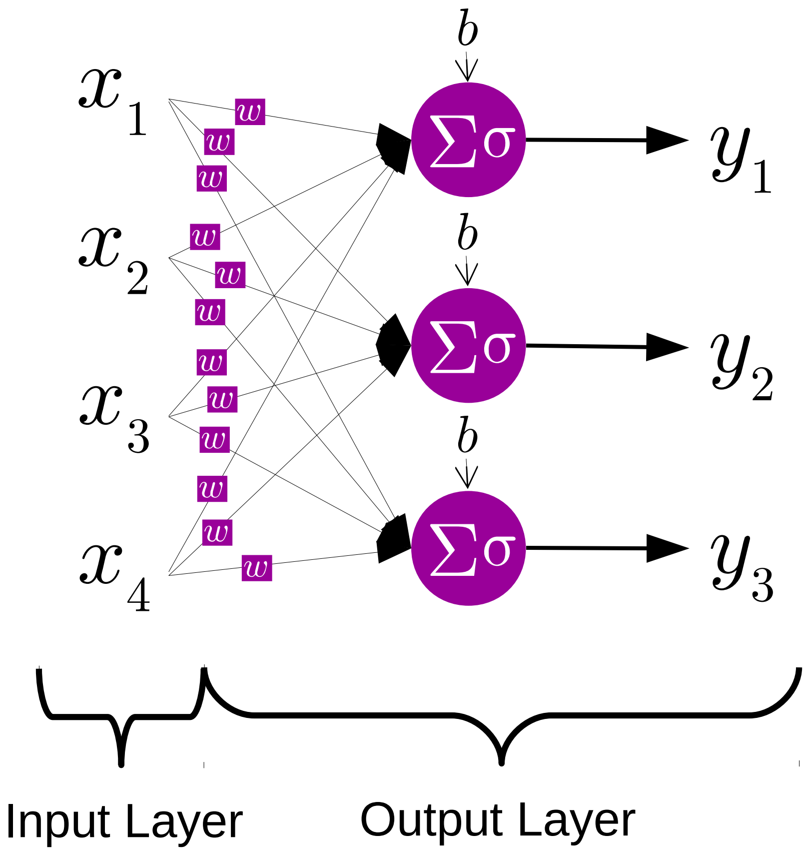


There are no recurrent and no feedback connections, which connect layers to themselves or later to earlier layers

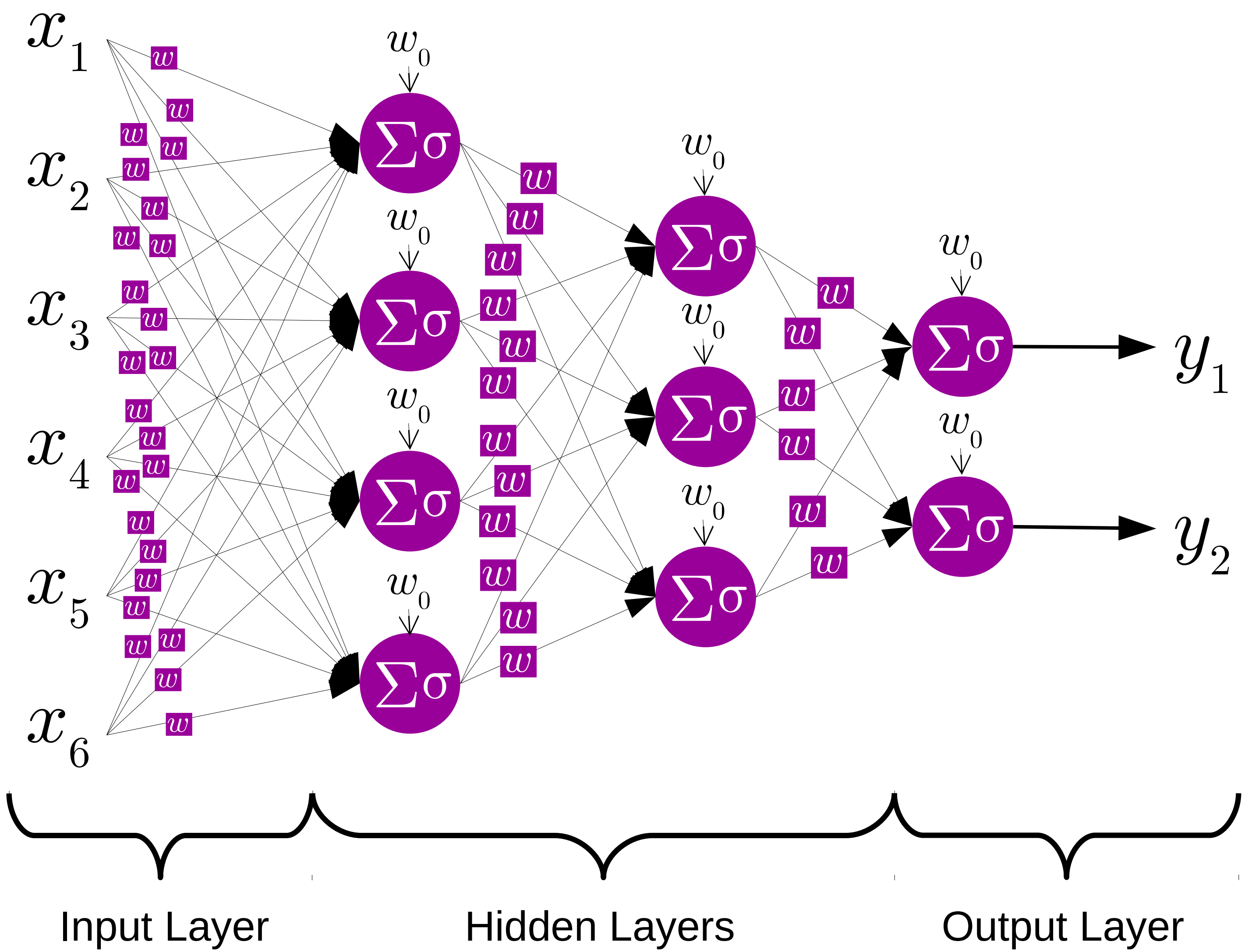
A two neuron network

For *m* nodes in layer 1 and *n* nodes in layer two, there are *m*n* connections and weights in a fully connected feed-forward network





Multilayer perceptron



Cybenko's Theorem

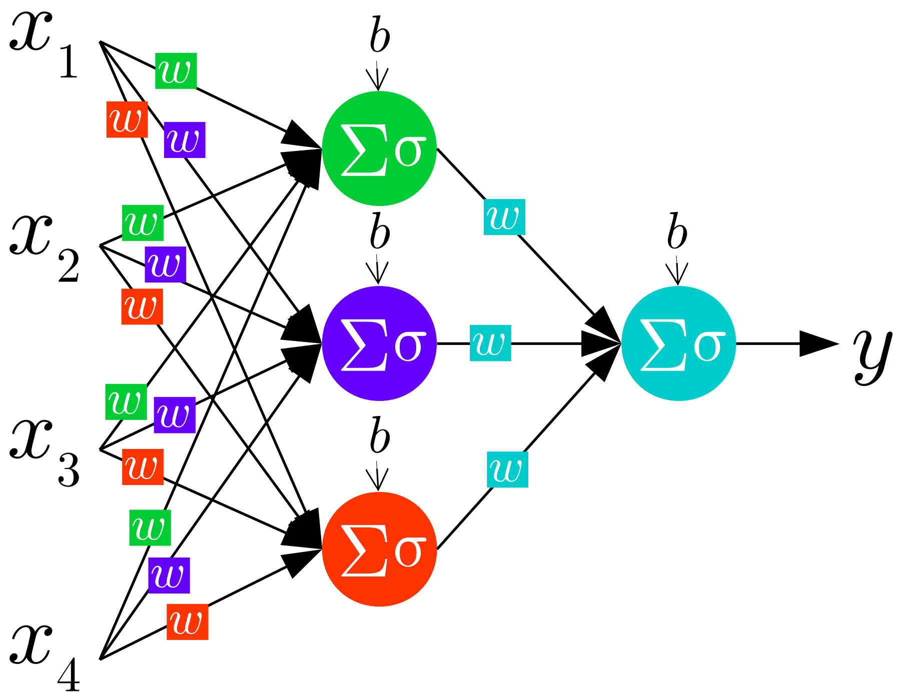
Any continuous function, can be approximated arbitrarily close with a feed-forward network with an input layer, one hidden layer with a finite amount of neurons with a non-linear activation function and a linear output layer

Network might be incredibly inefficient and hard difficult to train

Original paper by George Cybenko from 1989

Example network

- Four input variables
- One hidden layer with three artificial neurons
- A single output layer neuron



Forward Step

Terms and notation

We call the input to our network *input* and denote it with

$$\bar{x} \quad X$$

We call the weighted inputs (to any layer) *drive* and denote them with

$$d = \bar{x}^T \bar{w} \qquad \bar{d} = \bar{x}^T W \qquad D = X^T W$$

We call the output of the activation function *activation* and denote it with

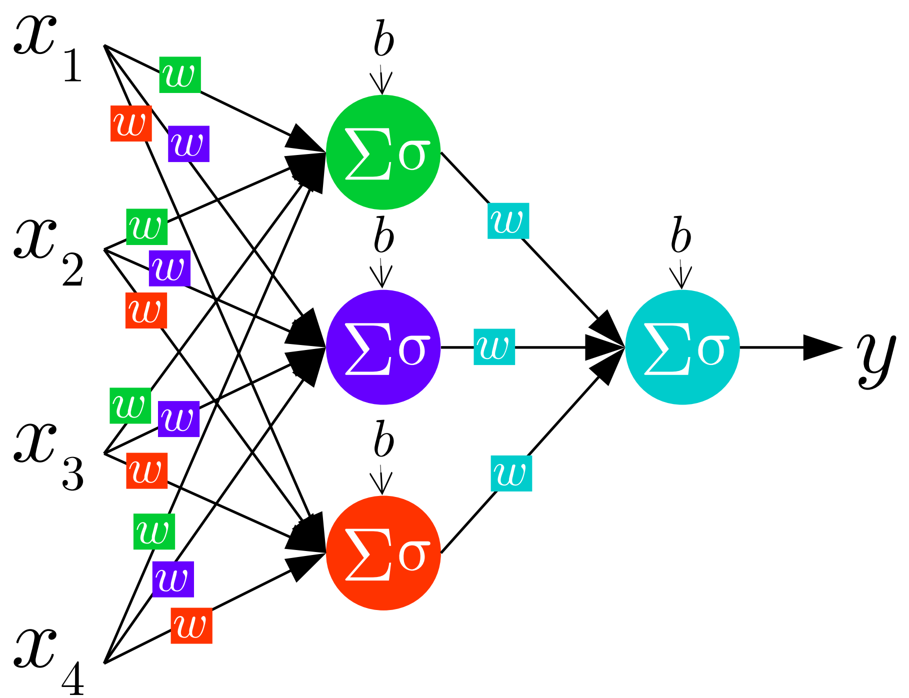
$$a = \sigma(d) \qquad \bar{a} = \sigma(\bar{d}) \qquad A = \sigma(D)$$

We call the output of the network *output* and denote it with

$$y = a_{\text{last\_layer}} \qquad \bar{y} = \bar{a}_{\text{last\_layer}} \qquad Y = A_{\text{last\_layer}}$$

Forward step, Layer 1

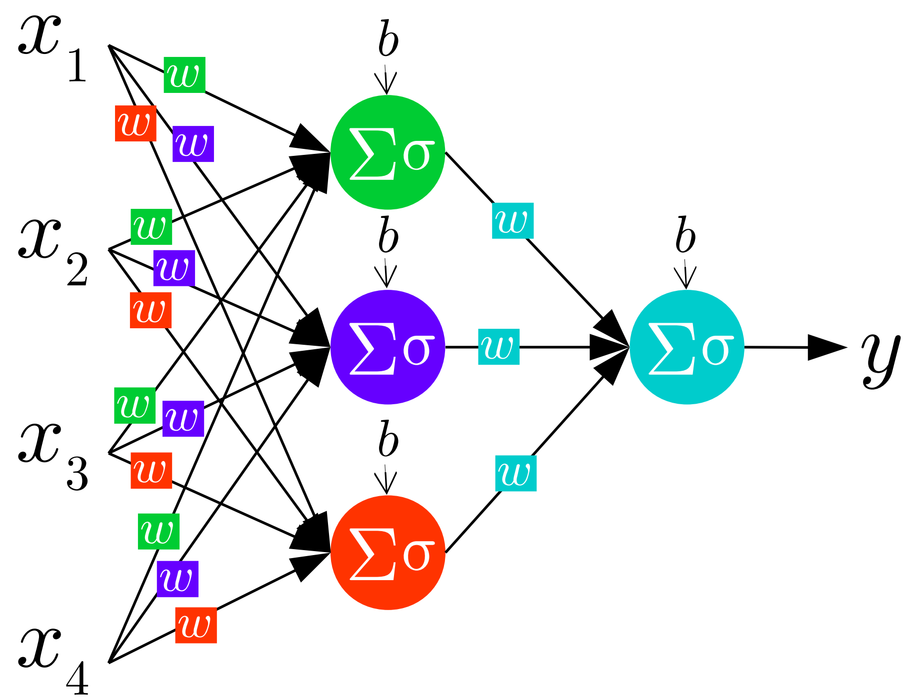
1) Input as vector and weights as matrix
2) Substitute biases
3) Multiply inputs with weights
4) Apply activation function



$$\bar{x} = \begin{bmatrix} 1 & x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T \quad W_1 = \begin{pmatrix} w_{01} & w_{02} & w_{03} \\ w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{pmatrix} \quad \bar{a_1} = \sigma(\bar{x}^T W_1)$$

Forward step, Layer 2

1) Substitute bias
2) Multiply layer 1 output with second weight matrix
3) Apply activation function



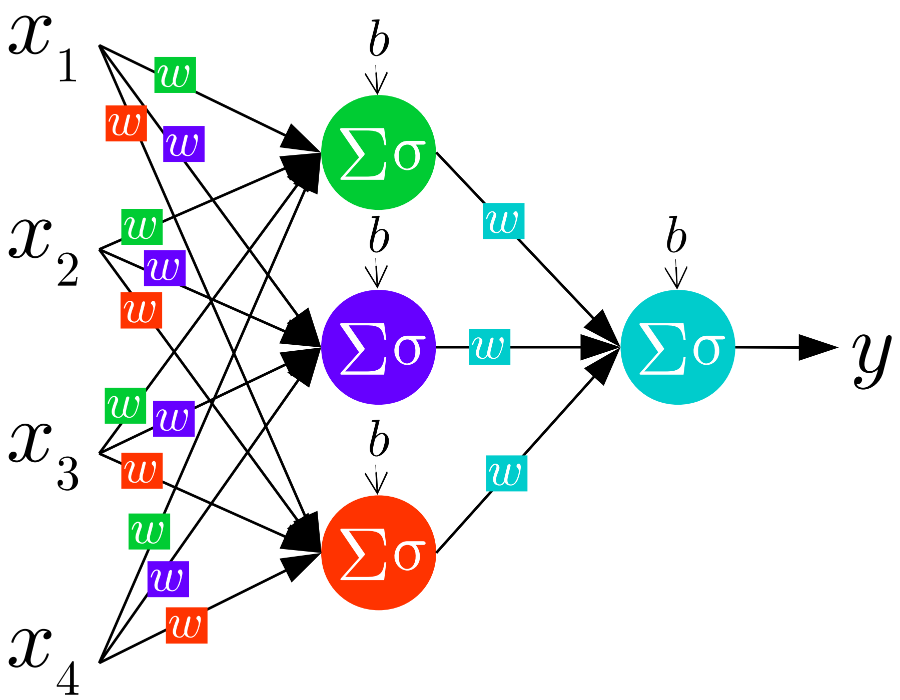
$$\bar{a_1} = \begin{bmatrix} 1 & a_{11} & a_{12} & a_{13} \end{bmatrix} \qquad \bar{w_2} = \begin{pmatrix} w_{01} \\ w_{11} \\ w_{21} \\ w_{31} \end{pmatrix} \qquad y = a_2 = \sigma(\bar{a_1}\bar{w_2})$$

Forward step

$$\bar{a_1} = \sigma(\bar{x}^T W_1)$$

Insert

$$y = a_2 = \sigma(\bar{a_1}^T \bar{w_2})$$



Network Function

$$y = \sigma(\sigma(\bar{x}^T W_1)\bar{w_2})$$

Dimensionality check

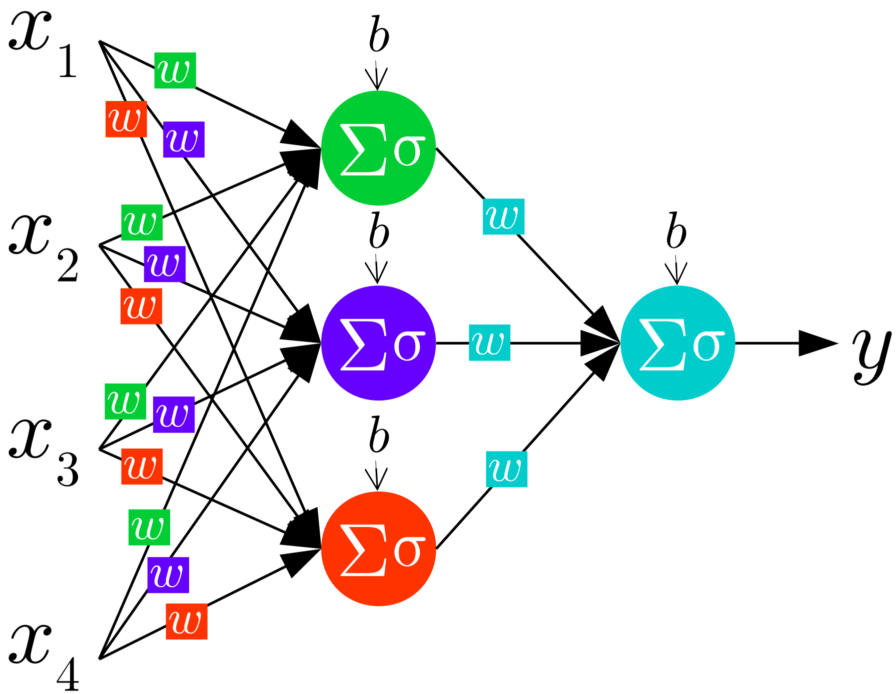
Input: 4x1
After bias substitution: 5x1

W_1: 4x3
After bias substitution: 5x3

a_1: 1x3
After bias substitution: 1x4

W_2: 3x1
After bias substitution: 4x1

a_2, y: 1x1



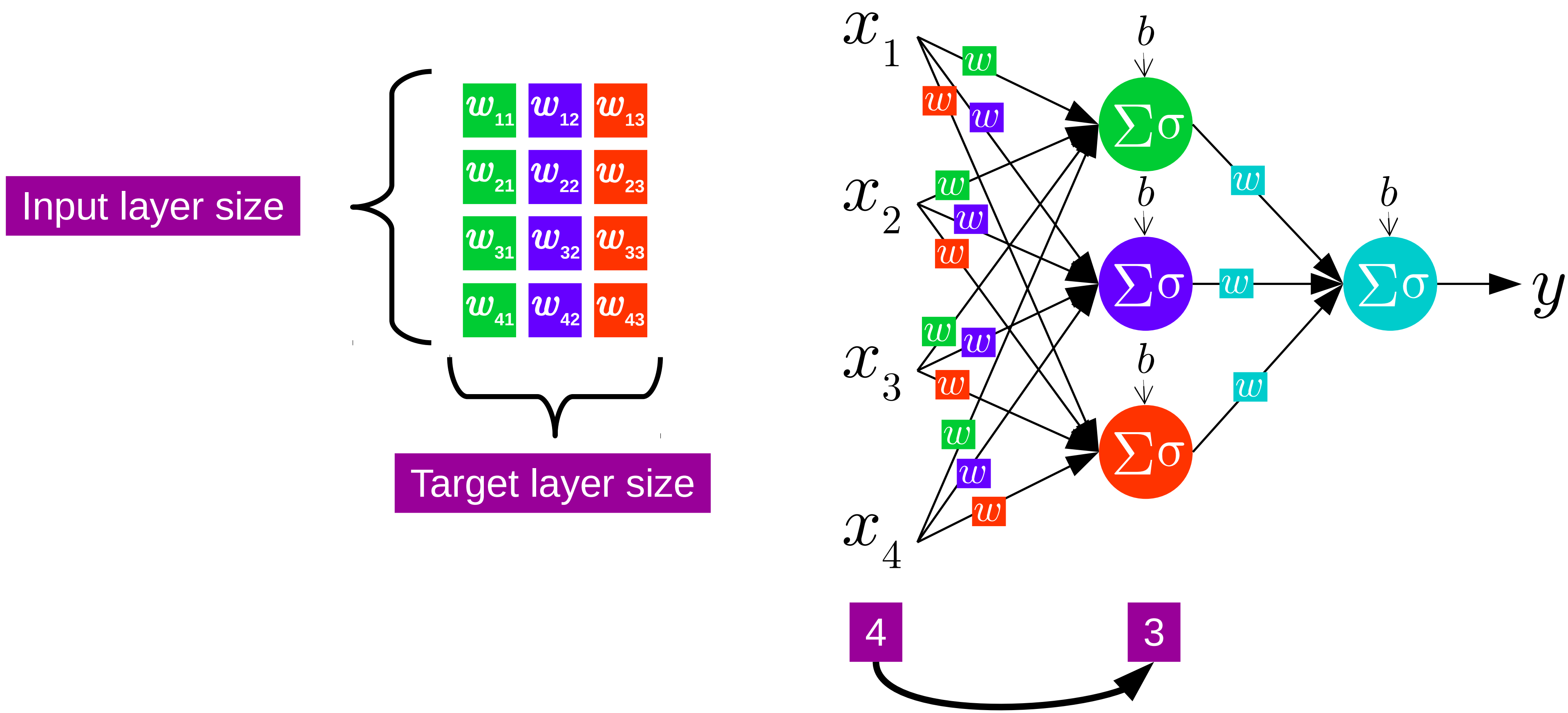
$$y = \sigma(\sigma(\bar{x}^T W_1)\bar{w_2})$$

$$\bar{a_1} = \sigma(\bar{x}^T W_1)$$ [1x5*5x3] = 3x1

$$y = a_2 = \sigma(\bar{a_1}\bar{w_2})$$ [1x4*4x1] = 1x1

Weight dimensionality

A MxN weight matrix, maps 1xM input values or activations onto a Nx1 drive vector



Supervised Learning

Supervised learning

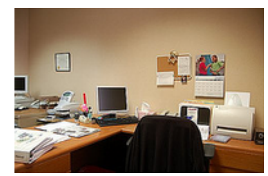
Try to infer a function from a labeled data set

After training: determine labels for unseen data

Network needs to generalize, pure memorization of training samples is insufficient

Data | Label


House


Office


Mountain


Street


Car

[2] *image-net.org*

Formalization

Feature tensor

$$\text{training\_data} = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$$

Label

Try to approximate the function g which maps *any* possible x to it's desired label $\hat{y}$, with a function f

$$\hat{y} = g(x) \qquad y = f(x;\theta)$$

Set of trainable parameters

Relation of input and output

Is there a $g(x)$ with $\hat{y} = g(x)$?

Are x and $\hat{y}$ related?

Genre ✓

Rating ?

Current value of Apple share ✗

Is there enough information in x to derive $\hat{y}$?

"Cat" ✓

Age ?

Name ✗

Formalization

In our case f is the network function, an the set of trainable parameters θ, are the weights and biases

$$\hat{y} = g(x) \qquad y = f(x; \theta)$$

Set of trainable parameters

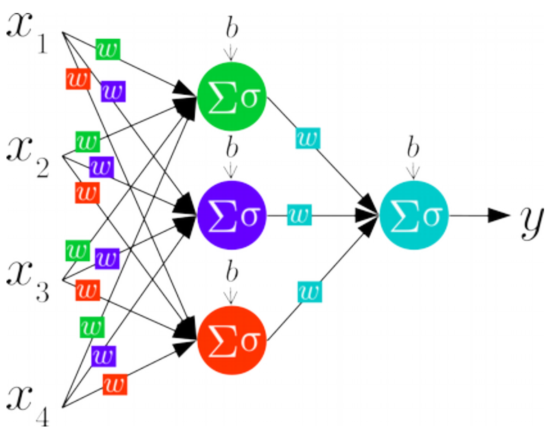
Goal: $g(x) = f(x; \theta) \Leftrightarrow \hat{y} = y$

How to represent $\hat{y}$ and how to design f to create an output of that representation type?

Output Activation Functions

Output activation functions

The output layer receives features that were extracted from previous layers



The activation function of the output layer performs the final transformation, from the extracted features and hence constitutes the shape of y

Linear unit

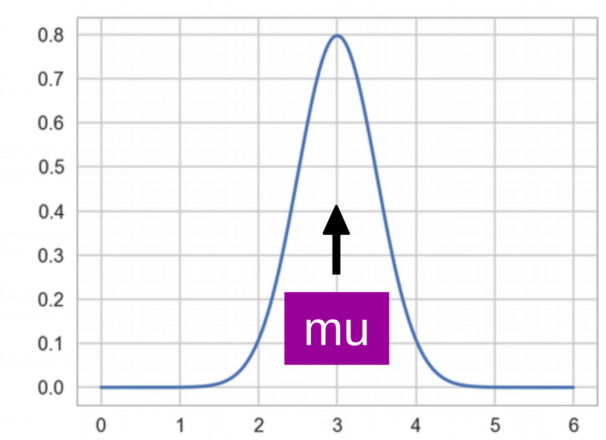
Linearly maps arbitrary input onto range]-∞, ∞[

Can be used to map evidence from the last hidden layer onto the mean of normal

$$p(y = \mu | x)$$

$$y = \bar{a}_n W + b$$

Last hidden layer activity



$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Bernoulli distribution

A Bernoulli distributed random variable takes value 1 with probability p and 0 with probability $q = 1 - p$

If x belongs to either one or another category y (binary outcome), we can think of it as a Bernoulli distributed random variable which is conditioned by x

$$p(y = 1|x)$$

Examples:

- Yes-no questions
- Positive or negative review
- Cancer or no cancer

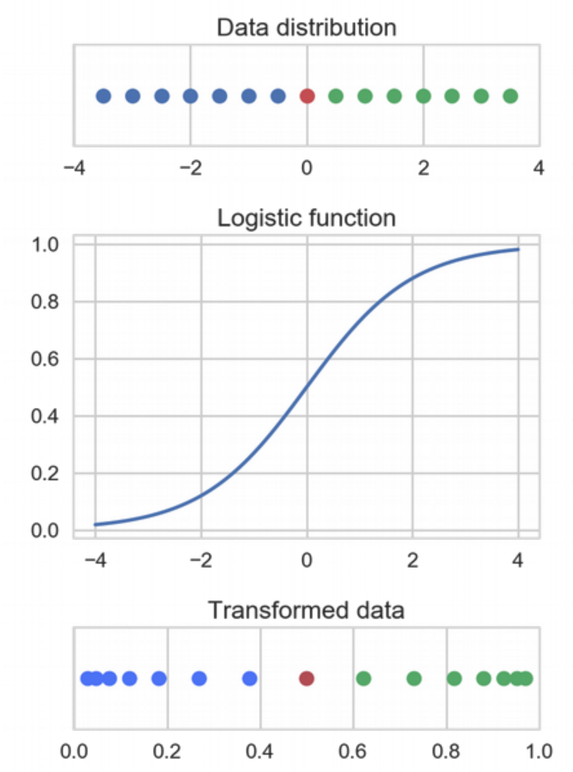
The logistic function

Maps arbitrary input onto range]-1, 1[

Large positive drive results in output close to 1

Large negative drive results in output close to 0

Maps evidence from last layer onto either one or another class in a graded fashion



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

Categorial distribution

A categorical distributed random variable takes one out of $k \in K$ possible **exclusive** outcomes

If x belongs to exactly one of K categories, we can think of it as a categorial distributed random variable which is conditioned by x

$$p(y = k|x)$$

$$\sum_i p(y = k_i) = 1$$

Examples:

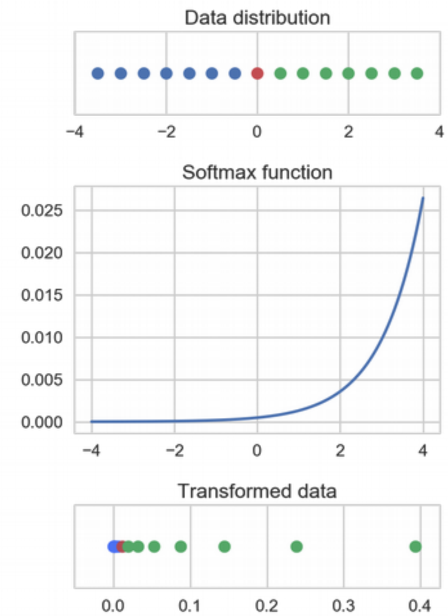
- Rolling a dice
- Categorization of animals
- Categorization of colored chocolate beans

The logistic function

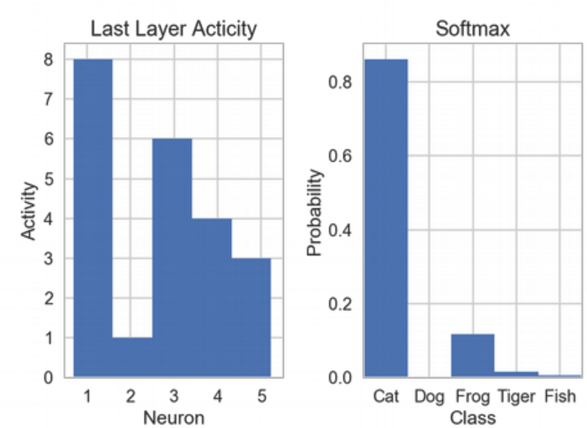
Maps a set of inputs onto range]0, 1[, such that the sum of the output values is 1

Individual values are weighted exponentially. Larger values become in relation to other values even larger and smaller ones smaller

Maps evidence from last layer onto a cumulative probability distribution



$$y_i = \frac{exp(x_i)}{\sum_n^N exp(x_n)}$$



THANK YOU!

3

3 vine.com

Sources

[1] Retrieved October 9th, 2016 from http://www.asimovinstitute.org/neural-network-zoo/

[2] Retrieved October 29th, 2017 from http://image-net.org/explore_popular.php

[3] Retrieved October 24th, 2016 from https://vine.co/v/iXWz6516Ipz