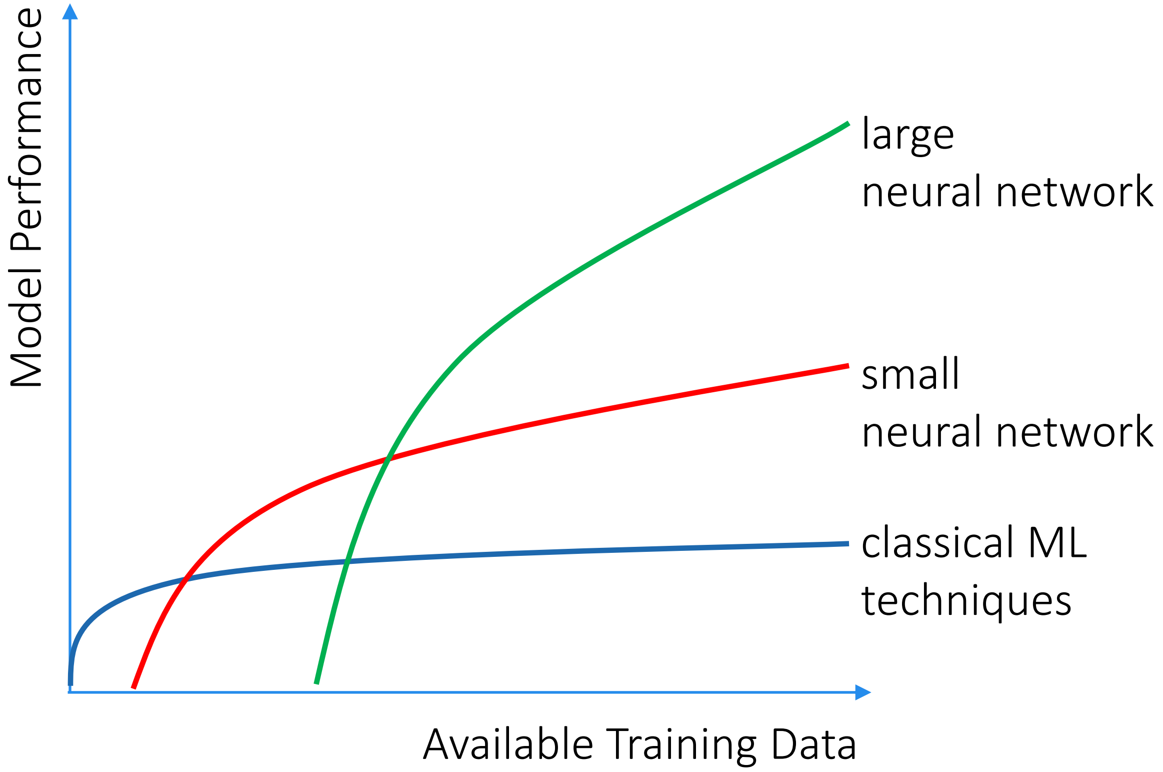
Deep Learning Details

Bert Gollnick

Deep Learning Details

Deep Learning Performance

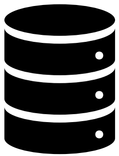
- Classical ML techniques work best for small datasets
- With increasing size of available data → neural networks outperform classical techniques



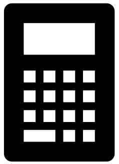
Deep Learning Details

Why did Deep Learning improve so much?

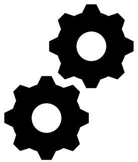
- Mainly four reasons why Deep Learning took off.



More
Data



Moore‘s Law
More computing
Power
GPU‘s



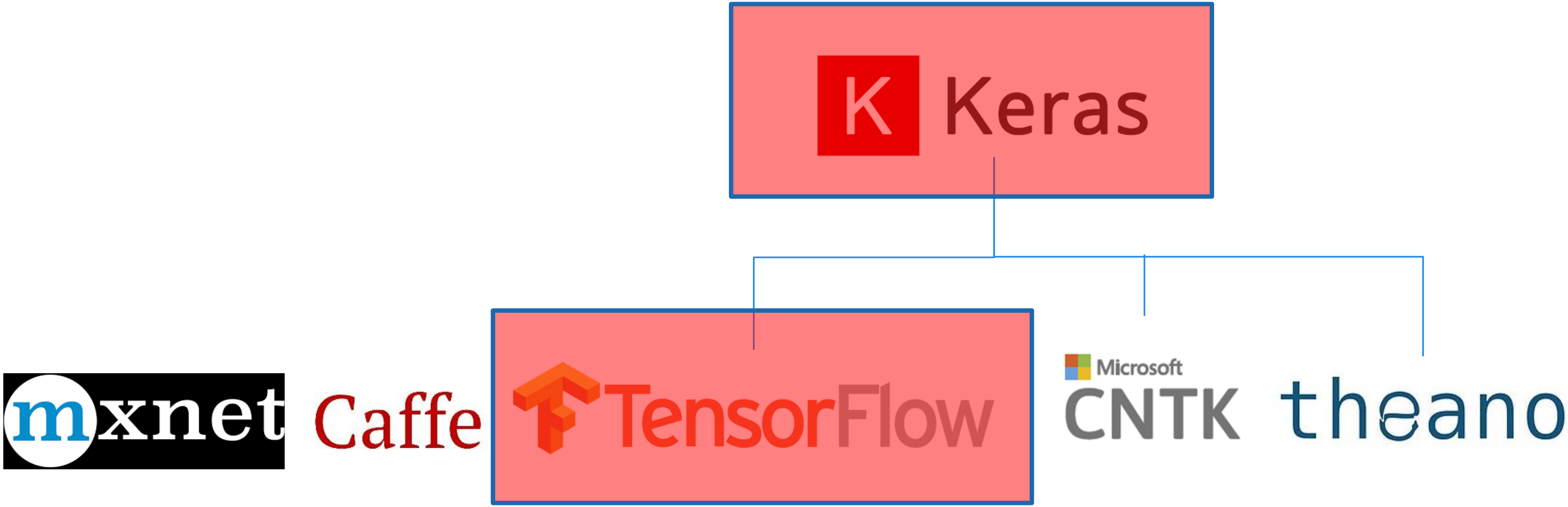
Better
Algorithms

Open
Source

Deep Learning Details

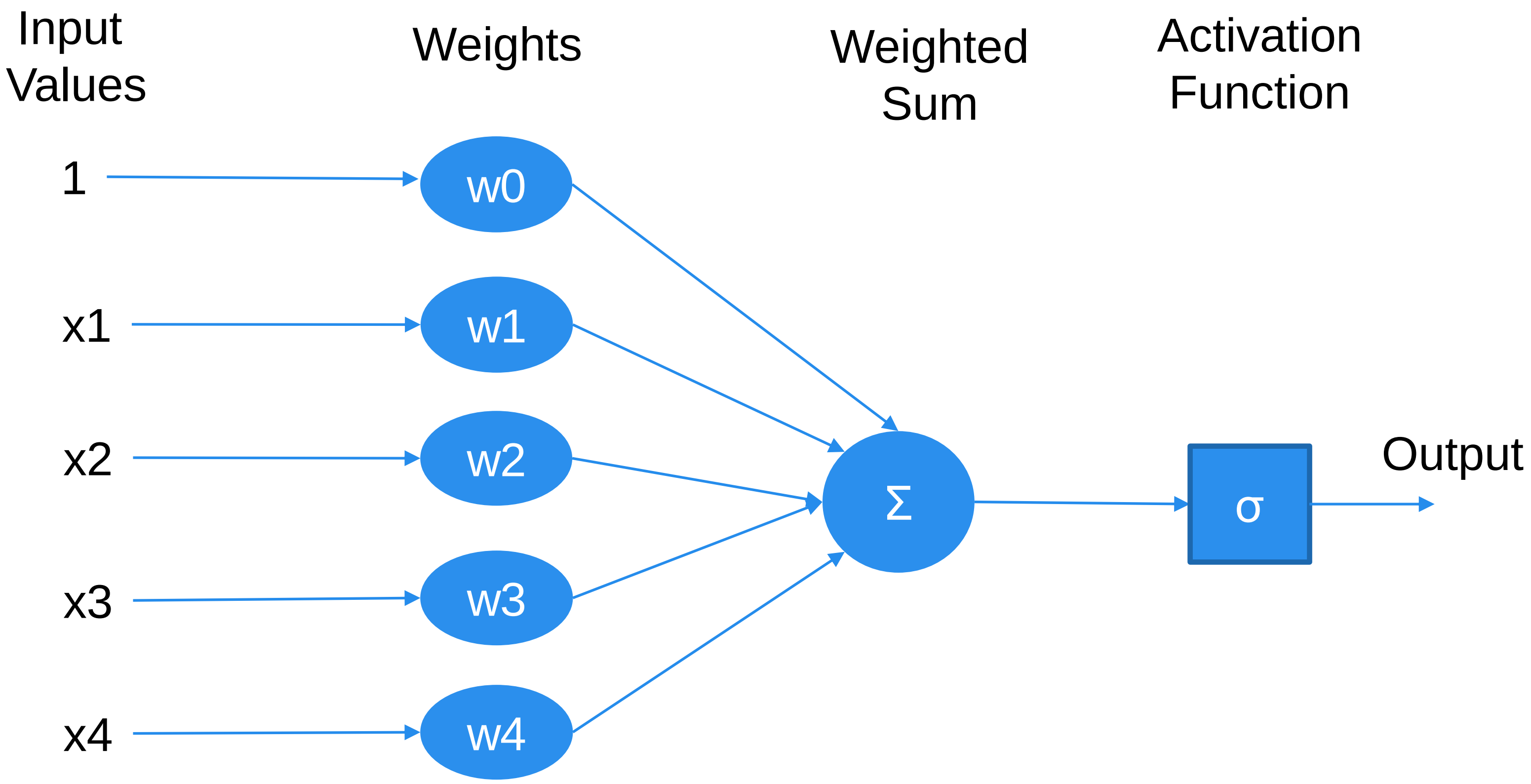
Frameworks

S



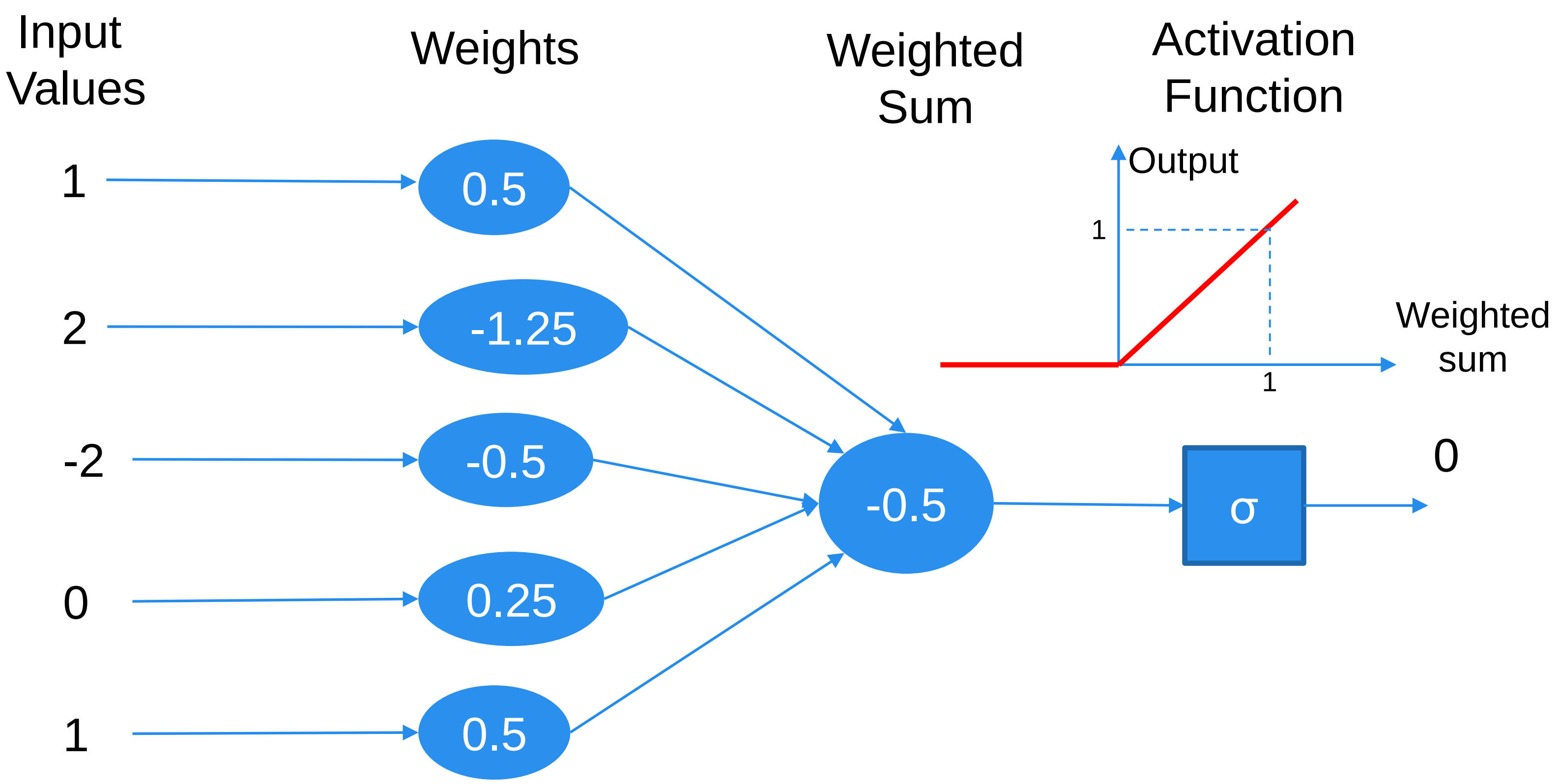
Deep Learning Details

Perceptron



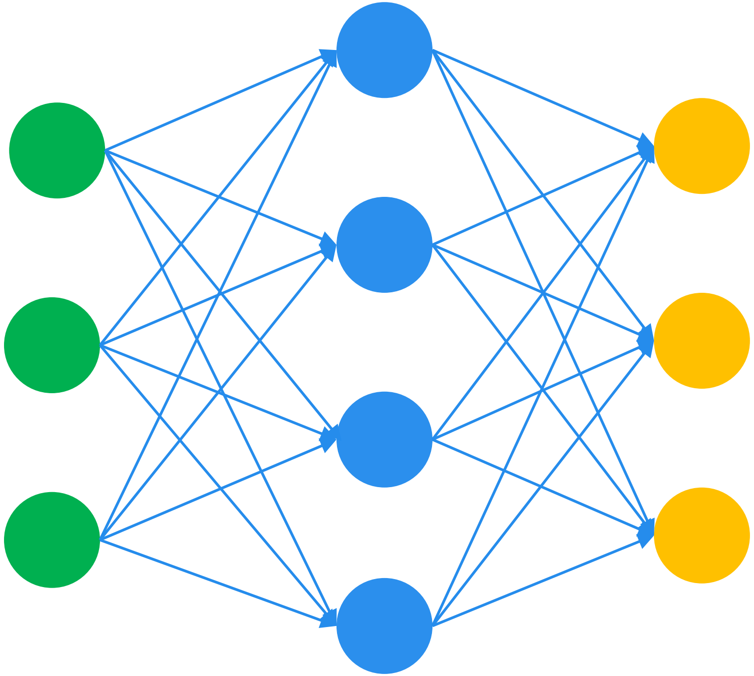
Deep Learning Details

Perceptron: Example



Deep Learning Details

Simple and Deep Neural Network

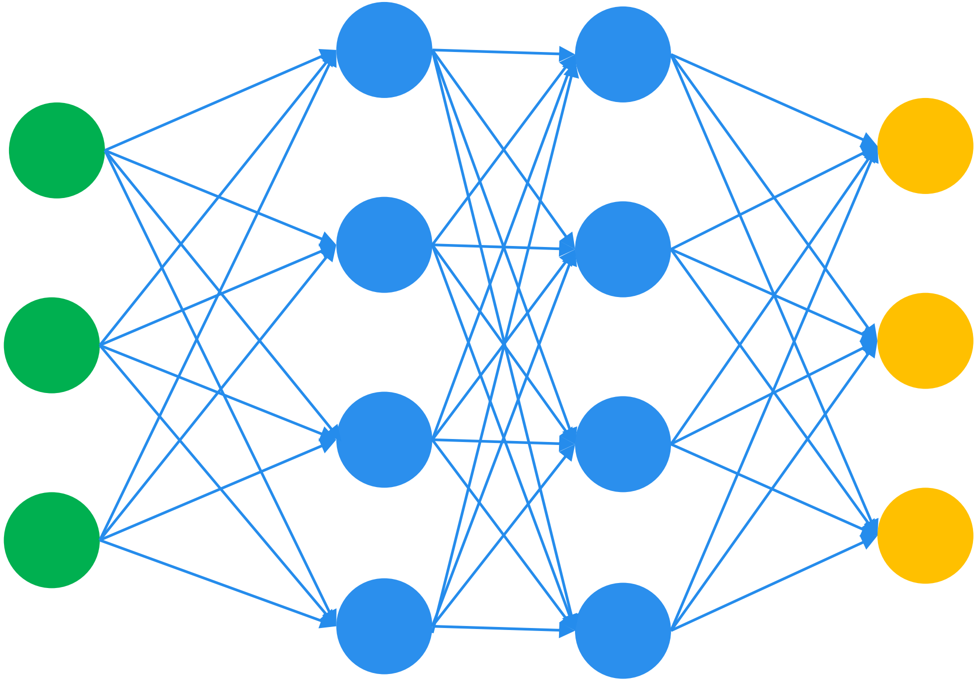


Simple Neural Network

Input Values
Hidden Nodes
Output Nodes

Deep Learning Details

Simple and Deep Neural Network



Deep Neural Network
Multi-Layer Perceptron

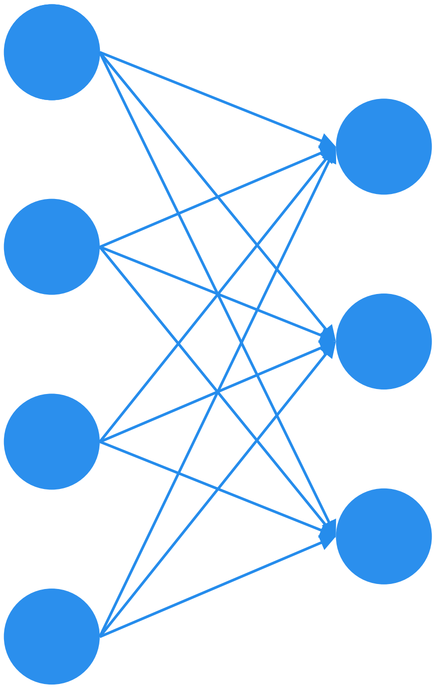
Input Values
Hidden Nodes
Output Nodes

Deep Learning Details

Layer Types

Dense Layer

- Each input layer is connected to each output layer
- Also called fully connected layer
- Usually non-linear activation function applied

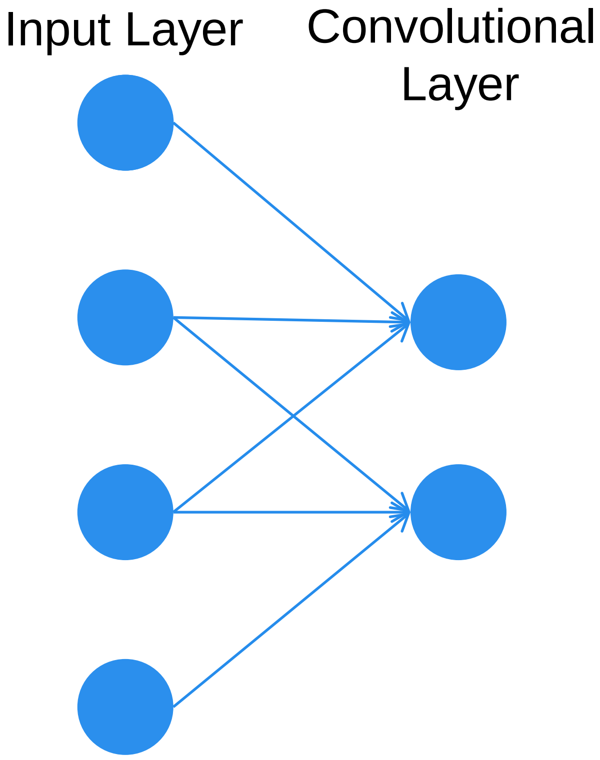


Deep Learning Details

Layer Types

1D convolutional layer

- Layer consists of filters
- Sequentially a subset of input layer is processed
- All nodes of input layer used



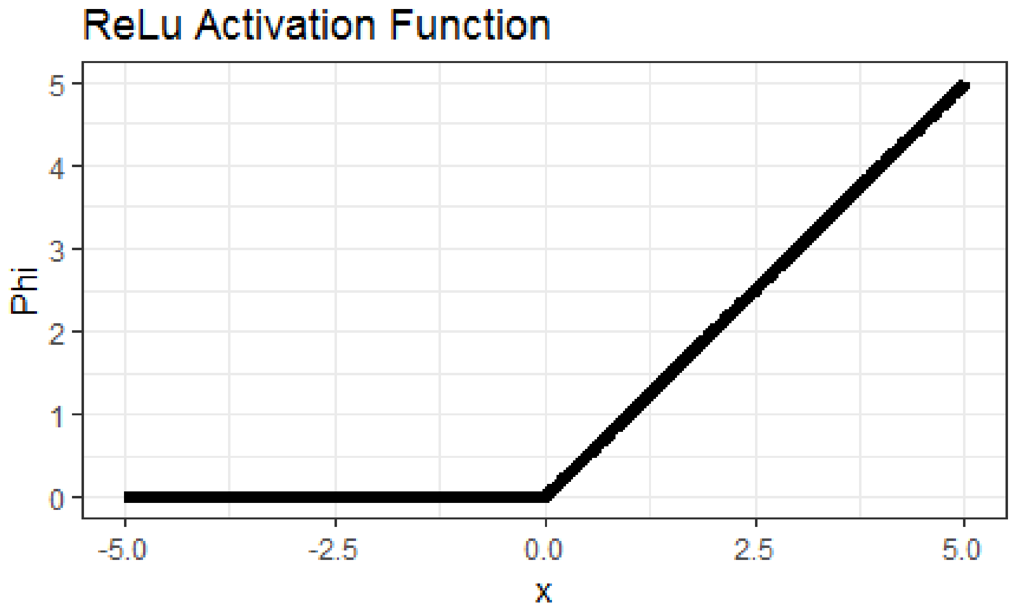
Deep Learning Details

Activation Functions

There are different activation functions.

Rectified Linear Unit (ReLu)

- Phi = max(0, x)
- Most common
- Non-linear



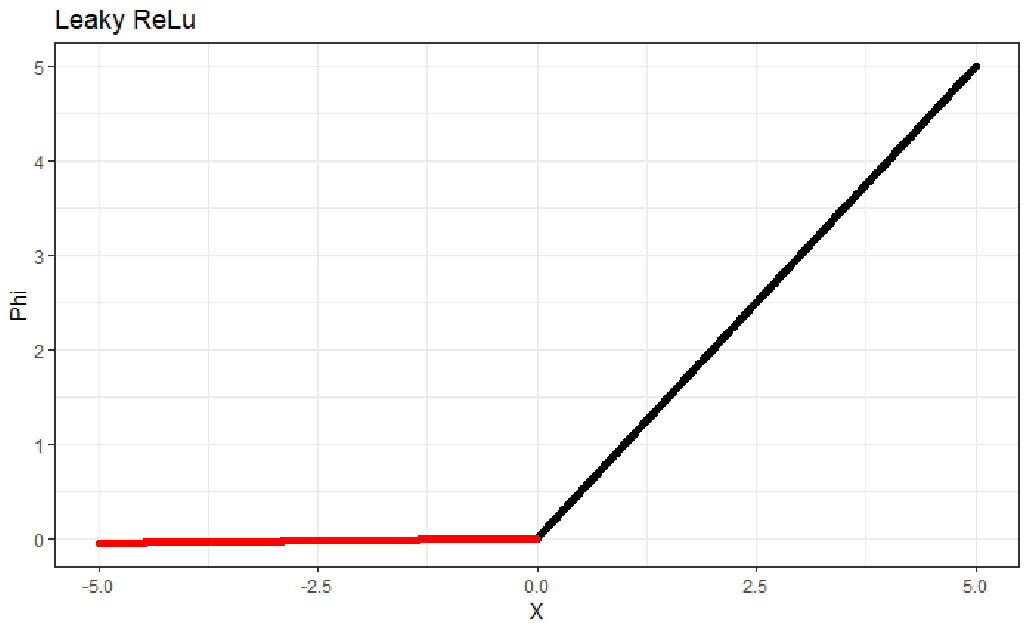
Deep Learning Details

Activation Functions

Leaky Rectified Linear Unit (Leaky ReLu)

- Phi(x) = { x if x>0; alpha * x otherwise

- alpha typically 0.01
- Instead zero for negative inputs, small gradient
- Gradient never zero



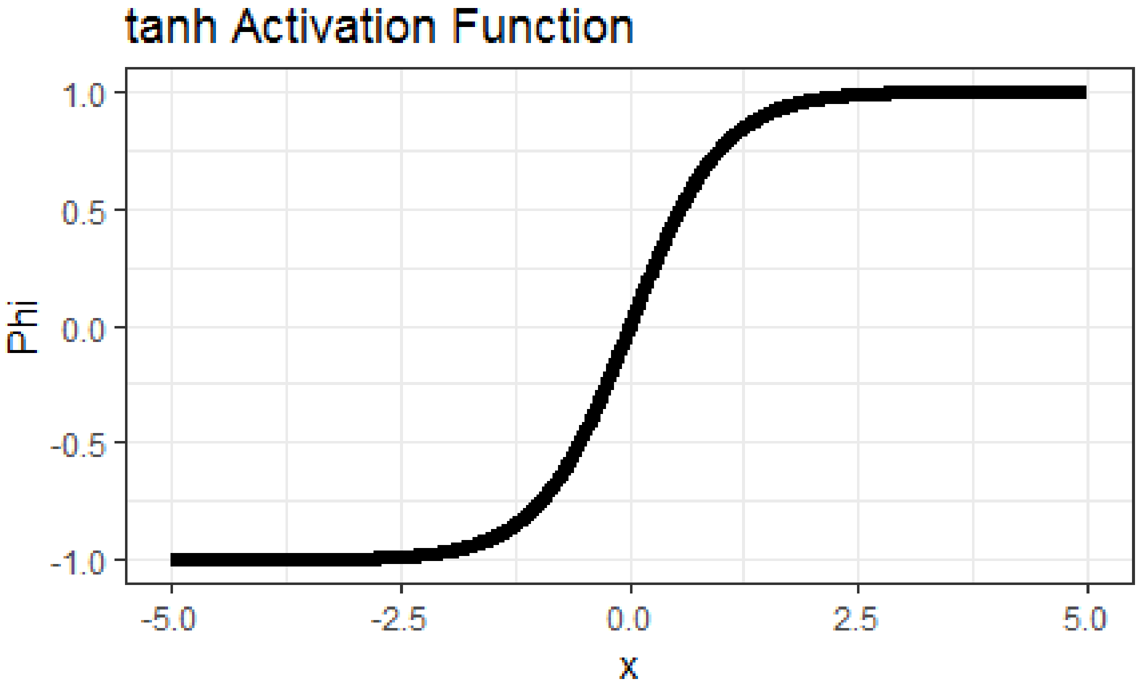
Deep Learning Details

Activation Functions

Hyperbolic Tangent (tanh)

- $\boldsymbol{Phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}}$

- Non-linear
- Relatively flat, except for small range
- Derivative small except for small range
- Might suffer vanishing gradient problem

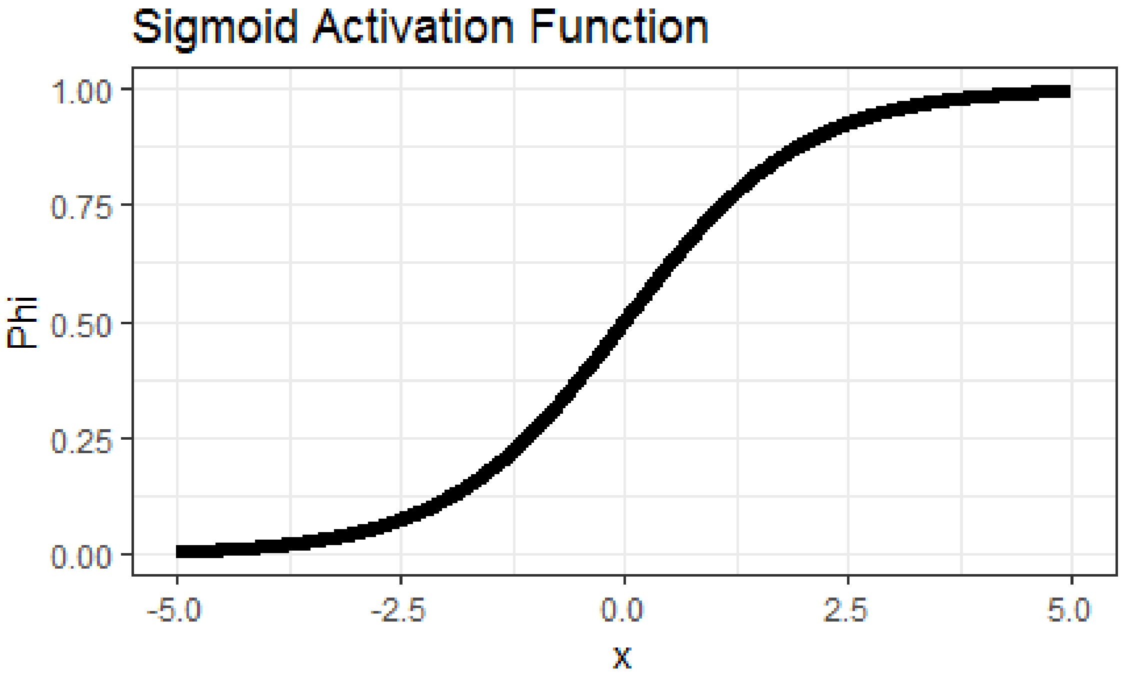


Deep Learning Details

Activation Functions

Sigmoid

- $Phi(x) = 1 / (1+e^{-x})$
- Non-linear
- Relatively flat, except for small range
- Derivative small except for small range
- Might suffer vanishing gradient problem
- Result range 0 to 1

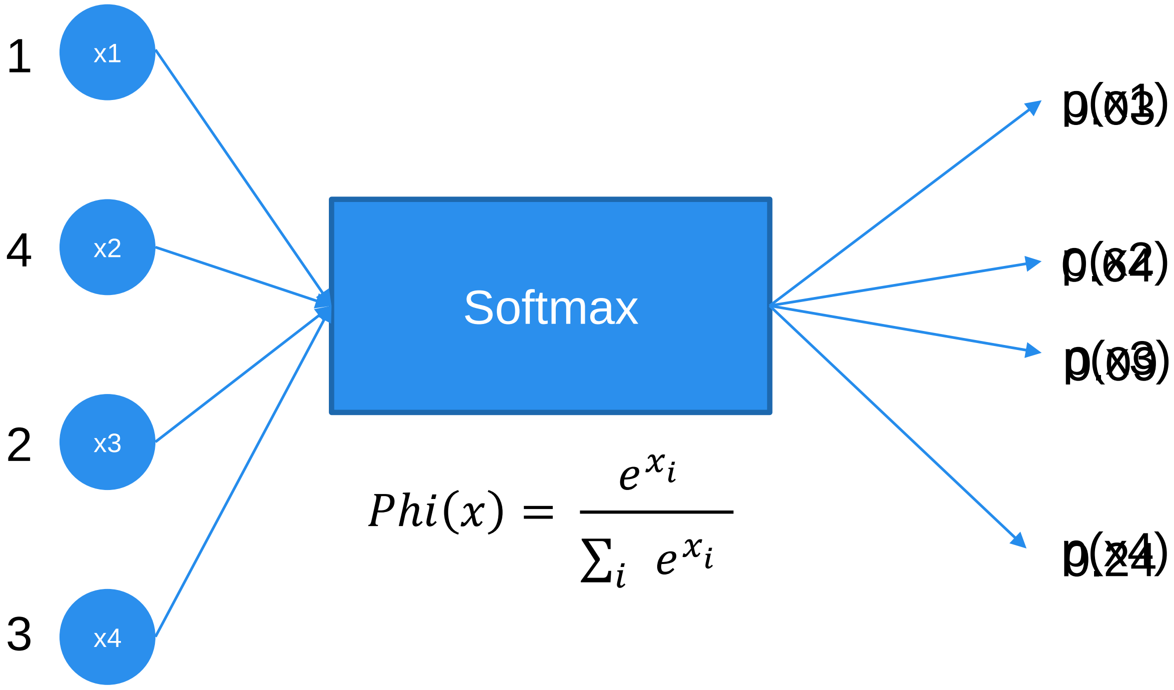


Deep Learning Details

Activation Functions

Softmax

- Used for multi-class prediction



$$Phi(x) = \frac{e^{x_i}}{\sum_i e^{x_i}}$$

Deep Learning Details

Loss Functions

- Evaluates model performance during training
- Gradual improvement due to optimizer
- Is minimized during training
- Multiple loss functions for one model possible (one for each output variable)

Regression

- Mean Squared Error $MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$
- Mean Absolute Error $MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n}$

Classification

- Cross Entropy

$$\mathrm{CE} = -(y_i \log \hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

Deep Learning Details

Optimizer

- Calculates updates of weights based on Loss Function
- implements gradient descent

Deep Learning Details

Wrapup

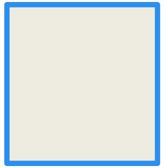
What you should know after this lecture:

- Simple Perceptron
- Multi-Layer Perceptron
- Most common layer types
- Most common activation functions
- Most common loss functions

Deep Learning Details

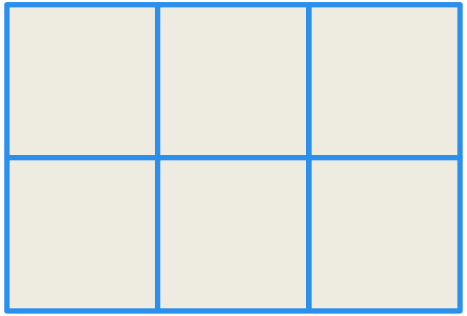
Tensor

- General representation of numeric objects

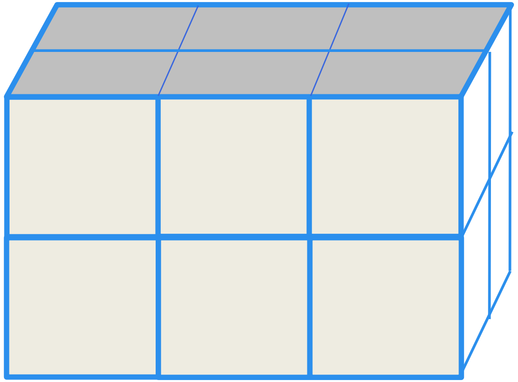


0-dimensional Tensor

1-dimensional Tensor



2-dimensional Tensor



3-dimensional Tensor