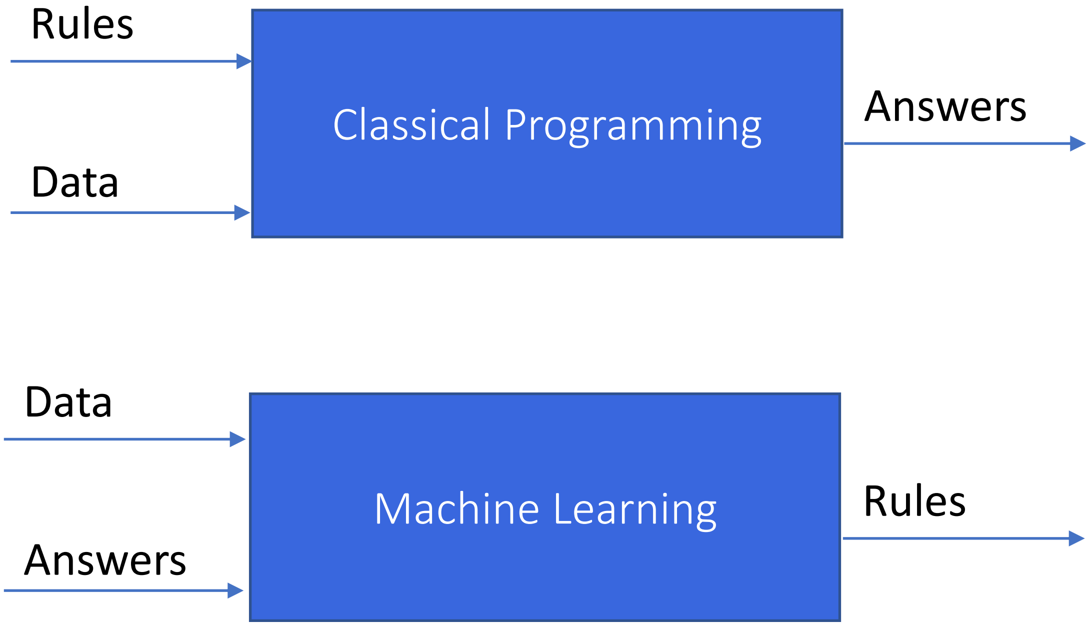
Deep Learning Workshop

Machine Learning 101

Machine Learning 101

Classical Programming and Machine Learning

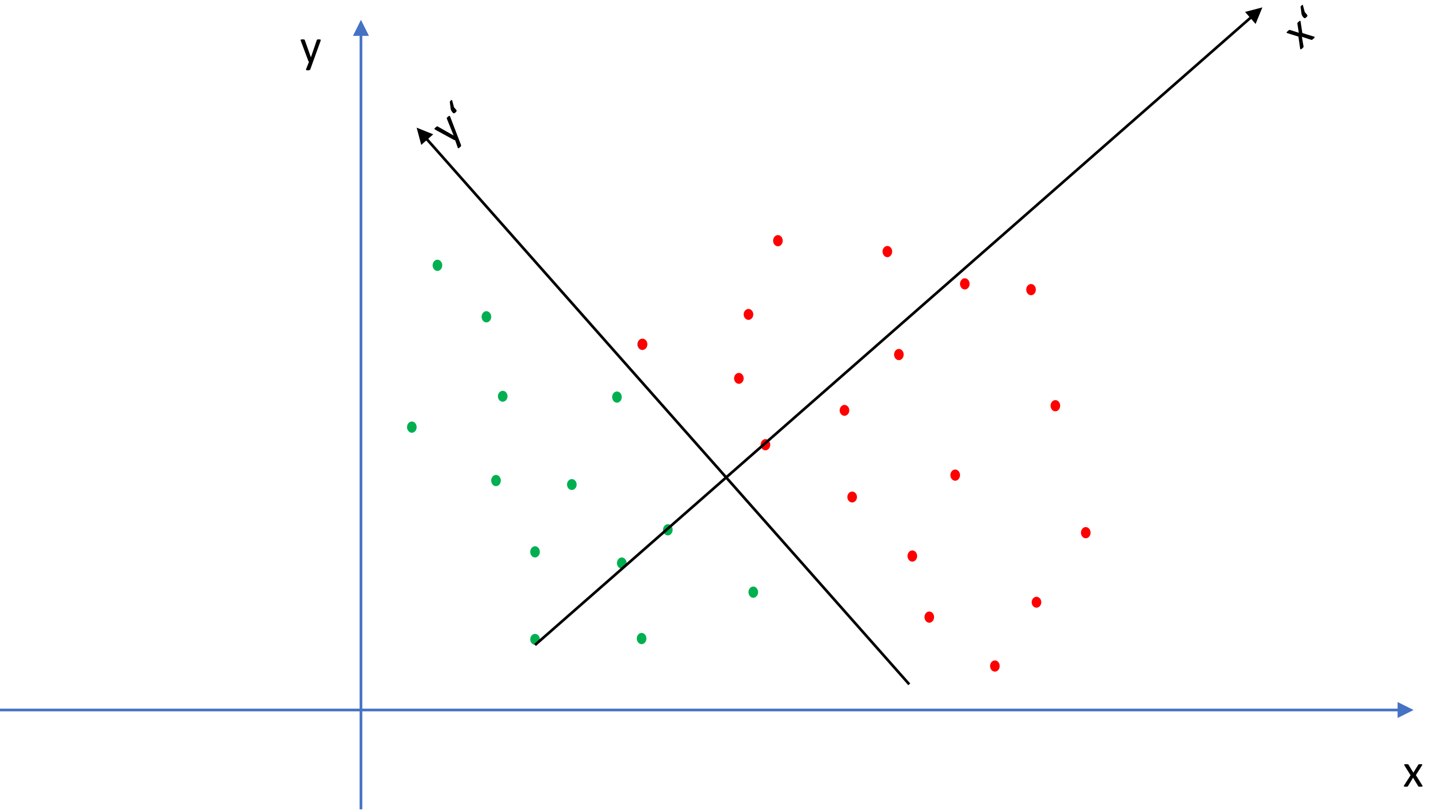


Inspired by: Francois Chollet and J.J. Allaire „Deep Learning with R and Keras“

gollnickdata.de

Machine Learning Overview

Data Transformation



Classification

Red, if x‘>0

Green, if x‘<=0

gollnickdata.de

Types of Machine Learning

Supervised, Unsupervised, Reinforcement Learning

Learning Type

Target Variable	Unsupervised	Supervised	Reinforcement Learning
Continuous		Regression	
Categorical	Clustering	Classification	

gollnickdata.de

Types of Machine Learning

Example: School Class

Supervised Learning
Task: Use Label / Target Variable
for Learning/Prediction

Name	Age	Learning Method	Class	Grade
Anton	14	A	Sport	2
Bert	15	B	Sport	2
Clare	13	A	Sport	3
Dave	16	B	Math	1
Emilia	15	A	Math	2
…				

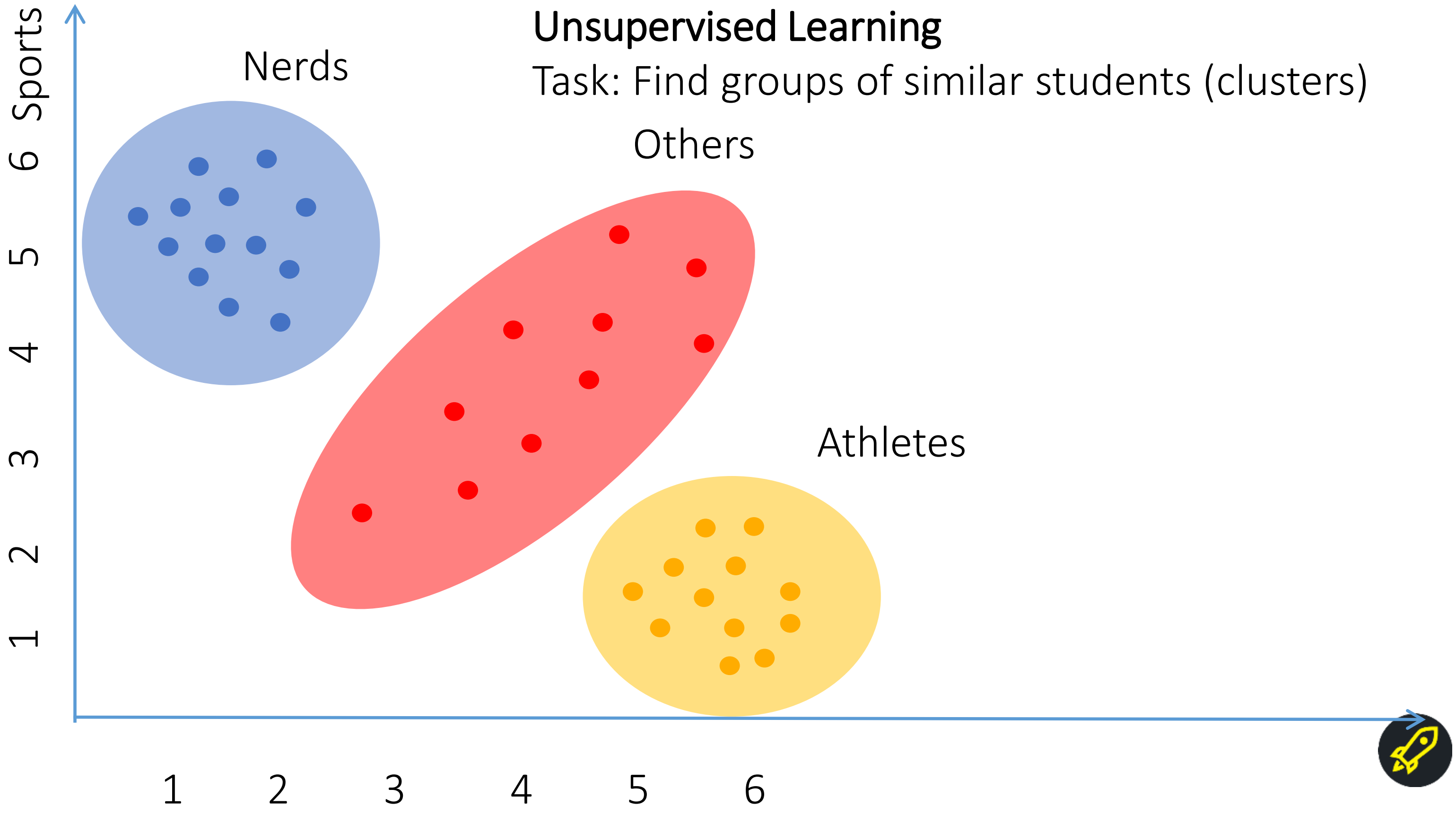
de

Types of Machine Learning

Example: School Class

Unsupervised Learning

Task: Find groups of similar students (clusters)



Types of Machine Learning

Example: School Class

Reinforcement Learning

-Assign Learning Method to each student **one by one**.

-Task: Find which learning method should be chosen in future

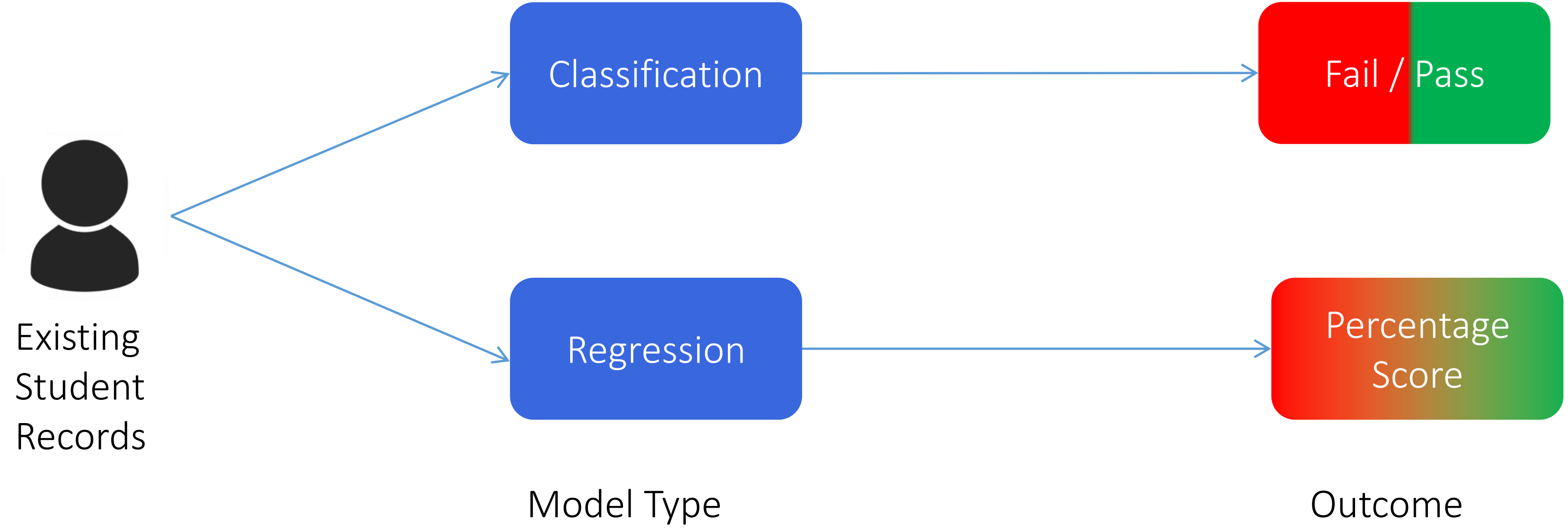
-RL Methods find faster solution than A/B tests.

Name	Age	Learning Method	Class	Grade
Anton	14	A	Sport	2
Bert	15	B	Sport	2
Clare	13	A	Sport	3
Dave	16	B	Math	1
Emilia	15	A	Math	2
…				

de

Types of Machine Learning

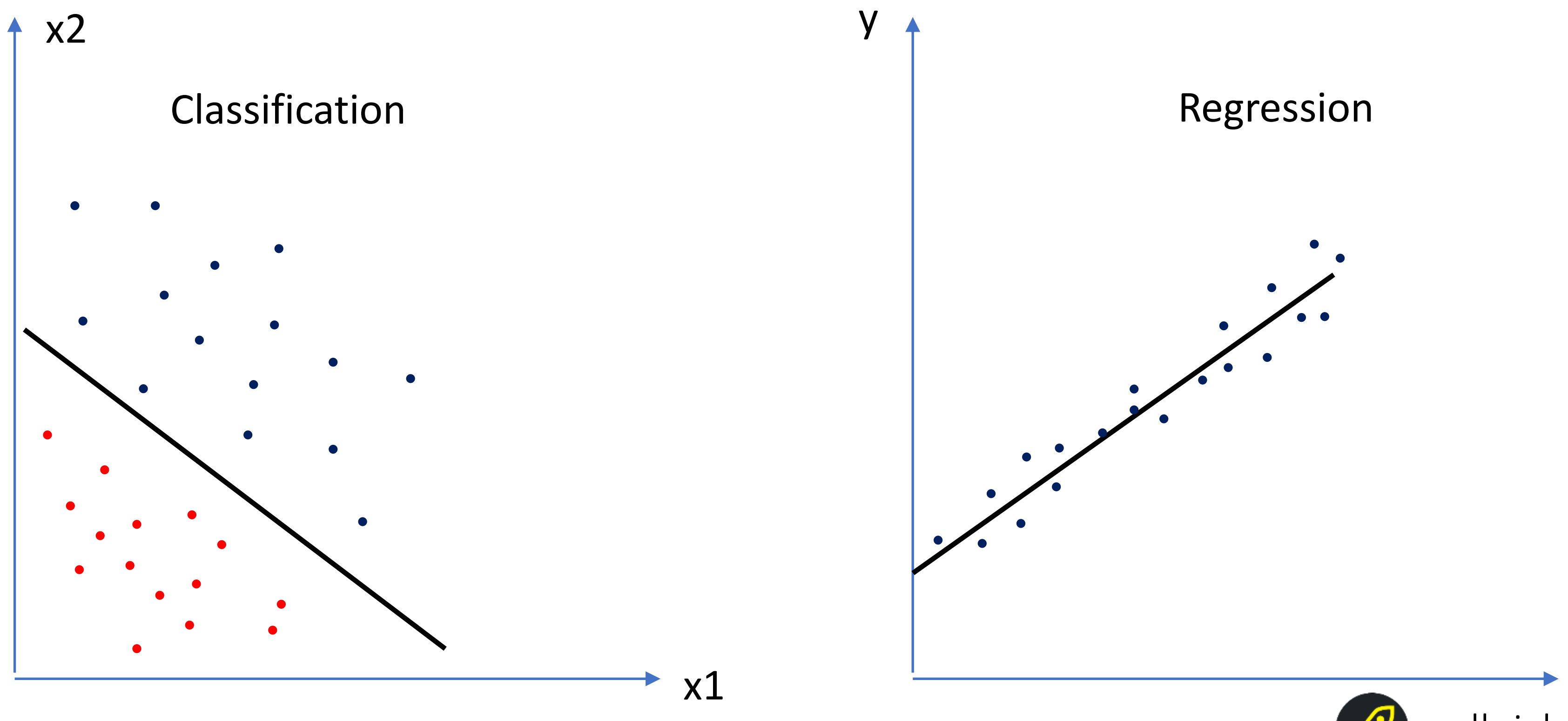
Example: Student Test Prediction



gollnickdata.de

Types of Machine Learning

Example: Classification and Regression Plot



gollnickdata.de

Types of Machine Learning

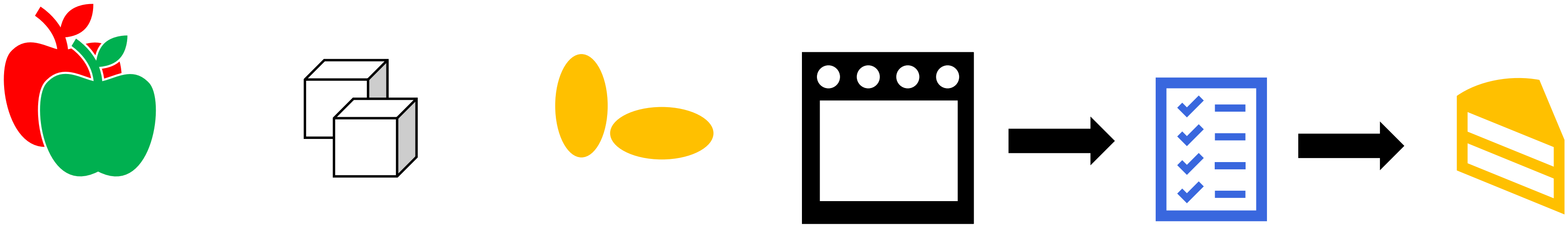
Example: Student Test Prediction

Property	Classification	Regression
Output / Target Variable	Discrete (class labels)	Continuous numbers
Examples	Fail / pass	Percentage scores
What is searched for?	Decision Boundary, Group membership	Best Fit Line
Evaluation Measure	Accuracy	Sum of squared errors (R^2)

Models

Models

High-Level Analogy



Apple Sugar Eggs Baking Oven Apple Pie

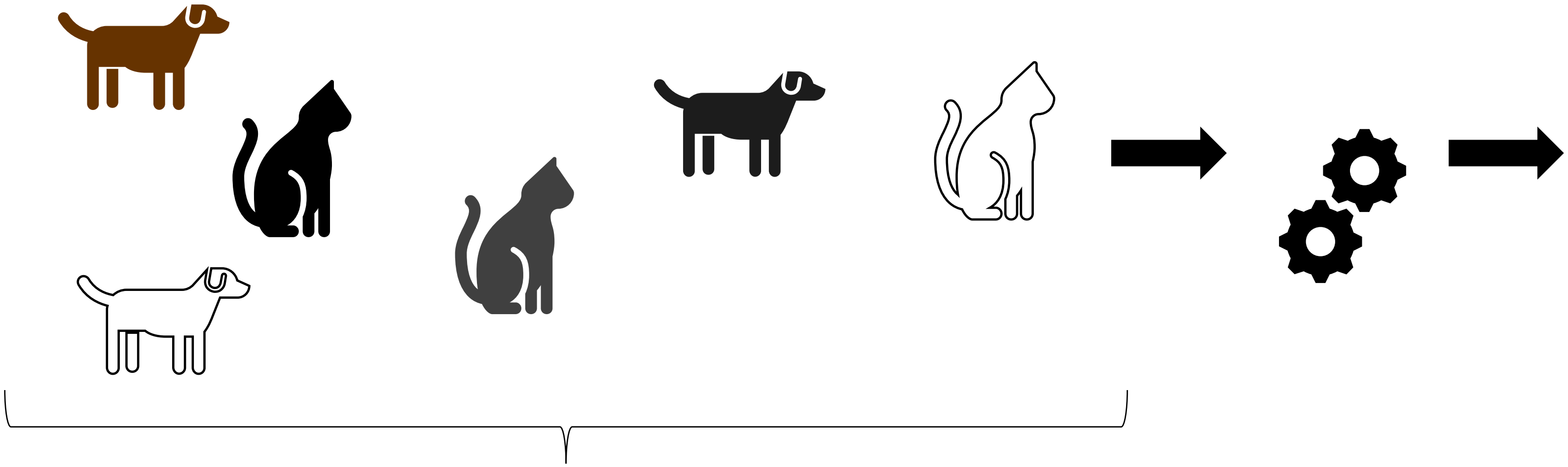
Ingredients + technical equipment
(Requirements)

Recipe

Result

Models

High-Level Analogy



Label:
„Dog“
„Cat“

Independent Variables

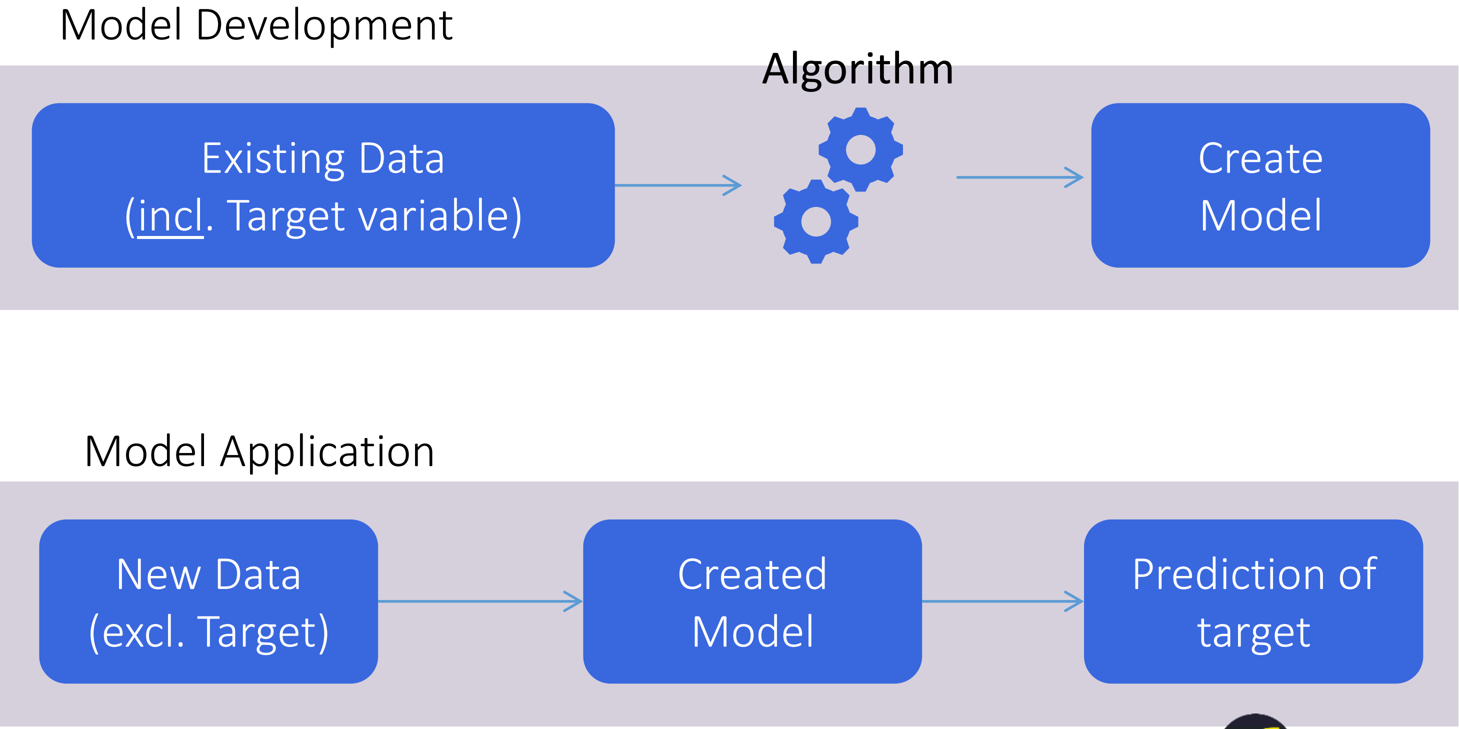
Model

Prediction

gollnickdata.de

Models

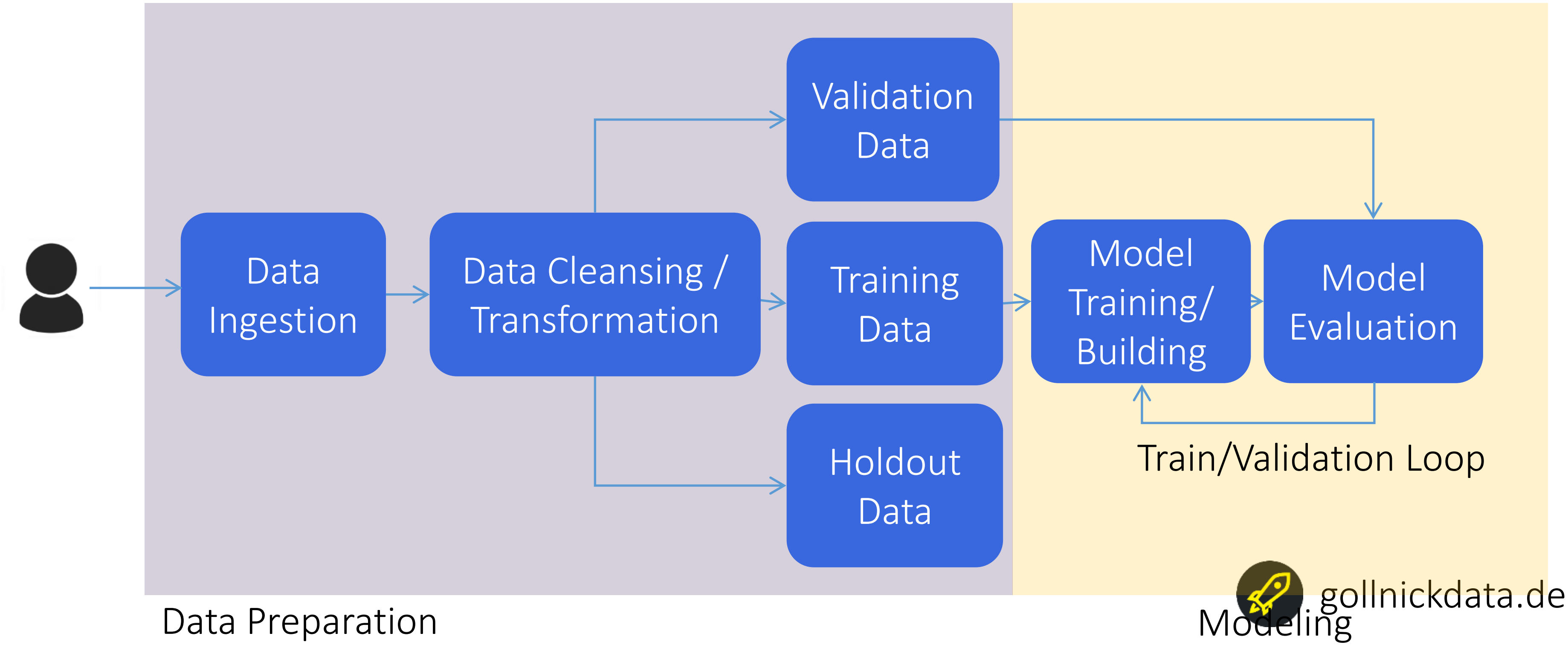
Model Development and -application



gollnickdata.de

Models

Detailed Model Development



gollnickdata.de

Models

Example

- Task: Target variable (dependent variable) should be predicted.
- Predictors (independent variables) are used to create a model based on an existing relationship between independent and dependent variable.
- Model „learns“ relationship
- Learned model can then be applied to new data.

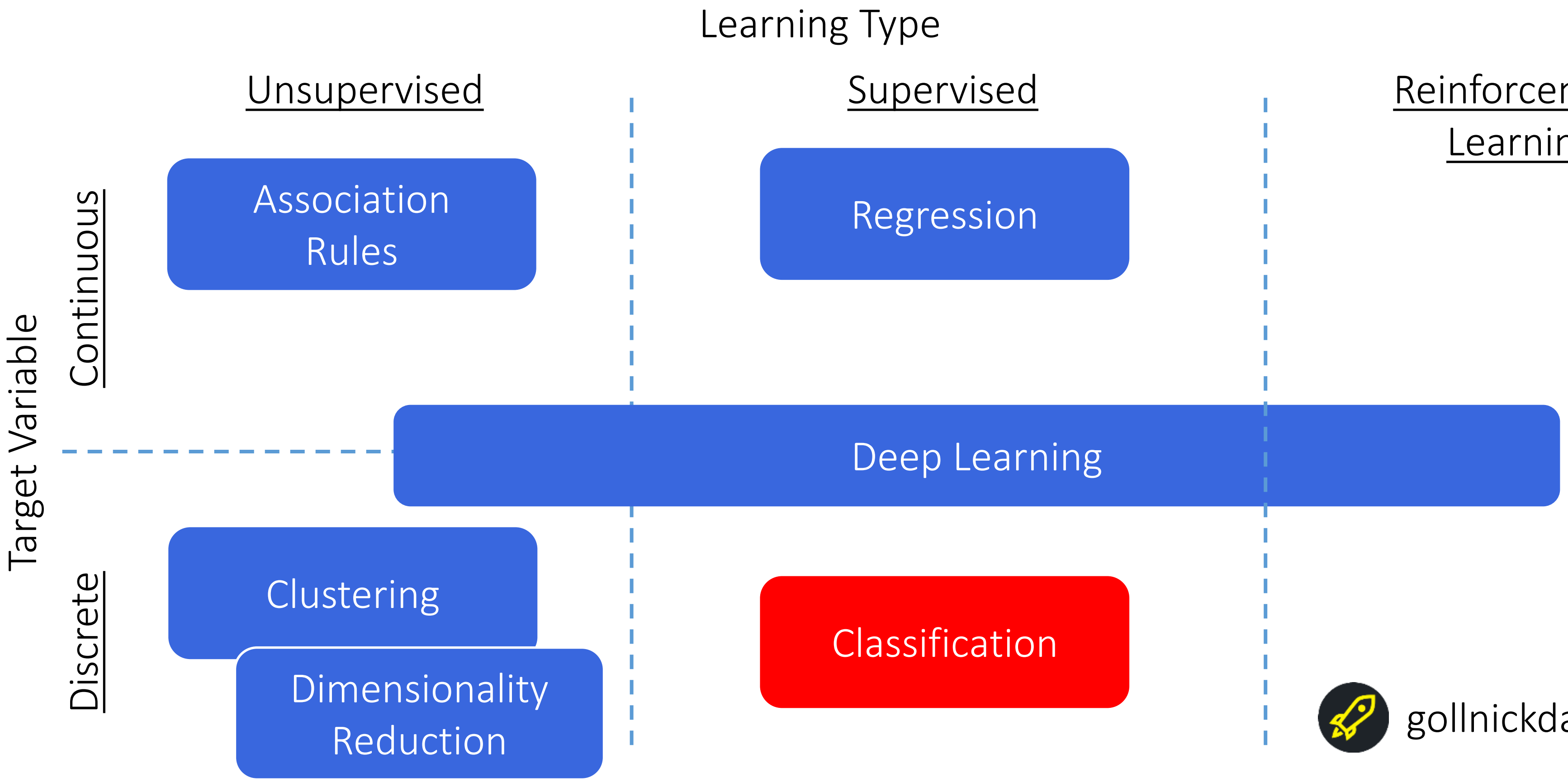
X1	X2	X3	X4	Y
Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
4.8	3.0	1.4	0.3	setosa
[illegible]	[illegible]	[illegible]	0.2	setosa
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	0.2	[illegible]
5.0	3.3	1.4	0.2	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
5.5	2.3	4.0	1.3	versicolor
6.5	2.8	4.6	1.5	versicolor
5.7	2.8	4.5	1.3	versicolor
6.3	3.3	4.7	1.6	versicolor
4.9	2.4	3.3	1.0	versicolor
6.6	2.9	4.6	1.3	versicolor

Independent Variables

Target

Our Focus in Today‘s Class

Classification



gollnickdata.de

Analysis Steps

Analysis Steps

Sample Steps

Data Preparation

- Data Import
- Missing Data Analysis
- Missing Data Treatment
- Categorical Data Treatm.
- Separate Dependent / Independent Feature
- Data Splitting
- Data Scaling

Modeling

- Model Training
- Create Predictions
- Model Evaluation

gollnickdata.de

Logistic Regression

Logistic Regression

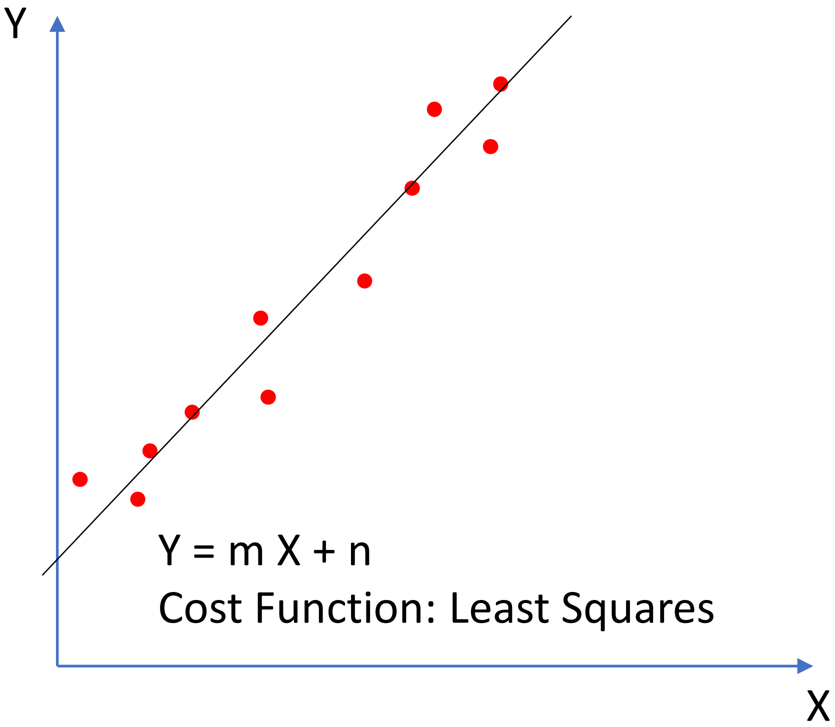
Introduction

- Suitable for classification tasks (don‘t get confused by „regression“)
- Only works for binary classifier
- Independent variables can be continous or discrete
- Related to classical regression

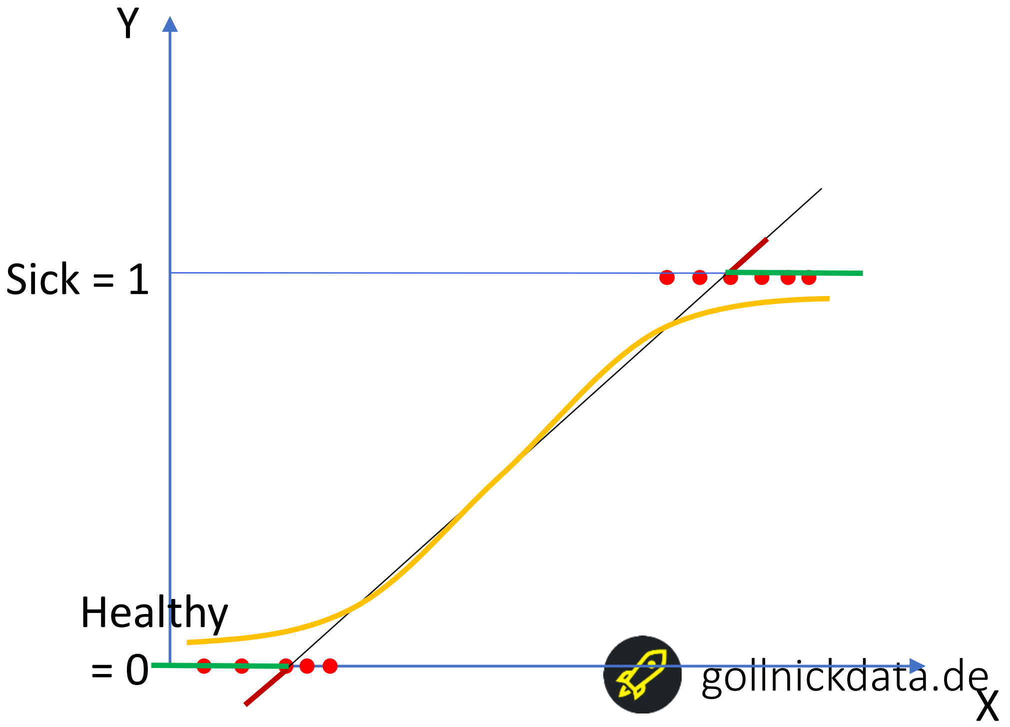
Logistic Regression

From Linear Regression to Logistic Regression

Linear Regression



Logistic Regression



gollnickdata.de

Logistic Regression

From Linear Regression to Logistic Regression

Logistic Regression

$$Y = mX + n$$

Transform Target Variable with Sigmoid Function

$$p = \frac{1}{1 + e^{-Y}}$$

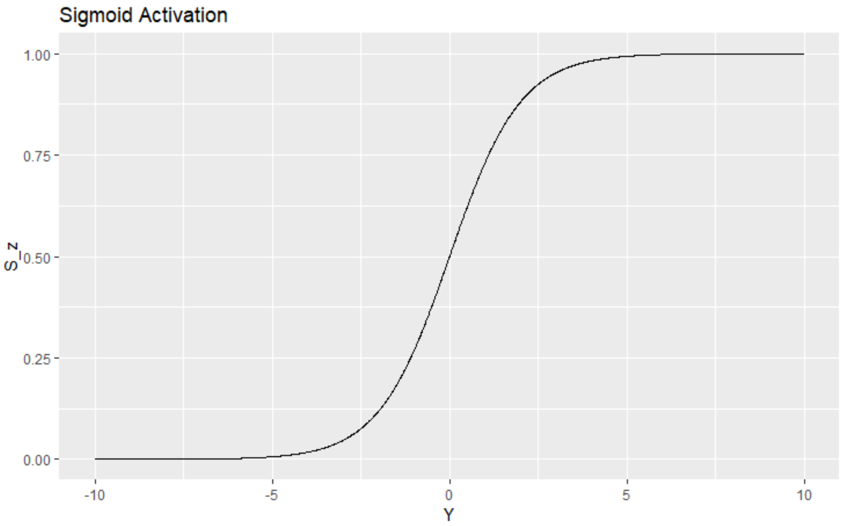
Rewrite Formula:

$$Y = ln\left(\frac{p}{1-p}\right)$$

Logit-Transformation of Target Variable:

$$Y = ln\left(\frac{p}{1-p}\right)$$

$$ln\left(\frac{p}{1-p}\right) = mX + n$$



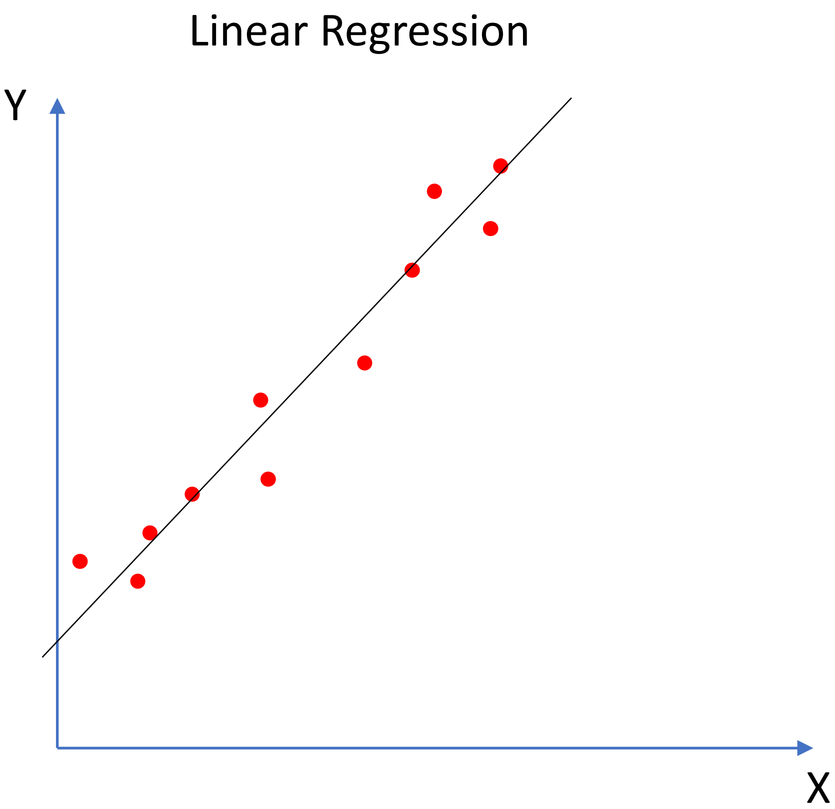
Sigmoid function maps results to 0 to 1 range.

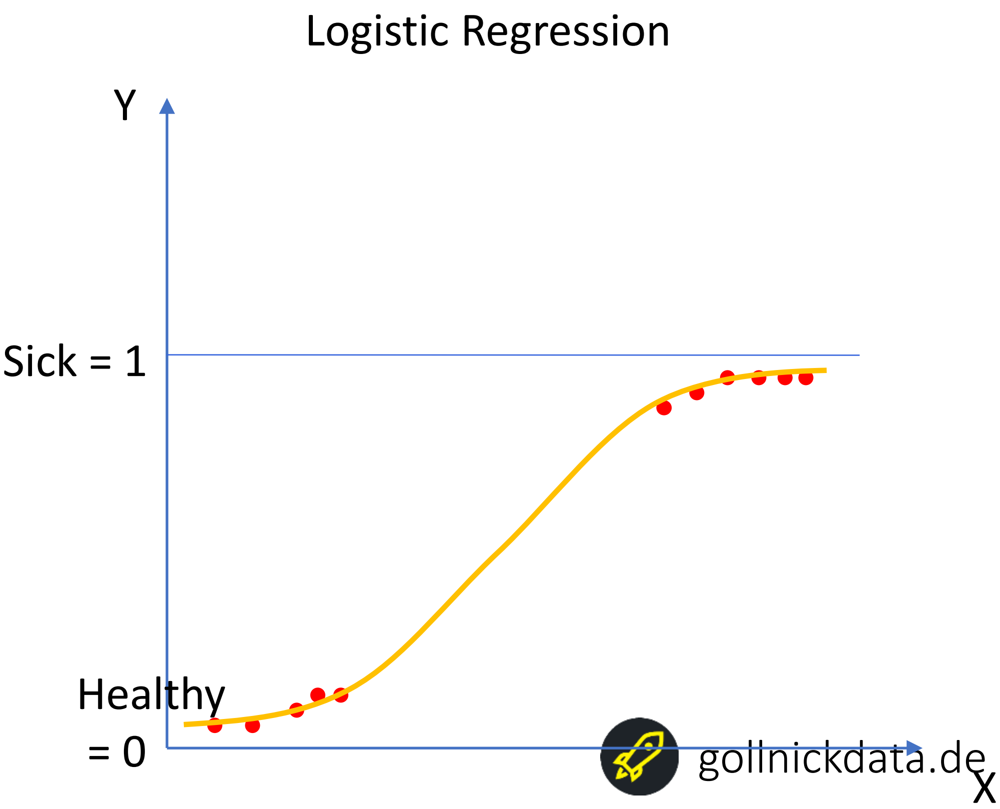
$$S(x) = \frac{1}{1 + e^{-x}}$$

gollnickdata.de

Logistic Regression

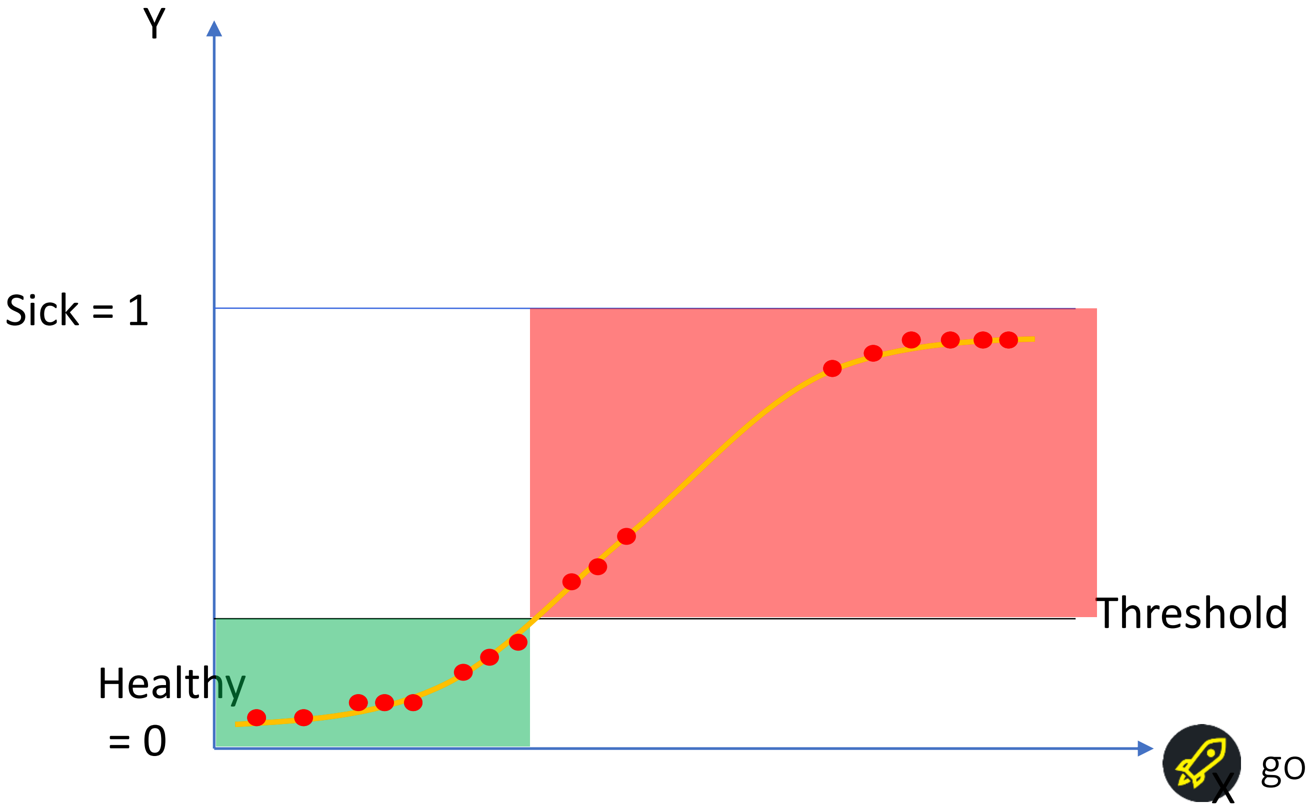
From Linear Regression to Logistic Regression





Logistic Regression

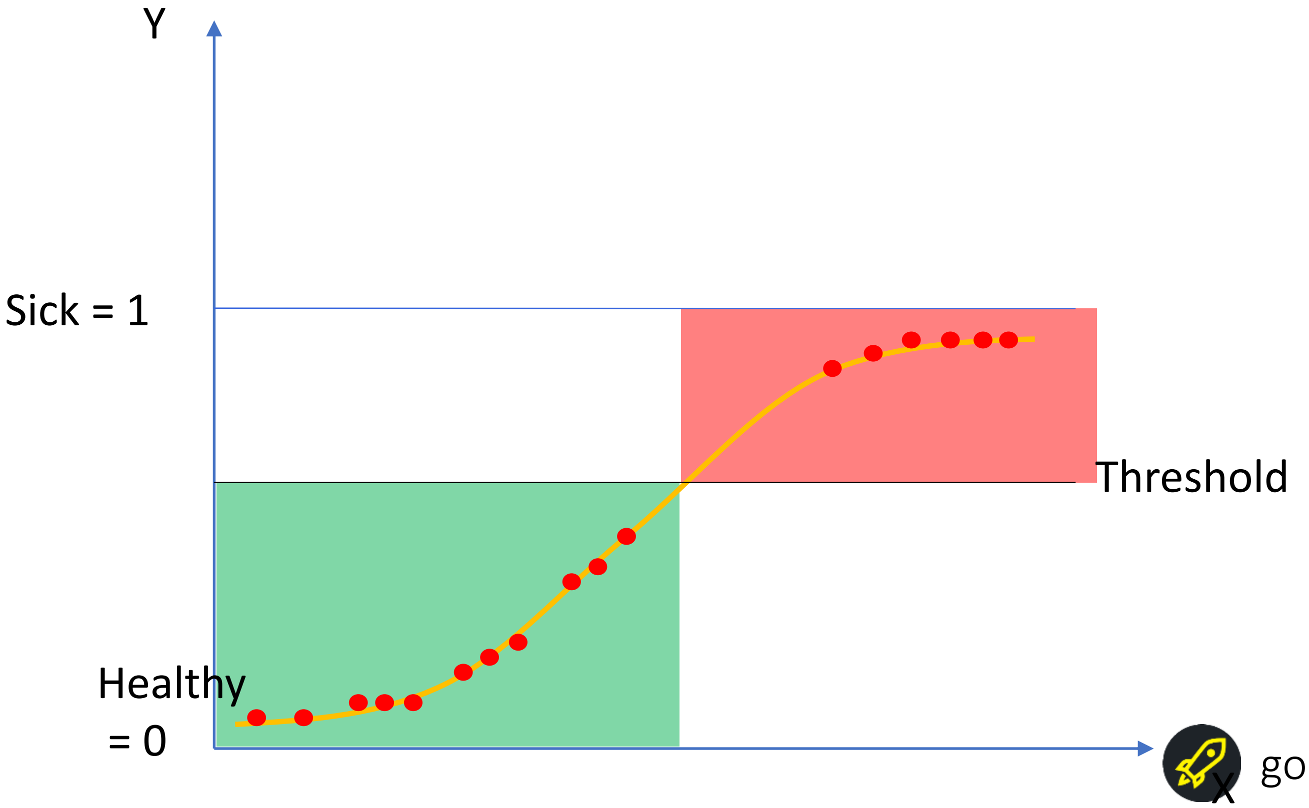
From Probabilities to Classes



gollnickdata.de

Logistic Regression

From Probabilities to Classes



Classification

Confusion Matrix

Example

		Event Occured?	
		Yes	No
Event Predicted ?	Yes	True Pos (Hit)	False Pos (Type I Error) (False Alarm)
	No	False Neg (Type II Error) (Miss)	True Neg (Correct Rejection)

True Outcome

Errors

gollnickdata.de

Confusion Matrix

Example: Tsunami

		Event Occured?	
		Yes	No
Event Predicted ?	Yes	Tsunami was observed, when it actually happened	A tsunami was predicted, but there was none
	No	There was a tsunami, but it was not predicted.	No tsunami occurred and nothing was Predicted

True Outcome

Less-critical Error

Critical Error

Confusion Matrix

Performance Measures: Accuracy

Numerator

Effect Observed? \ Effect Exists?	Yes	No
Yes	True Pos	False Pos
No	False Neg	True Neg

Denominator

Effect Observed? \ Effect Exists?	Yes	No
Yes	True Pos	False Pos
No	False Neg	True Neg

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Usually compared to baseline result or to compare models

gollnickdata.de

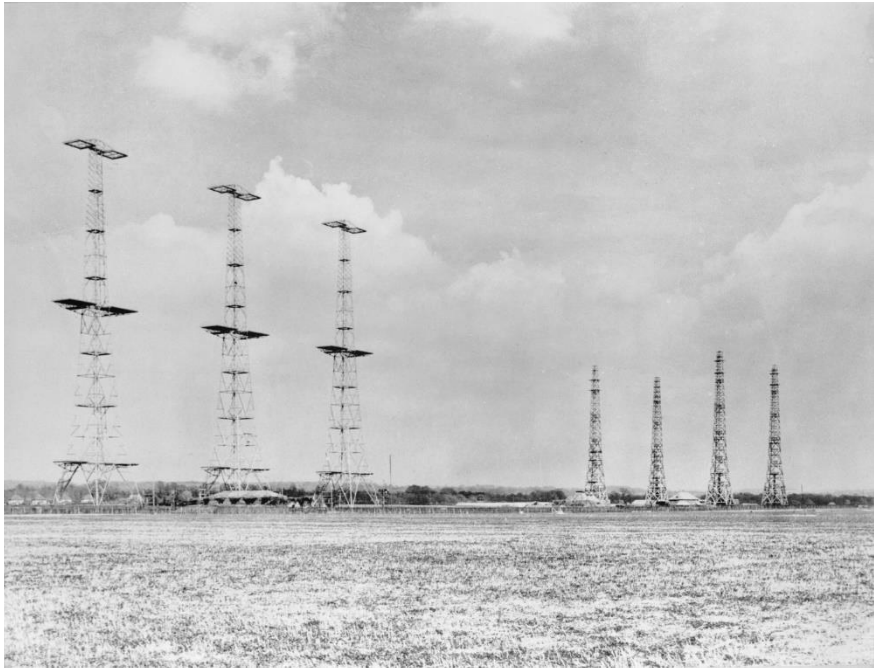
ROC Curve

ROC Curve - 101

Introduction

Receiver Operating Characteristics (**ROC**) Curve

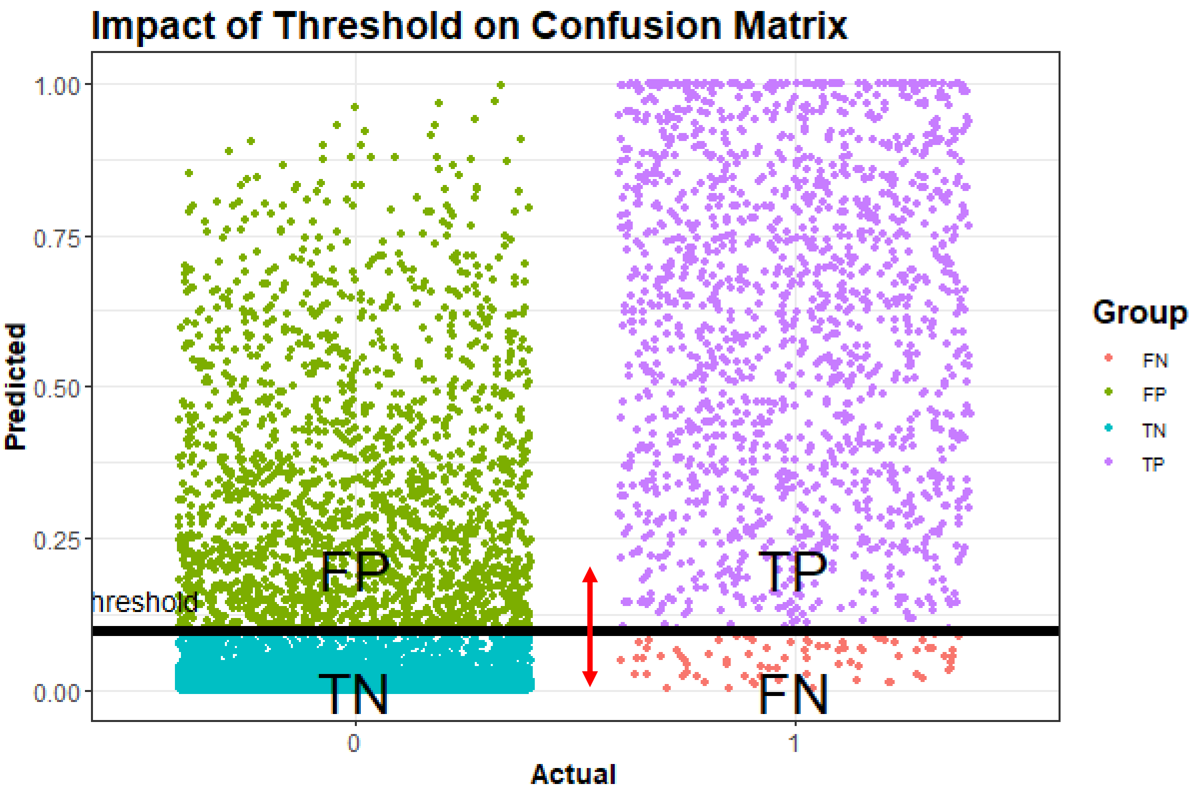
- First developed and used during WWII for detecting enemy objects in battlefields
- Later used in psychology, medicine, forecasting of natural hazards, …
- … and finally **model performance assessment**



Source: https://commons.wikimedia.org/wiki/File:
Chain_Home_radar_installation_at_Poling,_Sussex,_1945._CH15173.jpg

ROC Curve - 101

From Confusion Matrix to ROC Curve

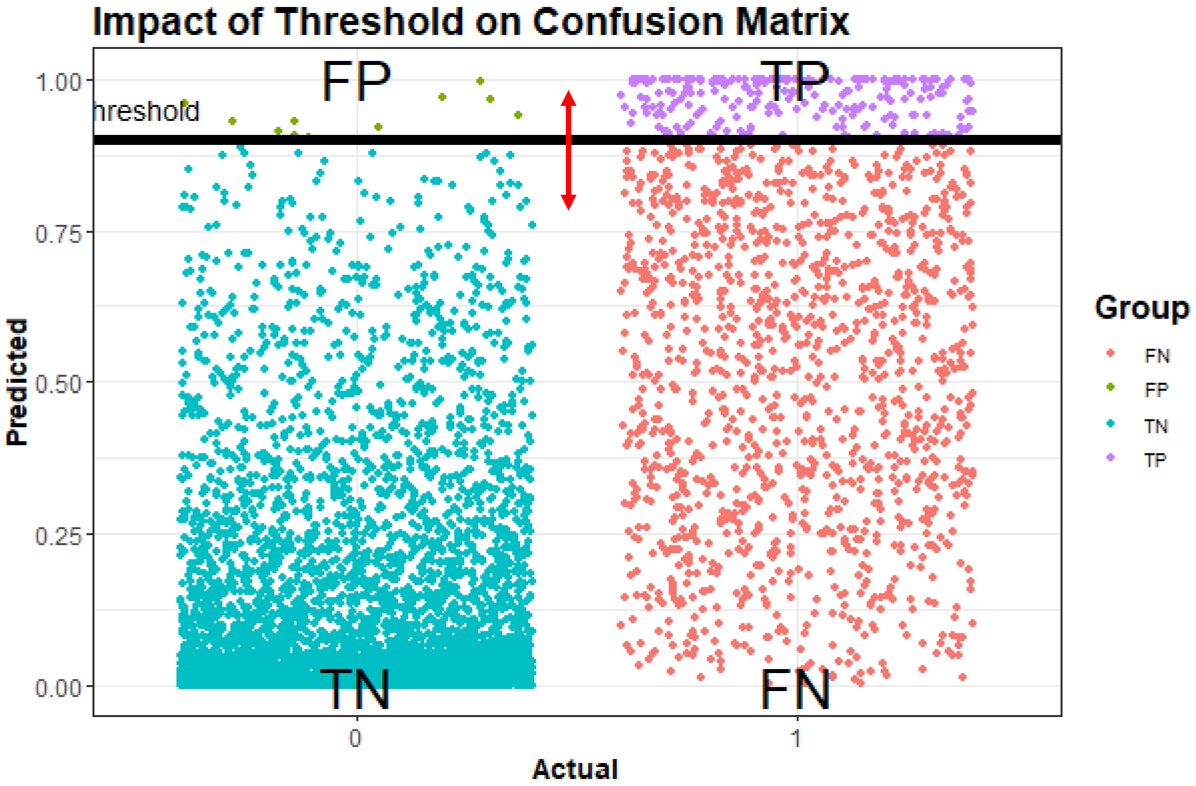


Actuals	PredNeg	PredPos
ActNeg	3117	1842
ActPos	84	1469

gollnickdata.de

ROC Curve - 101

From Confusion Matrix to ROC Curve



Actuals	PredNeg	PredPos
ActNeg	4948	11
ActPos	1305	248

gollnickdata.de

ROC Curve - 101

From Confusion Matrix to ROC Curve

		Predicted Class	
		Yes	No
Actual Class	Yes	True Pos (Hit)	False Neg (Type I Error)
	No	False Pos (Type II Error)	True Neg (Correct Rejection)

$$TPR = \frac{TP}{TP + FN}$$

→ Y Axis on ROC Curve

gollnickdata.de

ROC Curve - 101

From Confusion Matrix to ROC Curve

		Predicted Class	
		Yes	No
Actual Class	Yes	True Pos (Hit)	False Neg (Type I Error)
	No	False Pos (Type II Error)	True Neg (Correct Rejection)

$$FPR = \frac{FP}{FP + TN}$$

→ X Axis on ROC Curve

ROC Curve - 101

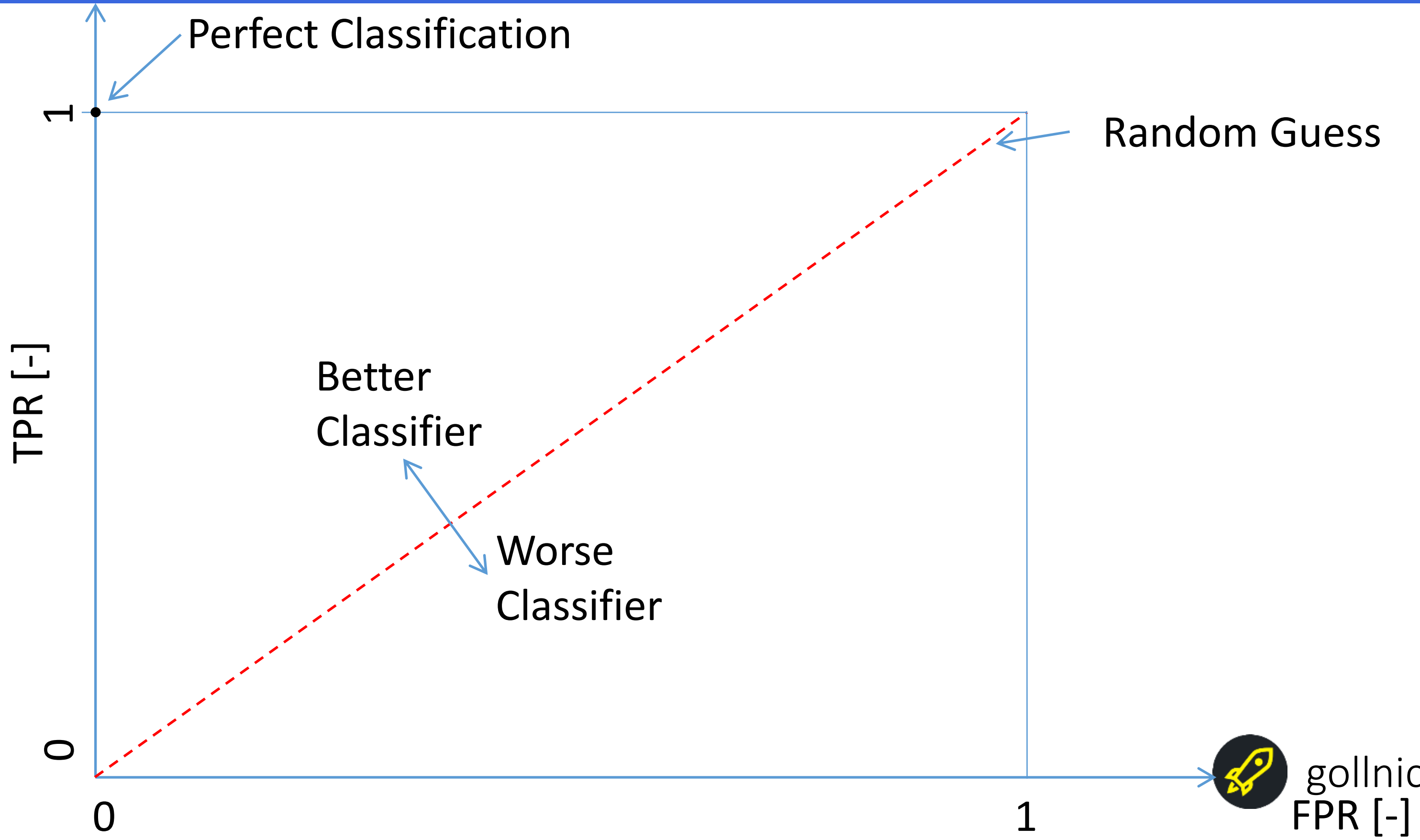
From Confusion Matrix to ROC Curve

Example

Threshold	TN	FP	FN	TP	FPR	TPR
0.01	1318	3641	3	1550	0.73	1
0.02	1776	3183	10	1543	0.64	0.99
…						
0.98	4958	1	1431	122	0	0.08
0.99	4958	1	1448	105	0	0.07

ROC Curve - 101

ROC Curve

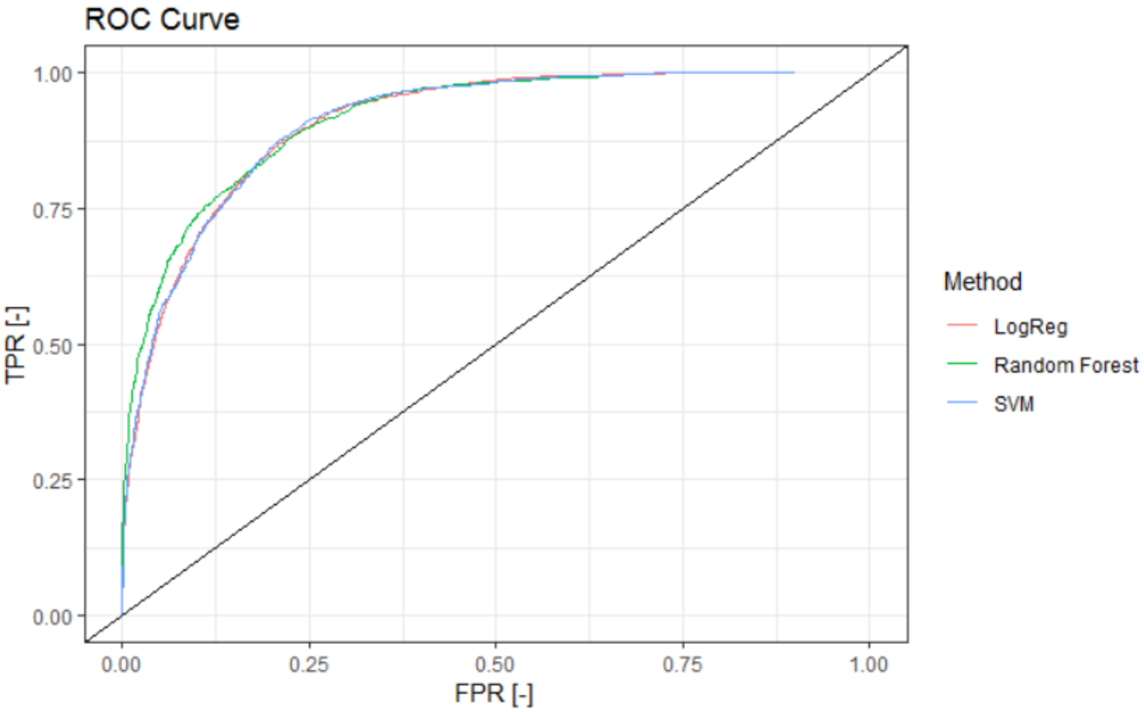


gollnickdata.de

ROC Curve - 101

Purpose

- Different methods can be compared



Source: own graph

Classification Types

Classification Types

Binary

- X: images of houses and / or trees
- y:
  - one label per image
  - two mutually exclusive classes
- Example Label encoding: 0 = tree, 1 = house

y = 0

y = 1

y = 0/1 ?

Classification Types

Multi-Class

- X: images of houses and / or trees
- y:
  - one label per image
  - more than two mutually exclusive

Example:

- Label encoding: 0 = tree, 1 = house, 2 = road

y = 0

y = 1

y = 0

y = 2

Classification Types

Multi-Label

- X: images of houses and / or trees
- y:
  - each image can have more than one class
  - more than two mutually exclusive classes

Example:

- Label encoding: 0 = tree, 1 = house, 2 = road

y = [0]

y = [1]

y = [0, 1]

y = [0, 2]

Classification Example

Classification Example

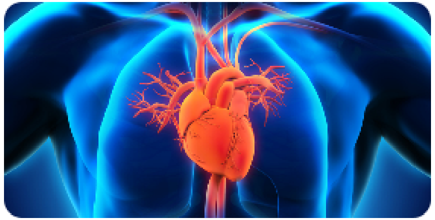
Dataset

FEDESORIANO · UPDATED 2 YEARS AGO

2325 | New Notebook | Download (9 kB)

Heart Failure Prediction Dataset

11 clinical features for predicting heart disease events.



Source: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction

Main Ingredients

Deep Learning Introduction

Section Overview

- Confusing process of interactions

High Level Understanding	Performance	Perceptron	Layers
Activation Functions	Loss Functions	Optimizer	

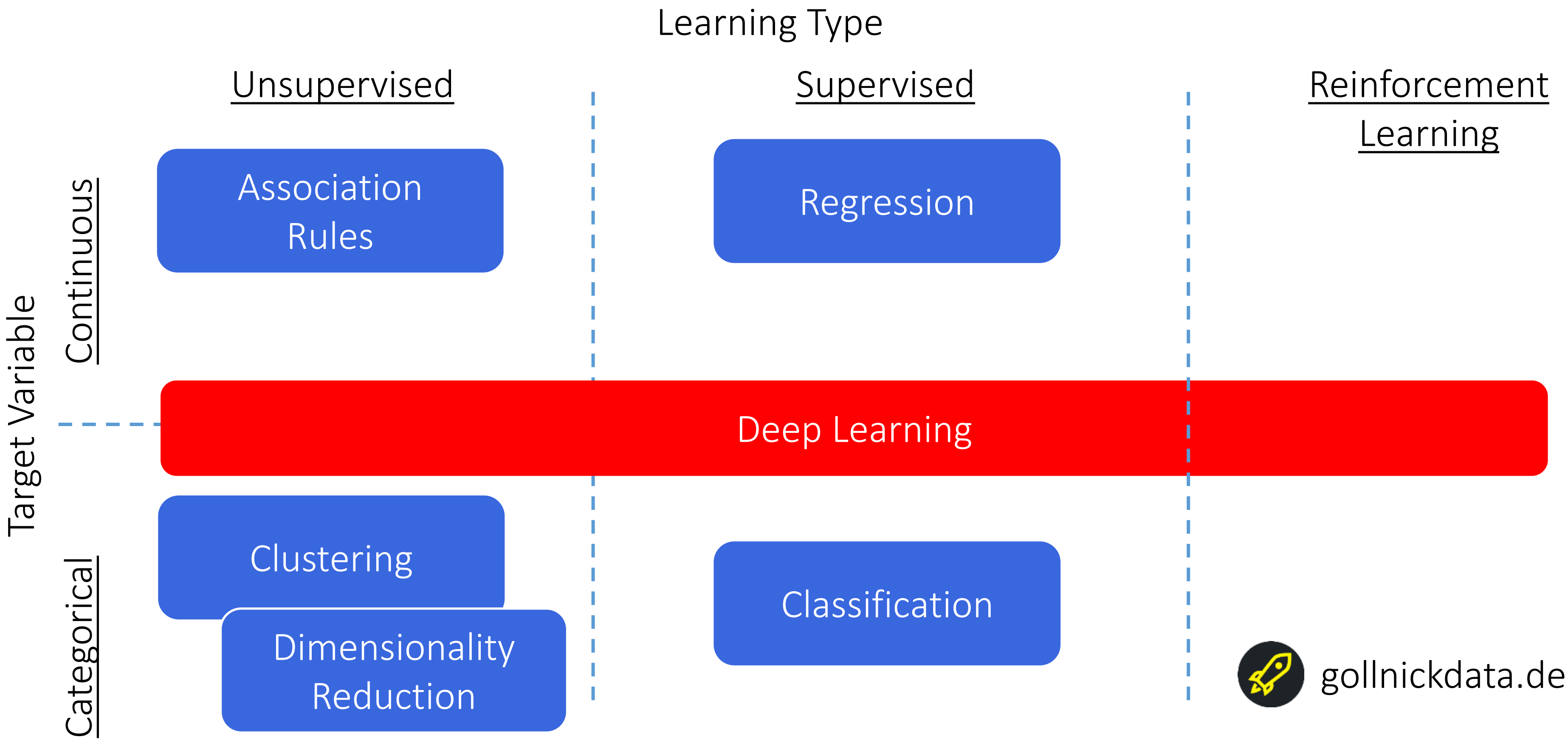
Deep Learning General Overview

Deep Learning General Overview

- type of machine learning
- Covers all learning types (supervised/unsupervised/reinforcement learning)
- Different architectures for different purposes
  - Fully-connected neural networks
  - Convolutional neural networks
  - Recurrent neural networks
  - …
- Inspired by the structure and functioning of human brain
- Use multiple layers for feature extraction
- Each layer uses data from previous layer
- Learn different levels of abstraction

Deep Learning General Overview

All Chapters



gollnickdata.de

Deep Learning General Overview

Computer Vision Tasks

Classification

- Algorithm recognizes a dog in the image

Classification and Object Detection

- Algorithm recognizes a dog in the image
- Algorithm detects rectangular location of dog

Object Detection

- recognizes a dog and cat in the image
- Algorithm detects rectangular location of dog/cat

Semantic Segmentation

- recognizes pixelwise location of dog/cat

gollnickdata.de

Deep Learning General Overview

Recurrent Neural Networks

Time Series Prediction
Forecasting

Text Generation
Machine Translation

gollnickdata.de

Deep Learning General Overview

GANs and Style Transfer

Style Transfer

Generative Adversarial Networks

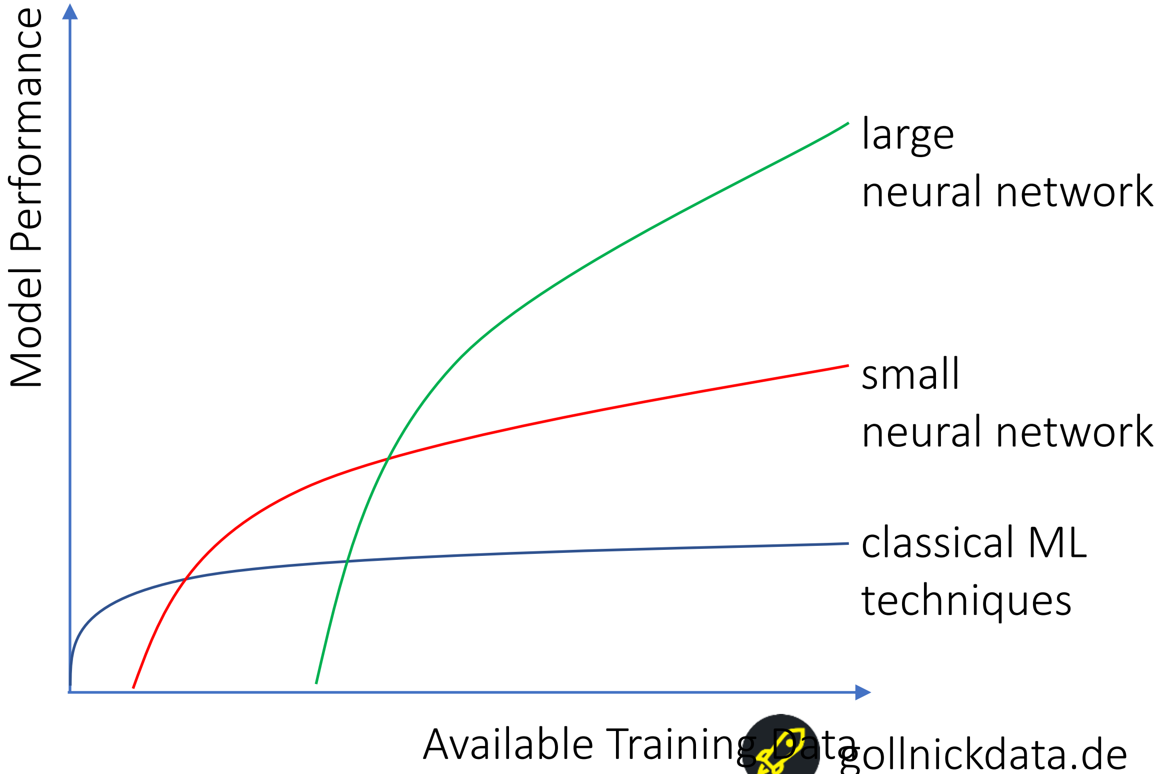
Sources:
[1] https://ndres.me/post/machine-learning-with-gifs-style-transfer/
[1] Karras „PROGRESSIVE GROWING OF GANS FOR IMPROVED QUALITY, STABILITY, AND VARIATION"

Deep Learning: Performance

Deep Learning: Performance

Deep Learning Performance

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



Deep Learning: Performance

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

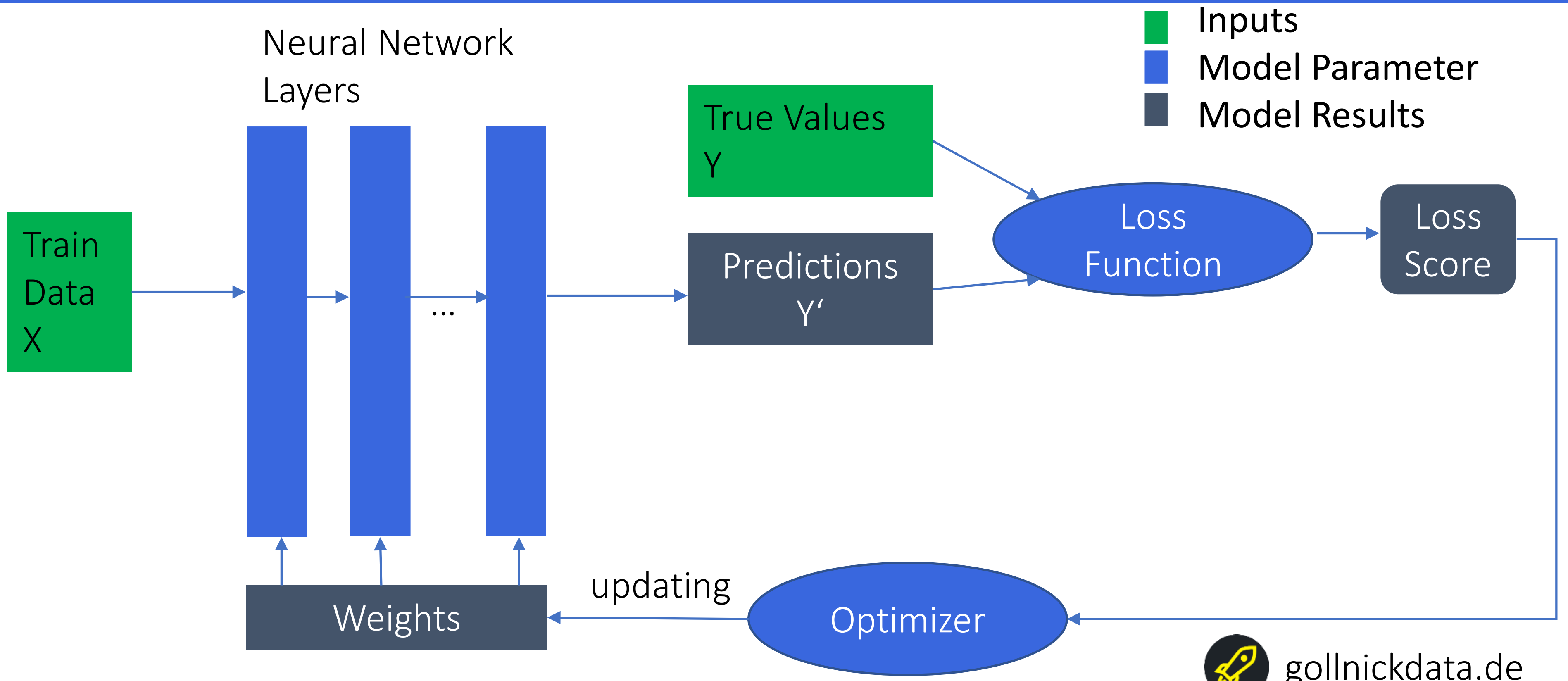
Better
Algorithms

Open
Source

Modeling Overview

Deep Learning Overview

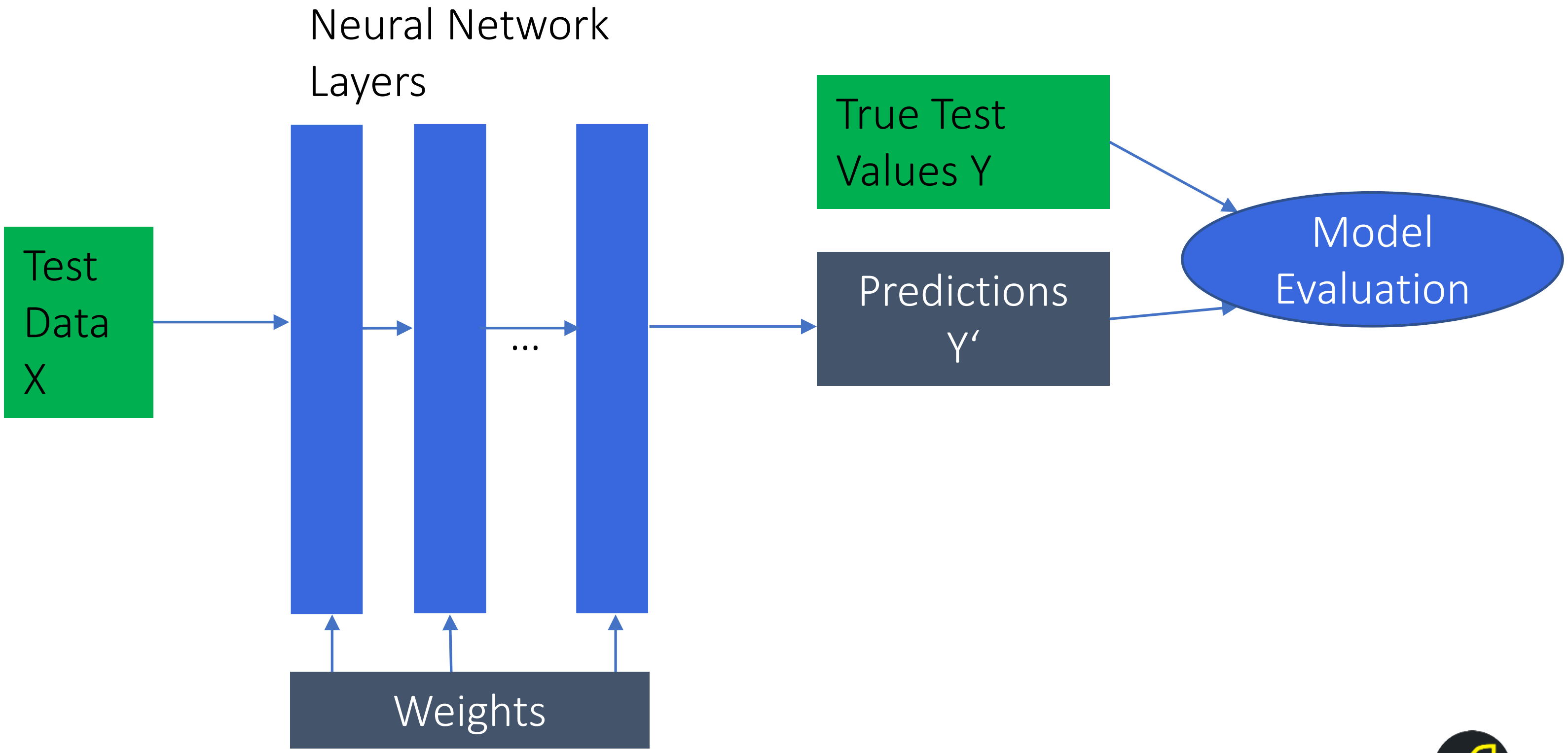
Model Creation Workflow



gollnickdata.de

Deep Learning Overview

Creating Predictions

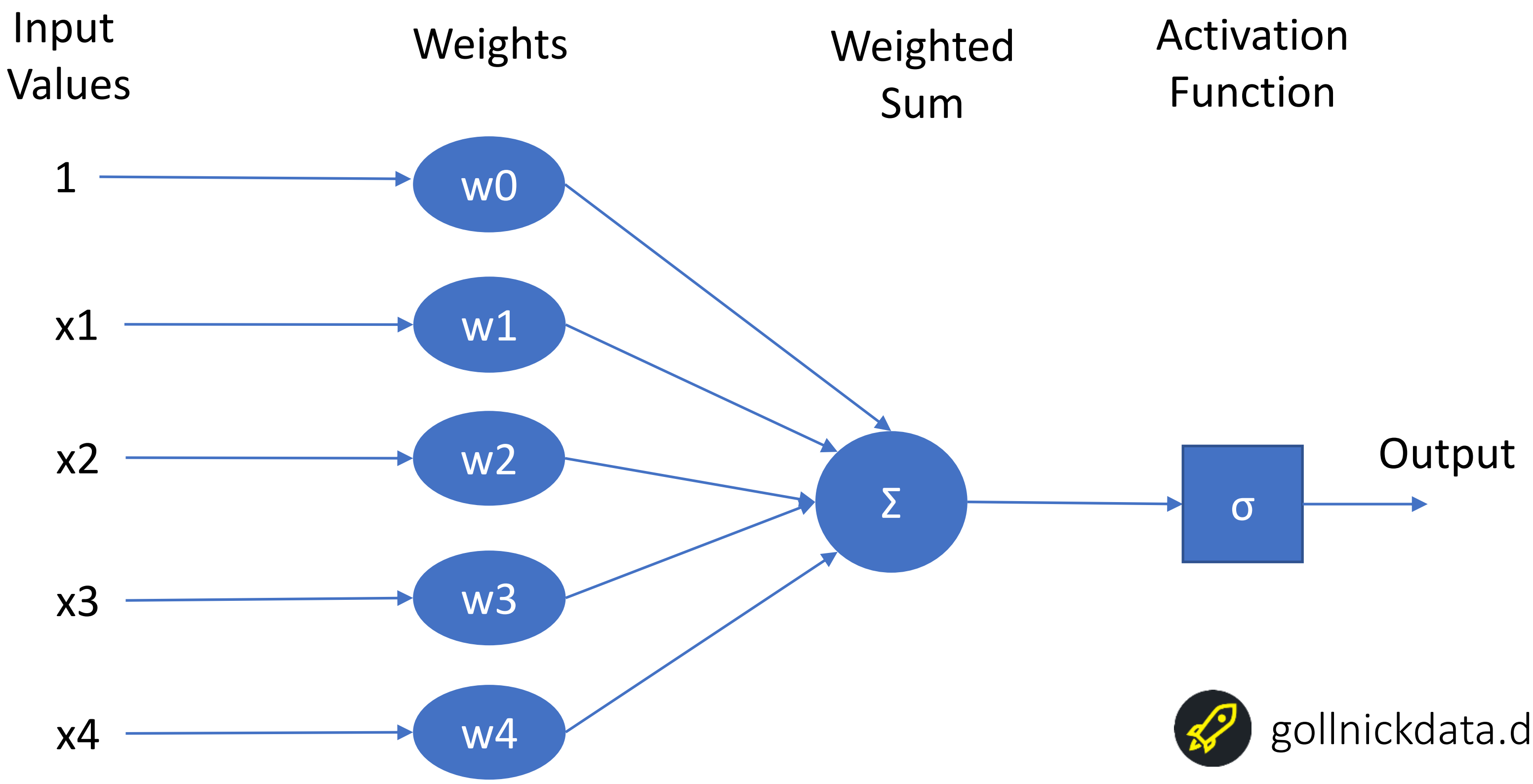


gollnickdata.de

From Perceptron to Neural Network

From Perceptron to Neural Network

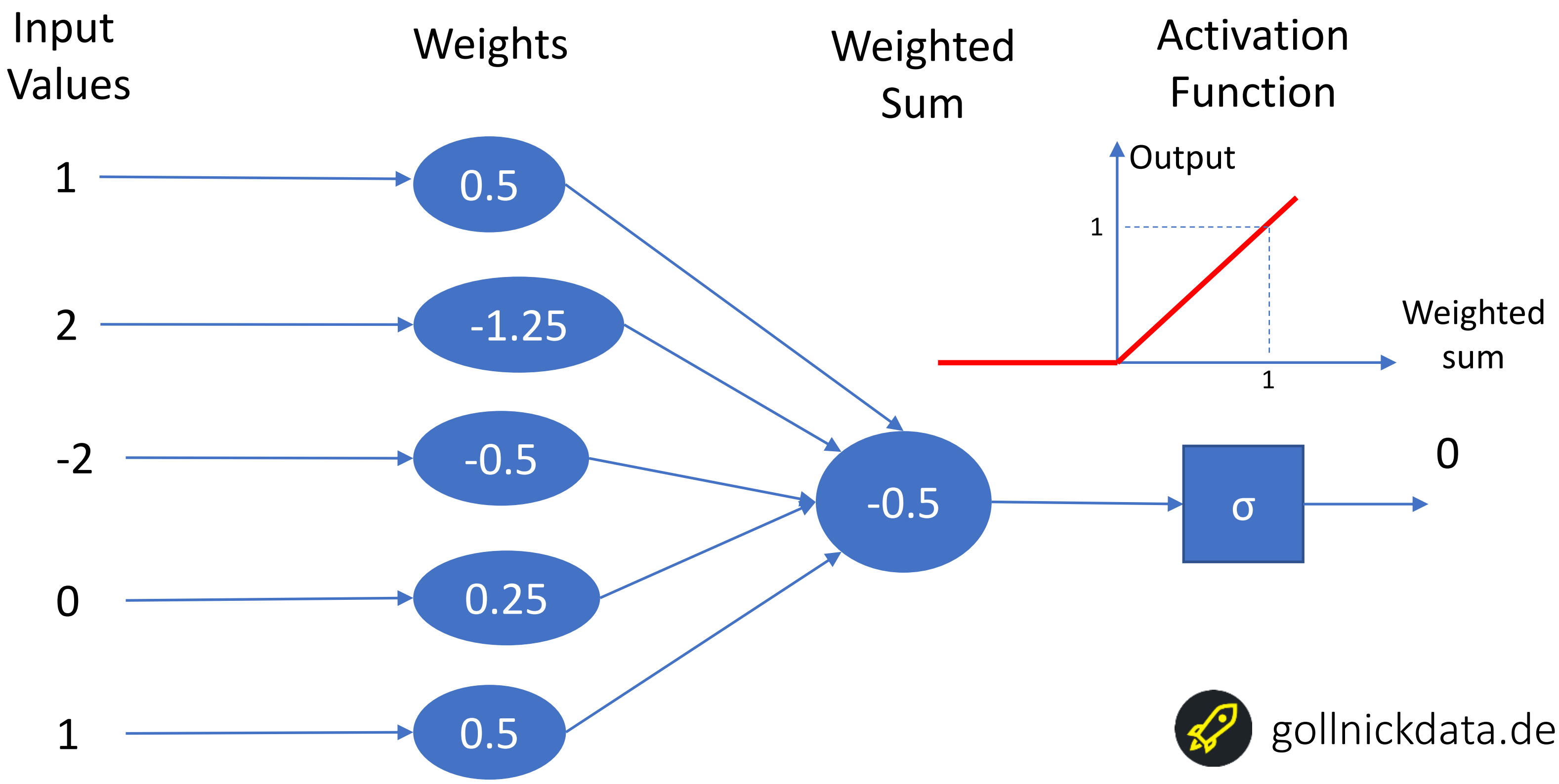
Perceptron



gollnickdata.de

From Perceptron to Neural Network

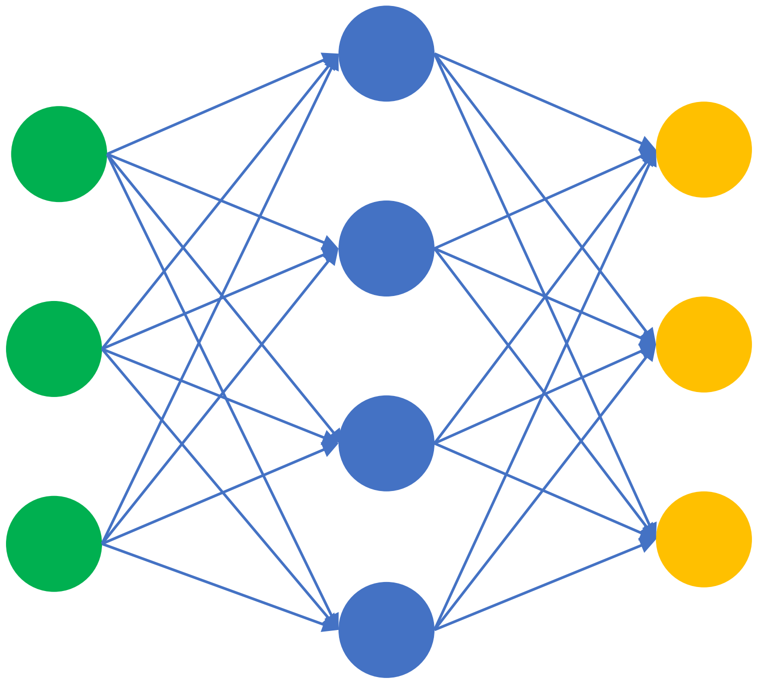
Perceptron: Example



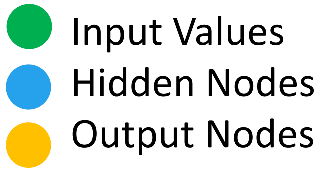
gollnickdata.de

From Perceptron to Neural Network

Simple and Deep Neural Network



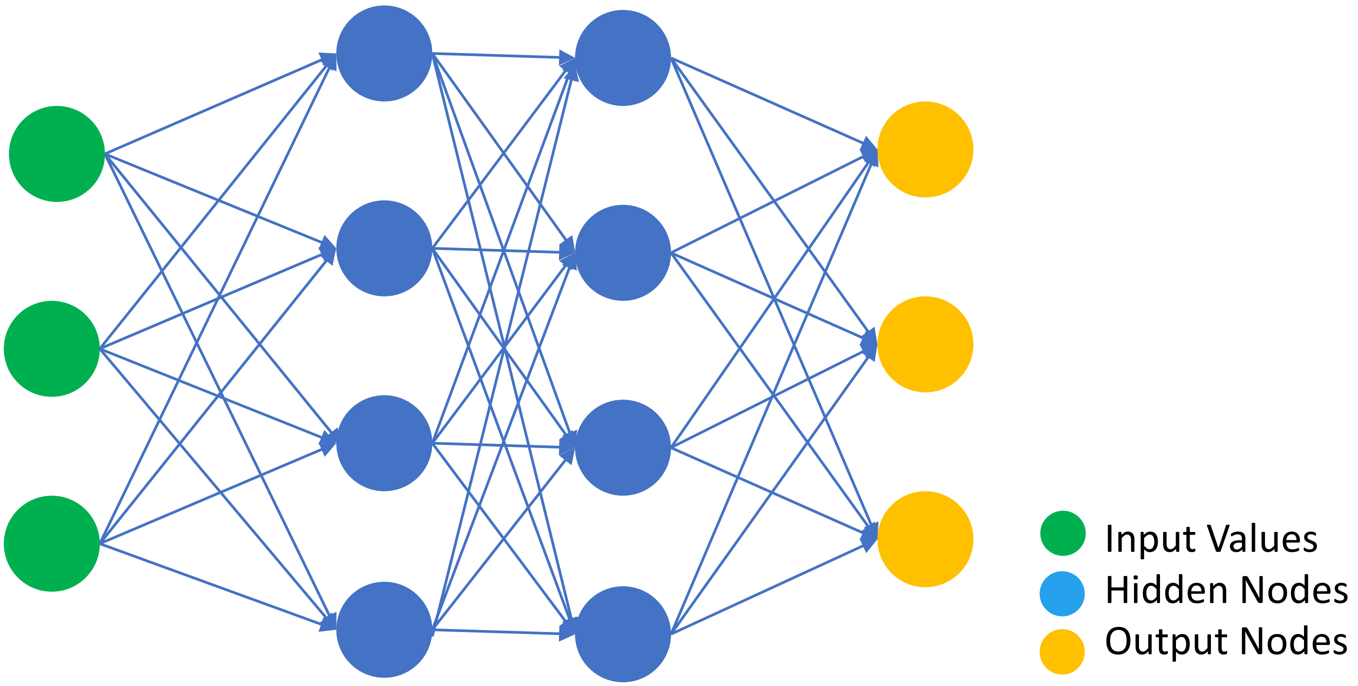
Simple Neural Network



gollnickdata.de

From Perceptron to Neural Network

Simple and Deep Neural Network



Deep Neural Network
Multi-Layer Perceptron

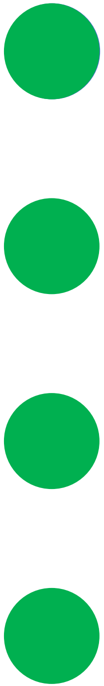
Deep Learning: Layer Types

Deep Learning: Layer Types

Layer Types

Input Layer

- Corresponds to independent variables
- Taken as batches
- Binned data
- Categorized data

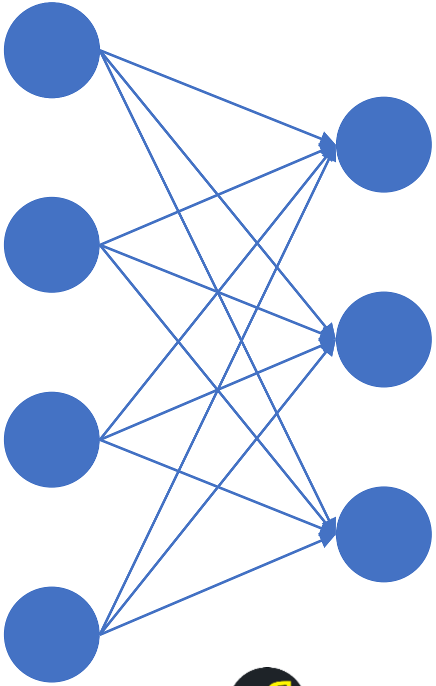


Deep Learning: Layer Types

Layer Types

Dense Layer

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



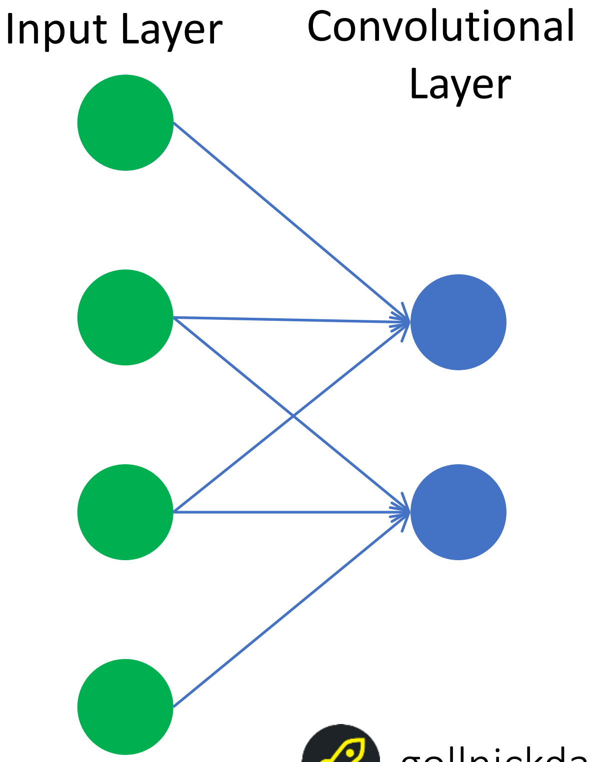
gollnickdata.de

Deep Learning: Layer Types

Layer Types

1D convolutional layer

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



gollnickdata.de

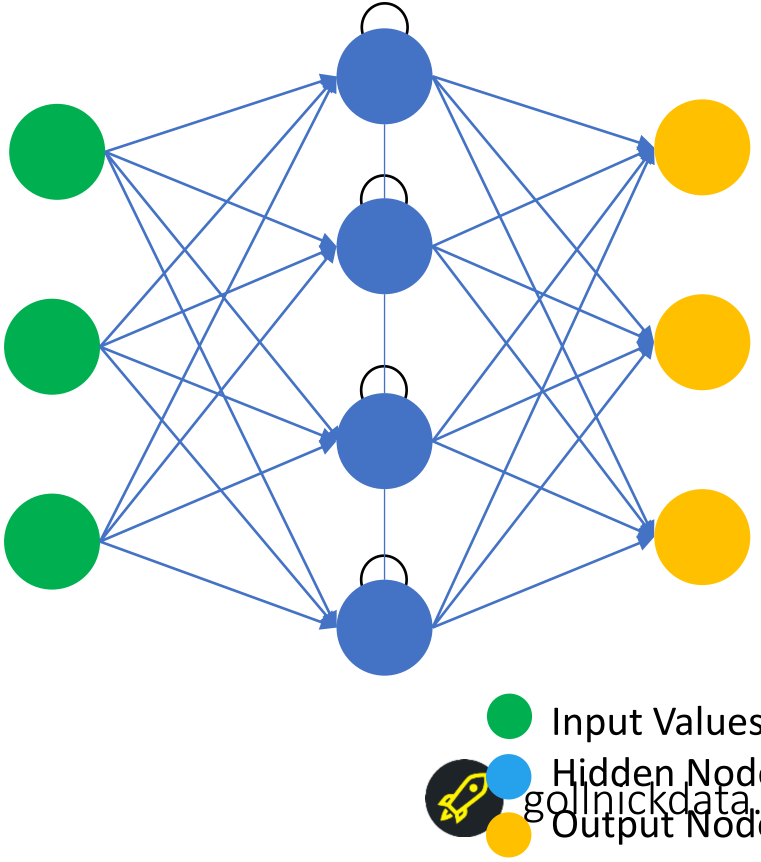
Deep Learning: Layer Types

Other Layer Types

Other Layer Types

- Recurrent Neural Networks
  - use recurrent cells
  - Receive their own output with a delay
  - applied when context plays a role

- Long short-term memory (LSTM)
  - use „memory cell“
  - Used for temporal sequences



Deep Learning Details

Layer Types

Output Layer

Problem Type	Nodes	Output Layer Activation
Regression	1	Linear
Multi-Target Regression	N (nr. of targets)	Linear
Binary Classification	1	Sigmoid
Multi-Label Classification	N (nr. of labels)	softmax

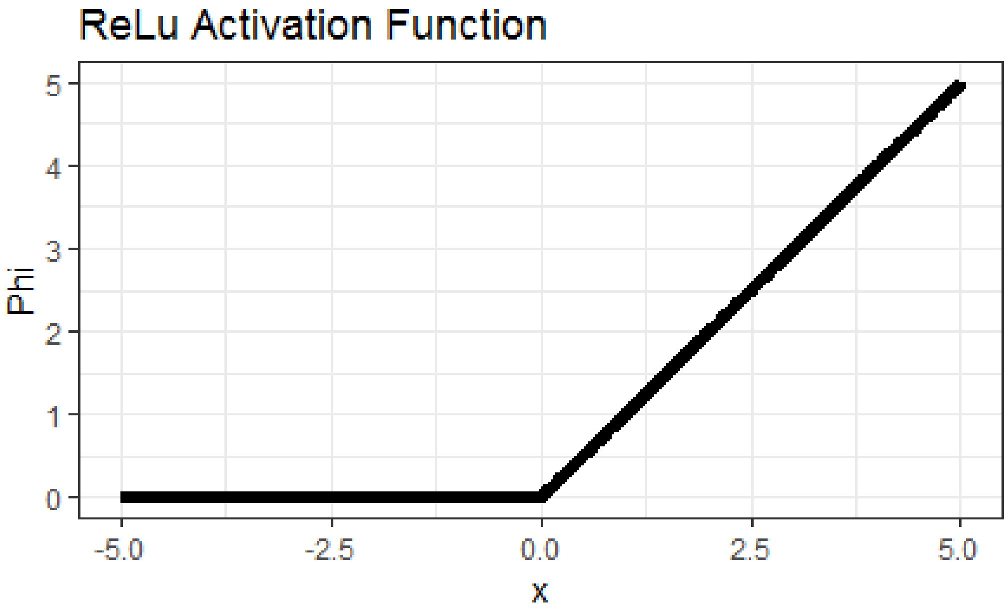
Deep Learning: Activation Functions

Deep Learning: Activation Functions

There are different activation functions.

Rectified Linear Unit (ReLu)

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

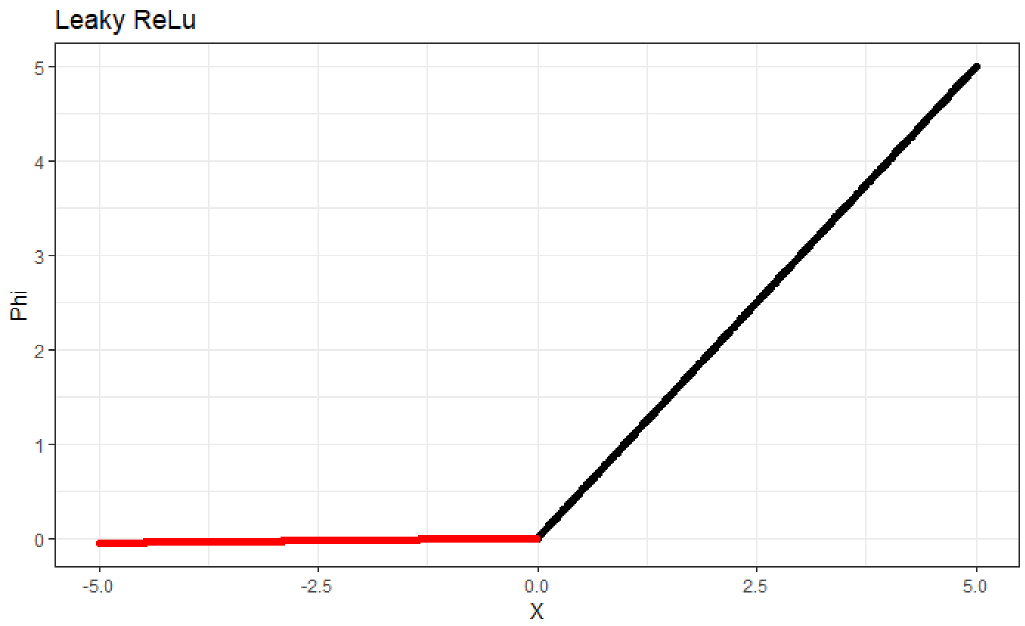


Deep Learning: Activation Functions

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

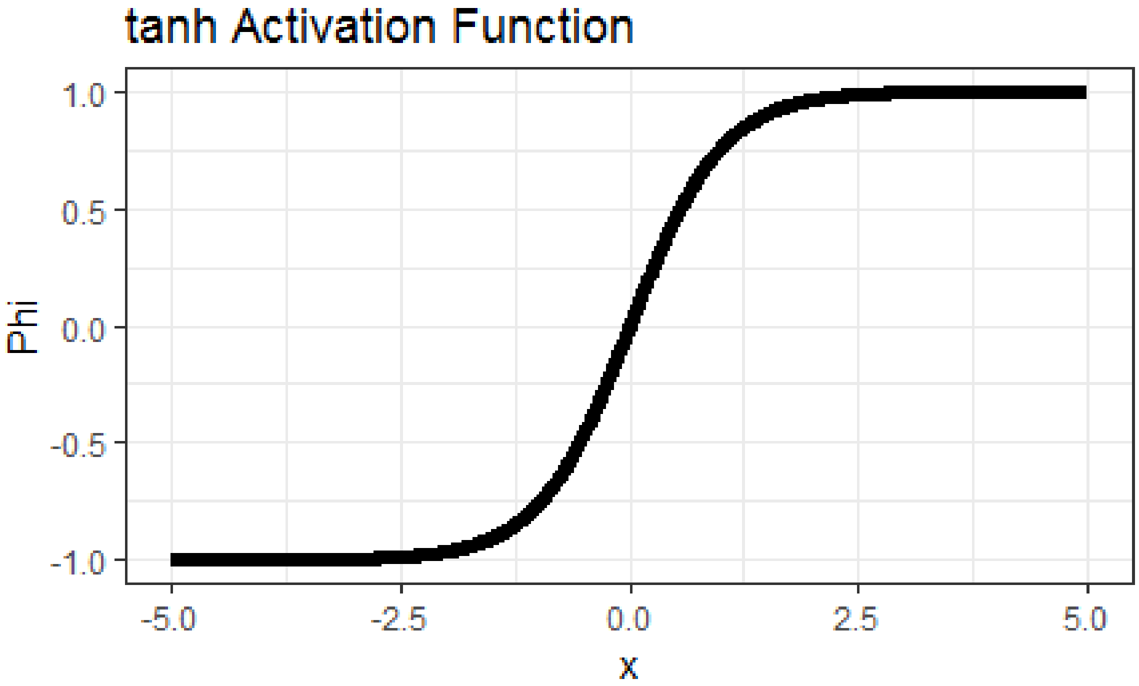


Deep Learning: Activation Functions

Hyperbolic Tangent (tanh)

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

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

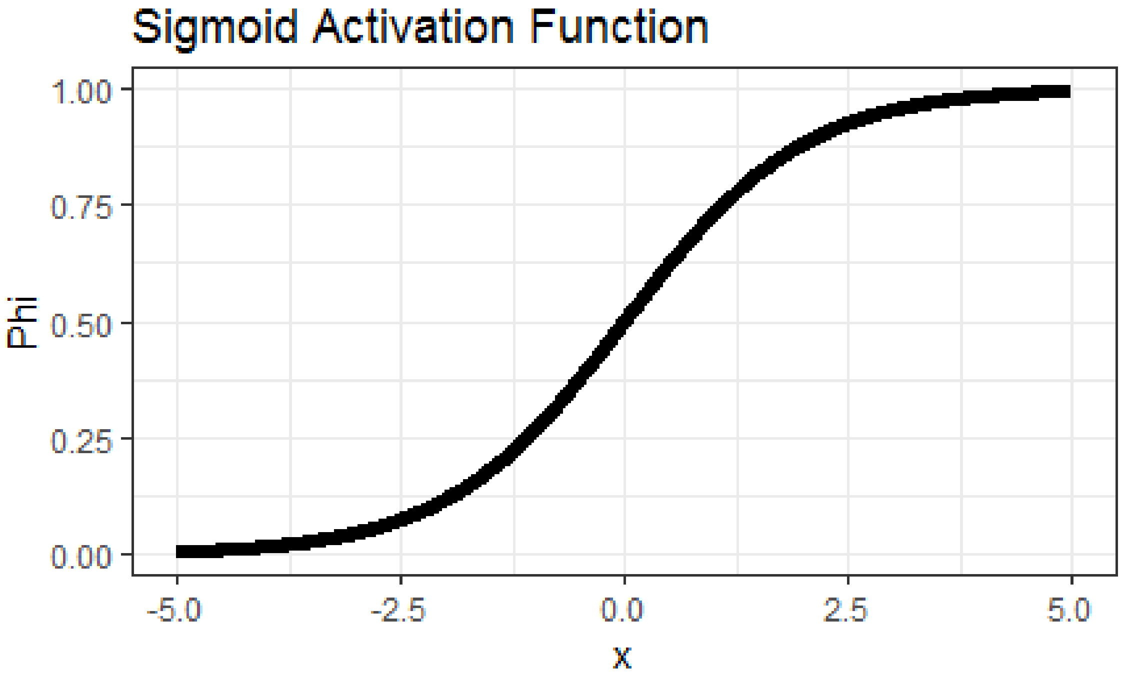


Deep Learning: Activation Functions

Activation Functions

Sigmoid

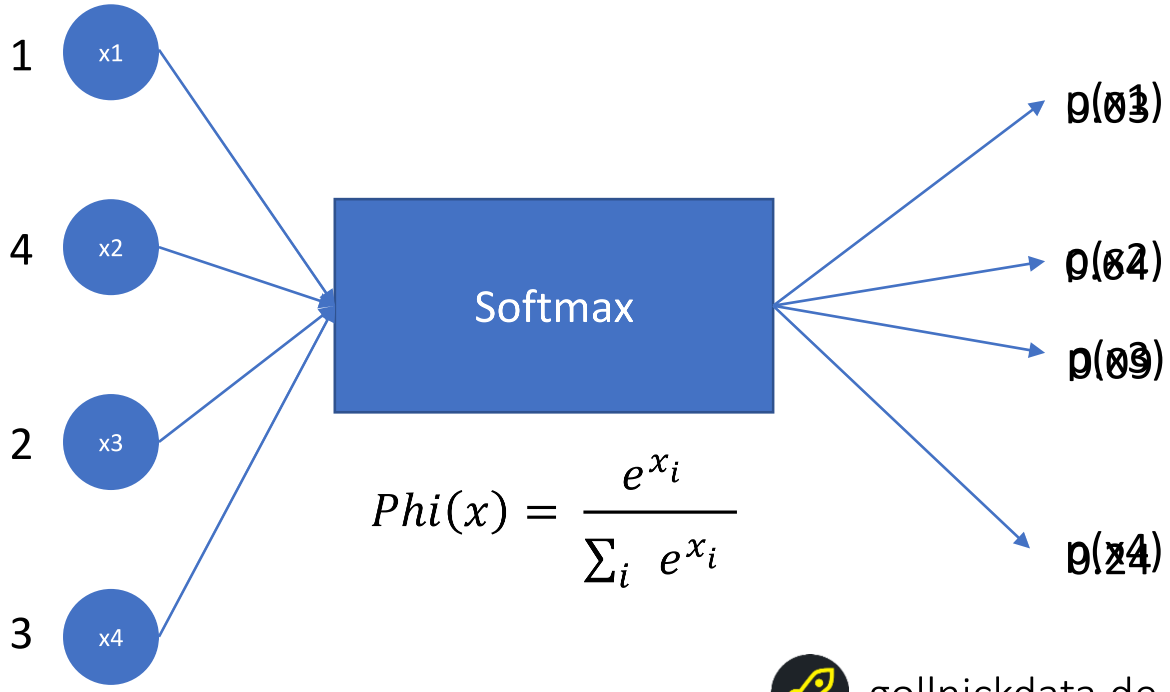
- Phi(x) = 1 / (1+e^{-x})
- Non-linear
- Relatively flat, except for small range
- Derivative small except for small range
- Might suffer vanishing gradient problem
- Result range 0 to 1



Deep Learning: Activation Functions

Softmax

- Used for multi-class prediction



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

Deep Learning: Loss Functions

Deep Learning: Loss Functions

Overview

- 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

Classification

Deep Learning: Loss Functions

Regression Loss Functions

Regression Losses

- Mean Squared Error $MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$
- Mean Absolute Error $MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n}$
- Mean Bias Error $MBE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{n}$
- Output layer has 1 node
- Typical activation function: linear

Deep Learning: Loss Functions

Binary Classification Loss Functions

Binary Cross Entropy

- Applicable for binary classification
- Most common

- Output layer has 1 node

- Typical activation function: sigmoid

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

Deep Learning: Loss Functions

Binary Classification Loss Functions

Hinge Loss

- Also called SVM loss
- Applicable for binary classification
- Used for maximum margin classifiers
- Output layer has 1 node
- Typical activation function: sigmoid

- $HingeLoss = \sum_{j \neq yi} \max(0, s_i - s_{yi} + 1)$

Deep Learning: Loss Functions

Multi-Label Classification Loss Functions

Multi-Label Cross Entropy

- Most common loss for multi-label classification
- Output layer has n nodes, where n is number of labels
- Typical activation function is softmax

Deep Learning: Optimizers

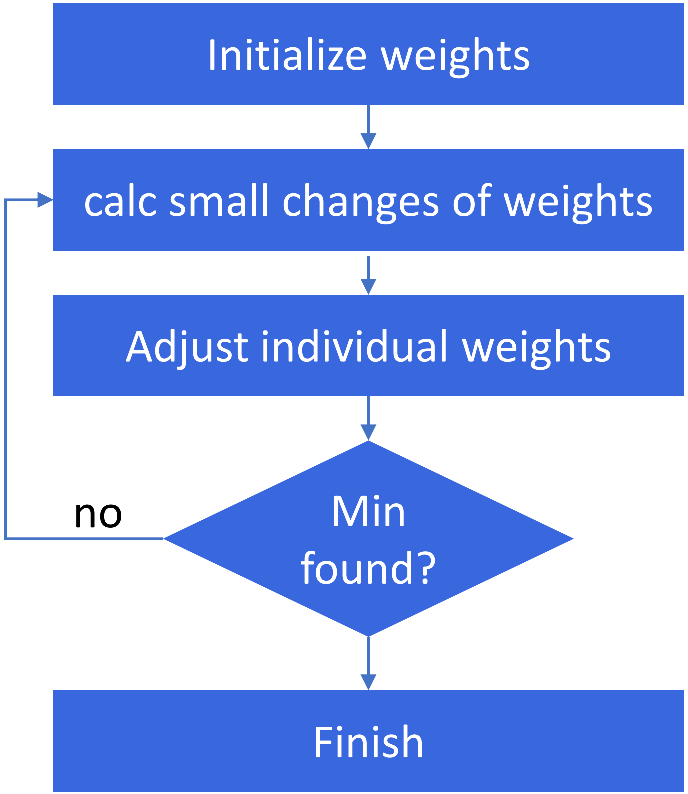
Deep Learning:Optimizer

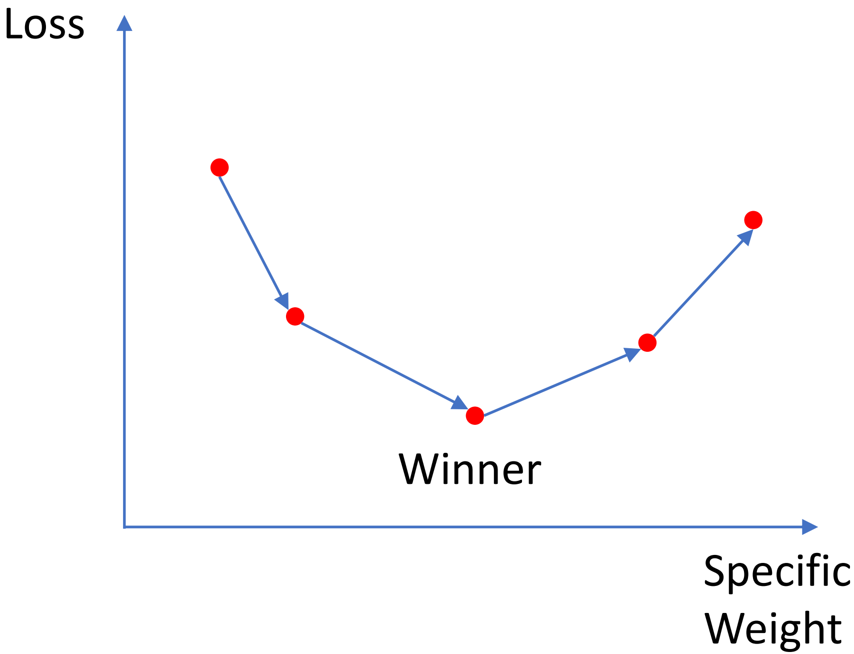
Overview

- During training weights of the model updated to **minimize loss** function
- But how? → **Optimizer**
- Calculates **updates** of **weights** based on Loss Function
- Brute force (check all combinations) → bad idea!
- Educated trial and error → good

Deep Learning:Optimizer

Gradient Descent



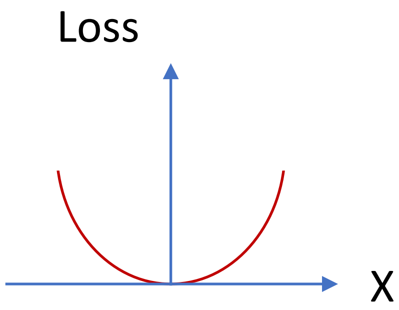


gollnickdata.de

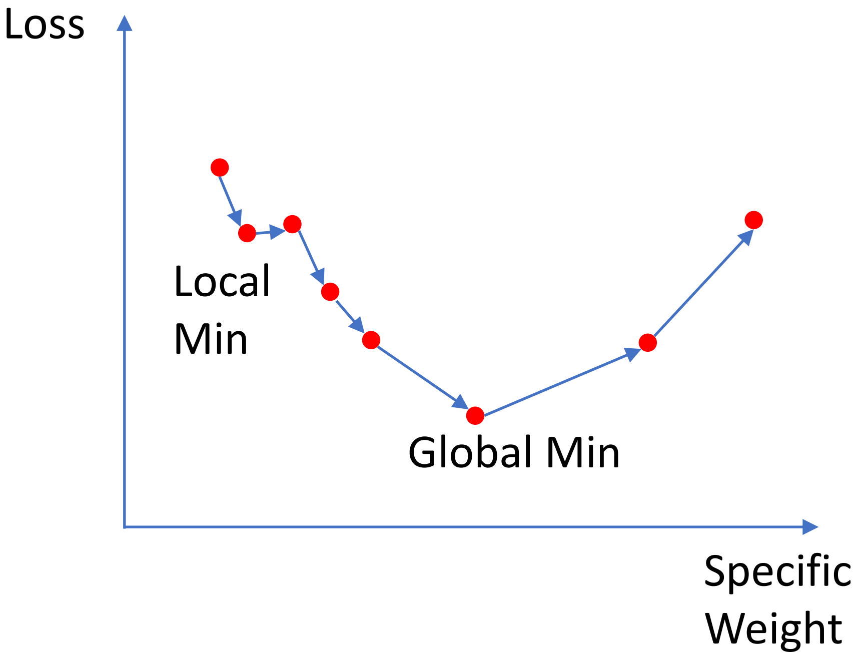
Deep Learning:Optimizer

Gradient Descent

- Problem: local minima
- Solution:
  - convex loss function



  - Learning rate

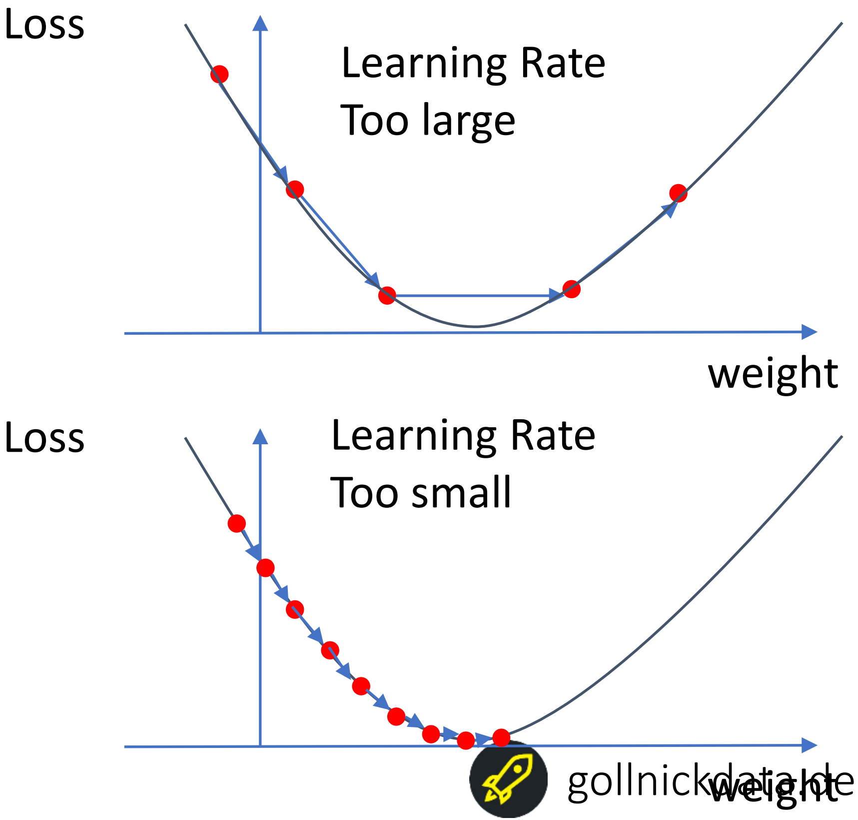


gollnickdata.de

Deep Learning:Optimizer

Learning Rate

- Size of weight changes
- High learning rate
  - Large steps
  - Risk of overshooting the minimum
- Low learning rate
  - Very precise
  - Time-consuming



Deep Learning:Optimizer

Other Optimizers

Adagrad

- Adapts learning rate to features → learning rate = f(weights)
- Works well for sparse datasets
- Learning rate decreases with time and gets sometimes too small
- Adaprop, RMSprop supposed to solve this

Adam

- **Ada**ptive **m**omentum estimation
- Applies momentum → includes previous gradients into current gradient calculation
- Widespread

More Optimizers

- Stochastic Gradient Descent, Batch gradient descent, …

PyTorch Tensors

PyTorch - Tensors

Introduction

- PyTorch structure to work with variables → PyTorch tensors
- Similar to numpy arrays, but more powerful
- Automatically calculates gradients
- Information about dependencies to other tensors

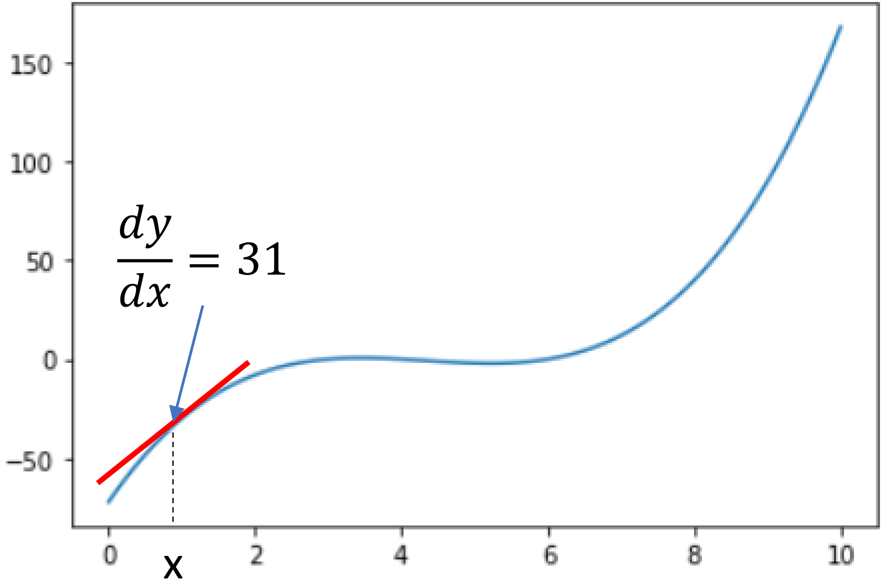
PyTorch - Tensors

Automatic Gradients

- Gradients are calculated automatically

```
# create a tensor with gradients enabled
x = torch.tensor(1.0, requires_grad=True)
# create second tensor depending on first tensor
y = (x-3) * (x-6) * (x-4)
# calculate gradients
y.backward()
# show gradient of first tensor
print(x.grad)
```

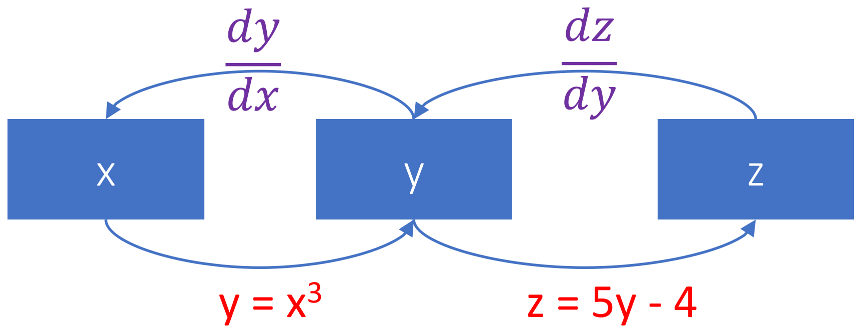
```
tensor(31.)
```



PyTorch - Tensors

Computational Graphs

- Simple network:
  - Input *x* is used to calculate *y*, which is used to calculate *z*.



Backpropagation

Forward Pass

Change of z based on change of x: $$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$ (Chain rule)

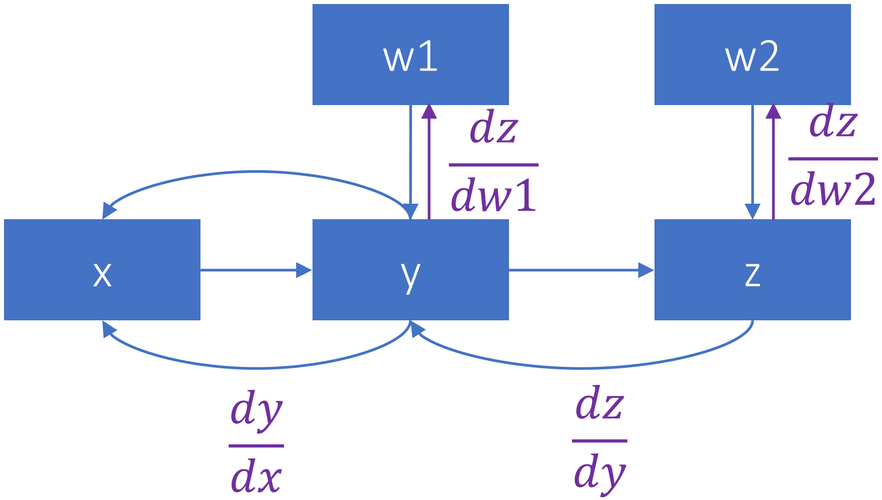
$$\frac{dz}{dx} = 5 * 3x^2$$

gollnickdata.de

PyTorch - Tensors

Computational Graphs

- Update of Weights
  - Calculated output z
  - True output t
  - Error E = $(z-t)^2$
  - Weights can be considered as nodes as well
  - z = f(y, w2)
  - Optimizer updates weights based on gradients

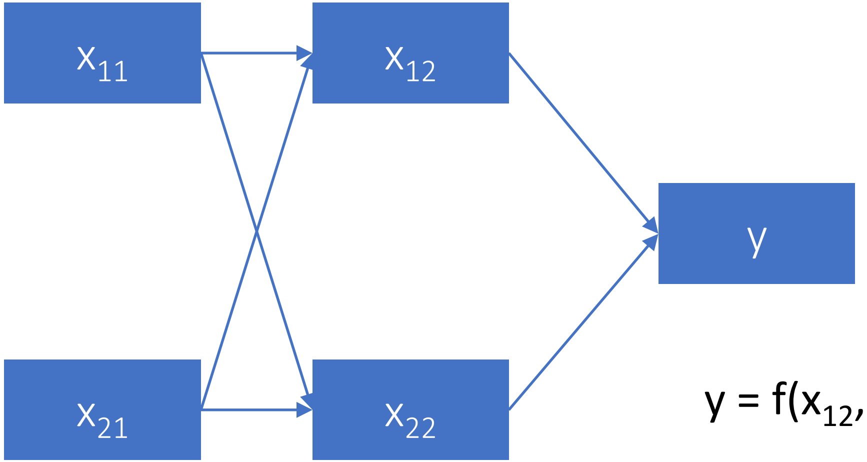


PyTorch - Tensors

Computational Graphs: Forward Pass

- More complex network with multiple inputs

$x_{12} = f(x_{11}, x_{21})$



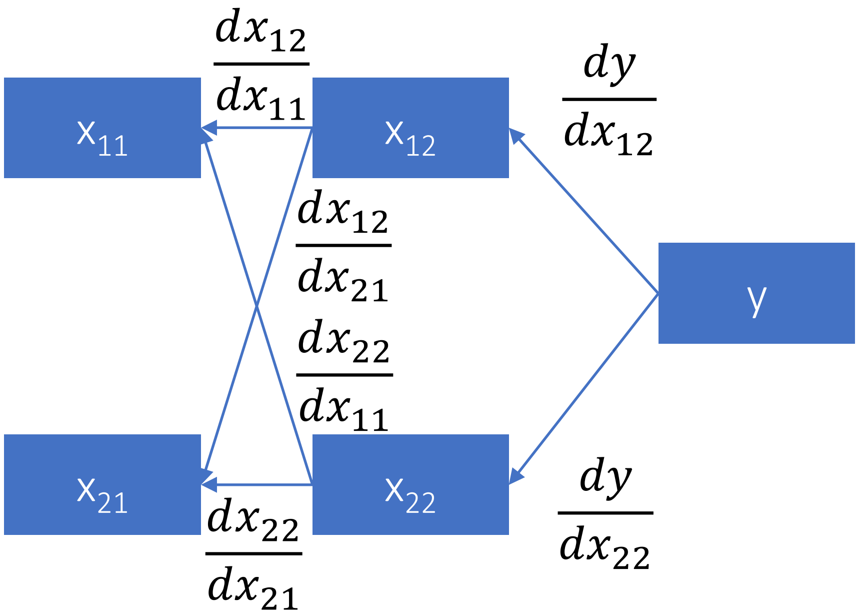
$y = f(x_{12}, x_{22})$

$x_{22} = f(x_{11}, x_{21})$

PyTorch - Tensors

Computational Graphs: Backpropagation

- More complex network with multiple inputs

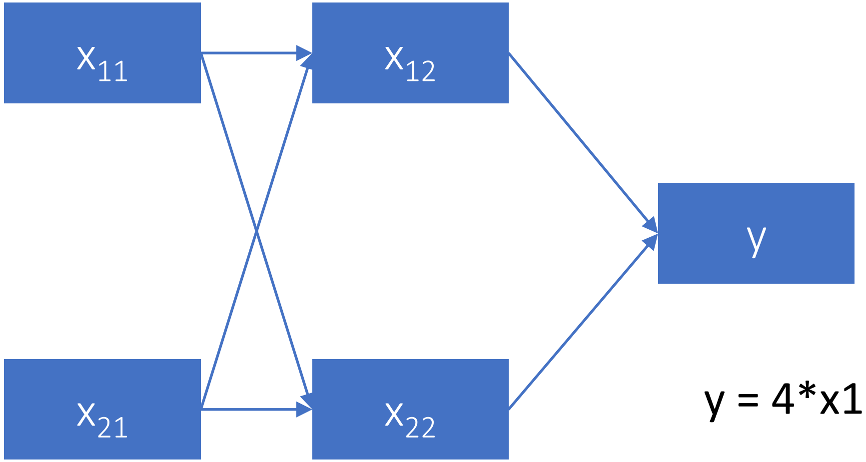


PyTorch - Tensors

Computational Graphs: Forward Pass

- Example

$x_{12} = 5*x_{11} – 3*x_{21}$



$y = 4*x12 + 3*x22$

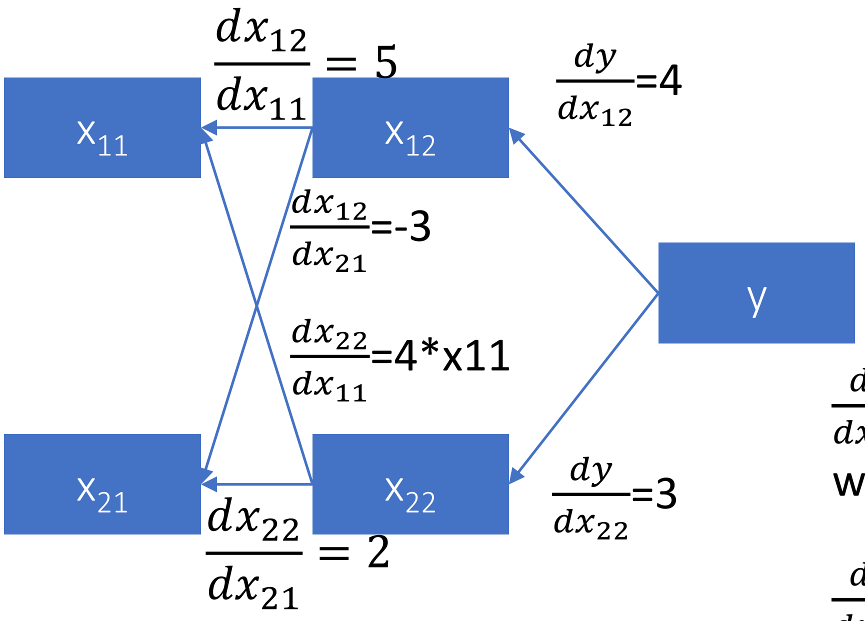
$x_{22} = 2*x_{11}^2 + 2*x_{21}$

gollnickdata.de

PyTorch - Tensors

Computational Graphs: Backpropagation

- More complex network with multiple inputs



$\frac{dy}{dx_{11}}$=5*4+4*x11*3=44

with x11=2

$\frac{dy}{dx_{21}}$=3*2+4*(-3)=-6

gollnickdata.de

Modeling: Section Overview

Modeling: Section Overview

Section Overview

PyTorch Model Training	nn.Module	Training Loop	Model Evaluation
Datasets	Dataloaders	Batches	Activation Functions
Hyperparameter Tuning	Saving / Loading Models		

Datasets and Dataloaders

Datasets and Dataloaders

Introduction

- Model training ideally should be separated from data preprocessing
  - Better readability and modularity
- Dataset and Dataloader
  - Interface to Pre-loaded datasets
  - Interface to custom datasets
- **Dataset**
  - Stores samples and labels
- **Dataloader**
  - Iterable wrapped around Dataset

Datasets and Dataloaders

Custom Dataset

- Requires three function implementations:
  - `__init__`
    - Runs once during instanciating the object
  - `__len__`
    - Returns number of samples
  - `__getitem__`
    - Loads samples from dataset, pre-processes them and returns them for given index

```
from torch.utils.data import Dataset, DataLoader
```

```
Run Cell | Run Above | Debug Cell
#%% Dataset and Dataloader
class LinearRegressionDataset(Dataset):
    def __init__(self, X, y):
        self.X = X
        self.y = y

    def __len__(self):
        return len(self.X)

    def __getitem__(self, idx):
        return self.X[idx], self.y[idx]
```

Datasets and Dataloaders

Dataloader

- Dataloader iterates through dataset
- Iterations return batches of data
- Features
  - allows for shuffling the data
  - custom sampling strategies

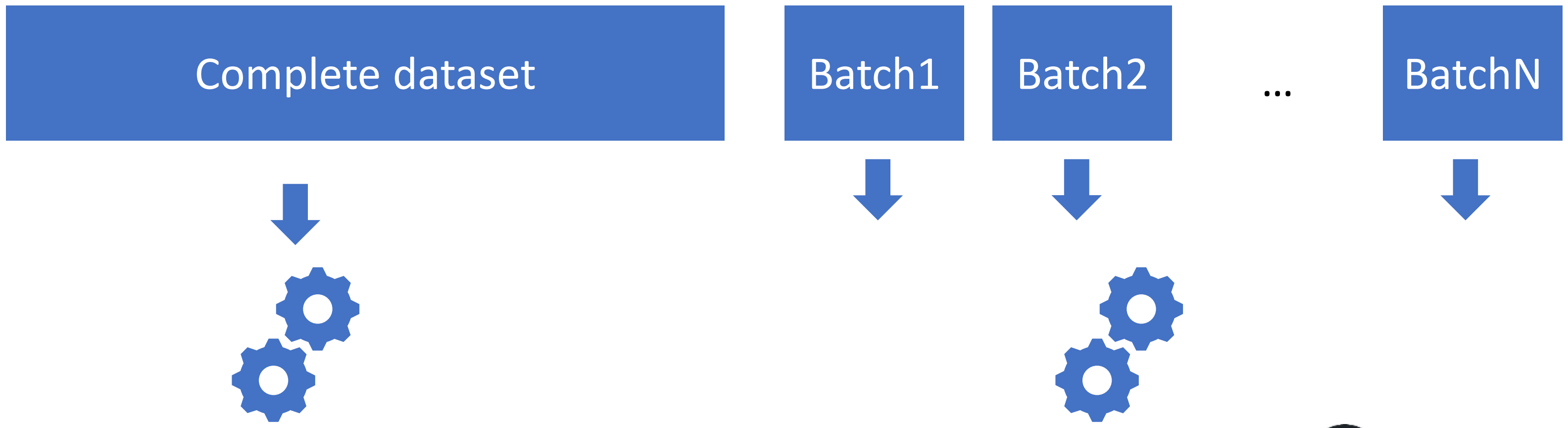
```
train_loader = DataLoader(dataset = LinearRegressionDataset
(X_np, y_np), batch_size=2)
```

Batches

Batches

Introduction

- Often datasets very large → passing the complete dataset in one step to the model training impossible
- Rather smaller bites provides to be consumed by model - batches



Batches

Batch Size

- Batch size number of simultaneous provided datasets provided to model.
- Defines speed of model training and stability of learning process.

Reasons for using smaller batch sizes

- Smaller batch sizes noisier and lower generalization error
- Easier to pass a batch of training data in memory (CPU or GPU)

Typical batch sizes

- 1 – 512
- Often multiple of two
- Good default: 32