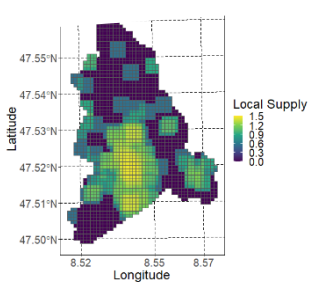
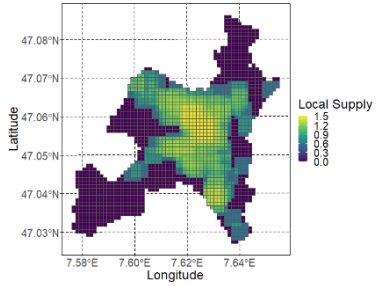
Gears of Neural Networks

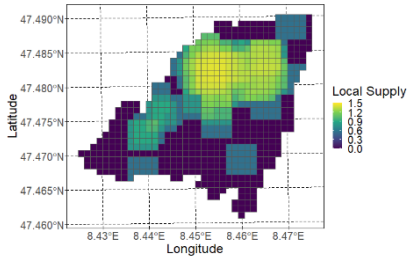
Dr. Yves Staudt



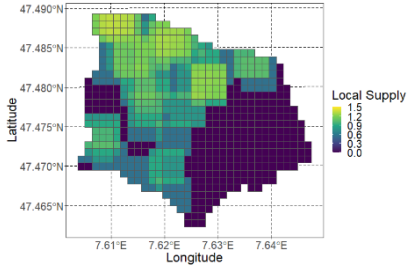
(a)
Grenchen



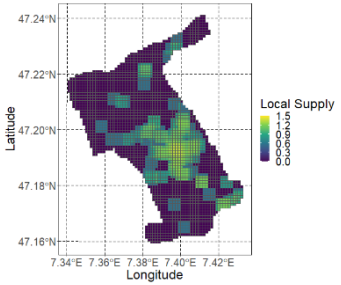
(b)
Oberglatt



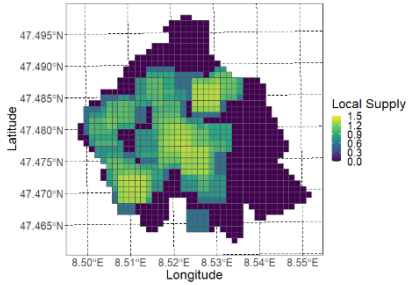
(c)
Reinach



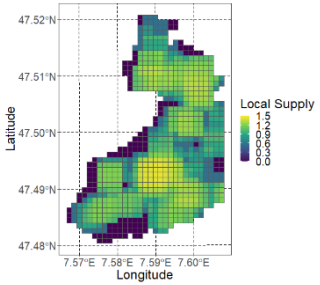
(d)
Rubigen



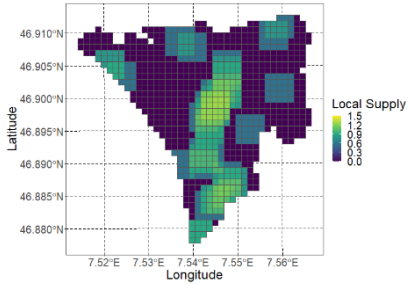
(e)
Buelach



(f)
Burgdorf



(g)
Dielsdorf



(h)
Dornach

Lernziel

Die Studierende sind in der Lage

- die Elemente eines Deep Learning Modells zu bestimmen und zu beschreiben.

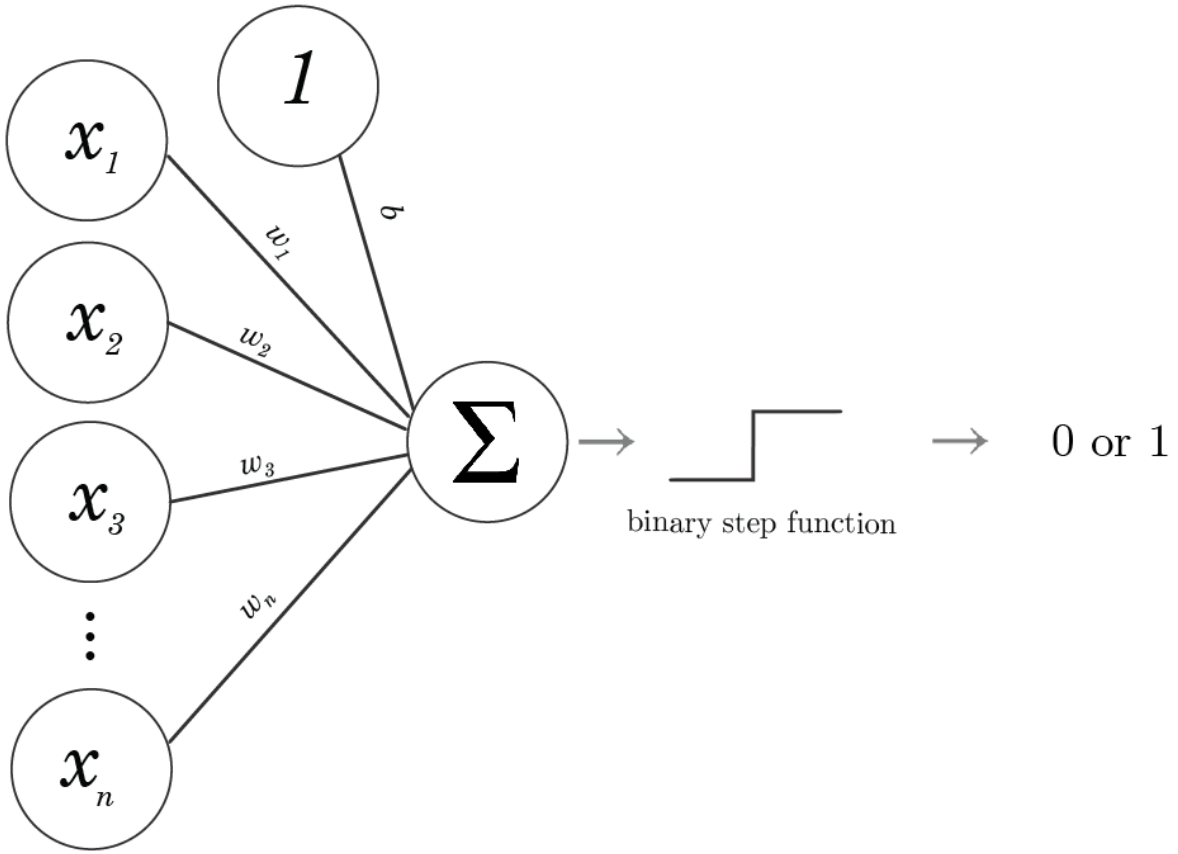
FH GR

Goal

The goal in deep learning is to optimize the parameters such that the model's predictions on the training data are as close as possible to the true labels

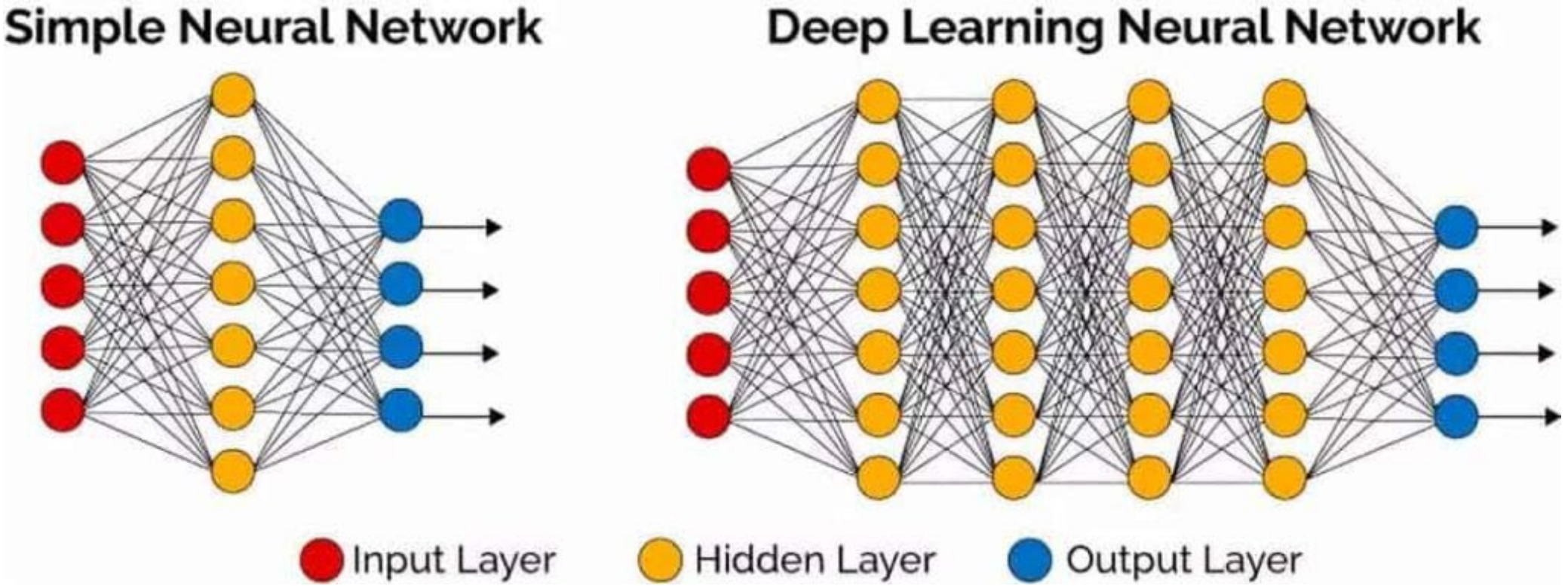
FH GR

Perceptron Erweiterung



Perceptron mit Aktivierungsfunktion

Deep Learning



Deep Learning

Initialization of the weights

Random initialization:

- Simplest method to randomly initialize the weights using a normal distribution with mean 0 and standard deviation 1
- This is also known as the Glorot initialization or Xavier initialization.

Uniform initialization:

- Common method to initialize the weights using a uniform distribution with a specific range, such as [-0.1, 0.1] or [-0.01, 0.01]

Zeroes:

- Setting all initial weights to zero

Ones:

- Setting all initial weights to one

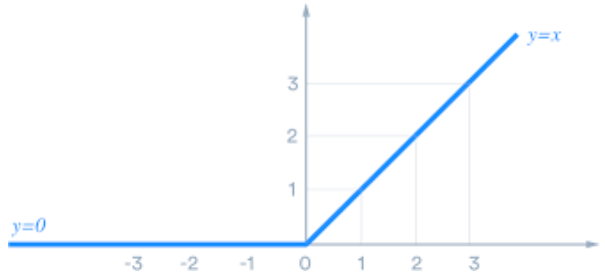
FH GR

Choice of Hidden Layers and Nodes

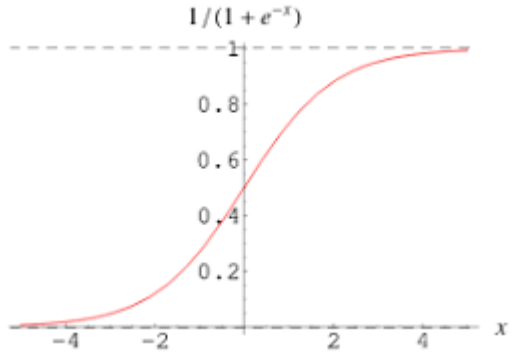
- The choice of the number of **hidden nodes** and **hidden layers** in feed forward neural networks depends on the complexity of the problem
- The higher the number of hidden nodes and layers, the higher the complexity
- Some general **guidelines**:
  - Start with a simple model – 1 hidden layer
  - **Common rule of thumb** for the number of hidden nodes: number of nodes between the size of the input layer and the output layer
- Most common problems can be solved with two hidden layers

FH GR

Common Activation Functions



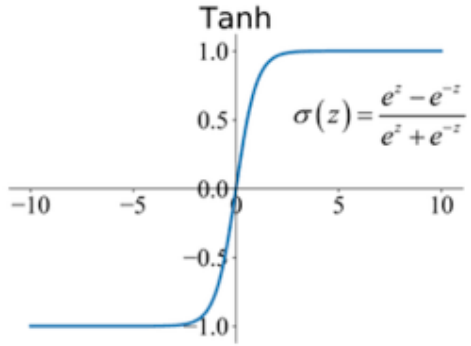
- Rectified Linear Unit (ReLU):
  - ReLU is the most popular activation function for deep learning.
  - It is a simple function that returns 0 for negative inputs and the input value for positive inputs.



- Sigmoid:
  - The sigmoid function is commonly used in feedforward neural networks.
  - It maps any input value to a value between 0 and 1,

    which can be interpreted as a probability.

- Hyperbolic tangent (tanh):
  - The hyperbolic tangent function is similar to the sigmoid function, but it maps input values to a range between -1 and 1.



FH GR

Common Activation Functions for Output

The choice of activation function for the output layer depends on the type of problem you are trying to solve.

Commonly used activation functions for the output layer are:

- **Sigmoid**:
  - The sigmoid function is commonly used in binary classification problems
  - Output is a probability value between 0 and 1.
- **Softmax**:
  - The softmax function is used in multi-class classification problems
  - Output is a probability distribution over multiple classes.
  - It normalizes the output values to ensure they sum to 1.
- **Linear**:
  - The linear activation function is used in regression problems
  - Output is a continuous value

FH GR

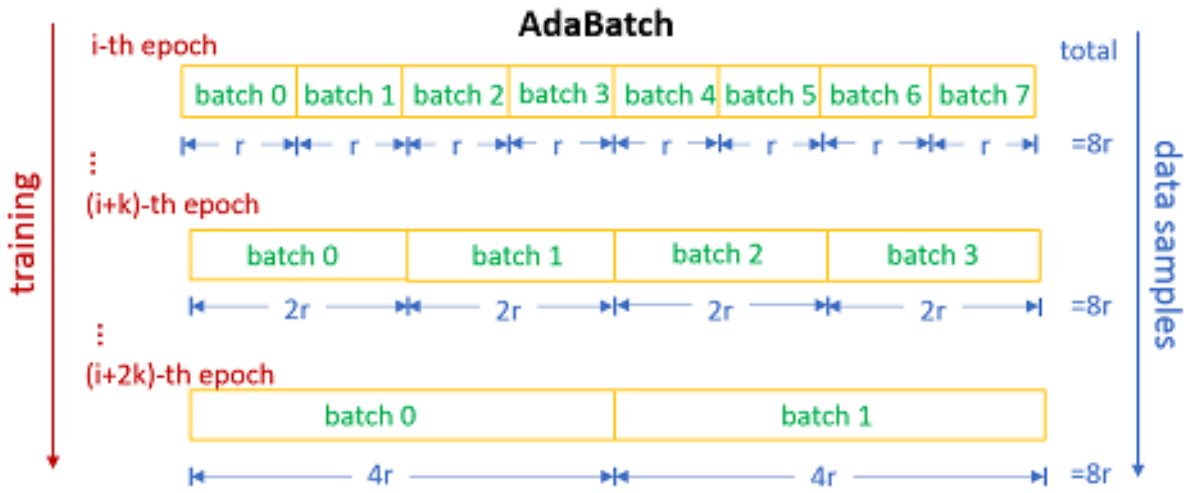
Epochs

- In deep learning, an epoch refers to a **single pass** over the **entire training dataset** during the training phase
- One epoch is complete when the model has seen and **processed every example** in the training set once
- During each epoch, the model **updates its parameters** (weights and biases) based on the gradients of the loss function with respect to those parameters
- After each epoch, the **model's performance** is typically **evaluated** on a separate validation set
- The number of epochs used to train a deep learning model is a **hyperparameter**
- If the number of epochs is **too low**, the model may **not be able to converge** to an optimal solution, resulting in poor performance.
- If the number of epochs is **too high**, the model may **overfit** to the training data
- The number of epochs needed to train a deep learning model depends on the **complexity** of the problem, the size of the dataset, and the architecture of the model

FH GR

Batches

- In deep learning, a batch refers to a **subset** of the **training data**
- Batches are used to **update** the model's **parameters** during **one iteration** of the optimization algorithm
- Batches allow the model to make **updates more frequently** and efficiently, while also reducing the **memory** requirements of the optimization process
- By passing each batch through the model, computing the loss function and gradients, and then updating the parameters using an optimizer such as stochastic gradient descent (SGD) or Adam



Choice of the Batches Size

- The choice of batch size is another **important** hyperparameter

- A **small** batch size can lead to **faster** convergence and **better** generalization performance
- A small batch size forces the model to learn more **robust features** that are relevant across different batches
- A smaller batch size can also **slow down** training and make the optimization process **less stable**
- The gradients computed on small batches can be more **noisy** and less **representative** of the true gradients on the entire training set

- A **larger** batch size can lead to **faster** training and more **stable gradients**, as the model has more information to work with during each update
- A larger batch size can also result in **overfitting**

FH GR

Needed Things for Compilation

For the compilation of a deep learning model we need the following three things:

- ***Loss function*** describes how the network will be able to **measure its performance** on the training data and how it will be able to **steer** itself in the **right direction**
- ***Optimizer*** is the **mechanism** through which the network will **update** itself based on the data
- ***Metrics*** monitor the **accuracy** of the model **during training and testing**

FH GR

Loss Functions for Regression

Mean Squared Error (MSE):

- Popular loss function used for regression problems.
- Measures the average squared difference between the predicted and actual values.

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Mean Absolute Error (MAE):

- Loss function used for regression problems.
- Measures the average absolute difference between the predicted and actual values.

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

FH GR

Loss Functions for Regression

Huber Loss:

- Loss function that is less sensitive to outliers than MSE.
- Used in regression problems where there are outliers in the data.

$$Huber\ Loss = \frac{1}{n} \cdot \sum_{i=1}^{n} L\delta(y_i - \hat{y}_i)$$

where

$$L\delta(x) = \begin{cases} 0.5 \cdot x^2 & if\ |x| \leq \delta \\ \delta \cdot (|x| - 0.5\ \delta) & if |x| > \delta \end{cases}$$

FH GR

Loss Functions for Classification

Binary Cross-Entropy:

Loss function used for binary classification problems.

Measures the difference between the predicted and actual values using the logarithmic loss.

$$CE = -\frac{1}{n} \cdot \sum_{i=1}^{n} [y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot (1 - \log(\hat{y}_i)]$$

Categorical Cross-Entropy:

Loss function used for multi-class classification problems.

Measures the difference between the predicted and actual values using the logarithmic loss.

$$CE = -\frac{1}{n} \cdot \sum_{i=1}^{n} \sum_{j=1}^{k} y_{ij} \cdot \log(\hat{y}_{ij})$$

Where i is the i-th sample and j the j-th class

FH GR

Metrics

- The loss function is used to optimize the parameters of the neural network during training
- Evaluation metrics are used to measure the performance of the model on the training and test data
- Evaluation metrics provide insights into its strengths and weaknesses
- In regression problems, the loss function and evaluation metrics are the same
- In classification problems this is not the case

FH GR

Metric for Classification - Confusion Matrix

		Real classification	
		0	1
Prediction	0	TN	FN
	1	FP	TP

Metric for Classification – Accuracy

Accuracy formula:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

FH GR

Metric for Classification – Precision/Recall/False Positive Rate/True Negative Rate

Precision also called Predictive Positive Rate:

$$PPV = \frac{TP}{TP + FP}$$

Recall also named Sensitivity or True Positive Rate:

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

False Positive Rate:

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

True negative rate also called Specificity:

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

FH GR

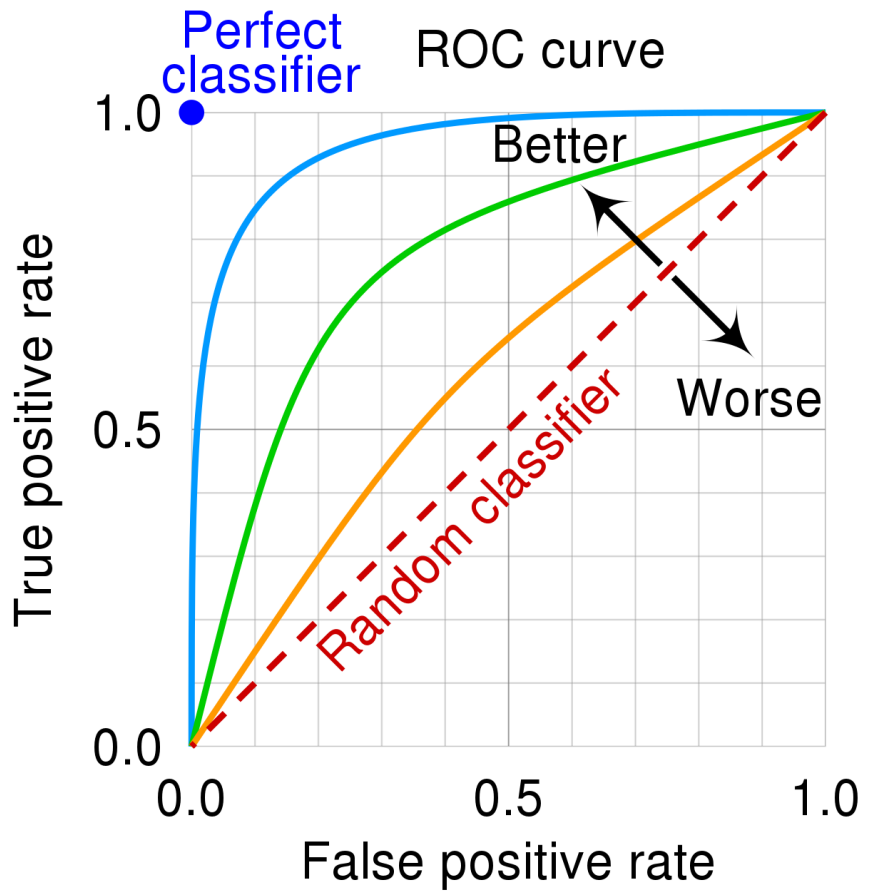
Metric for Classification – F1 Score

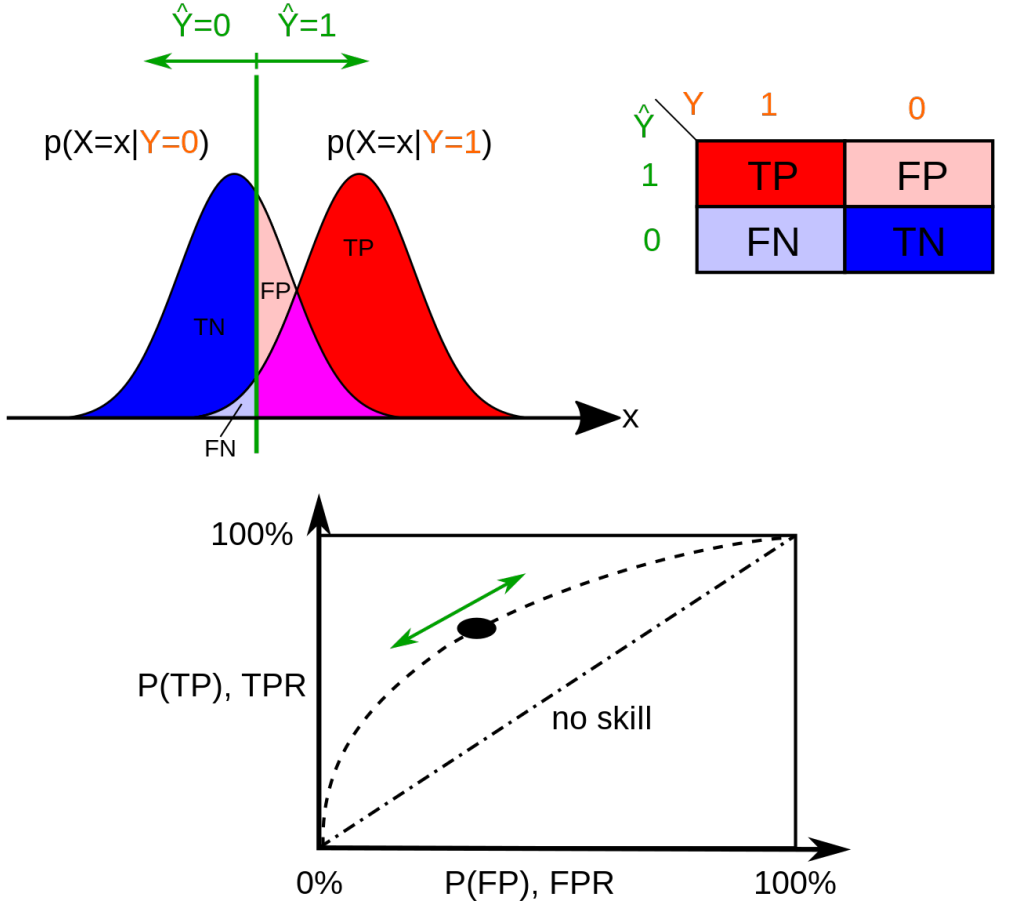
$$F_1 = 2\frac{precision \cdot recall}{precision + recall} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

- F1 score reaches its best value at 1
- F1 score worst score at 0

FH GR

Receiver Operating Characteristic



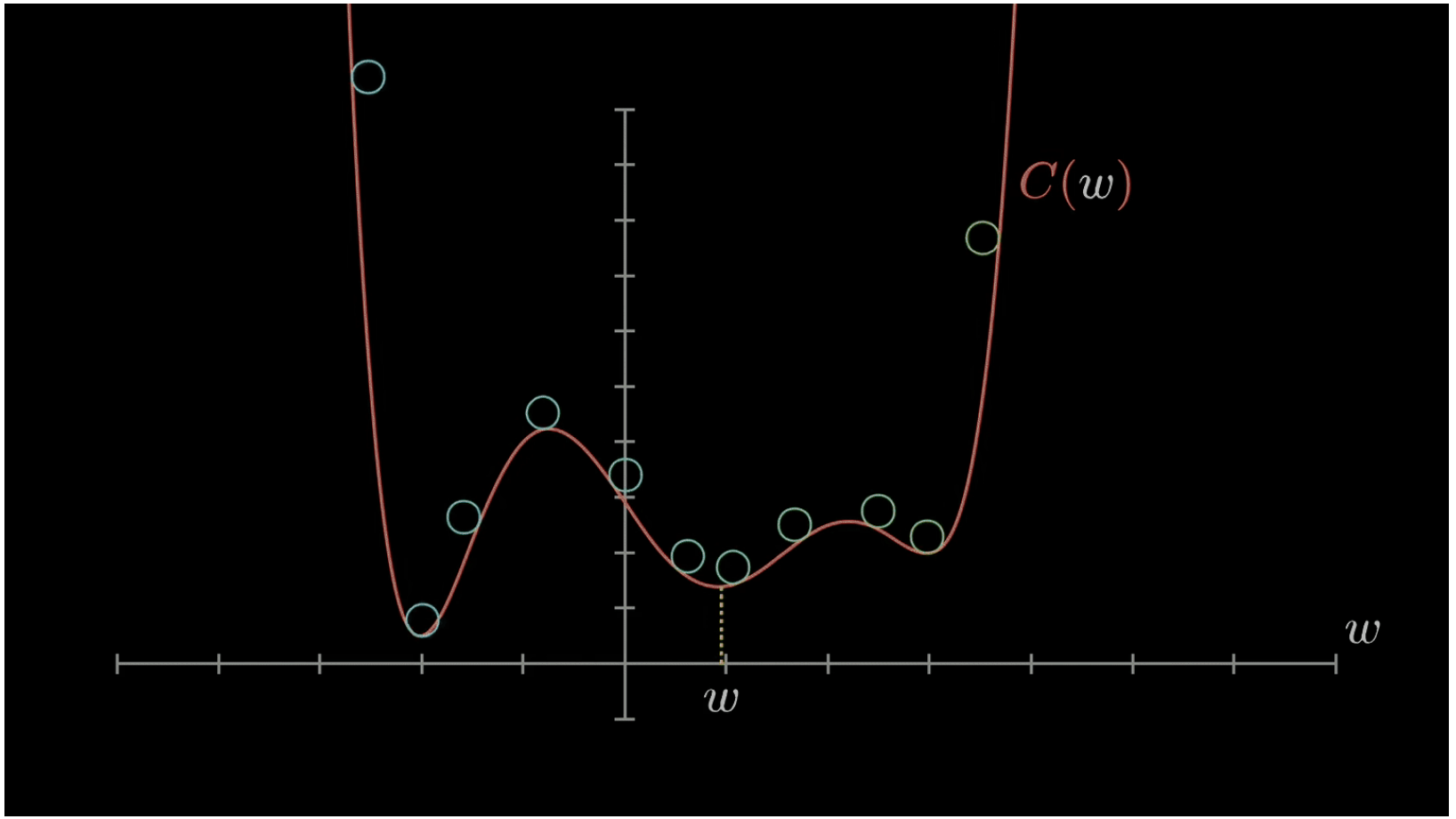


Area under the ROC Curve

- The area under a ROC curve (AUC) is a single scalar value that measures the overall performance of a classifier.
- AUC is within [0.5-1.0]
- AUC = 1: represents a perfect classifier
- AUC is a robust measure

FH GR

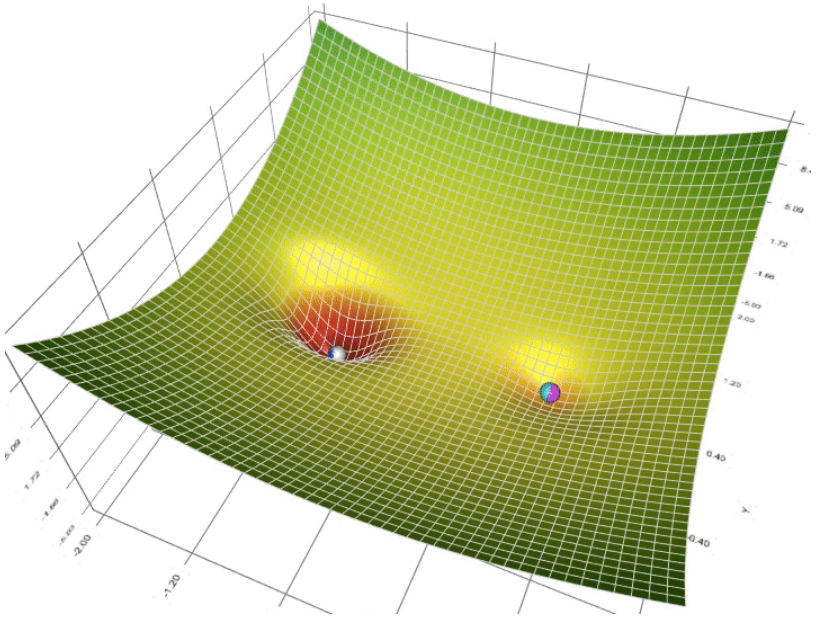
Gradient Descent



FH GR

Schwierigkeit Gradient Descent

- Das Konvergieren in ein lokales Minimum einer Kostenfunktion
  - Durch wiederholtes Anwenden des gradiellen Abstiegs



Learning Rate

- The learning rate is a critical hyperparameter in deep learning
- The learning rate controls the **step size** used to update the weights during training
- The learning rate determines how quickly or slowly the model learns from the data
- The learning rate have a significant **impact** on the **performance** of the model
- If the learning rate is too **high**, the model may fail to converge or even **diverge**
- If the learning rate is too **low**, the model may **converge very slowly** or get stuck in a **local minimum** of the loss function
- A to low learning rate can **prevent** the model from finding the **global minimum** of the loss function.

FH GR

Application of Learning Rate

To find an **appropriate** learning rate, it's common to start with a relatively high learning rate and **gradually reduce** it during training

To gradually reduce the learning rate is known as **learning rate annealing** or **decay**

Some common strategies for learning rate **annealing** include:

- **Step decay**: The learning rate is reduced by a fixed factor after a certain number of epochs
- **Exponential decay**: The learning rate is reduced exponentially over time
- **Adaptive** learning rate: The learning rate is adjusted dynamically based on the gradients of the loss function

FH GR

Optimizer

There exists several optimizers
The most common are:

- Stochastic Gradient (SGD)
- Adaptive Moment Estimation (Ada)
- Root Mean Square Propagation (RMSprop)
- Adaptive gradient (Adagrad)

FH GR

Stochastic Gradient Descent (SGD)

- The most basic optimizer
- Good starting point.
- SGD works by adjusting the weights of the neural network in the direction of the negative gradient of the loss function with respect to the weights
- Variants of SGD: mini-batch SGD, momentum SGD

FH GR

Adaptive Moment Estimation (Adam)

- Adapative learning rate optimizer
- Very popular in recent years.
- Uses a combination of the first and second moments of the gradients to adapt the learning rate for each parameter.

FH GR

Root Mean Square Propagation (RMSprop)

- Adaptive learning rate optimizer
- Uses a moving average of the squared gradient to normalize the gradient values

FH GR

Adaptive Gradient (Adagrad)

- Adaptive learning rate optimizer
- Adapts the learning rate for each parameter based on the history of the gradients for that parameter.

FH GR

Fragen

Darstellung eines Fragesymbol aufgerufen von der Webseite https://www.qnigge.de/news/detail/modul-v/#images am 12.07.2021.

Fachhochschule Graubünden
Pulvermühlestrasse 57
7000 Chur
T +41 81 286 24 24
info@fhgr.ch

Vielen Dank für Ihre Aufmerksamkeit.
Grazia fitg per l'attenziun.
Grazie per l'attenzione.

Fachhochschule Graubünden
Scola auta spezialisada dal Grischun
Scuola universitaria professionale dei Grigioni
University of Applied Sciences of the Grisons