Exkurs: Linear Algebra

Matrix: $\begin{bmatrix} 5 & 12 & 6 \\ 3 & 0 & 14 \end{bmatrix} = \underset{2\times3}{A}$ $\quad a_{12} = 12$ $\quad$ 2 Dimensions

Scalar: $[15] = \underset{1\times1}{A}$ $\quad$ 0 Dimensions

Vector: $\begin{bmatrix} 5 \\ -2 \\ 4 \end{bmatrix} = \underset{3\times1}{V}$ $\quad$ 1 Dimensions

Transpose:

$$X = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad X^T = [1\ 2\ 3] \quad (X^T)^T = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Dot Product:

$$m \times n \cdot n \times k = m \times k$$

$$3 \times 1 \cdot 1 \times 4 = 3 \times 4$$

NOW AIRING: Udemy Deep Learning

Training an Algorithm:

- Data
- Model
- Objective Function → minimize error
- Optimization Algorithm

Linear Model: $f(x) = w \cdot x + b$

- x: Input
- w: coefficient (weight)
- b: Intercept (Bias)

Example: Houseprice = sqft(x) · weight − bias

Input Model is determined by W & B

$$Y = X \mid W + B$$

$n \times m = n \times k * k \times m \quad 1 \times m$

- n: Observation
- m: Output var
- Obs
- Input var
- In
- Out
- Out

Objective Function:

def: Measure used to evaluate how well the Output matches the desired correct Values

Lower Loss = Lower Error

Regression: L2-norm $= \sum_i (y_i - t_i)^2$ (Input: 1.19, 1.21)

- y_i: output
- t_i: target

Classification: Cross-Entropy $= L(y, t) = -\sum_i t_i \ln y_i$ (Inputs: Cats, Dogs)

Optimization Algorithms

Gradient descent:

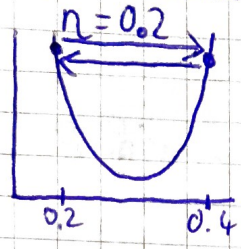
RL-Examples: $f'(x) = 10x + 3$

$x_0 = 4$ ⎫ $10x_0 + 3 = 43$

$x_1 = 4 - \eta 43$ ⎭

η (eta) ⟶ Learning rate

if learningrate too big we will start oscillate

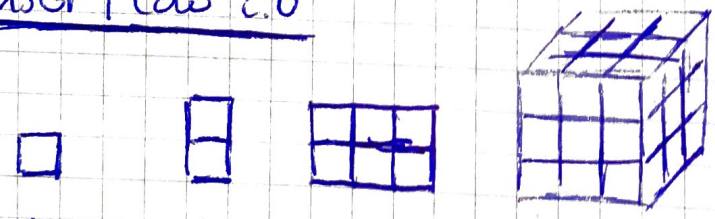


the minimum is reached when:

$x_{i+1} = x_i - \eta f'(x_i)$

$x_{i+1} = x_i$ ⟶ no more updates

TensorFlow 2.0

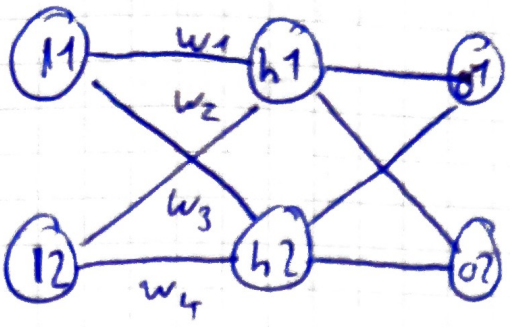


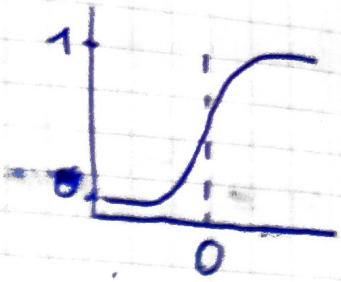
Skalar Tensor Rank1 | Vector Tensor Rank2 | Matrix Tensor Rank3 | Tensor

data → preprocess → save in .npz ↑ Multidimensional Files easyer and faster to work with

Epoch = iteration over the full dataset

Forward Pass





I H O

Input Output

$$net_{h_1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

Implementation:

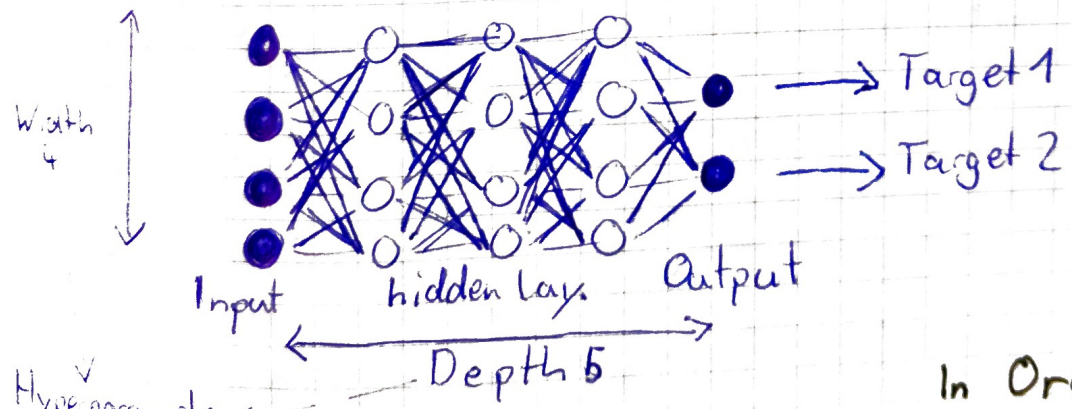
Scaling is very important!!!

Thorax and Elbow needs to be scaled clearly

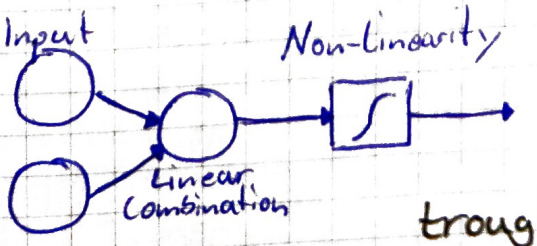
Maybe write own Trainer, Keras might be enough

Lecture by Jeroen

What is a deep Net?



Hyperparameter



In Order to have deep nets and find complex relationships trough arbitrary functions, we NEED non-linearities

non-linearities are called ACTIVATION FUNCTION

↳ also: Transfer functions

The last activation function of a DNN is often a Softmax: $\frac{e^{a_i}}{\sum_j e^{a_j}} = \frac{E}{n}$

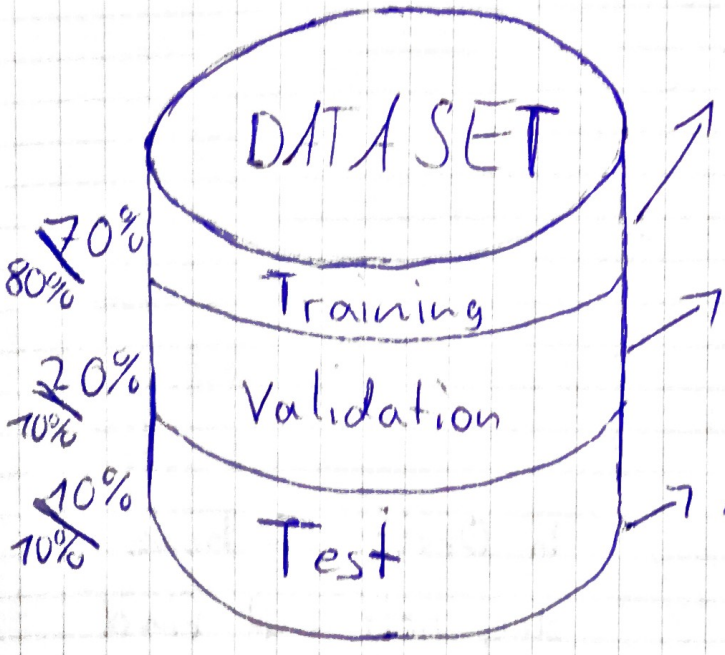
↳ This gives back PROBABILITIES that add up to 1. Which makes the Softmax very convenient for the last hidden Layer

What's **Overfitting**:

Our training has focused on the particular trainingset so much, it has "missed the point"

What's **Underfitting**:

The model has not captured the underlying logic of the data.

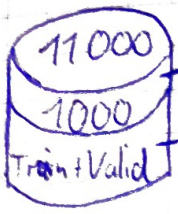


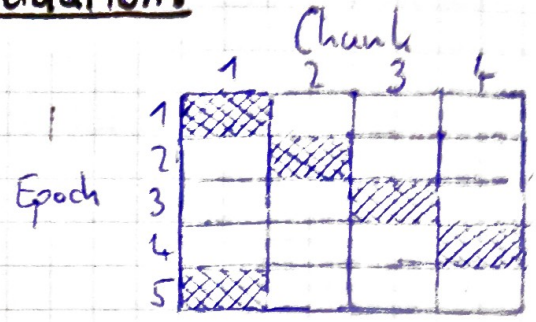
That's where training happens

Helps us prevent overfitting

Measures final predictive Power of the model

N-fold cross-validation:





Train

Valid

Pros:

Utilized more data

↓

We have a Model

Red flag

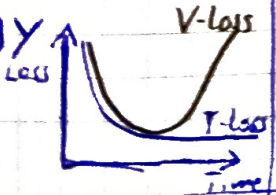
Cons:

Possibly overfitted a bit

tradeoff

Only used if not enough data!!!

Early Stopping

	Pros	Cons
1 Preset number of Epochs	- Eventually, solves the Problem	- No guarantee to reach min - Might not minimize - Naive
2 Stop when updates become too small	- We're sure the loss is minimized - Saves computing power	- Might still overfit
3 Validation set Strategy 	- We're sure the loss is minimized - Saves computing Power - Prevents overfitting	- Might iterate through validationset widerily
4 best parts of 2 and 3	all of them	NONE

PREPROCESSING

Why?:
- Compatibility
- Orders of magnitude
- Generalization

One-Hot Encoding

	bread	yogurt	muffin
🍞	1	0	0
🥛	0	1	0
🧁	0	0	1

1. Standardization:

$$\text{standardized variable} = \frac{x - \mu}{\sigma}$$

x: original variable; μ: mean of original variable; σ: standard deviation of original variable

2. Normalization:

5	100
120	-10
-1	0

normalize → using L2-norm

.042	.995
.999	-0.1
-0.008	0

std = standard deviation

3. PCA:

principal components analysis

xr, yr, zr → rotation where mean = 0, std = 1

4. Whitening:

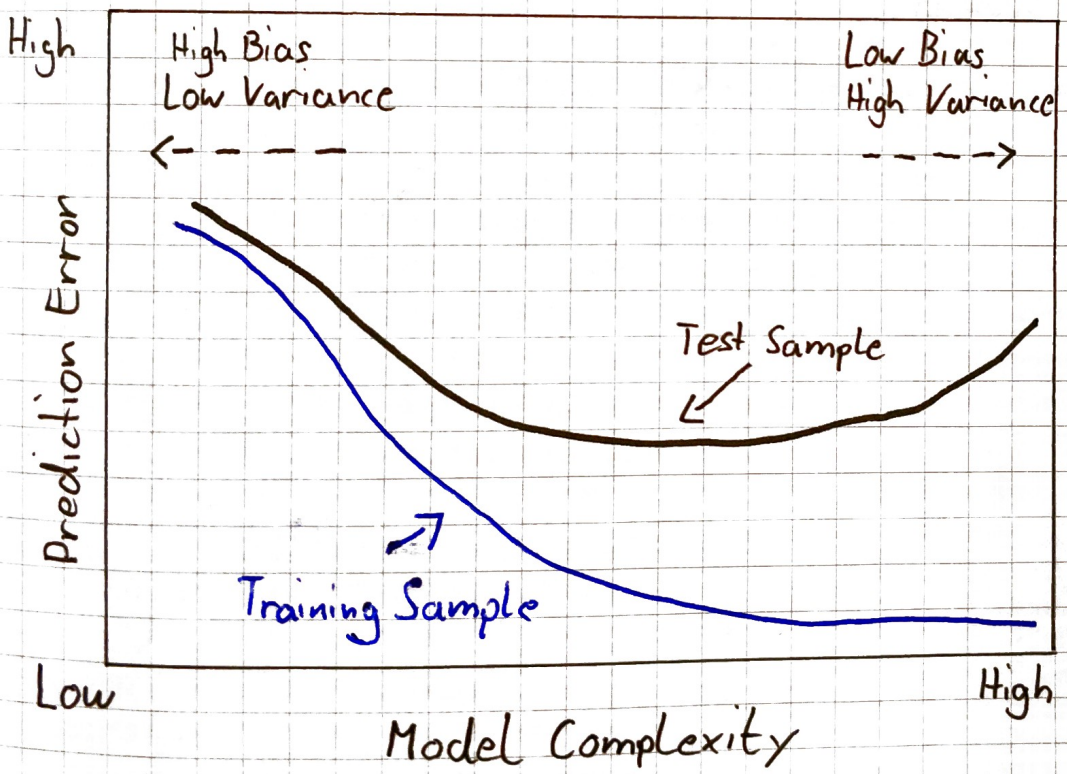
⋮

Week 4: Evaluation Methodology

Regression: Estimate the relationship between a target variable and the Features (House vs. Room)

Classification: predict the most likely class given the features of a datapoint

Validation



Evaluation

- Machine Learning is a Black-box
  - We make many assumptions
    - About the Model and it's type
    - About interpretation of Data
  - **Validation**: did we learn the model correct?
  - **Evaluation**: did we learn the correct model?
- Rules of thumb for sound Machine Learning
  - Data: Check the quality of your data
  - Model: Law of Parsimony → **LAW OF PARSIMONY** The principle that the simplest Explanation of an event or observation is the preferred explanation
  - Validation
  - Evalation
    - measure effectiveness
    - Compare against baseline (sanity check)
- Effectiveness
  - the degree to which a learned model is successful in producing a desired result
- Efficiency
  - Computation time
  - Memory use

Evaluation Metrics

- Regression/continuous scale
  - Mean Squared Error

$$\frac{1}{m}\sum_{i=1}^{m}(\hat{y}-y)^2$$

- Reduces variance to data points
- Distant points are more important

- Root Mean Squared Error

$$\sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{y}-y)^2}$$

- More explainable than MSE because it is on the same scale as the target variable

---

- Mean Absolute Error

$$\frac{1}{m}\sum_{i=1}^{m}|\hat{y}-y|$$

- Is more tolerant to outliers

- Regression/continous scale

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(\hat{y}-y)^2}{\sum_{i=1}^{m}(y-\bar{y})^2}$$

Percentage of variance in the target variable that is explained by the features