Artificial neural networks and TensorFlow

Agenda

- Installation of TensorFlow
- Artificial neural networks - theory
- Work with TensorFlow and MNIST - practice

Installation

Installation

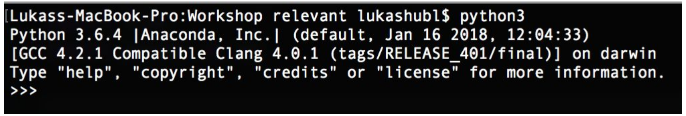
- python3: run the installation file
- pandas: pip install pandas
- matplotlib: pip install matplotlib
- tensorflow:
  - MAC: pip install tensorflow-1.7.0rc1-py3-none-any.whl
  - WIN: pip install tensorflow-1.7.0rc1-cp36-cp36m-win_amd64.whl
  - Ubuntu: pip install tensorflow-1.7.0rc1-cp36-cp36m-linux_x86_64.whl
  - Generell: pip install tensorflow

Setup check

Python 3.5.6

Command: “python3” respectively “python”

Output:



```
[Lukass-MacBook-Pro:Workshop relevant lukashubl$ python3
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 12:04:33)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

If you have further questions, feel free to ask me

MacOS, Windows & Linux

TensorFlow

Download: https://github.com/lukashubl/ann_tf_ws/blob/master/setupCheck.py

Command: “python3 setupCheck.py”

Output: The script will tell you if TensorFlow is installed properly on your machine

MacOS, Windows & Linux

Theory

Basics

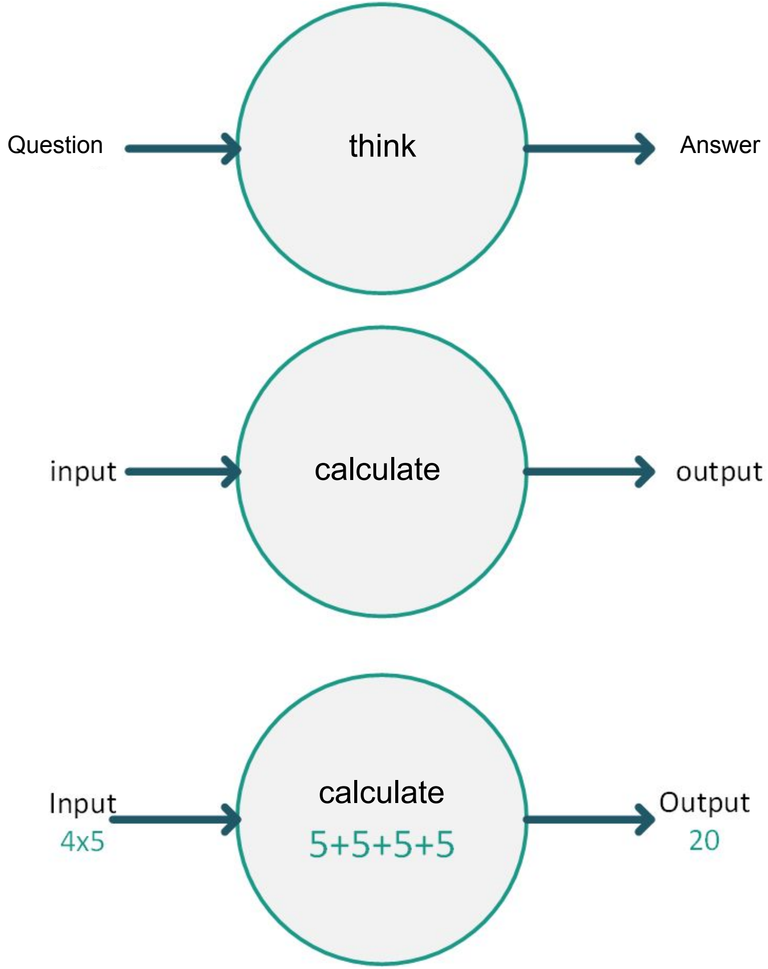
What is an artificial neural network?

Why should we use artificial neural networks?

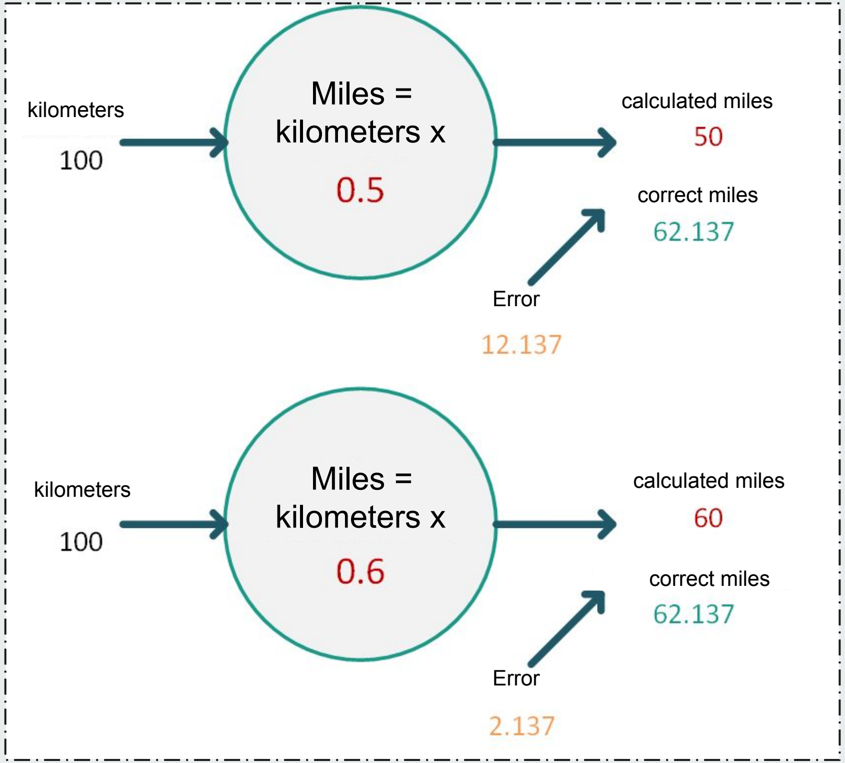
How can we develop them?

Comparison

Human vs Computer



Simple Predictor



kilometers

100

Miles = kilometers x 0.5

calculated miles 50

correct miles 62.137

Error 12.137

kilometers

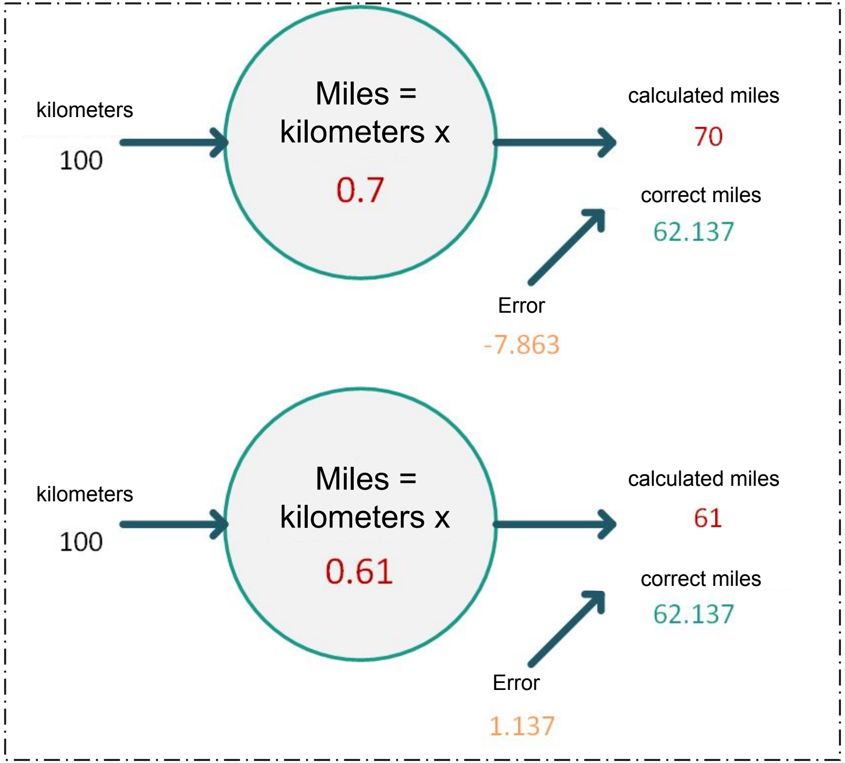
100

Miles = kilometers x 0.6

calculated miles 60

correct miles 62.137

Error 2.137



kilometers

100

Miles = kilometers x 0.7

calculated miles 70

correct miles 62.137

Error -7.863

kilometers

100

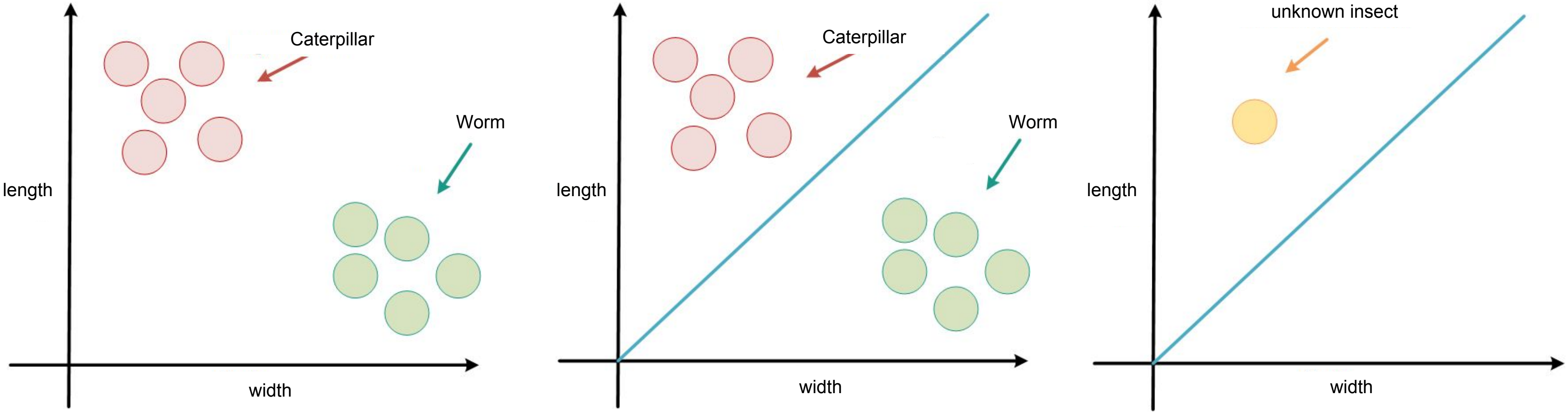
Miles = kilometers x 0.61

calculated miles 61

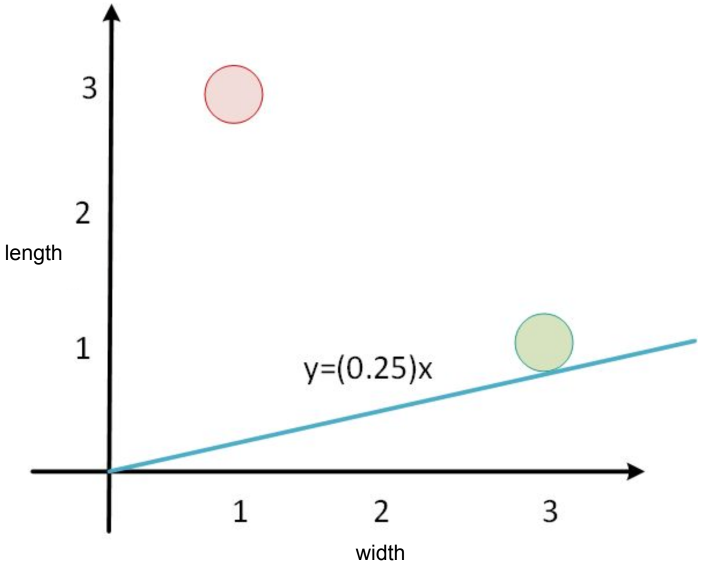
correct miles 62.137

Error 1.137

Simple classifier



Simple classifier



x	y	Type
1	3	Worm
3	1	Caterpillar

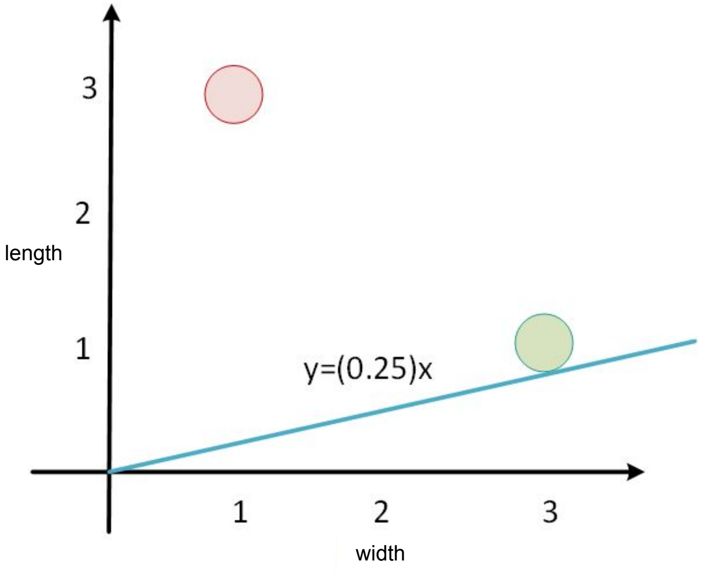
y=Ax

y=0.25x

y=(0.25)*(3.0)=0.75

error=(correct result - function output)
E=1.1 - 0.75 = 0.35

Simple classifier



$y=Ax$

$targetY=(A+\Delta A)x$

$targetY-actualY=(A+\Delta A)x -Ax$

$E=targetY-actualY$

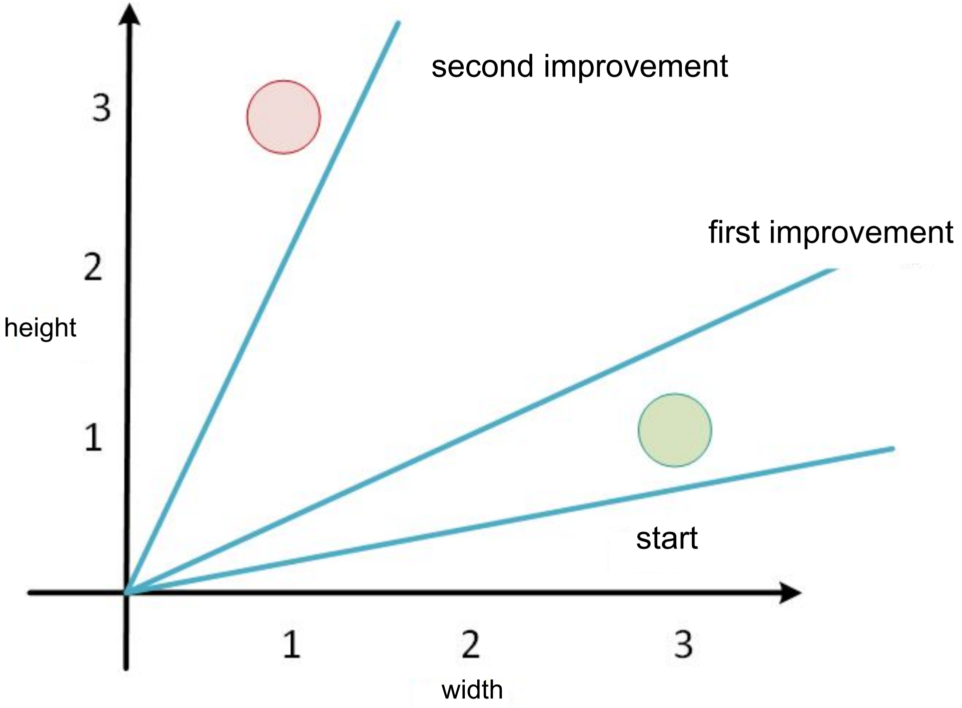
$E=Ax+(\Delta A)x -Ax$

$E=(\Delta A)x$

$\Delta A=E/x$

Simple classifier

x	y	Type
1	3	Worm
3	1	Caterpillar



ΔA = E / x

ΔA = 0.35/3 = 0.1167

y = 0.3667x

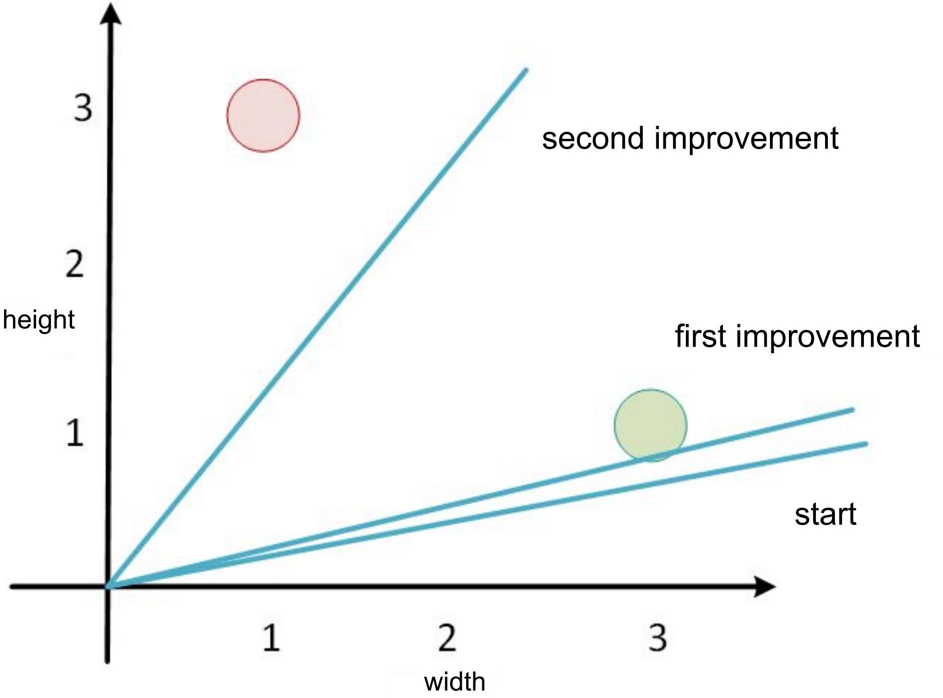
y = 0.3667 * 1.0 = 0.3667

E = 2.9 - 0.3667 = 2.5333

ΔA = 2.5333 / 1.0 = 2.5333

y = 2.9x

Simple classifier



x	y	Type
1	3	Worm
3	1	Caterpillar

Learning rate L

ΔA = (E / x) L

L = 0.5

ΔA = (0.35 / 3) 0.5 = 0.0583

y = 0.3083x

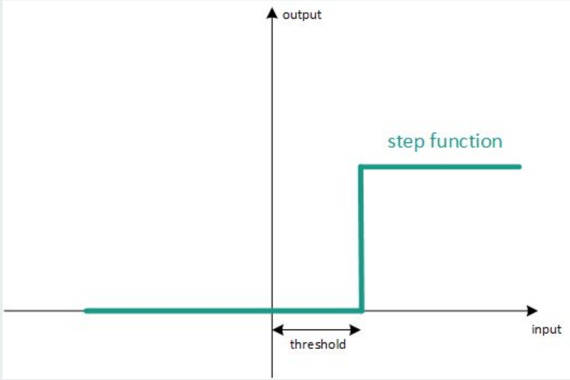
y = 0.3083 * 1.0 = 0.3083 -> E = 2.5917

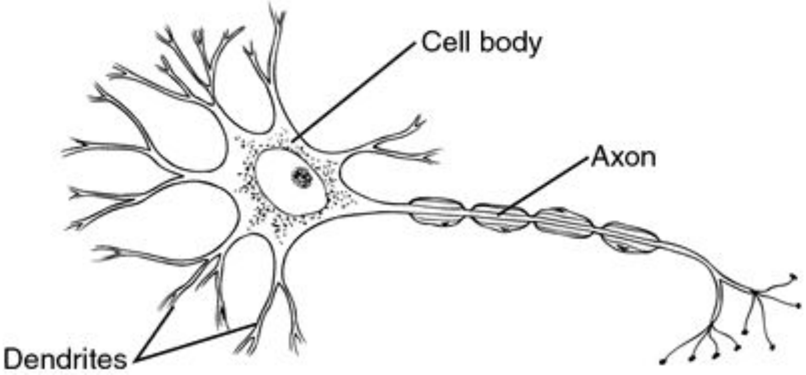
ΔA = (2.5917 / 1.0) * 0.5 = 1.2958

y = 2.9x

Biological neural network

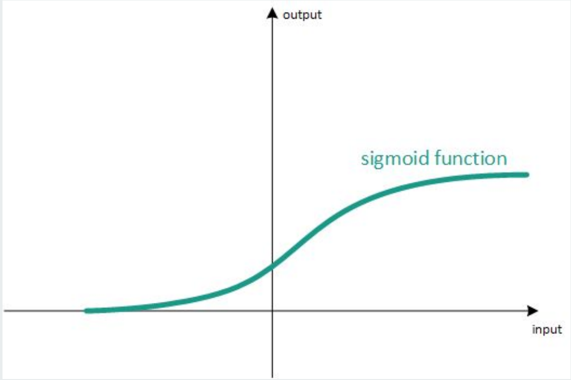
- Dendrites = recipient
- Axon = sender
- Cell body = decision maker
- Threshold exceeded -> Axon is firing
- Axon and other neurons are connected via synapses

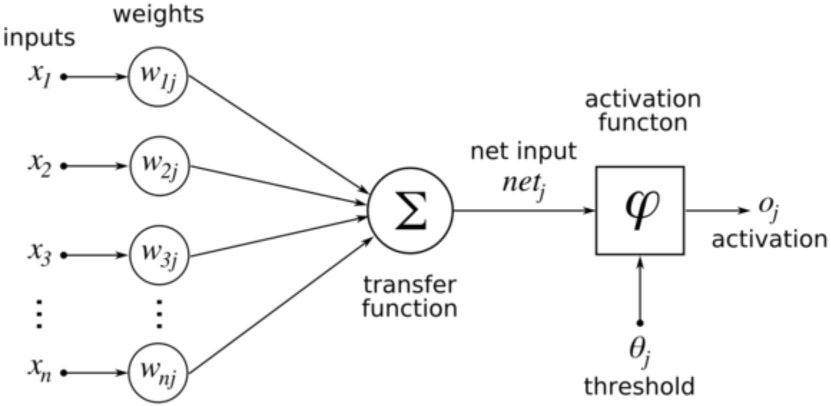


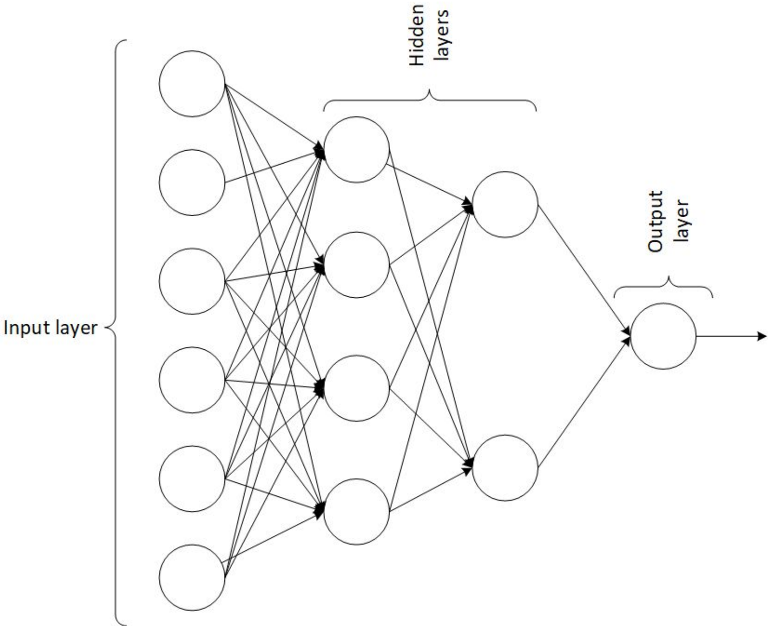


Biological neural network

- Weighted inputs = Dendrites
- Activation function = Threshold
- Output = Axon
- The sum of all weighted inputs is used as the activation functions input







Simple artificial neural networks

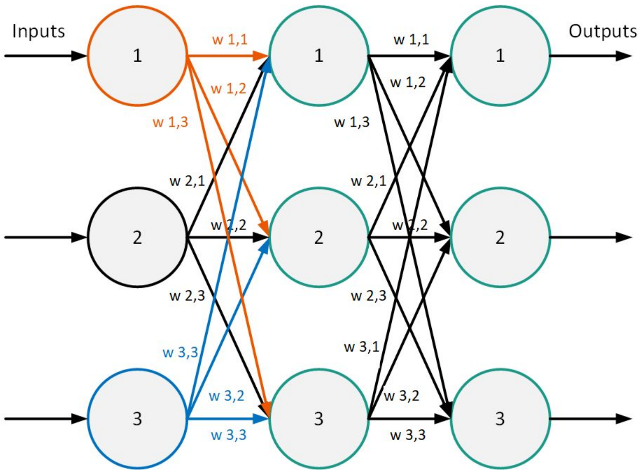
- Each node of a layer is connected with every node of the next layer
- Leads to simple matrix multiplications
- Signal gets passed through the network

$$X_{hidden} = W_{input\_hidden} \cdot I$$

$$X_{hidden} = \begin{pmatrix} 0.9 & 0.3 & 0.4 \\ 0.2 & 0.8 & 0.2 \\ 0.1 & 0.5 & 0.6 \end{pmatrix} \cdot \begin{pmatrix} 0.9 \\ 0.2 \\ 0.1 \end{pmatrix}$$

$$X_{hidden} = \begin{pmatrix} 1.16 \\ 0.42 \\ 0.62 \end{pmatrix}$$

$$O_{hidden} = sigmoid \begin{pmatrix} 1.16 \\ 0.42 \\ 0.62 \end{pmatrix} = \begin{pmatrix} 0.761 \\ 0.603 \\ 0.650 \end{pmatrix}$$



w 1,1	w 1,2	w 1,3
w 2,1	w 2,2	w 2,3
w 3,1	w 3,2	w 3,3

0.9	0.3	0.4
0.2	0.8	0.2
0.1	0.5	0.6

Inputs
0.9
0.1
0.8

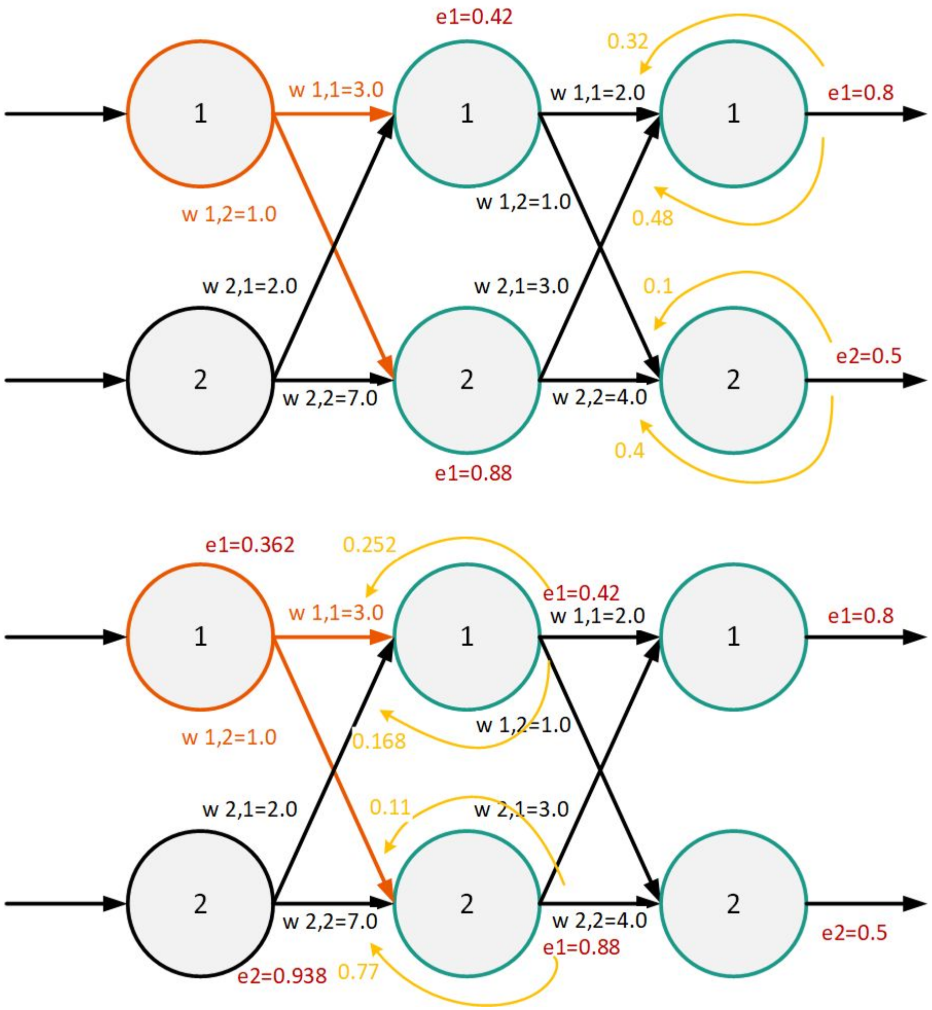
Simple artificial neural networks

- Error backpropagation

- Split up the error based on the weights
- Sum up the errors

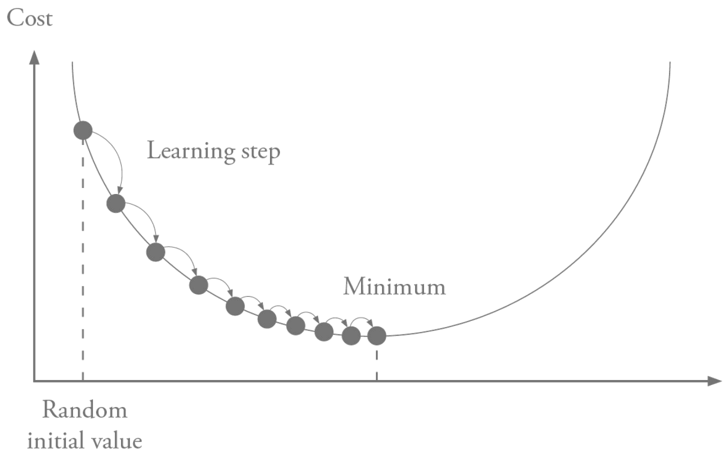
- Error backpropagation to previous layers

- Don’t panic - TensorFlow is supporting :-)



Updating weights

- Most complicated part
- Step by step
- Evaluation via the gradient Δ
- Again - thankfully there's TensorFlow



$$\frac{\delta E}{\delta w_{jk}} = -(t_k - O_k) \cdot afunc(\sum w_{jk} \cdot O_j)(1 - afunc(\sum w_{jk} \cdot o_j)) \cdot o_j$$

$$w_{jk\,neu} = w_{jk\,old} - \alpha \cdot \frac{\delta E}{\delta w_{jk}}$$

$$\Delta w_{jk} = \alpha \cdot E_k \cdot O_k(1 - O_k) \cdot O_j^T$$

TensorFlow-Workshop

Agenda

Getting used to MNIST data

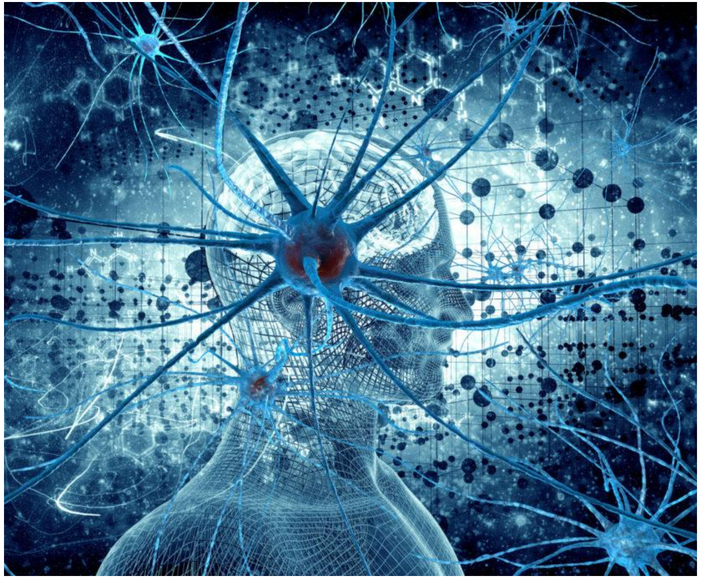
Creating a model

Defining the assessment criteria

Training the model

Evaluating the results

Visualizing and reusing the model



MNIST classification in TensorFlow

MNIST

Model

Assessment

Training

Evaluation & Visualisation

Real-time application

MNIST

- What is MNIST
- MNIST data structure
- Code

MNIST database

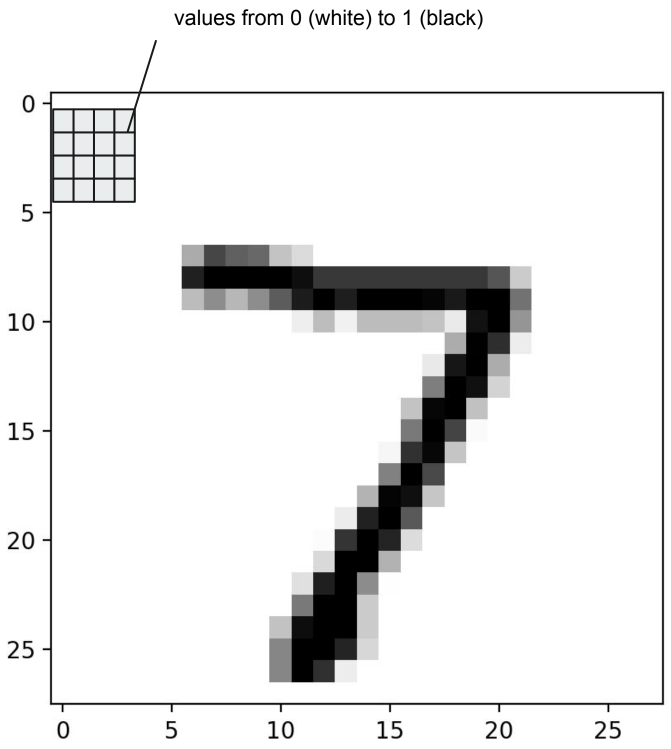
- Modified Institute of Standards and Technology
- Images of handwritten digits
- 60.000 training-records
- 10.000 test-records

MNIST database

Why are test- and training-records separated?

MNIST daabase

- Bitmap
- 28 x 28 pixels
- Values from 0 to 1
- Jedes Bild hat dazugehöriges Label



Coding time

MNIST Klassifizierung in TensorFlow

MNIST

Model

Assessment

Training

Evaluation & Visualisation

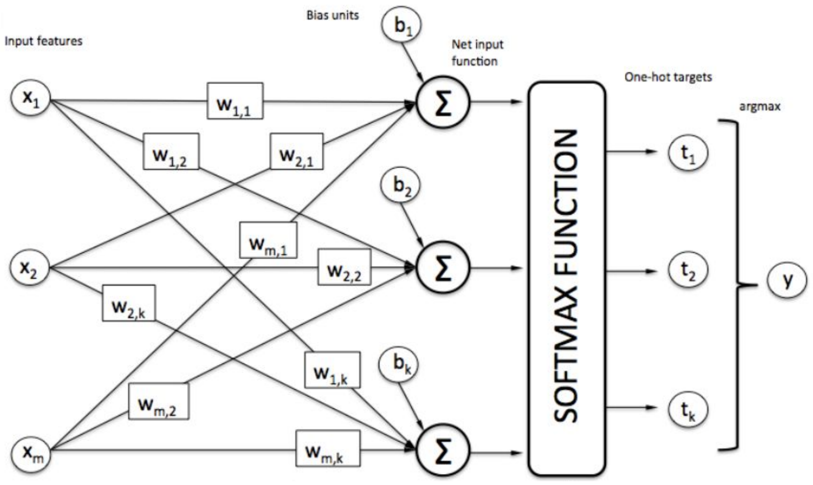
Real-time application

Modell

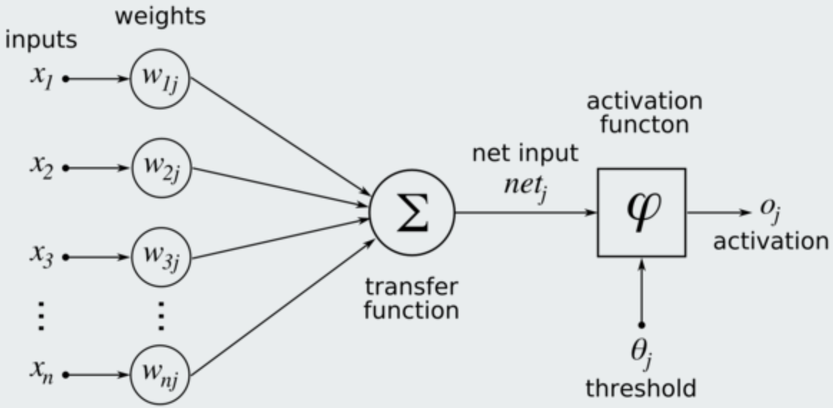
- The model’s job
- Softmax
- Code

Model

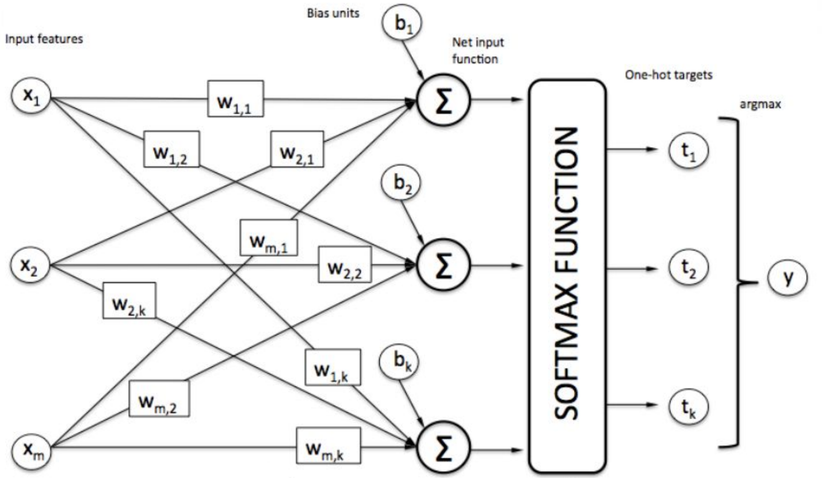
- provides output based on input
- has to be trained
- Logistic regression
- Softmax regression



Model



Logistic regression

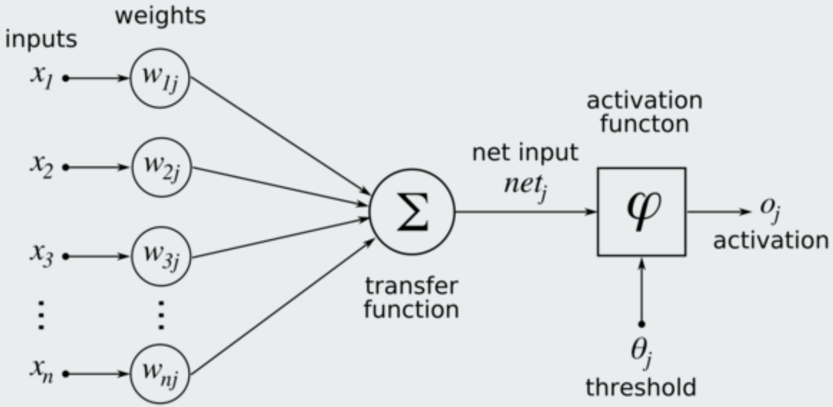


Softmax regression

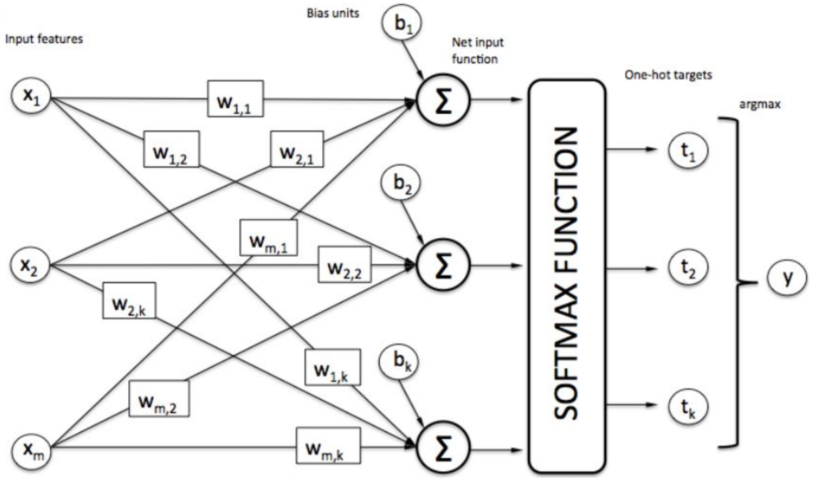
Model

What’s the main difference between these two types of models?

Model



Logistic regression



Softmax regression

Coding time

MNIST Klassifizierung in TensorFlow

MNIST

Model

Assessment

Training

Evaluation & Visualisation

Real-time application

Assessment criteria

- What’s the value we have to optimize
- Cross-entropy
- Code

Assessment criteria

- How can the weights be optimized?
- Which value is deciding?
- Necessary in order to be able to learn
- Cross-entropy

Cross-entropy

- Difference between the real distribution function and the one from the model
- Cross-entropy as deciding value
- When we are able to minimize that value the prediction equals the label - result is correct

Example: digit 7

0 0 0 0 0 0 1 0 0 Label
0 0 0 0 0 0 1 0 0 Prediction

$-[0 \times \ln(0) + 0 \times \ln(0) + \dots + 1 \times \ln(1) + \dots] = 0$ Error

0 0 0 0 0 0 0 1 0 0 Label
,1 ,1 ,1 ,1 0 0 0 ,6 0 0 Prediction

$-[4 \times 0 \times \ln(0{,}1) + 5 \times 0 \times \ln(0) + 1 \times \ln(0{,}6)] = 0{,}51$ Error

0 0 0 0 0 0 0 1 0 0 Label
1 0 0 0 0 0 0 0 0 0 Prediction

$-[0 \times \ln(1) + 8 \times 0 \times \ln(0) + 1 \times \ln(0)] = \infty$ Error

Coding time

MNIST Klassifizierung in TensorFlow

MNIST

Model

Assessment

Training

Evaluation & Visualisation

Real-time application

Training

- Gradient descent
- Parameter
- Code

Training

What we already achieved:

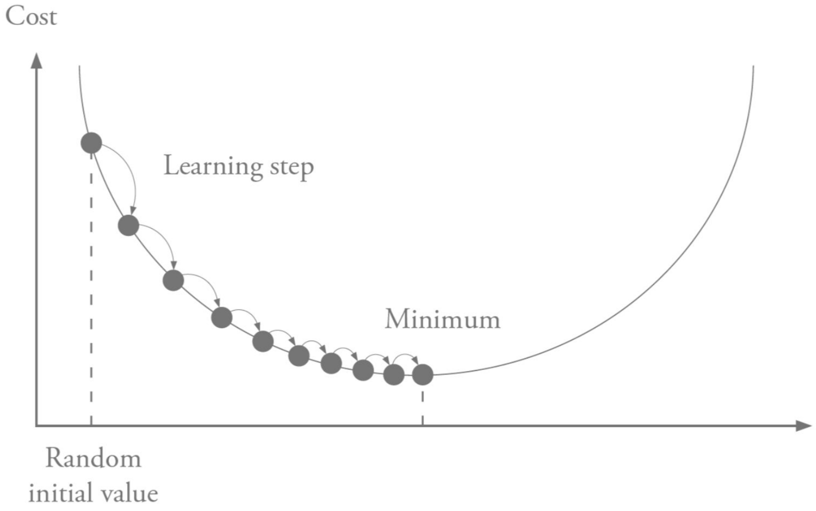
- Loading MNIST data
- Creating the model
- Assess the results

Next step:

- Improving the model by training

Gradient descent

- Example: Mountaineer
- Mountain → error-function (Cross-Entropy)
- Mountaineers position → error
- Step downwards→ learning-iteration



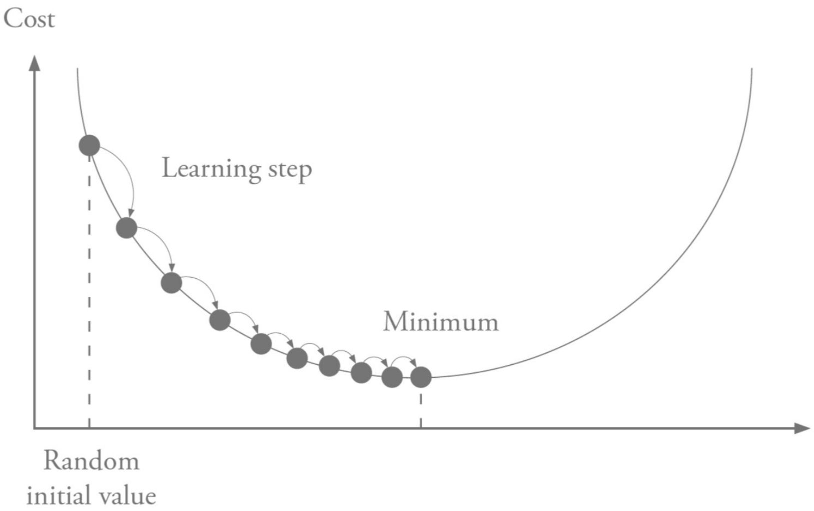
Initial weights

What do they precise?

- Mountaineer’s starting position
- Initial error we have to optimize

Learning rate

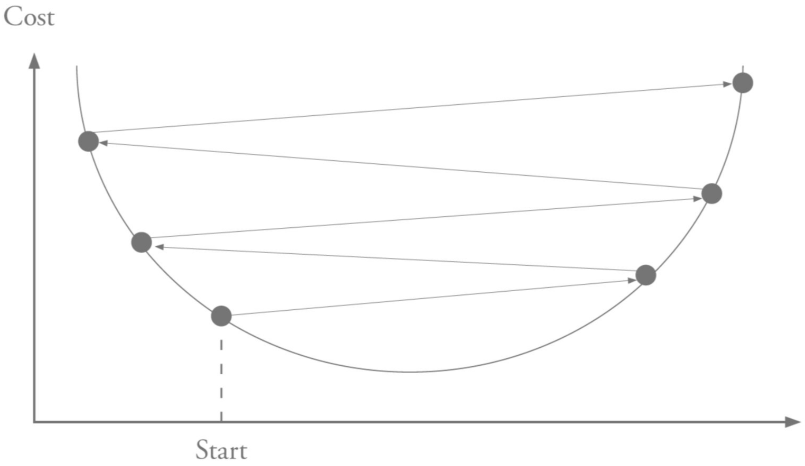
- Defines the speed of learning
- The mountaineer’s steps size

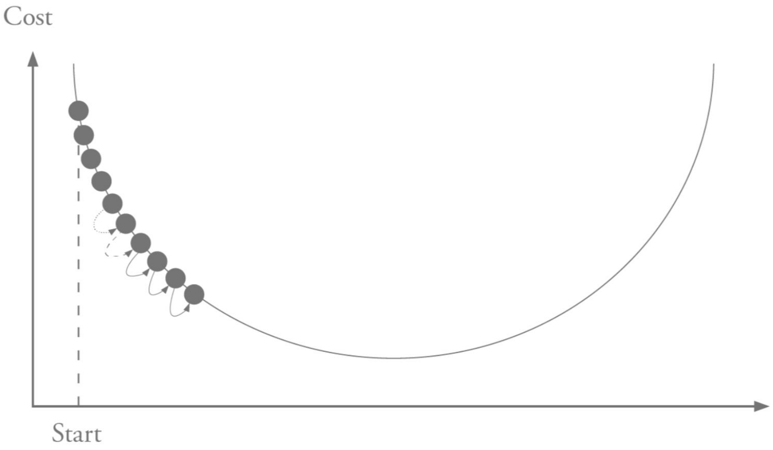


Learning rate

What happens if the learning rate is

- too big?
- too small?





Coding time

MNIST Klassifizierung in TensorFlow

MNIST
Model
Assessment
Training
Evaluation & Visualisation
Real-time application

Evaluation

- Accuracy
- TensorBoard
- Code

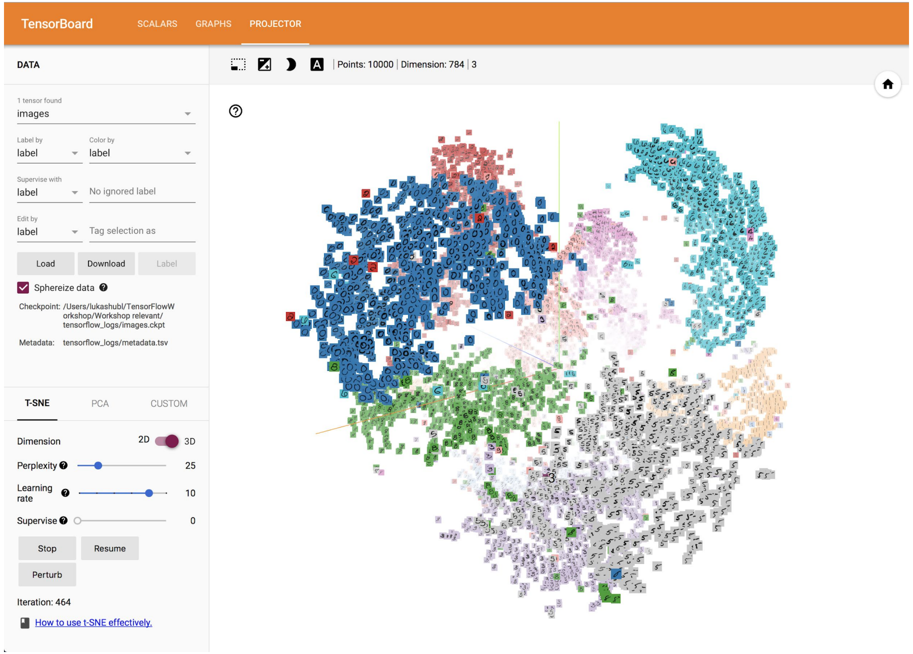
Evaluation

- Accuracy

- How many records are correnctly classified

TensorBoard

- Visualisations of all kinds
- Graphs
- Scalars
- Projectors



Points: 10000 | Dimension: 784 | 3

1 tensor found: images

Label by: label

Color by: label

Supervise with: label

No ignored label

Edit by: label

Tag selection as

☑ Sphereize data

Checkpoint: /Users/lukashubl/TensorFlowWorkshop/Workshop relevant/tensorflow_logs/images.ckpt

Metadata: tensorflow_logs/metadata.tsv

Dimension: 2D ● 3D

Perplexity: 25

Learning rate: 10

Supervise: 0

Iteration: 464

How to use t-SNE effectively.

Coding time

MNIST Klassifizierung in TensorFlow

MNIST → Model → Assessment → Training → Evaluation & Visualisation → Real-time application

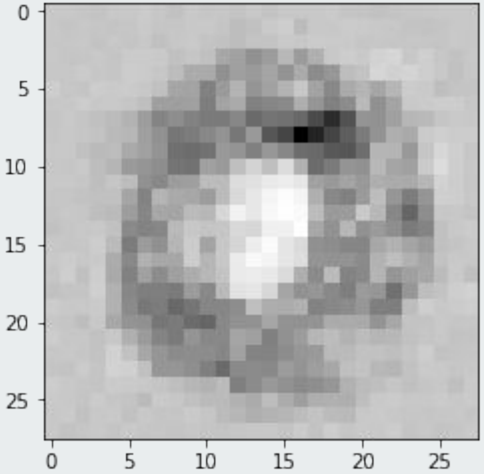
Deploying the model in a real-time application

- Saving the model
- Reusing the model
- Code

Coding time

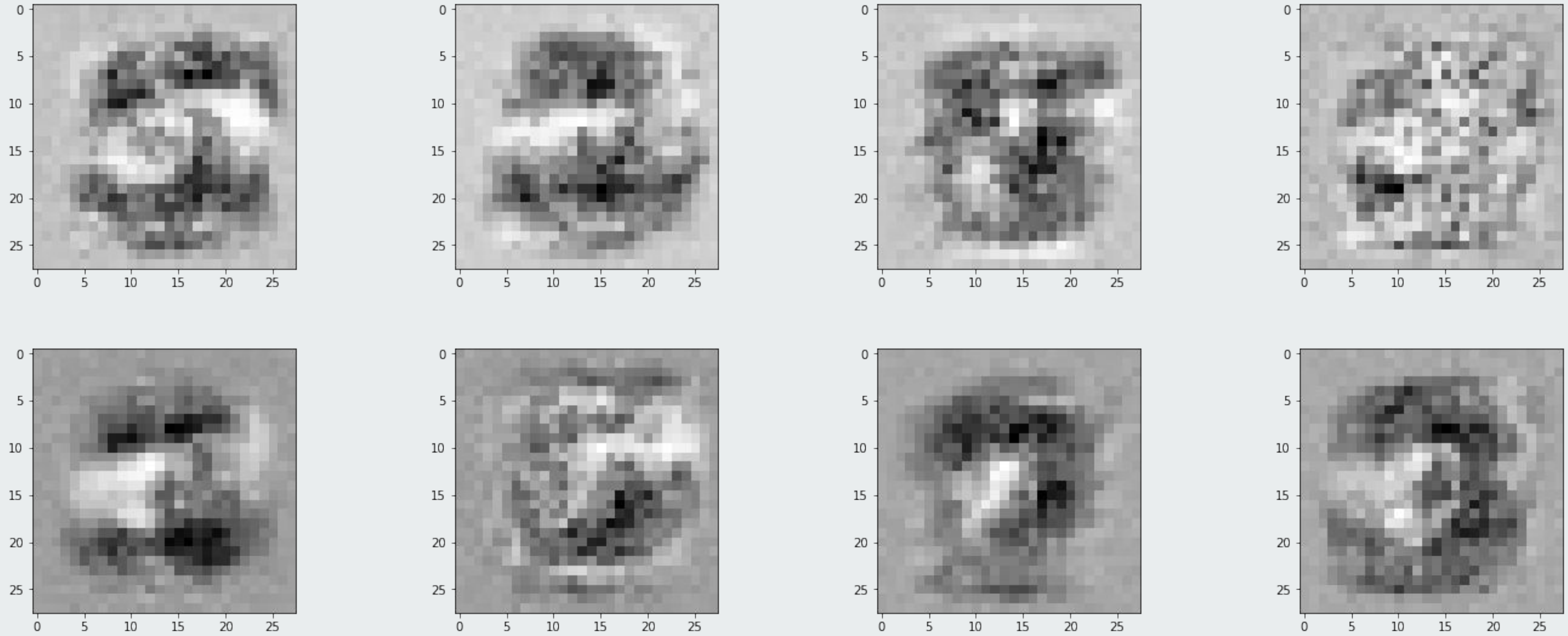
Thank you for your attention

Fun Facts

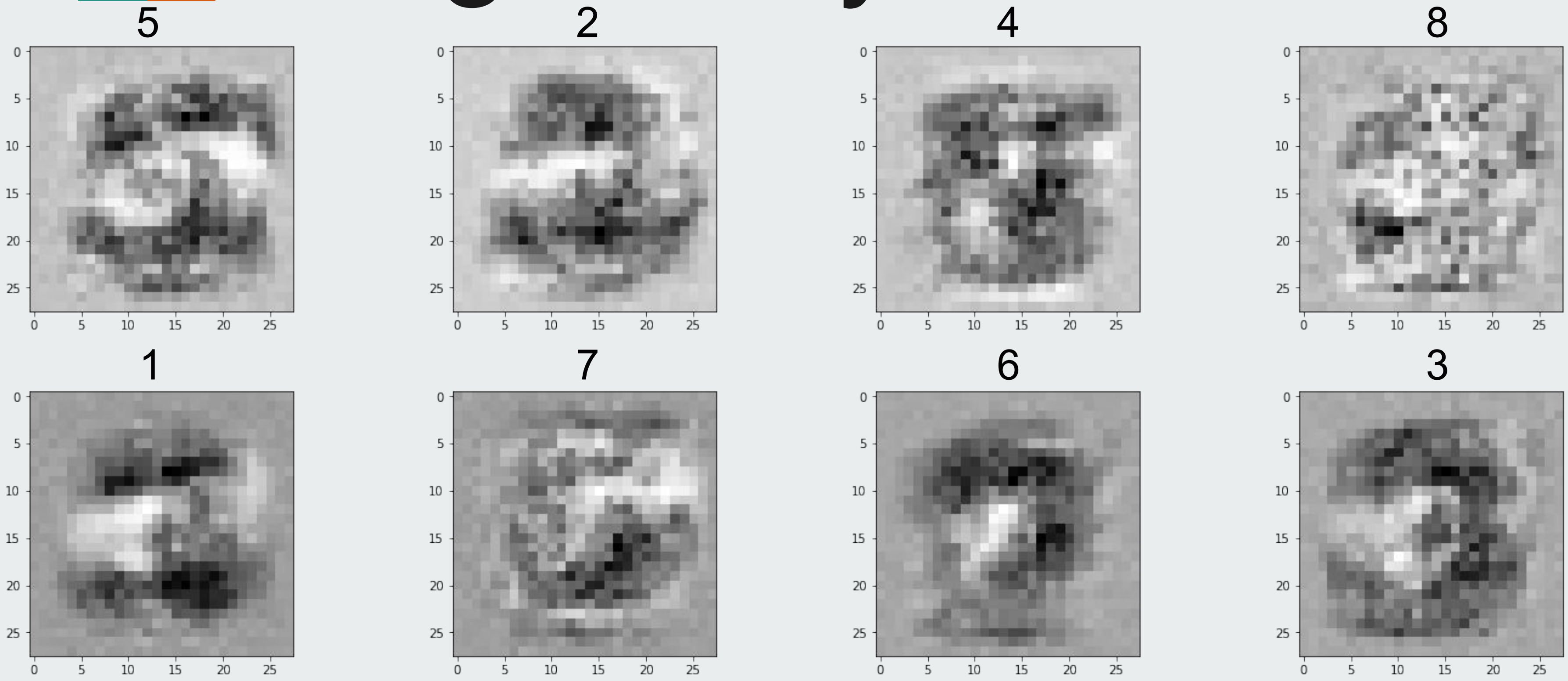


What digit is it?

What digits can you see?



What digits can you see?



In austria we would say: “Jetzt ist’s wirklich vorbei - Danke”