Deep Learning
SS 2023

Prof. Dr. Rainer Herrler
Phone: 09721/ 940-8710
Email: rainer.herrler@fhws.de

Pictures from Wikipedia / Pixabay
Some Pictures generated with Dall-E or Stable Diffusion

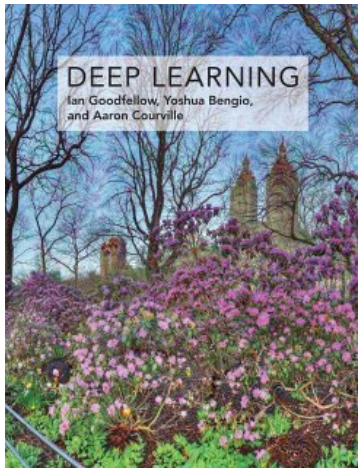
21

Literature

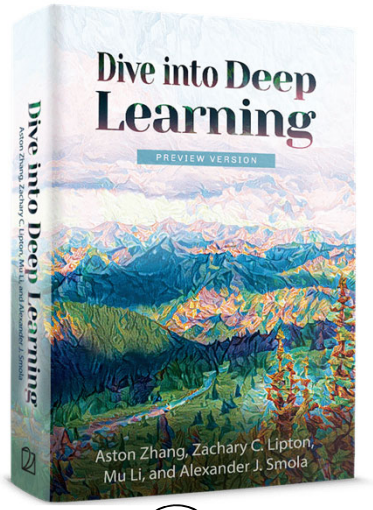
Deep Learning

1. Deep Learning
   Ian Goodfellow, Yoshua Bengio

2. Dive into Deep Learning
   https://d2l.ai/



1

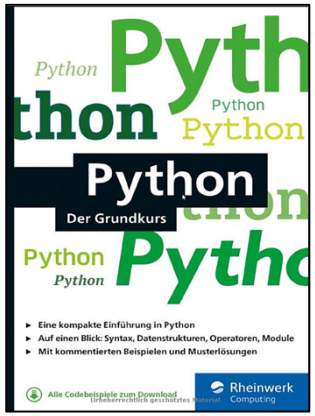


2

Python

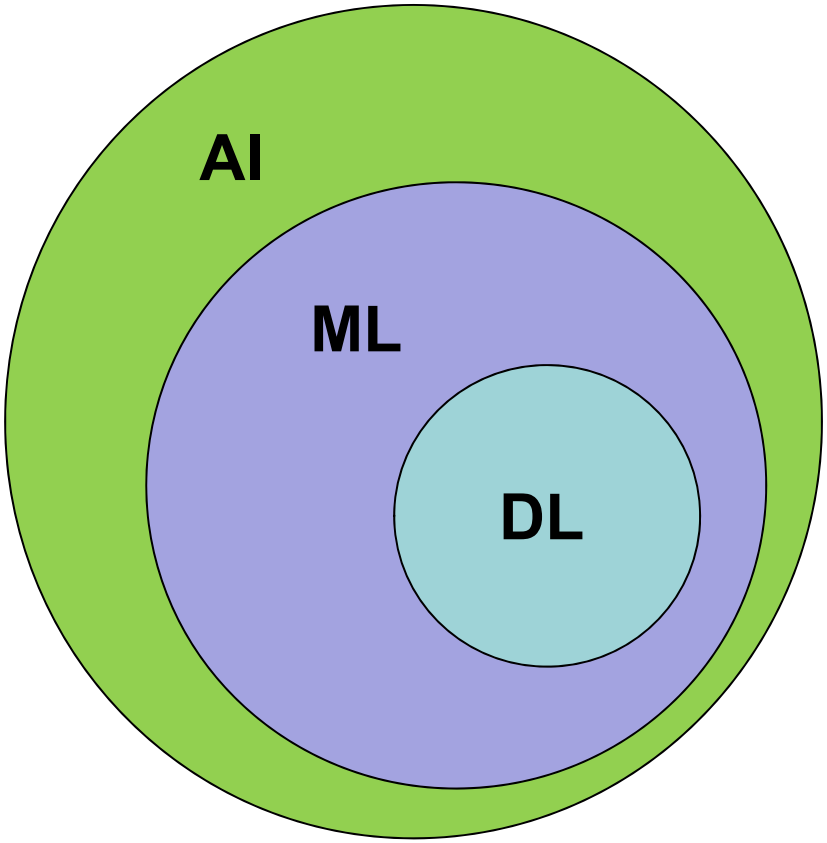
3. Additional book to learn python
   Michael Kofler

4. W3 Schools
   https://www.w3schools.com/python/



3

How does DL diifer from AI and ML ?



- AI (Artificial Intelligence)
  -
  -
  -
- ML (Machine Learning)
  -
  -
  -
- DL (Deep Learning)
  -
  -
  -

What is the plan for this Semester

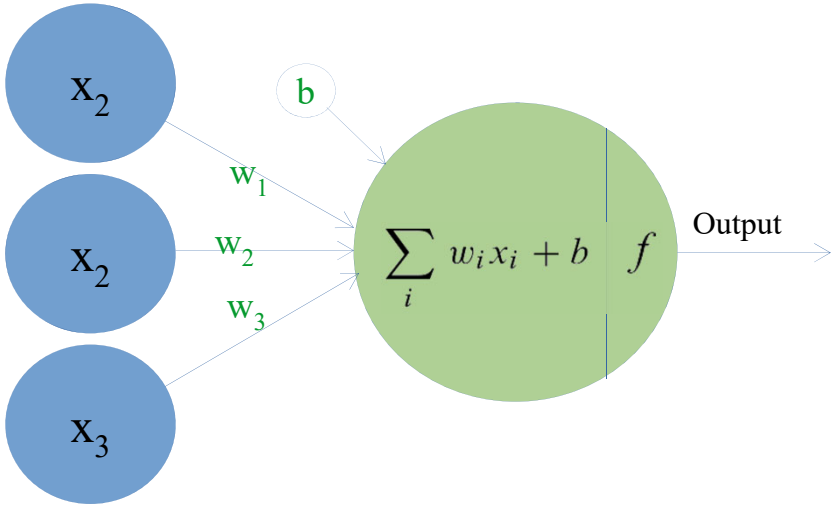
- We should get familiar in building neural networks with state of the art frameworks
- We learn how to create different network structures and when they are useful:
  - Recurrent Networks
  - Autoencoder Networks
  - Convolutional Neuronal networks
  - …
- We learn how to find and use existing trained models
- We learn how to fine tune existing models for own tasks
- We learn how to train models with GPU's or TPU's in the cloud.
- …

Short Repetition of SS2022 Machine Learning

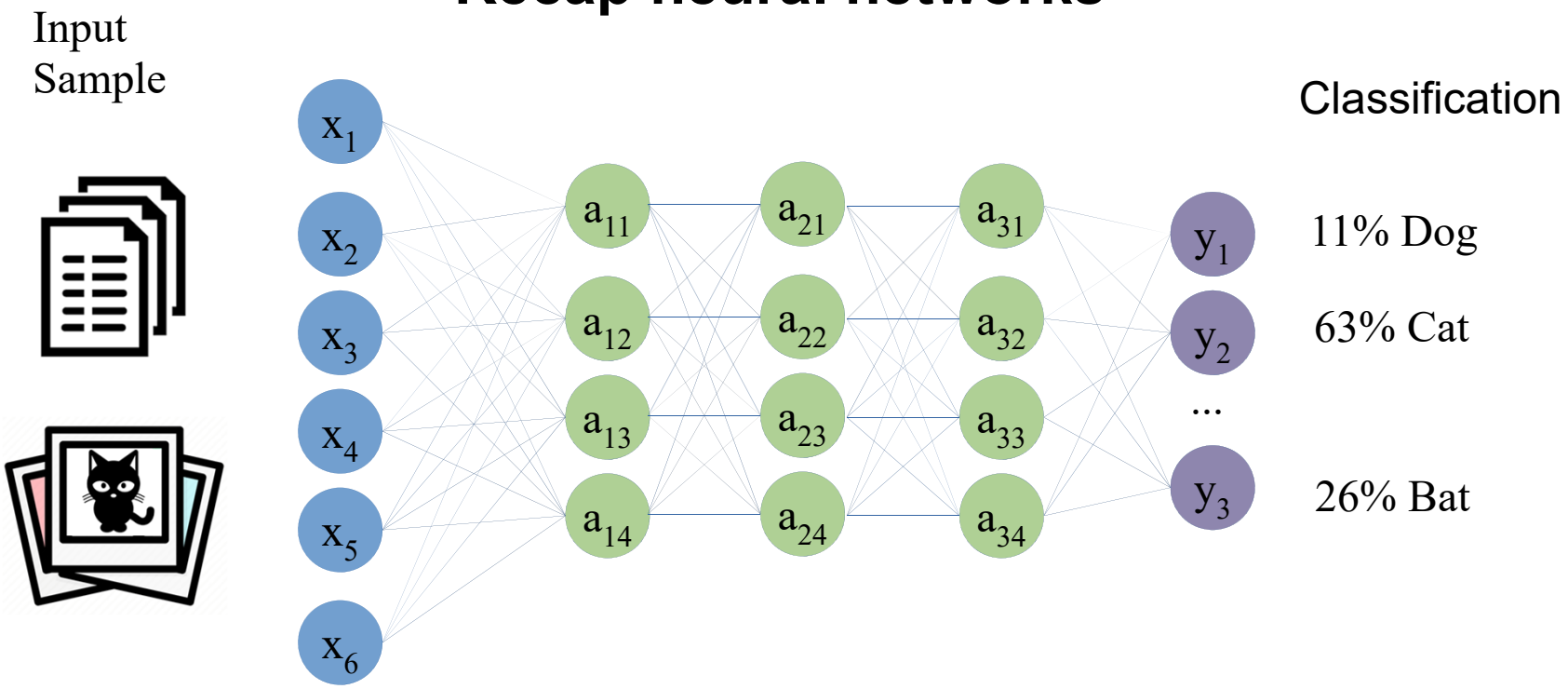
What should we remember from the ML Lecture

- Basic knowledge of Python/Conda/Jupyter
- Basic structure of supervised learning
- Difference between regression and classification
- Understanding of Overfitting / Underfitting
- How to work with training data
  - Normalisation
  - Splitting in Test Training and Validation Set
- How to evaluate models
  - Accuracy / Precision / Recall /Cooeficient of determination
- Idea of gradient descent
- …

Recap neural networks



Recap neural networks



Questions regarding the last slide

- Name all elements off the last slide!
- What formula do we have in the neurons?
- Which dimension does the weight matrix have in the first layer of the example?
- How many bias values are needed for the first layer ?
- What type of classification to we have here ?
- What possibilities for activation functions do we have ?
- Can a network be trained on 25*25 pixes and then do inference on 100*100 pixels ?

What is Forward and Backward Propagation?

- Forward Propagation
  - Done during Training and Application
  - Often called “predict”-Step or inference
  - Comparatively fast (one pass, one prediction)
- Backpropagation
  - Done during Training (fit-Step)
  - Includes Forward propagation
  - Important to adjust the weights
  - Many iterations needed
  - Computationally expensive

Loss and Score What is the Difference ?

- Loss
  - Describes how close we are to the ground truth value during training
  - Used to track prediction quality during training
  - the key value to minimize for out optimizer
  - Loss of a single training example, loss of a batch, loss of the whole training set (usually averaged)
  - Different losses common AE, MSE, LogLoss, HingeLoss
- Score
  - Describes how good our predictions are on either test/training or validation set
  - Common Score Functions: Accuracy, Precision, Coefficient of determination (Bestimmtheitsmaß)

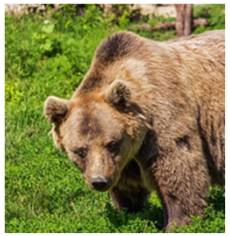
Loss functions in detail

- Multinomial SVM Loss (new)
- Softmax + LogLoss

Loss Functions

Suppose we have a neural network with a ReLU or Identity function as the last layer and get the following results.





Cat	3.2	1.3	2.2
Mouse	5.1	4.9	2.5
Bear	-1.7	2.0	-3.1

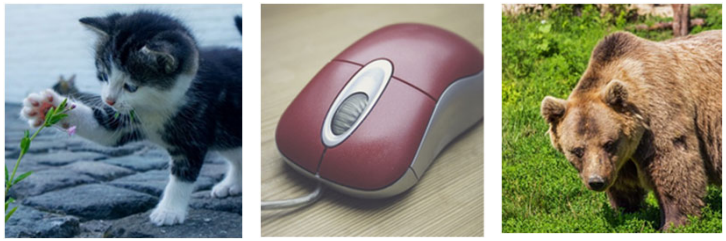
N Training examples
K Classes $\{1,2,..k\}$
$x_i \in X$ Training example
y_i true label with range K

Selection of predicted class by argmax for each result vector.

How good is that prediction ?

Loss Functions – Multiclass SVM Loss

Suppose we have a neural network with a ReLU or Identity function as the last layer and get the following results.



Cat	**3.2**	1.3	2.2
Mouse	5.1	**4.9**	2.5
Bear	-1.7	2.0	**-3.1**

N Training examples
K Classes $\{1,2,..k\}$
$x_i \in X$ Training example
y_i true label with range K
$s = h_\Theta(x_i)$ Score vector

Loss over all training examples

$$L = \frac{1}{\mathrm{N}} \sum_i L_i(h_\Theta(x_i), y_i)$$

Loss of a single Training example

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Applying the loss formula to the cat prediction



Cat	**3.2**	1.3	2.2
Mouse	5.1	**4.9**	2.5
Bear	-1.7	2.0	**-3.1**
Loss			

The blue frame corresponds to the prediction for the cat image $h_\Theta(x_i)$
For the cat training sample The prediction value for cat should be highest.
So all other prediction values are compared to it they should be smaller with a distance of 1 to not get a loss.

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$= \max(0, 5.1 - 3.2 + 1)$ // loss for telling that mouse is more likely than cat
$+ \max((0, -1.7 - 3.2 + 1)$ // no loss for giving bear a small value
$= \max(0,2.5) + \max(0, -3.9)$
$= 2.9$

Applying the loss formula to the mouse image



Cat	**3.2**	1.3	2.2
Mouse	5.1	**4.9**	2.5
Bear	-1.7	2.0	**-3.1**
Loss	2.9		

For the mouse image the predictions are consistent with expected results. The mouseclass has the highest value and other values follow with a safety margin. There should be no loss

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$= \max(0, 1.3 - 4.9 + 1) + \max(0, 2.0 - 4.9 + 1)$$
$$= \max(0, -2.6) + \max(0, -1.9)$$
$$= 0$$

Applying the loss formula to the bear image

			
Cat	**3.2**	1.3	2.2
Mouse	5.1	**4.9**	2.5
Bear	-1.7	2.0	**-3.1**
Loss	2.9	0.0	

For prediction for the bear class is completely screwed up. Cat and Mouse have higher values. Lets see what the loss calculation says.

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$= \max(0, 2.2 - (-3.1) + 1) + \max(0, 2.5 - (-3.1) + 1)$$
$$= \max(0, 6.3) + \max(0, 6.6)$$
$$= 12.9$$

Taking the mean of all losses (full batch)



Cat	**3.2**	1.3	2.2
Mouse	5.1	**4.9**	2.5
Bear	-1.7	2.0	**-3.1**
Loss	2.9	0.0	12.9

$$L = \frac{1}{\mathrm{N}} \sum_i L_i(h_\theta(x_i), y_i)$$

$$= \frac{1}{3}(2.9 + 0.0 + 12.9)$$

$$= \mathbf{5.27}$$

SVM Loss - Check your understanding



Cat	**3.2**	1.3	2.2
Mouse	5.1	**4.9**	2.5
Bear	-1.7	2.0	**-3.1**
Loss	2.9	0.0	12.9

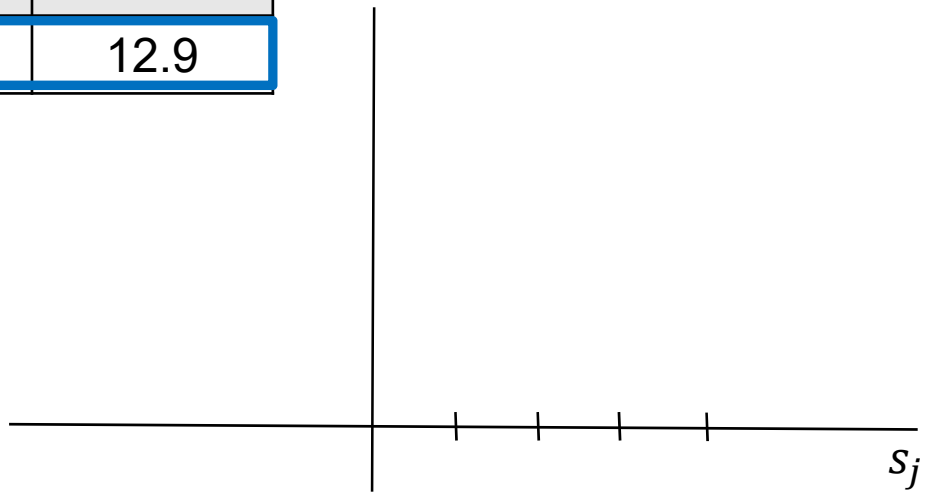
- What happens if the scores of mouse are modified just a little ?
- What is the range of the loss ?

SVM Loss – Why also called Hinge loss



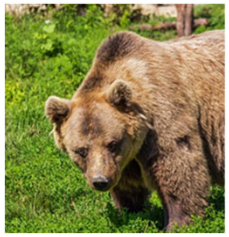
Cat	**3.2**	1.3	2.2
Mouse	5.1	**4.9**	2.5
Bear	-1.7	2.0	**-3.1**
Loss	2.9	0.0	12.9

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$



Softmax + LogLoss





Cat	3.2	1.3	2.2
Mouse	5.1	4.9	2.5
Bear	-1.7	2.0	-3.1

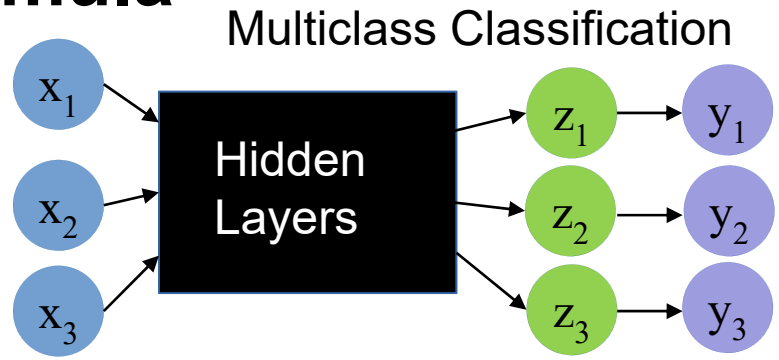
N Training examples
K Classes $\{1,2,..k\}$
$x_i \in X$ Training example
y_i true label with range K

These unnormalized values and SVM loss also work for optimizing the network, but they are not very intuitive. Instead we prefer something like this

Cat	0.13	0.03	0.42
Mouse	0.87	0.92	0.57
Bear	0.00	0.05	0.00

How do we get these probabilities?

Calculating probabilities with the softmax formula

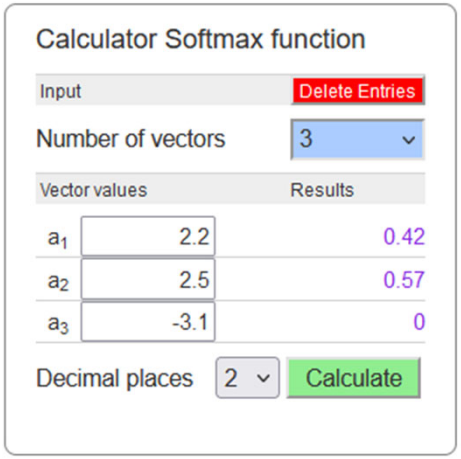


Softmax Activation:

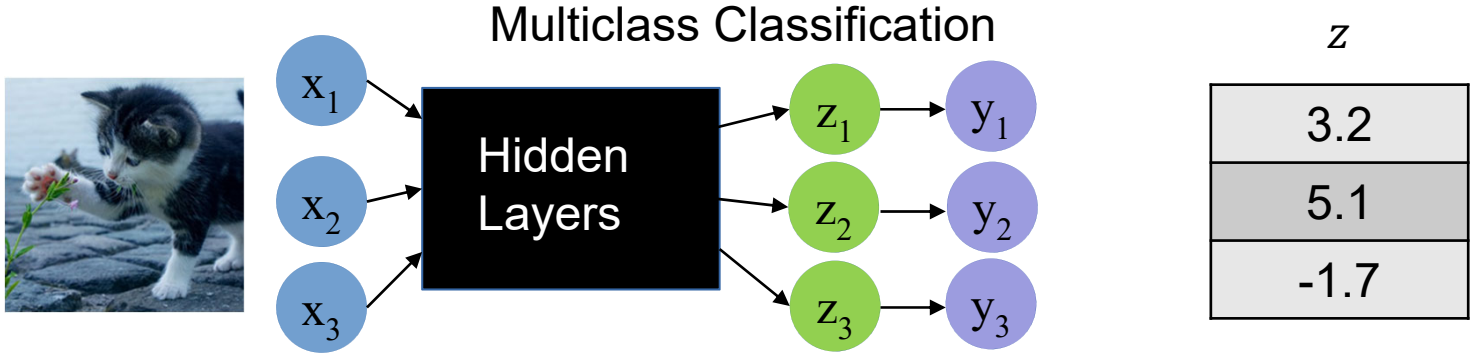
$$softmax(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$$

$$z_i \in [-\infty, \infty]$$

https://www.redcrab-software.com/en/Calculator/Softmax



Calculating probabilities with the softmax formula



The scores z are something like the “unnormalized log probabilities” because

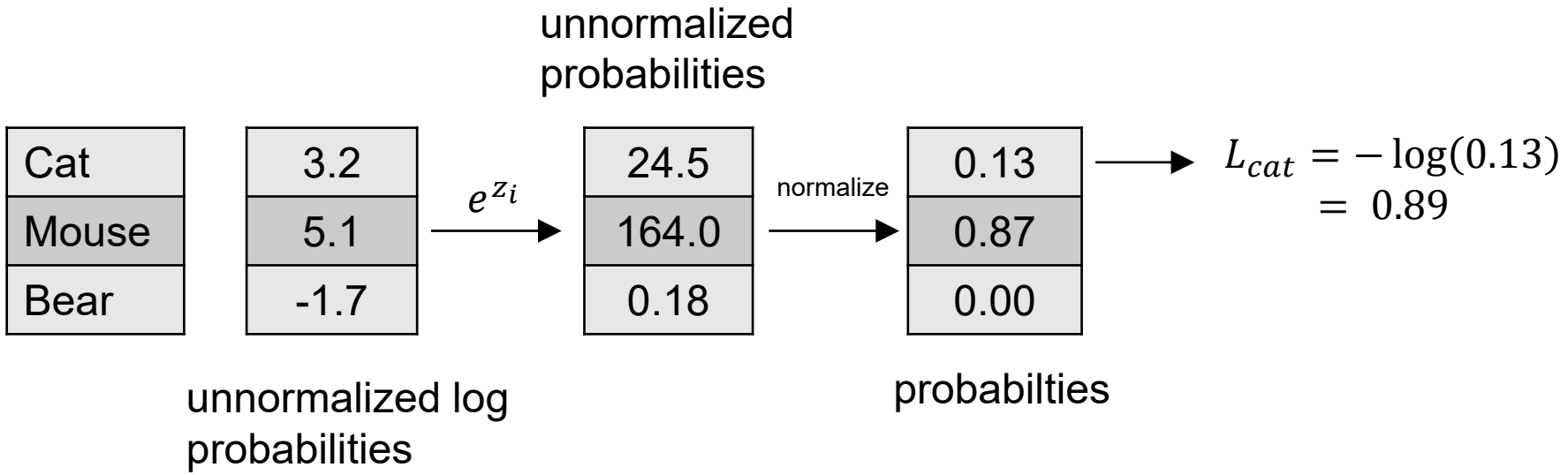
$$P(Y = i | X = x) = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$$

Since we want to maximize the probability for the true class y, that corresponds to minimize the negative log of the probability of the correct class

$$L_i = -\log(P(Y = y_i | X = x_i))$$

Calculating the loss

$$L_{cat} = -\log\left(\frac{e^{z_{cat}}}{\sum_{k=1}^{K} e^{z_k}}\right)$$



Check your understanding

$$L_{cat} = -\log\left(\frac{e^{z_{cat}}}{\sum_{k=1}^{K} e^{z_k}}\right)$$

Questions:

- What is the range of the loss that we can get for a single class?
- What happens if the scores of a class are modified just a little ?

Loss + Regularization

- In the previous we had the following Loss Function

$$L = \frac{1}{\mathrm{N}} \sum_{i} L_i(h_\Theta(x_i), y_i)$$

- Compare to a typical formula that we defined last year:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x[i]), y[i])$$

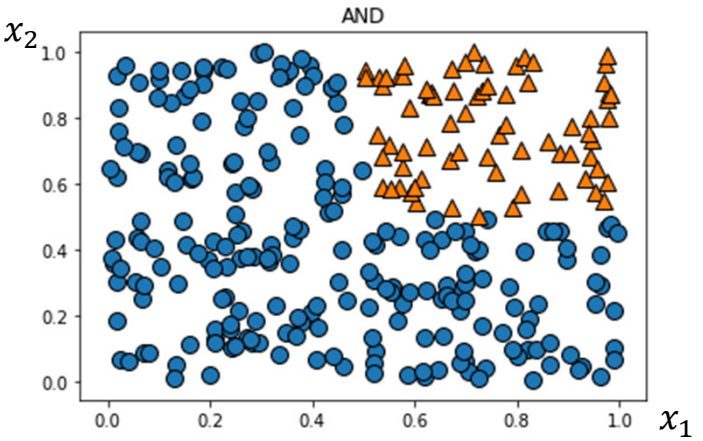
How does regularization work ?

How does regularization work

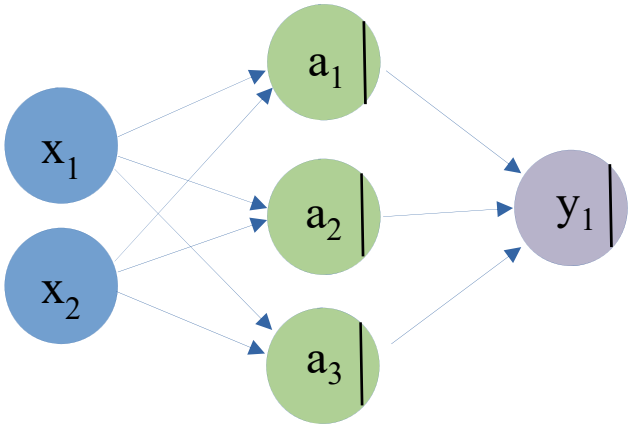
θ_0	θ_1	$\|\theta\|_1$	$\|\theta\|_2^2$
4	0	4	16
2	2	4	8
1	3	4	10
-1	5	6	26

- Cost function just including “data loss”
  - $J(\Theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\Theta(x[i]), y[i])$
- Regularization with L2-Norm
  - $J(\Theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\Theta(x[i]), y[i]) + \alpha \cdot \|\theta\|_2^2$
- Regularization with L1-Norm
  - $J(\Theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\Theta(x[i]), y[i]) + \alpha \cdot \|\theta\|_1$

Forward propagation with a simple example



$x_1 > 0.5$ and $x_2 > 0.5$



$$a = f(\Theta_1 \cdot x)$$

$$y = f(\Theta_2 \cdot a) = f(\Theta_2 \cdot f(\Theta_1 \cdot x))$$

Updating anaconda

Setting up a anaconda environment
If not installed any more please go to: https://www.anaconda.com

If still installed you should update to the newest version

1. conda update conda
2. conda update anaconda
3. conda create –n DL python=3.10 jupyter numpy
4. conda activate DL
5. jupyter notebook

```
    Uninstalling importlib-metadata-1.7.0:
      Successfully uninstalled importlib-metadata-1.7.0
Successfully installed absl-py-0.15.0 astunparse-1.6.3 cached-property-1.5.2 cachetools-4.2.4 clang-5.0 flatbuffers-1.12
 gast-0.4.0 google-auth-1.35.0 google-auth-oauthlib-0.4.6 google-pasta-0.2.0 grpcio-1.48.2 h5py-3.1.0 importlib-metadata
-4.8.3 keras-2.6.0 keras-preprocessing-1.1.2 markdown-3.3.7 numpy-1.19.5 oauthlib-3.2.2 opt-einsum-3.3.0 protobuf-3.19.6
 pyasn1-0.4.8 pyasn1-modules-0.2.8 requests-oauthlib-1.3.1 rsa-4.9 tensorboard-2.6.0 tensorboard-data-server-0.6.1 tenso
rboard-plugin-wit-1.8.1 tensorflow-2.6.2 tensorflow-estimator-2.6.0 termcolor-1.1.0 wheel-0.37.1 wrapt-1.12.1

(DL) C:\>
```

Whats happening behind the scenes?

- “Conda create –n [name]”
  - creates an python environment with a name and initial packages
  - In the example of last page we installed numpy, python with a special version and jupyter.
  - Some packages have dependent packages which are derived automatically (e.g. the package anaconda has many dependencies)
  - All you install inside an environment is limited to that environment only
  - You can use “conda install” or “pip install” to install new packages
- “Conda activate [name]”
  - Activates a certain environment, you can see the active envirmonment in the prompft
- “Conda deactivate”
  - Deactivates the named environment and switches to the “base environment”
- “jupyter notebook”
  - Starts jupyter on a free local port and opens the page in the browser
  - Current directory is root sandbox for the jupyter instance
  - Usually can just be accessed from your computer except you set a password

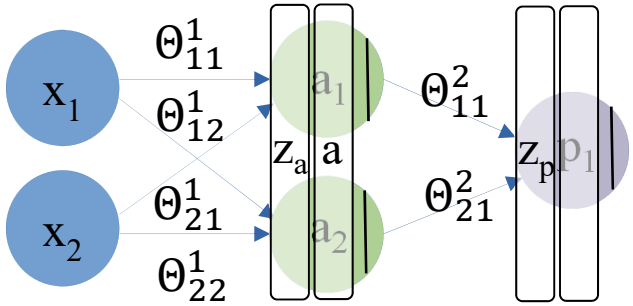
Implementing a neural network with numpy

DL_001_ForwardPropagation.ipynb

Forward Propagation

Suppose we have **just one** Training Example (x,y).

- $z_a = \Theta^1 x$
- $a = \sigma(z_a)$
- $z_p = \Theta^2 a$
- $p = \sigma(z_p)$



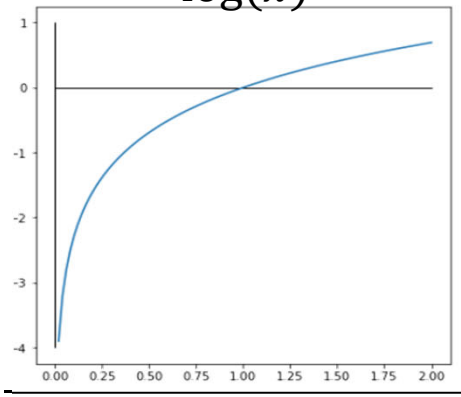
This also is in a similar way for multiple Training Examples (a Batch)

- $Z_a = \Theta^1 X$
- $A = \sigma(Z_a)$

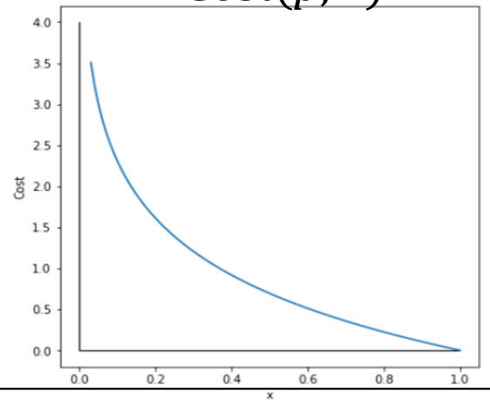
Cost function of binary classification

- $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x[i]), y[i])$
- $Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$

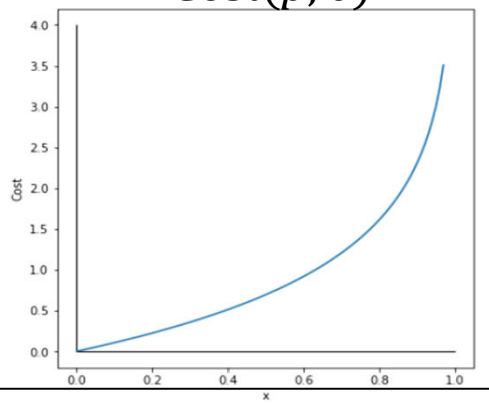
$\log(x)$



$\text{Cost}(p, 1)$



$\text{Cost}(p, 0)$



Whats did we learn?

Whats next ?

- Backpropagation
- Gradient Descent
- Different Optimizers