Machine Intelligence:: Deep Learning
Week 2

Oliver Dürr

Institut für Datenanalyse und Prozessdesign

Zürcher Hochschule für Angewandte Wissenschaften

Winterthur, 26. Feb. 2019

Organizational Issues: Times

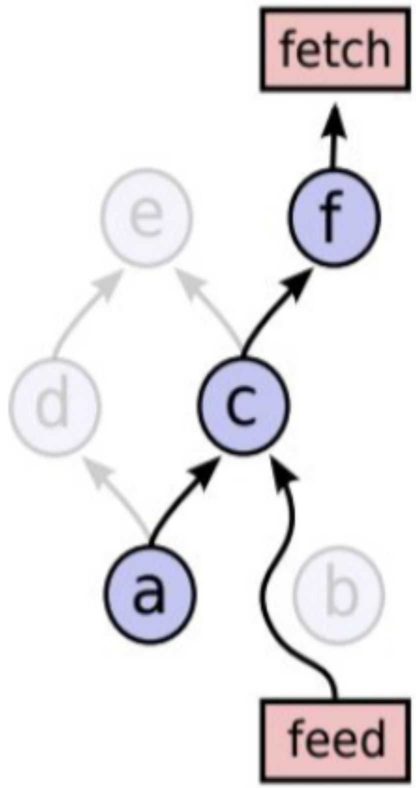
- First 3 times (total 30 minutes break in between)
  - 13:30 – 15:00
  - 15:30 – 17:00

- Please interrupt us if something is unclear!

- Projects

Recap from last week

Recap: Feed and Fetch




```
res = sess.run(f, feed_dict={b:[2]})
```

fetch
(the numeric value)

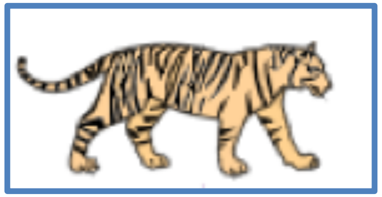
Fetch
f (symbolic)

symbolic

values

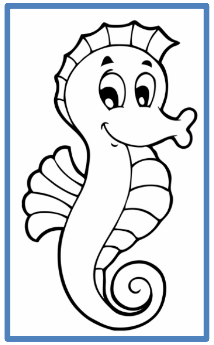
Prinzipielle Funktionsweise: Training Bild Klassifikation

Wahre Klasse | Vorhergesagtes Label

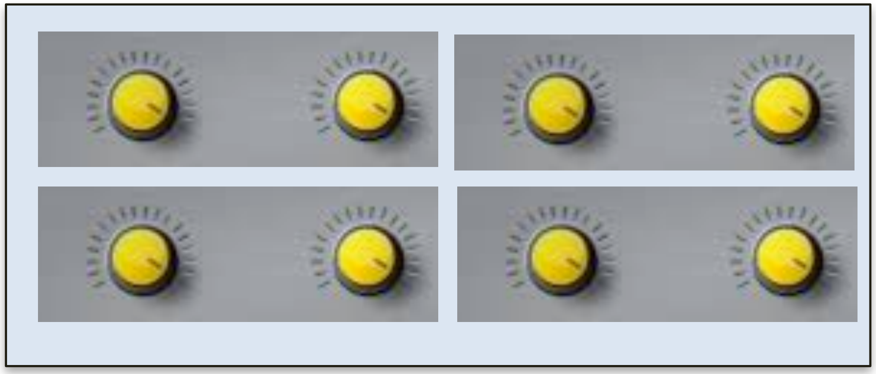


Tiger → Seehund 👎

Tiger → Tiger 👍



Seepferd → Seepferd 👍



...

Typisch 1 Mio. Trainingsdaten

Trainingsprinzip:
Parameter werden so eingestellt, dass möglichst wenige Fehler in den Trainingsdaten gemacht werden.

Prinzipielle Funktionsweise: Training Lineare Regression

	Wahrer Blutdruck	Vorhergesagter Blutdruck
Alter: 55	152	130
Alter: 35	155	151
Alter: 55	140	131

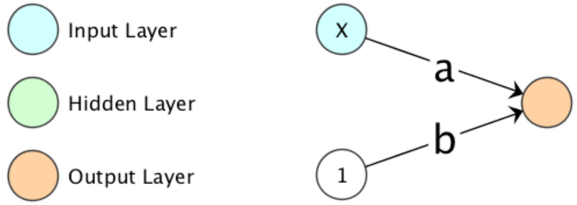
Das einfachste "Deep Learning" Model hat 2 Parameter.

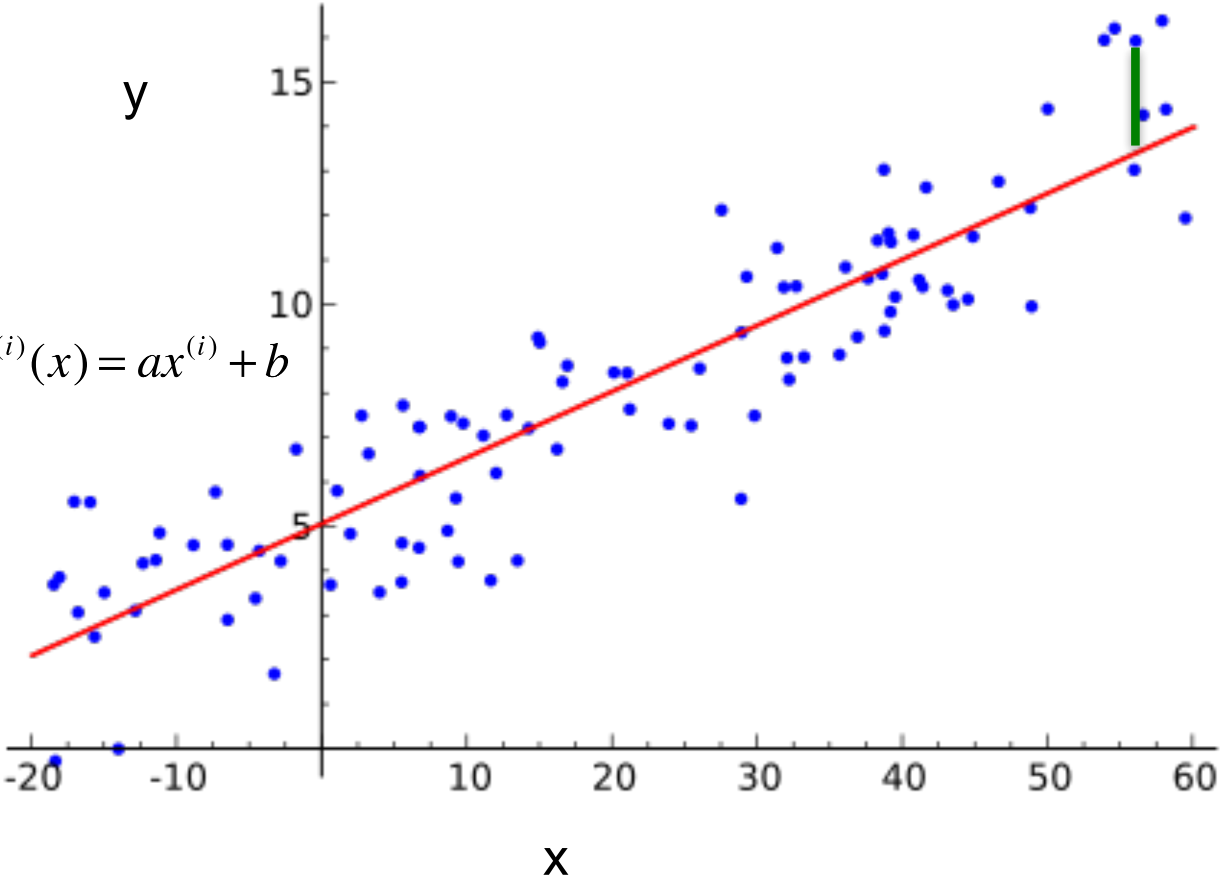
...

33 Trainingsdaten aus einer Studie von nordamerikanischen Frauen

Loss for linear regression: sums of squared error

$$\text{loss} = \frac{1}{N}\sum_{i=1}^{N}(ax^{(i)} + b - y^{(i)})^2$$





$$\hat{y}^{(i)}(x) = ax^{(i)} + b$$

Optimization

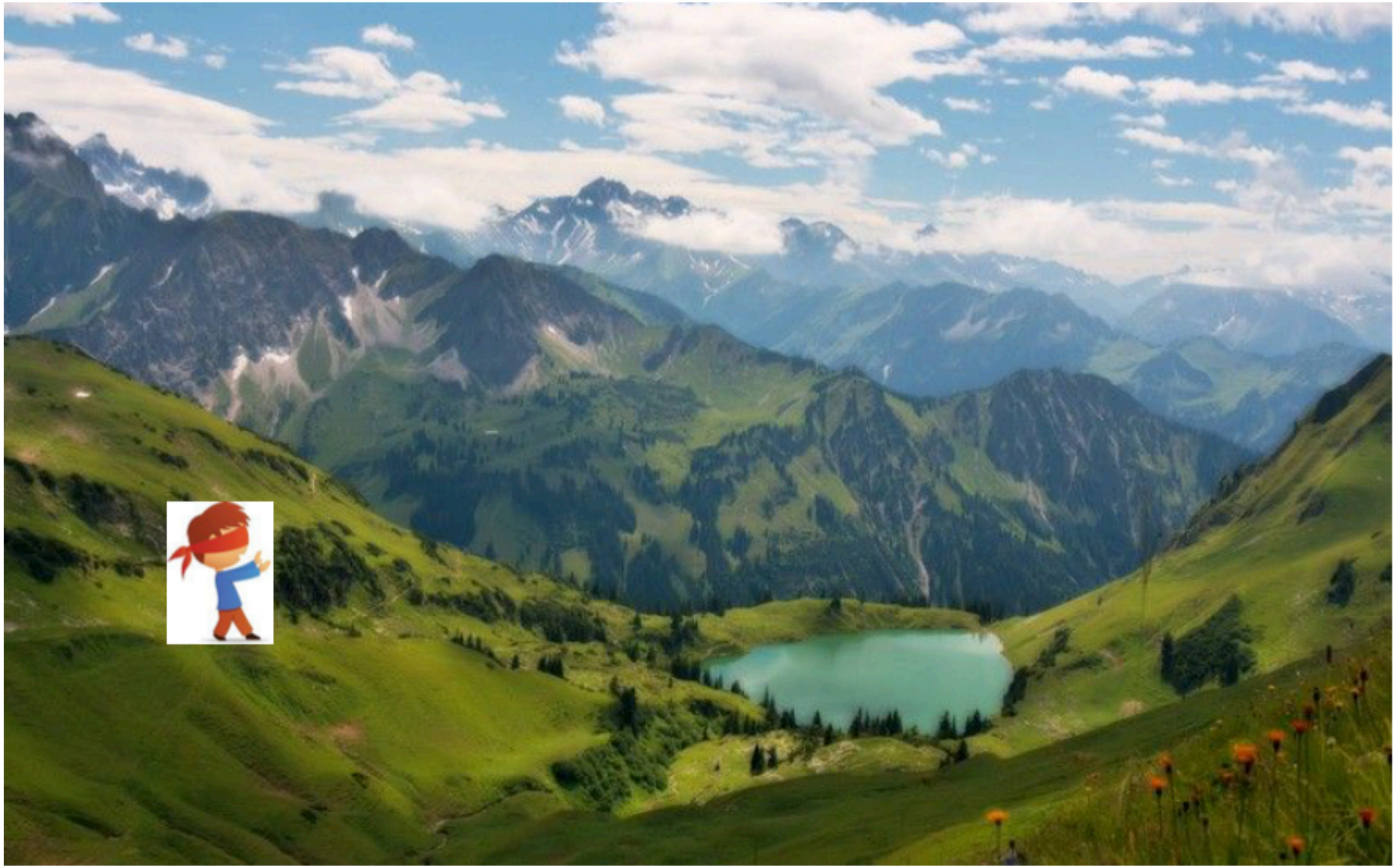


Figure shows a 2 dimensional loss function. In DL Millions!
We just know the current value (blind)

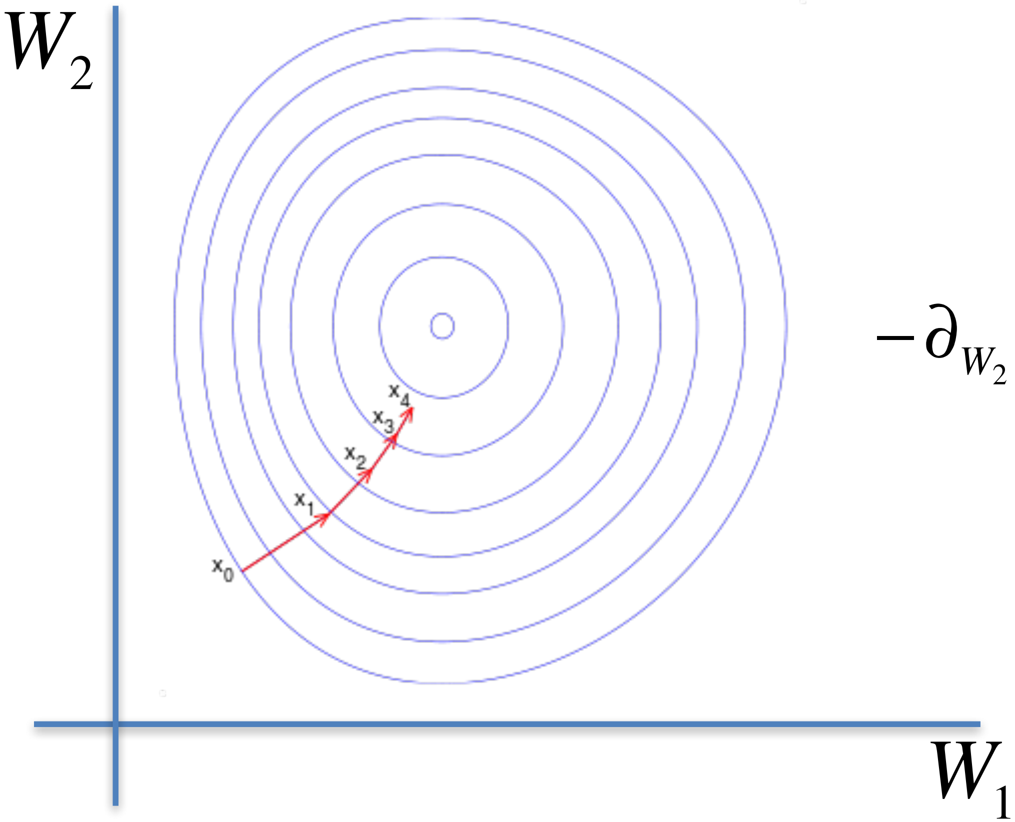
Slide from cs229

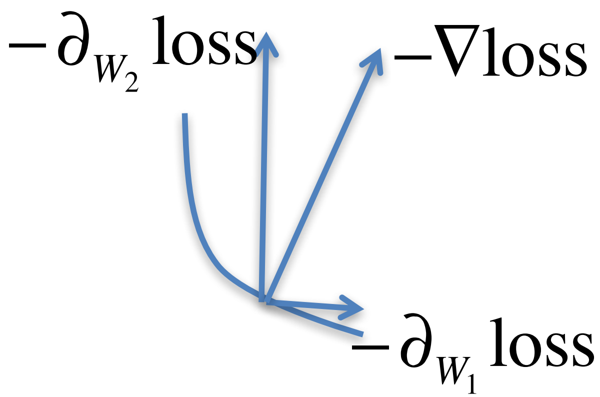
Optimization

- Gradient Descent

  Gradient is perpendicular to levels

$$W_i^{t+1} = W_i^{t} - \varepsilon \, \partial_{W_i} \mathrm{loss}$$





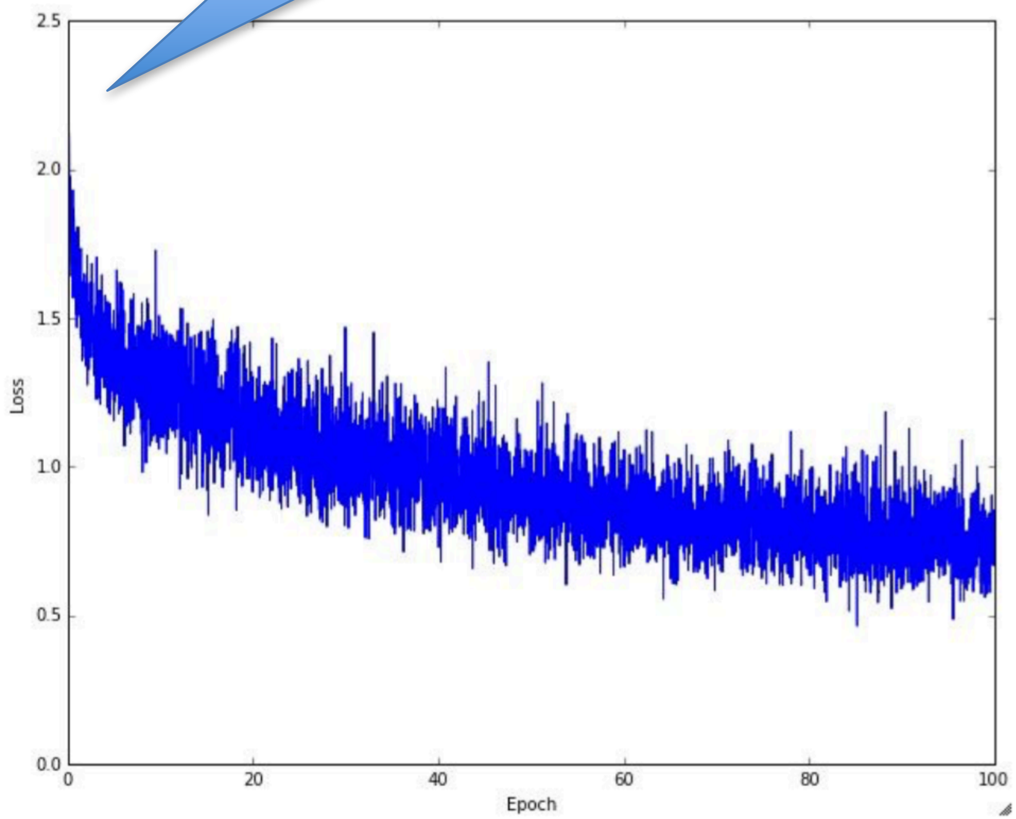
Slide credit: wikipedia

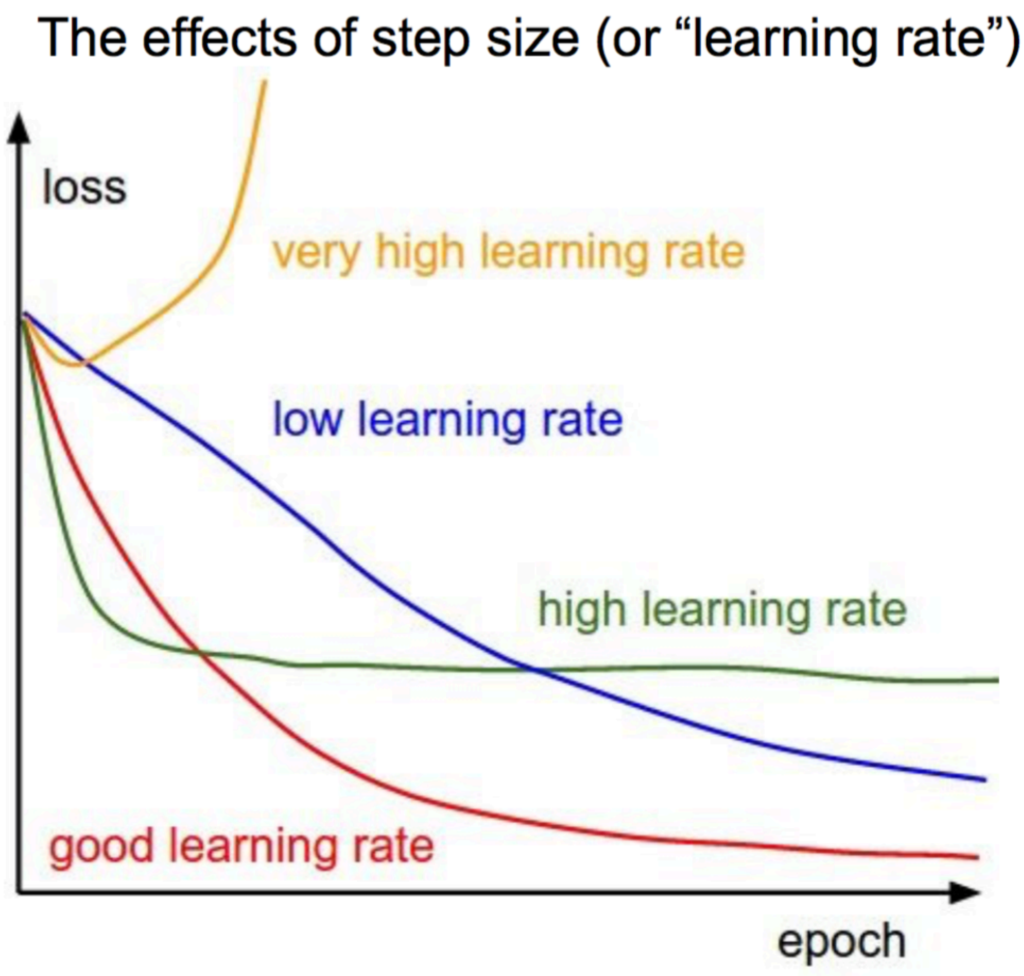
Effect of learning rate

https://developers.google.com/machine-learning/crash-course/fitter/graph

Gradient Descent in DL

Always have a look at the learning curves



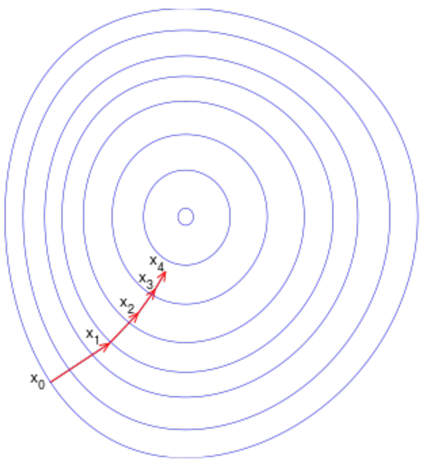


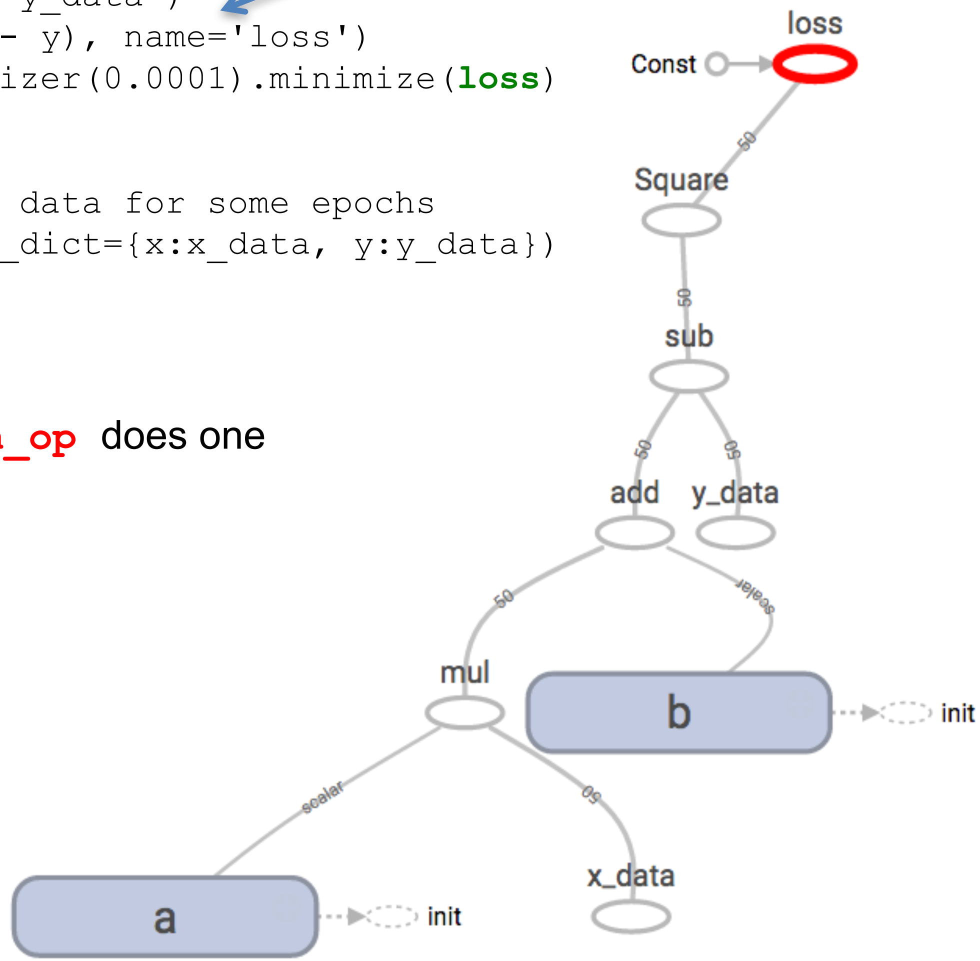
Computation done with a graph

loss= $\frac{1}{N}\sum_{i=1}^{N}(ax^{(i)} + b - y^{(i)}) = \frac{RSS}{N}$

```
tf.reset_default_graph()
a = tf.Variable(1.0, name = 'a')
b = tf.Variable(1.0, name = 'b')
x = tf.placeholder('float32', [N], name='x_data')
y = tf.placeholder('float32', [N], name='y_data')
loss = tf.reduce_mean(tf.square(a*x + b - y), name='loss')
train_op = tf.train.GradientDescentOptimizer(0.0001).minimize(loss)
with tf.Session() as sess:
    …
    for e in range(epochs): #Fitting the data for some epochs
        res = sess.run([train_op,…],feed_dict={x:x_data, y:y_data})
```

TF does all the hard work for you.
Symbolically calculates gradient. Running `train_op` does one gradient step.





Feeding and Fetching the graph

Matrix Multiplication in TensorFlow (Rest)

c) Now use a placeholder for m2 to feed-in values. You must specify the shape of the m2 matrix (rows, columns).

Besprechung der Aufgabe

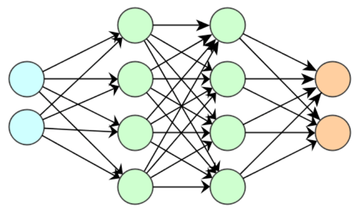
- **Linear regression in TensorFlow**

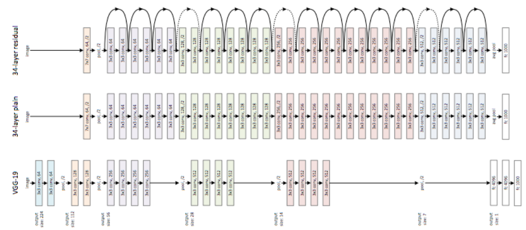
- a) Open the notebook Linreg_with_slider and run the fist 4 cells and try to minimize the loss by adjusting the parameters a and b.

- b) Run the next two cells and feed your adjusted parameters through the graph. You have to modify cell 6 a bit.
- Do not do c, d)

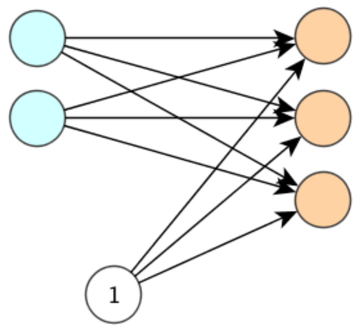
End of Recap

Learning Objectives





- Increase our knowledge in TF
- Foundations of DL
  - **Loss Function (what to minimize)**
    - Loss Function for Regression
      - Mean Squared Error
    - Loss Function for classification
      - Binary cross entropy loss for logistic regression
      - Cross entropy loss for multinomial logistic regression
    - Two principles to construct loss functions
      - Maximum Likelihood Principle
      - Cross Entropy [time permitting]



We use networks with no hidden layers to explain basics. Loss function and gradient descent stay the same for real networks.

Tuning a neural network: a loss function

- Neural networks are models which have parameters
- We have (training $i = 1 \ldots N\_\text{training}$) data in pairs $x^{(i)}$ and $y^{(i)}$

$x^{(i)}$→model parametrized with weights→ $\hat{y}^{(i)}$

Depending on the weight the model produces a different $\hat{y}^{(i)}$

- Examples (your task what are the x’s what are the y’?)
  - Facerec.: Faces and Names
  - Age Prediction: Faces and Age (numerical problem)

- How to tune the nobs that the output of the model $\hat{y}^{(i)}$ matches the “true” value $y^{(i)}$? We optimize a loss.

- To understand the principle, we start with something dead simple: good old linear regression

Linear Regression as Max Likelihood Optimization

Die Maximum-Likelihood (ML) Methode

Sie haben einen Würfel mit z.B. l=2 roten Seiten.

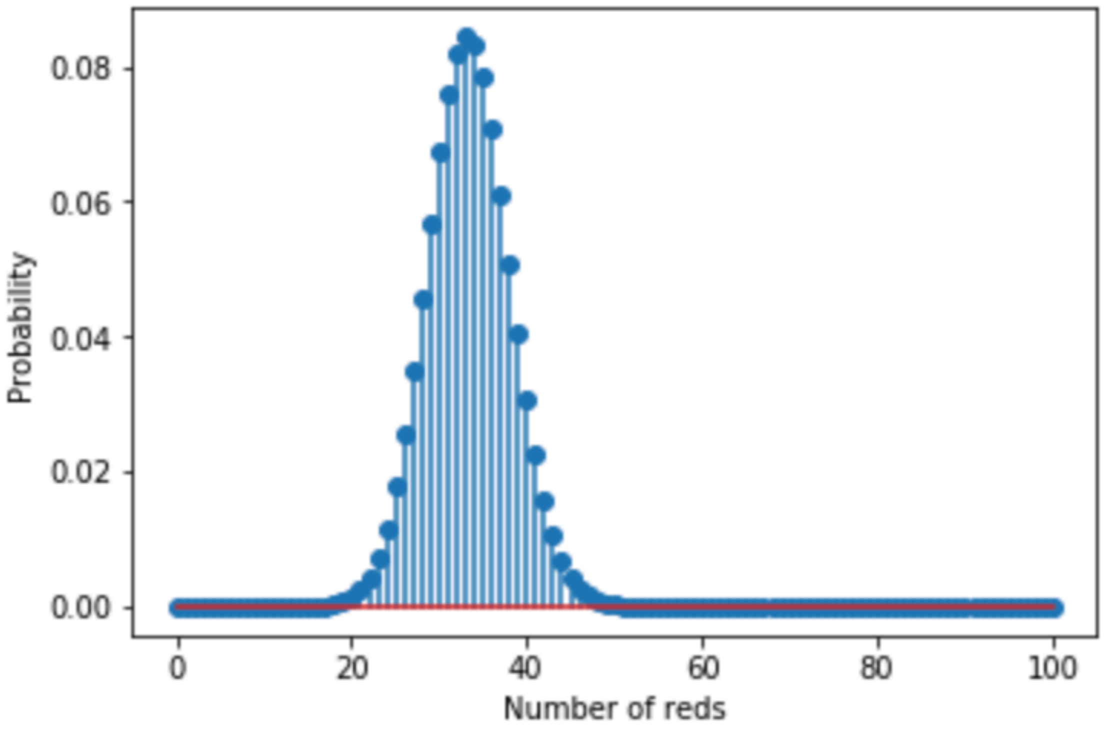
Wie Wahrscheinlich ist es, dass Sie bei 100 Würfen:

- *Keine rote Seite gewürfelt haben*
- *34 rote Seiten gewürfelt haben*
- *99 rote Seiten gewürfelt haben*

$$P(X = k \mid p) = \binom{100}{k} \cdot p^k \cdot (1-p)^{100-k}$$

$$p = \frac{2}{6}$$

Ergebniss




```
from scipy.stats import binom
reds = np.asarray(np.linspace(0,100,101), dtype='int') #The numbers 0 to 10 as integers
p_reds = binom.pmf(k=reds, n=100, p=2/6) #The probability of 0,1,2...,throws
plt.stem(reds, p_reds)
plt.xlabel('Number of reds')
plt.ylabel('Probability')
```

Umdrehen der Argumentation

- Wir haben 34 mal rot gesehen, welche Wert des Parameters erklärt die Daten am Besten (hat die grösste Wahrscheinlichkeit)
-

Die Maximum-Likelihood (ML) Methode

Sie haben einen Würfel mit z.B. l=2 roten Seiten.

Wie Wahrscheinlich ist es, dass Sie bei 100 Würfen:

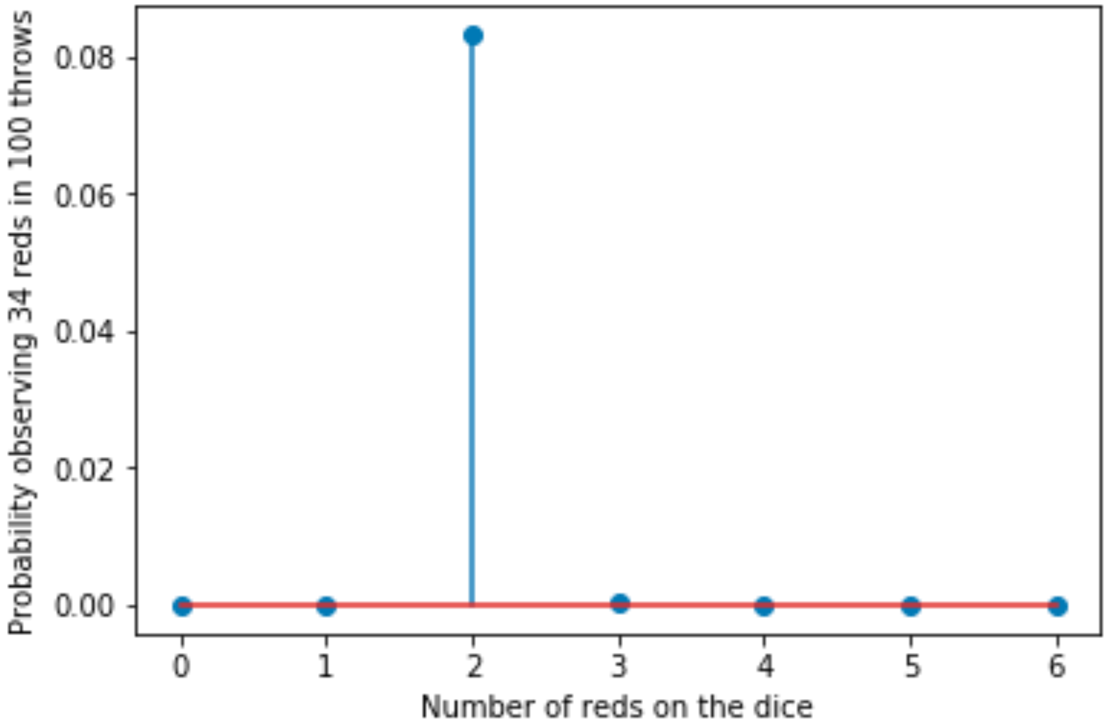
- *Keine rote Seite gewürfelt haben*
- *34 rote Seiten gewürfelt haben*
- *99 rote Seiten gewürfelt haben*

$$P(X = k \mid p) = \binom{100}{k} \cdot p^k \cdot (1-p)^{100-k}$$

$$p = \frac{2}{6}$$

In R ausprobieren.

Varying p

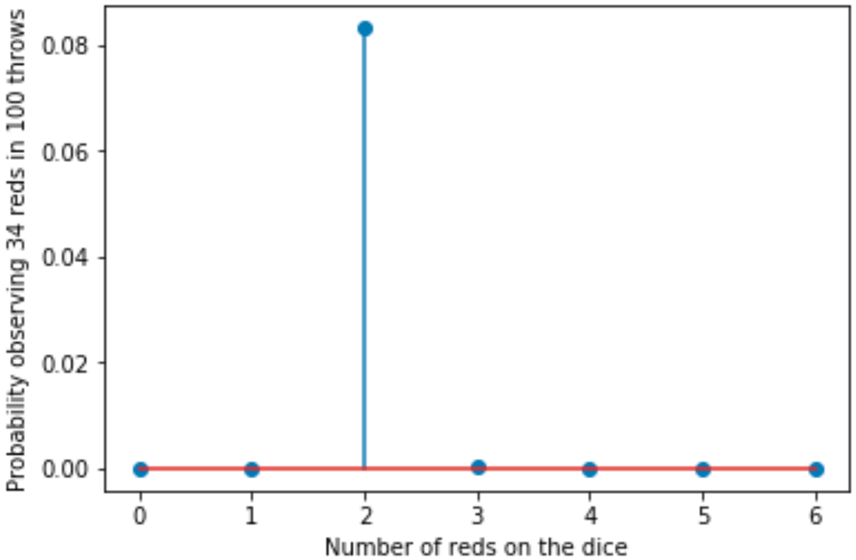



```
# Solution
from scipy.stats import binom
num_reds = np.asarray(np.linspace(0,6,7), dtype='int')
p_num_dollar = binom.pmf(k=34, n=100, p=num_reds/6)
```

ML-Methode Was haben wir getan?

$$L(X \mid \theta) = P(X = k \mid p) = \binom{100}{k} \cdot p^k \cdot (1-p)^{100-k}$$

Fest   Der Parameter Variiert



ML-Methode Was haben wir getan?

$$L(X \mid \theta) = P(X = k \mid p) = \binom{100}{k} \cdot p^k \cdot (1-p)^{100-k}$$

Fest

Der Parameter
Variiert

Wir haben das p so gewählt, dass es dem Maximum der Wahrscheinlichkeit genügt.

Anmerkung: P(X=k|p) (als Funktion von p) ist keine Wahrscheinlichkeit im engerem Sinne, denn z.B.

```
sum(dbinom(x = 34, prob = c(1:6)/6, size = 100))=0.083
```

Maximum Likelihood
(one of the most beautiful ideas in statistics)

Ronald Fisher in 1913
Also used before by Gauss, Laplace

Likelihood / “probability“
(often known)

$$M(\theta) \longrightarrow \text{Data}$$

Tune the parameter(s) θ of the model M so that (observed) data is most likely

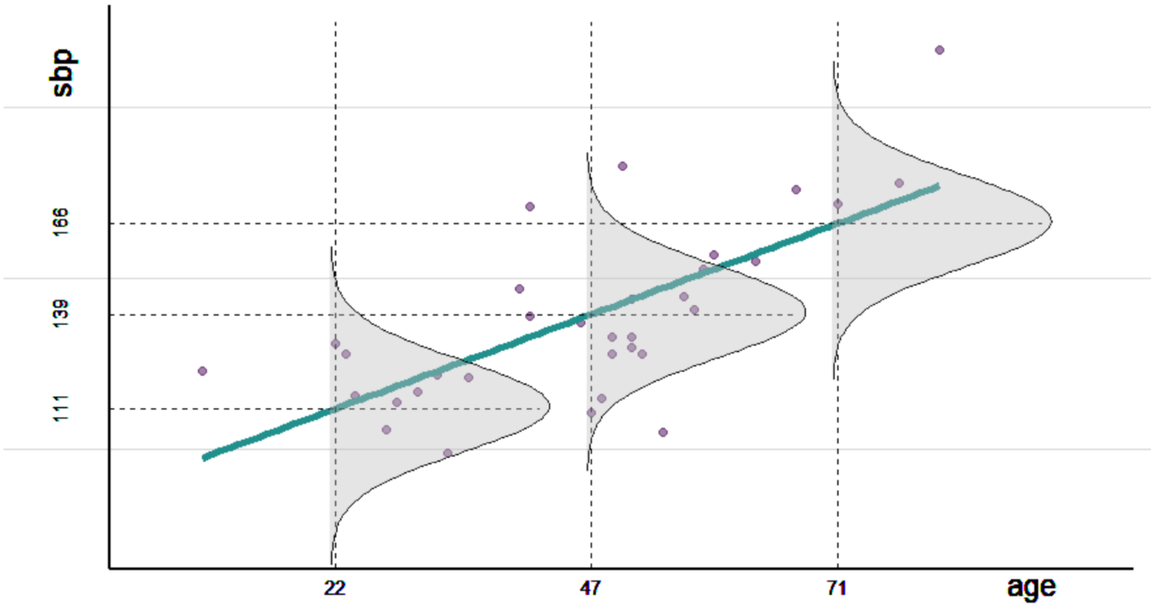
What‘s the likelihood of the data for log. regression...

Lineare Regression als MaxLikelihood

Step 1: Probability Distribution for Data y given x.

$$p_{\text{model}}(y^{(i)} \mid x^{(i)}; \theta)$$

Here: Gaussian with $\mu = a * x + b$ and $\sigma = constant$



$$Y_{x_i} \quad \sim N(\mu_{x_i}, \sigma^2)$$

See Blackboard

Derivation of the MSE

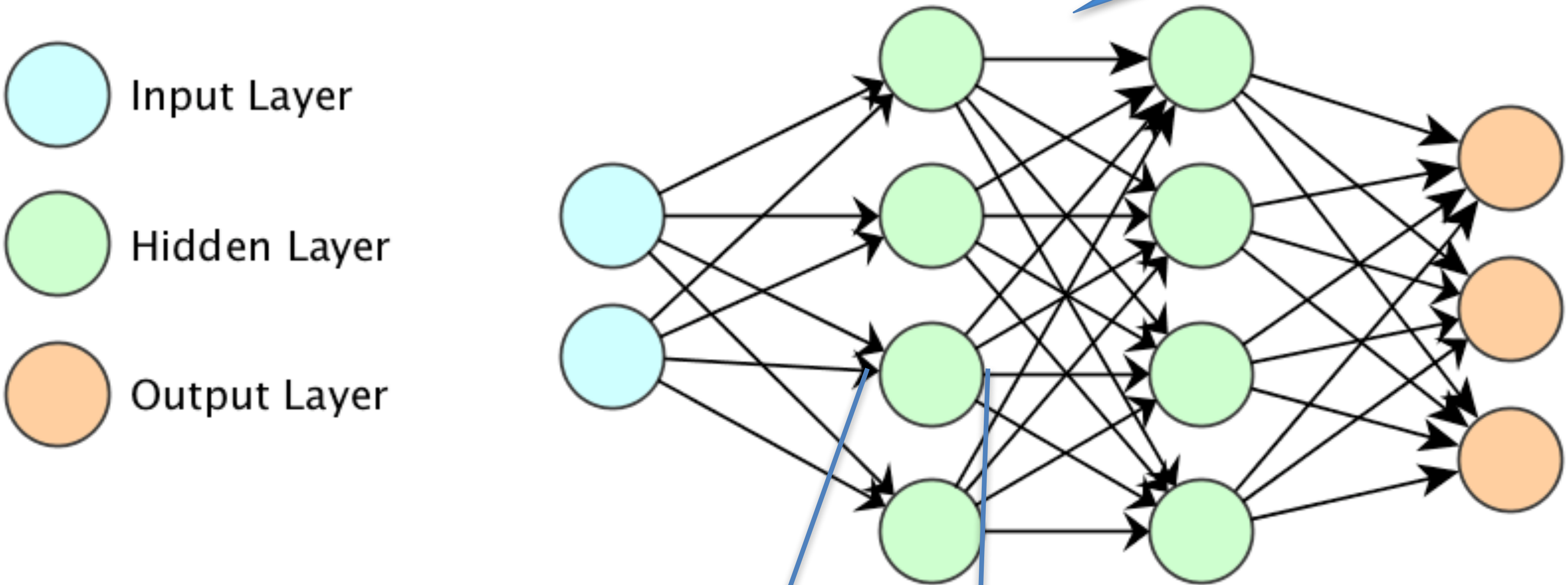
See also blackboard

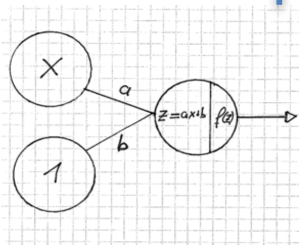
$$
\begin{aligned}
\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left\{ \prod_{l=1}^{L} \Pr(y_l \,|\, \mathbf{x}_l, \boldsymbol{\theta}) \right\} \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left\{ \prod_{l=1}^{L} \mathrm{Norm}_{y_l}(\mathbf{w}^T \mathbf{x}_l + b, \sigma^2) \right\} \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left\{ \prod_{l=1}^{L} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(\mathbf{w}^T \mathbf{x}_l + b - y_l)^2}{2\sigma^2} \right) \right\} \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{ -\log \left(\prod_{l=1}^{L} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(\mathbf{w}^T \mathbf{x}_l + b - y_l)^2}{2\sigma^2} \right) \right) \right\} \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{ \sum_{l=1}^{L} -\log \left(\frac{1}{2\pi\sigma^2} \right) + \frac{1}{2\sigma^2} (\mathbf{w}^T \mathbf{x}_l + b - y_l)^2 \right\} \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{ \frac{1}{2\sigma^2} \sum_{l=1}^{L} (\mathbf{w}^T \mathbf{x}_l + b - y_l)^2 \right\}
\end{aligned}
$$

Logistic Regression

Fully Connected Networks

Real networks of course are larger. But this captures the basic structure



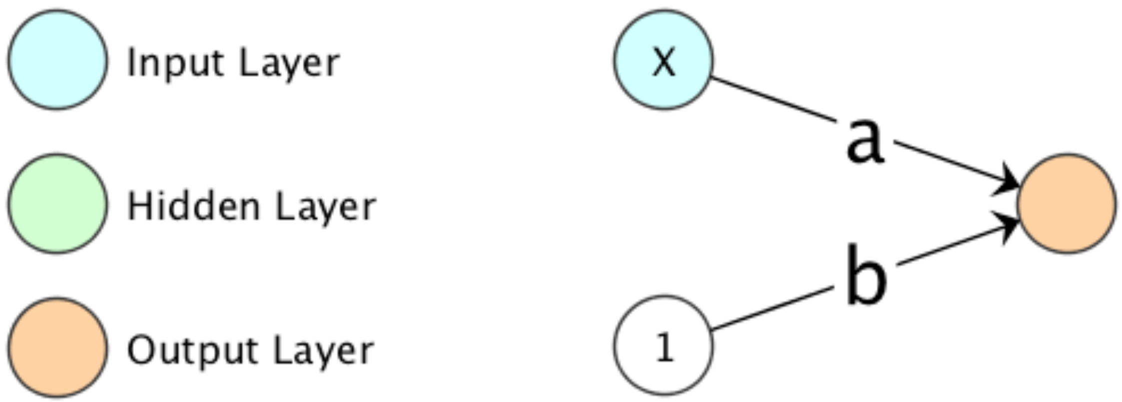


We start with....

1-D Logistic Regression

The building blocks of a neural network:
- logistic regression: the mother of all networks

- The first building block

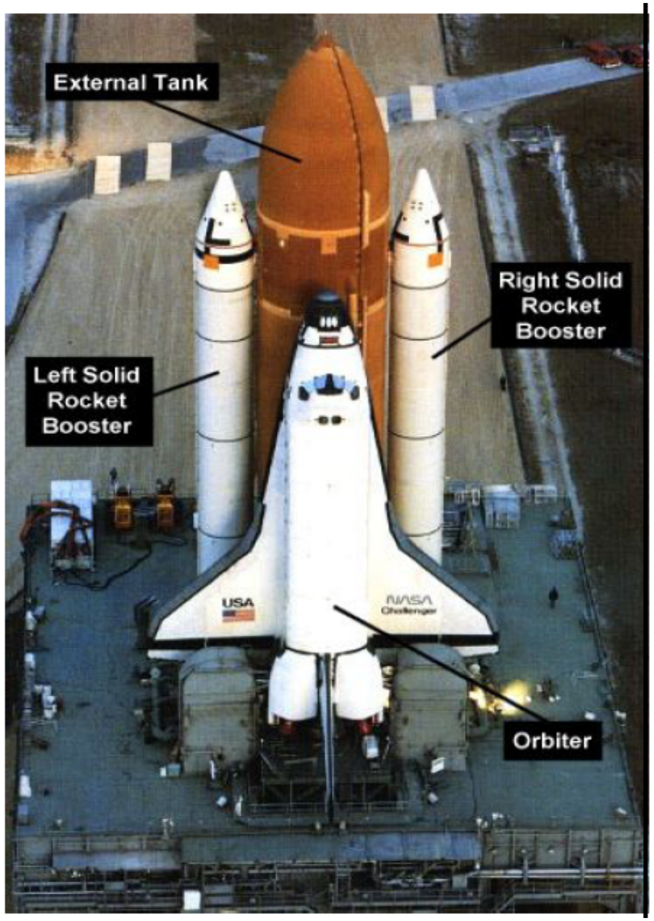


- In the following, have a look at logistic regression and derive the cost-function (log-likelihood) which we maximise.

- Logistic Regression by it self is a method used since many years in statistics (David Cox 1958) and should be part of the ML toolbox

Logistic Regression [motivation]

Some Background on probabilistic modelling

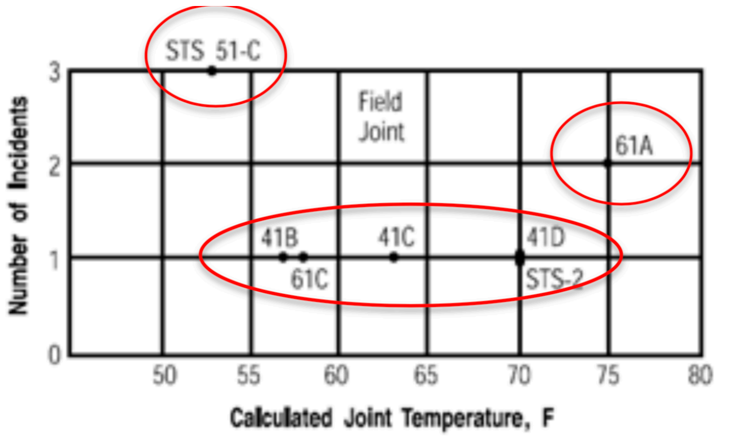


The challenger space shuttle exploded 73 seconds the start in 1986. One of bearings in the booster has been broken.

Statistik & Challenger Desaster [side track]

- On the day of the challenger launch it was cold: 31°F.
- In 7 from 23 flights there have been problems with the booster bearings

Ambient temperature	Number of O–rings damaged	$\hat{p}$
53°	2	.333
57°	1	.167
58°	1	.167
63°	1	.167
70°	1	.167
70°	1	.167
75°	2	.333



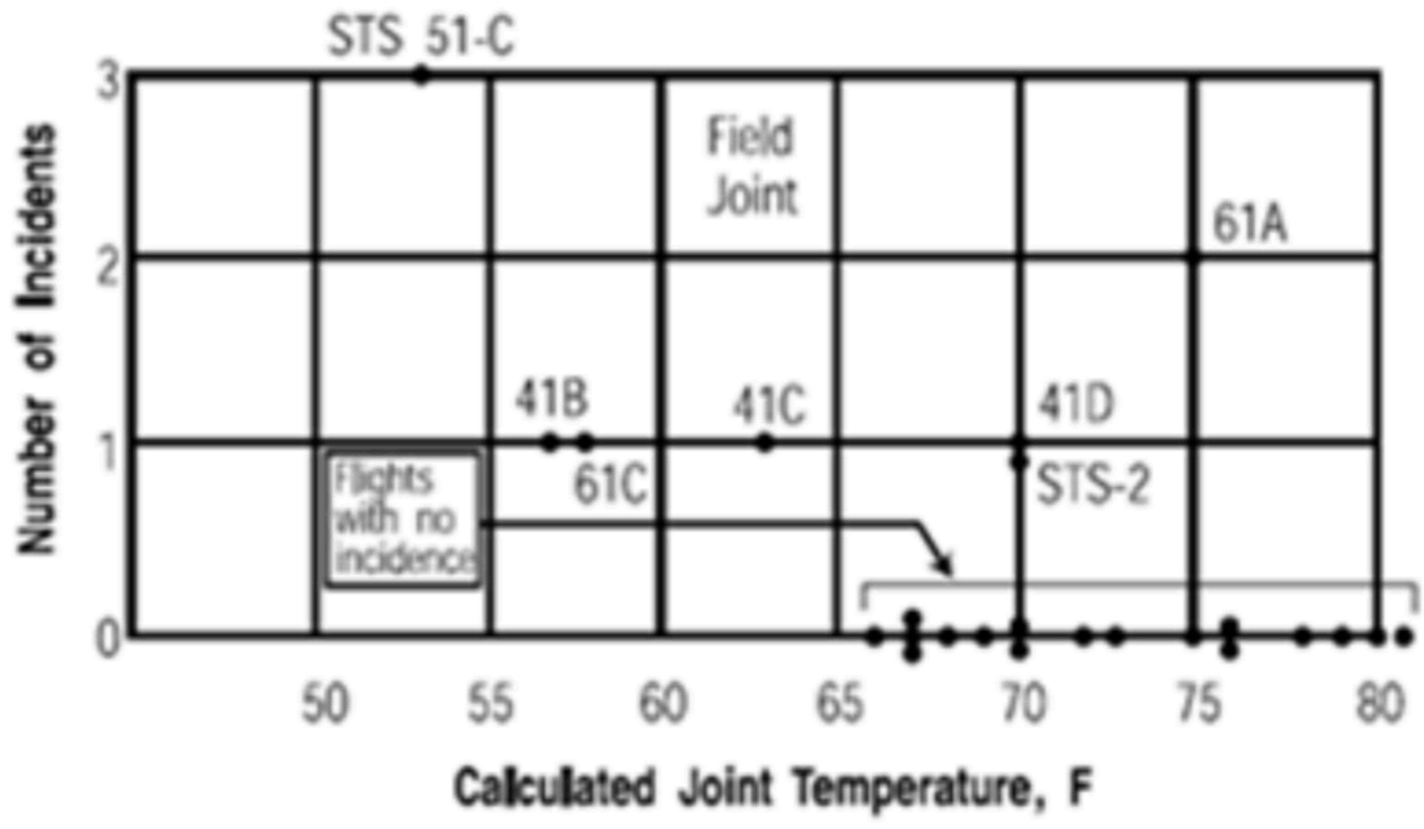
Plot of flights with incidents of O-ring thermal distress as function of temperature.

Figures from: [PRESIDENTIAL COMMISSION on the Space Shuttle Challenger Accident](https://history.nasa.gov/rogersrep/v4part3.htm)
(https://history.nasa.gov/rogersrep/v4part3.htm)

- Is there an increased risk of failure at low temperatures?
  - NASA Engineer: „...I can't get a correlation between O-ring erosion, blow-by an O-ring, and temperature. “
- Would you launch (give reasons)?

Statistik & Challenger Desaster [side track]

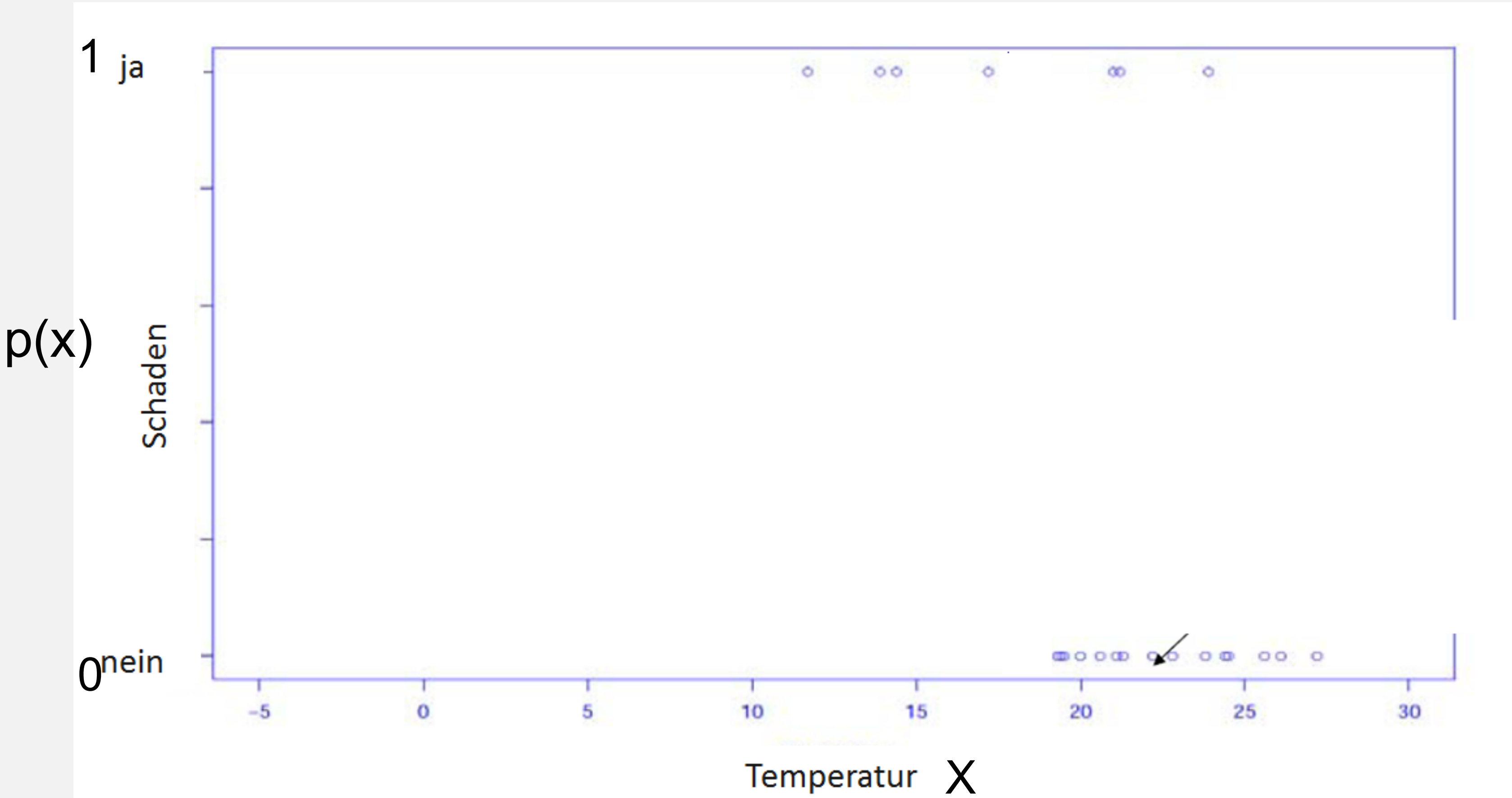
- There is information in the successful flights



Plot of flights with and without incidents of O-ring thermal distress.

Modelling logistic regression

$p(X) = Pr(Y = 1|X)$ Prob. for one (or more) o-ring to be defect at a given temperature X

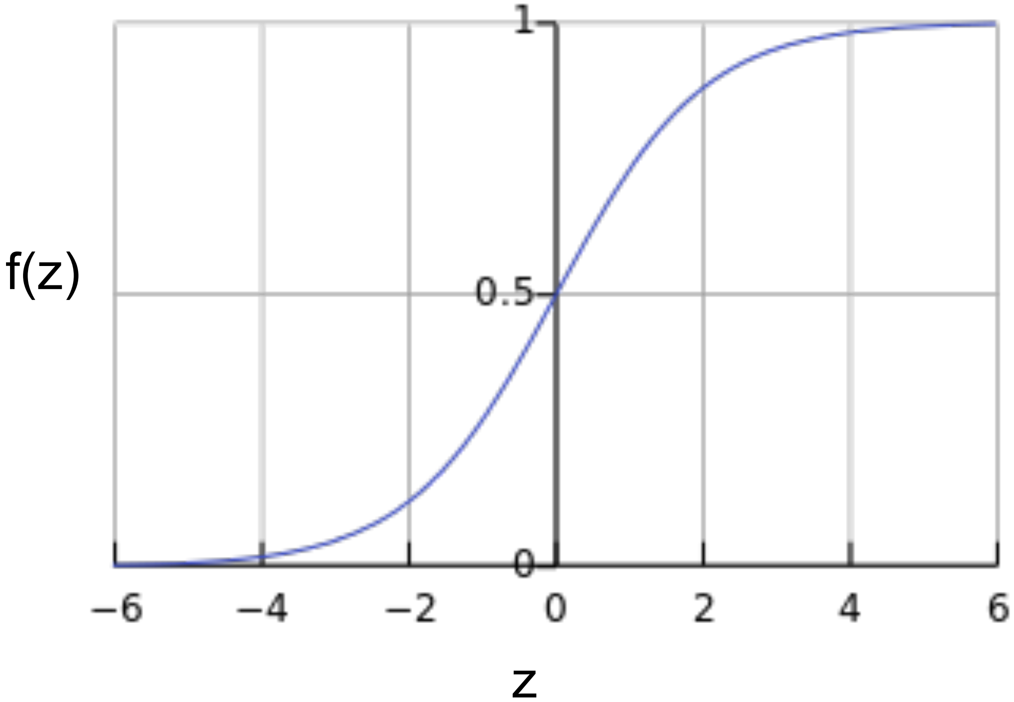


1. Draw a line $p(X) = a X + b$ which fits data best (linear regression)
2. Question: Why is linear regression wrong?

Sigmoids to the rescue

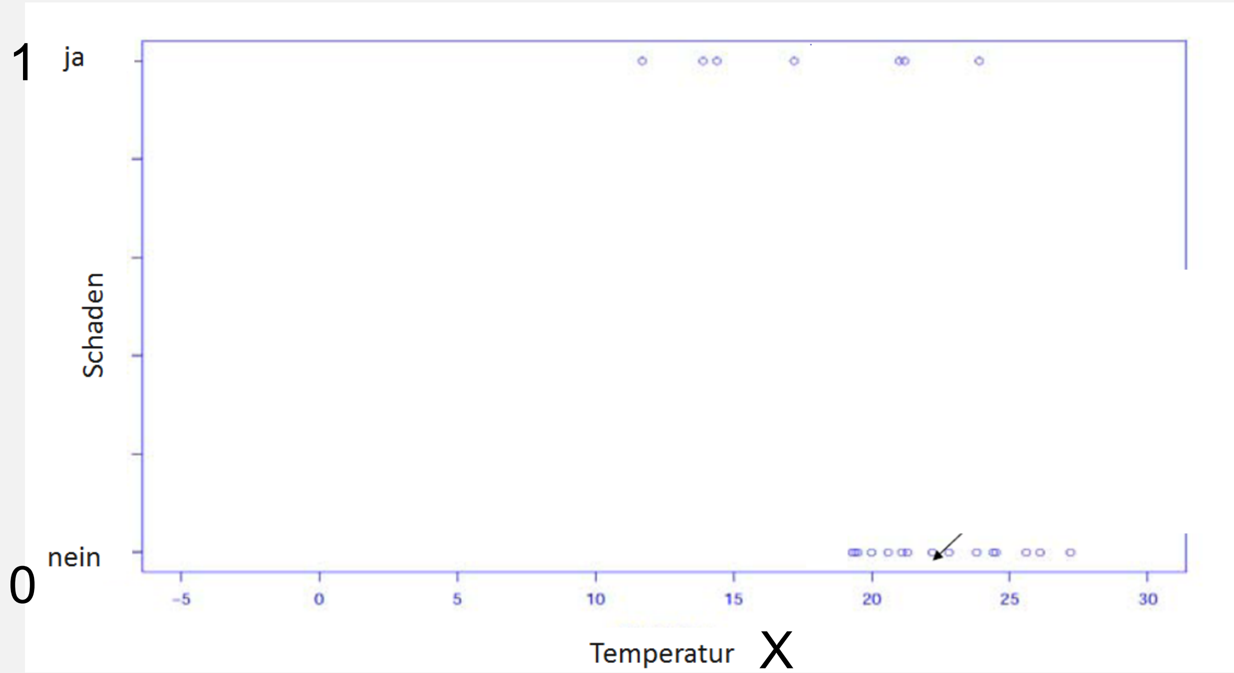
- With linear regression we have values outside [0,1]
- We do a sigmoid transformation to fix this (a.k.a. logistic curve)

$$f(z) = (1 + e^{-z})^{-1} = \sigma(z)$$



Modelling logistic regression

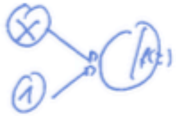
p(X) = Pr(Y = 1|X) Prob. for a O-ring to be defect at a given temperature X

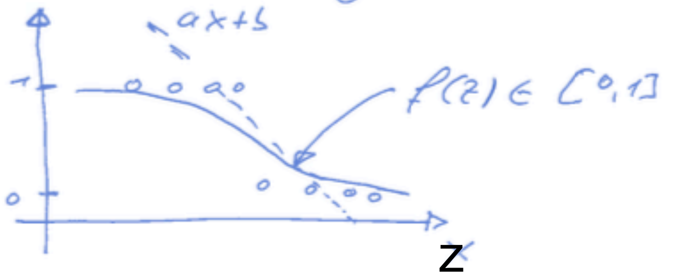


Task: Use the sigmoid function to bend your results of the linear regression.

Logistic Regression

Logistic Regression





$z = a \cdot x + b$ bend to 0, 1 with sigmoid.

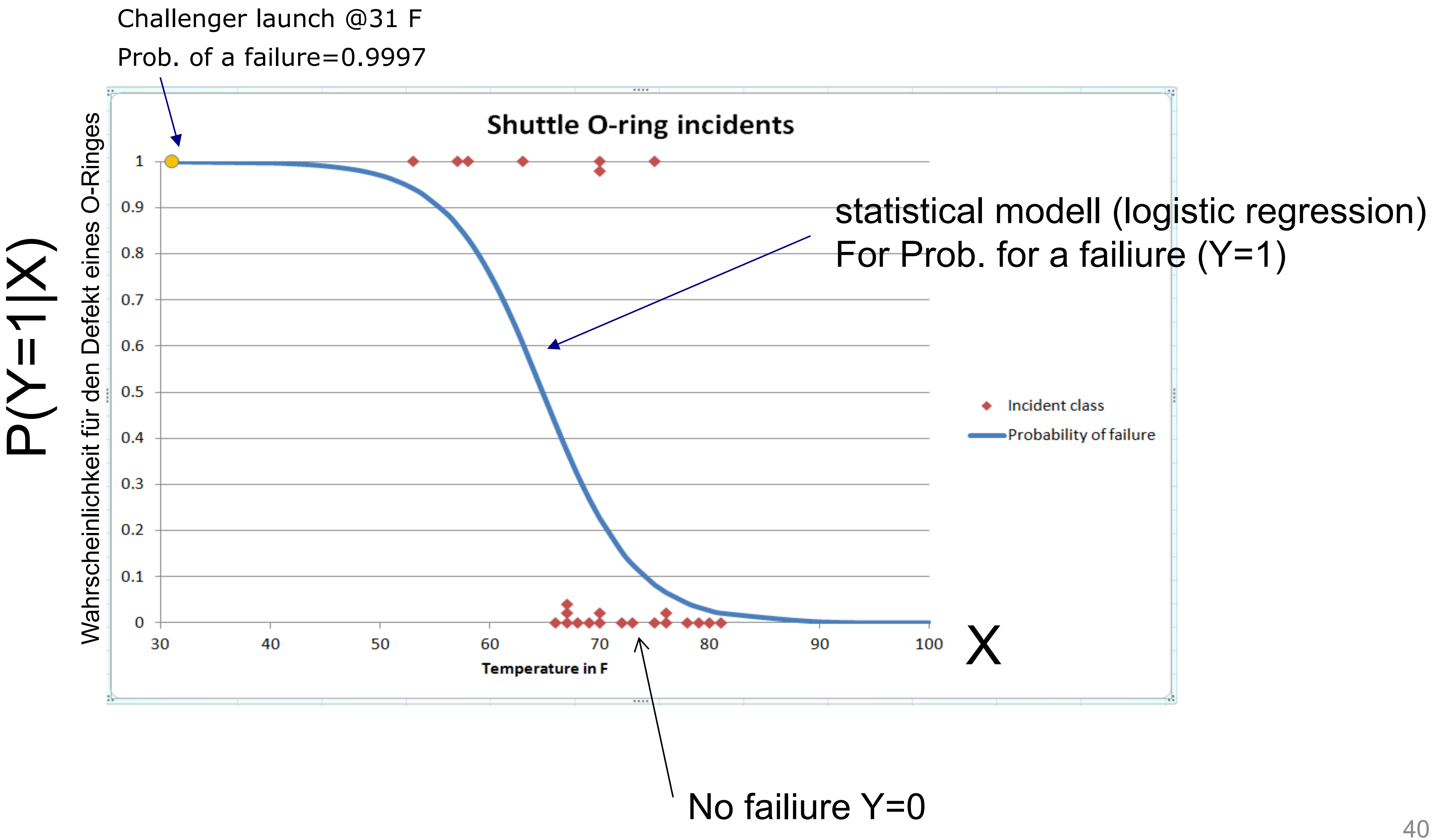
$$f(z) = \frac{1}{1+e^{-z}} = p(Y=1|X)$$

$f(z)$ stat. model contains a, b fit to all the data points (x_i, y_i).

Nur das erklaeren

Logistic Regression: Example challenger O-rings

Predict if O-Ring is broken, depending on temperature



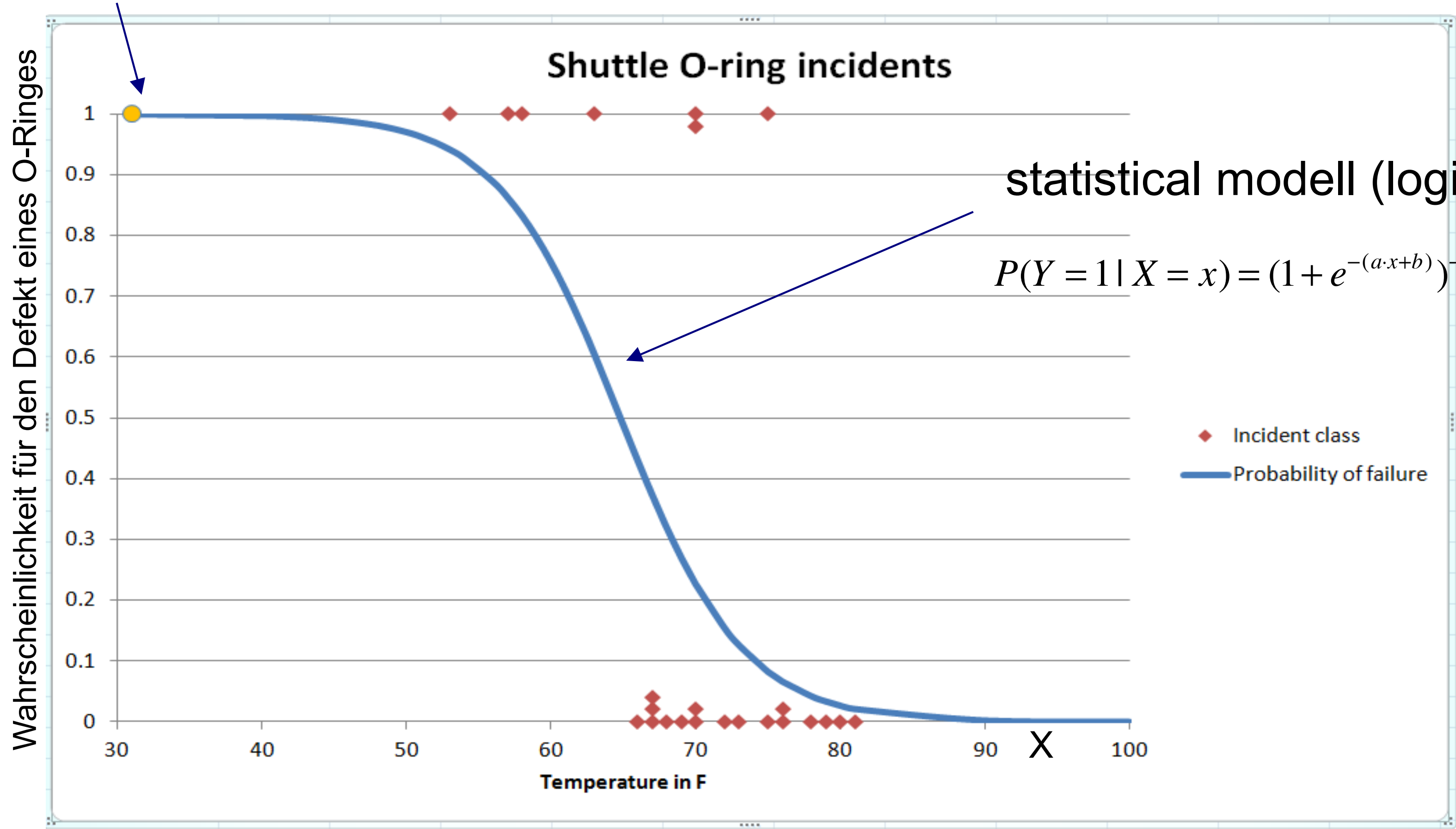
Logistic Regression

Predict if O-Ring is broken, depending on temperature

Challenger launch @31 F

Prob. of a failure=0.9997

P(Y=1|X)



X

statistical modell (logistic regression)

$$P(Y=1 \mid X=x) = (1+e^{-(a \cdot x+b)})^{-1} = (1+e^{-z})^{-1} = f(x)$$

$$z \in]-\infty,\infty[\rightarrow [0,1]$$

How do we determine the parameters (a,b) of the model? M(β)

Maximum Likelihood
(one of the most beautiful ideas in statistics)

Ronald Fisher in 1913
Also used before by Gauss, Laplace

Likelihood / “probability“
(often known)

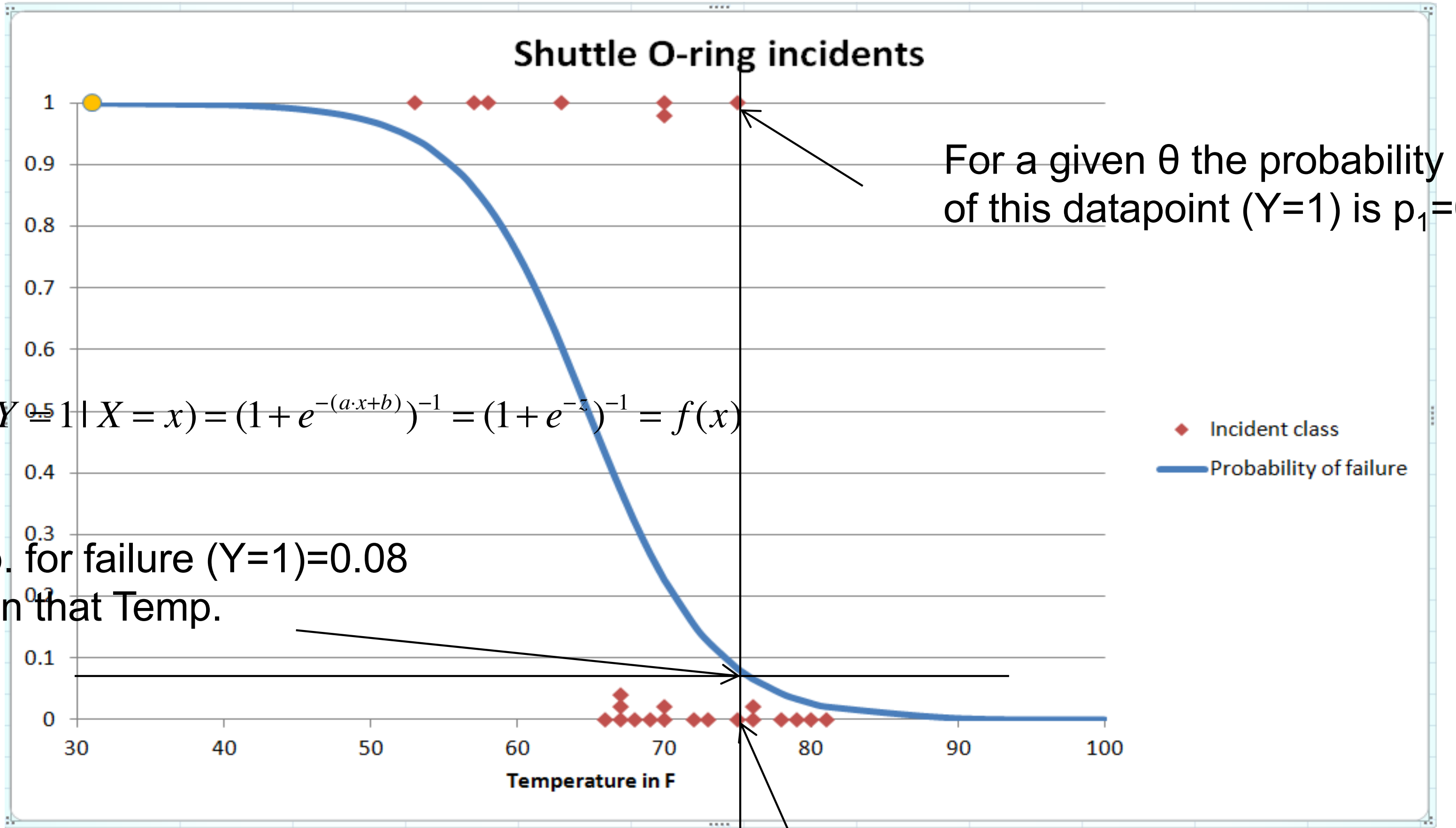
$$M(\theta) \xrightarrow{\hspace{4em}} \text{Data}$$

Tune the parameter(s) θ of the model M so that (observed) data is most likely

What‘s the likelihood of the data for log. regression...

Likelihood: Probability of a single observation

Two data points Y=1 (failure) and Y=0 (OK)



Y

For a given θ the probability of this datapoint (Y=1) is p_1=0.08

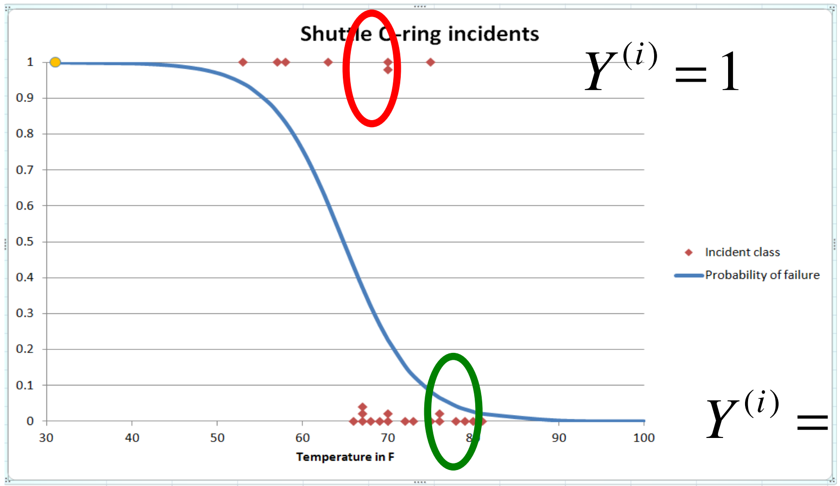
$$p_1 = P(Y = 1 | X = x) = (1 + e^{-(a \cdot x + b)})^{-1} = (1 + e^{-z})^{-1} = f(x)$$

Prob. for failure (Y=1)=0.08
Given that Temp.

For a given θ the probability of this datapoint (Y=0) is
1 - 0.08 = 92%

Prob. of all data points is the product of the individual data points...
(if iid).

Likelihood: Probability of the training set



$Y^{(i)} = 1$

$Y^{(i)} = 0$

Training Data i = 1...N

$X^{(i)}, Y^{(i)}$

$$p_1(X) = P(Y=1 \mid X) = (1 + e^{-(a \cdot x + b)})^{-1} = (1 + e^{-z})^{-1} = f(x)$$

Probability to find Y=1 for a given values X (single data point) and a, b

$$p_0(X) = 1 - p_1(X)$$

Probability to find Y=0 for a given value X (single data point)

Likelihood (probability⁺ of the training set given the parameters)

$$L(a,b) = \prod_{i \in All\ ones} p_1(x^{(i)}) \; * \prod_{i \in All\ Zeros} p_0(x^{(j)})$$

Let's maximize this probability

Maximizing the Likelihood

Likelihood (prob of a given training set) want to maximized wrt. parameters

$$L(a,b) = \prod_{i \in All\ ones} p_1(x^{(i)}) \quad * \prod_{i \in All\ Zeros} p_0(x^{(j)})$$

Taking log (maximum of log is at same position)

This is like a if-then

$$-nJ(\theta) = L(\theta) = L(a,b) = \sum_{i \in All\ ones} \log(p_1(x^{(i)})) \quad + \sum_{i \in All\ zeros} \log(p_0(x^{(i)})) = \sum_{i \in All\ Training} y_i \log(p_1(x^{(i)})) + (1 - y_i) \log(p_0(x^{(i)}))$$

Same as cross-entropy loss used already for linear regression

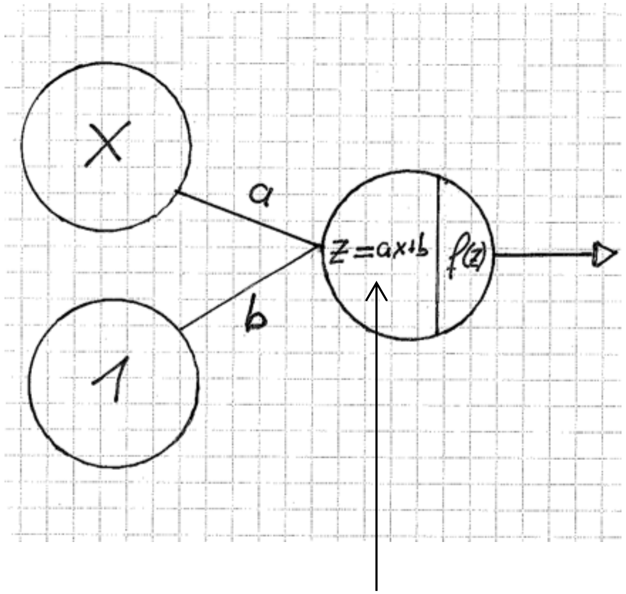
$$\text{loss} = -\frac{1}{N} \sum_{n=1}^{N} \log(p_{\text{model}}(y^{(i)} \mid x^{(i)}; \theta)) = -\frac{1}{N} \left(\sum_{i \in All\ ones} \log(p_1(x^{(i)})) \quad + \sum_{i \in All\ zeros} \log(p_0(x^{(i)})) \right)$$

Neg. log likelihood loss (general)
This is the prob. the model evaluates for the true class $y^{(i)}$ of training example $x^{(i)}$

For logistic regression

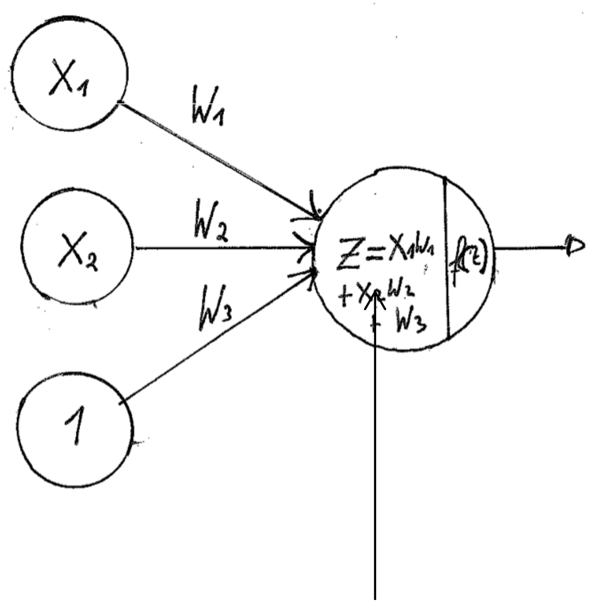
Logistic Regression in the neural net speak

1-D log Regression



$$z = ax + b$$

N-D log Regression



GPUs love linear algebra!

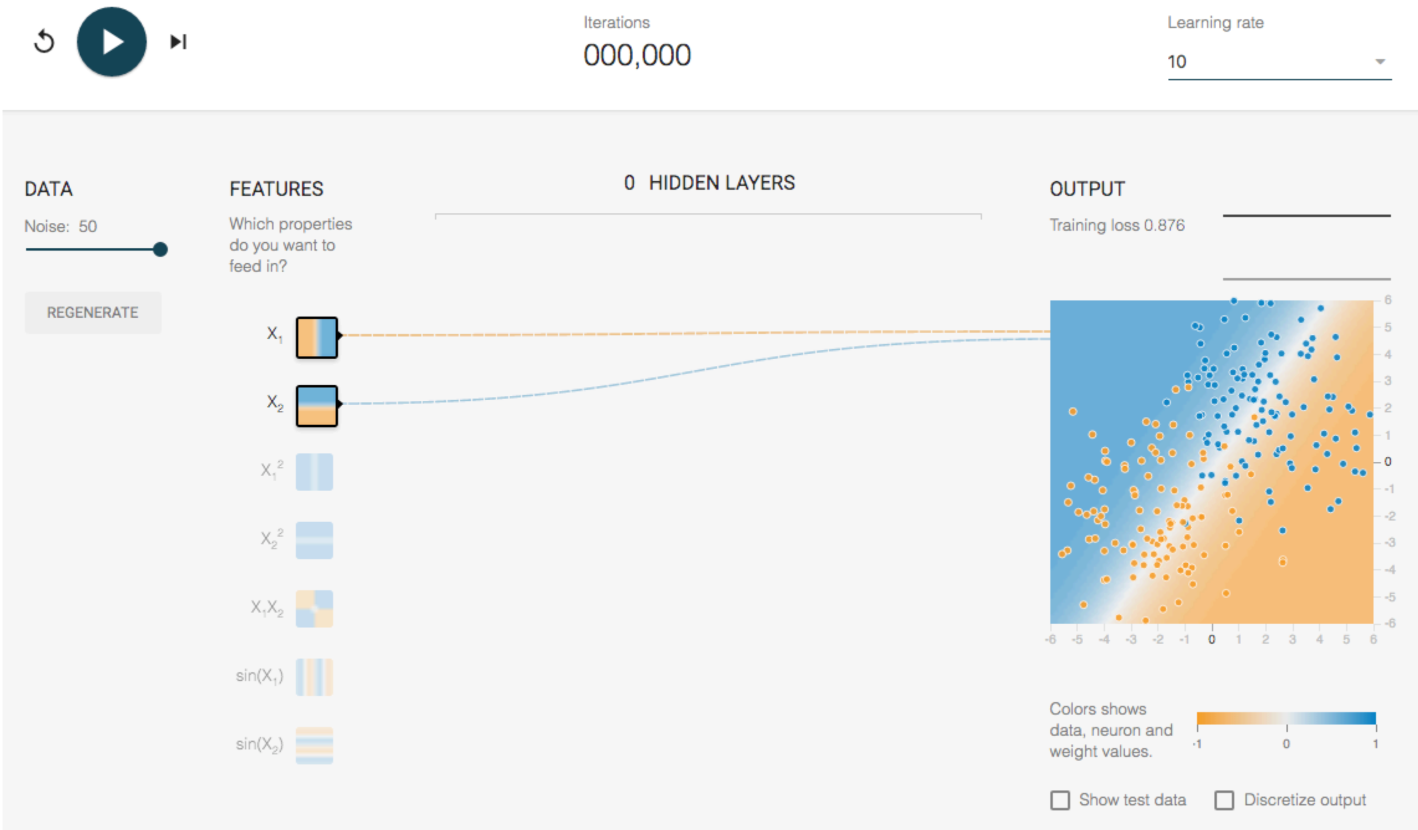
$$z = x_1 W_1 + x_2 W_2 + 1 \cdot W_3 = \vec{x}\,\boldsymbol{W}$$

$$p_1(x) = P(Y = 1 | X = x) = [1 + \exp(-x\,W)]^{-1} = f(\vec{x}\,\boldsymbol{W})$$

f called
Logit non-linearity

Logistic Regression

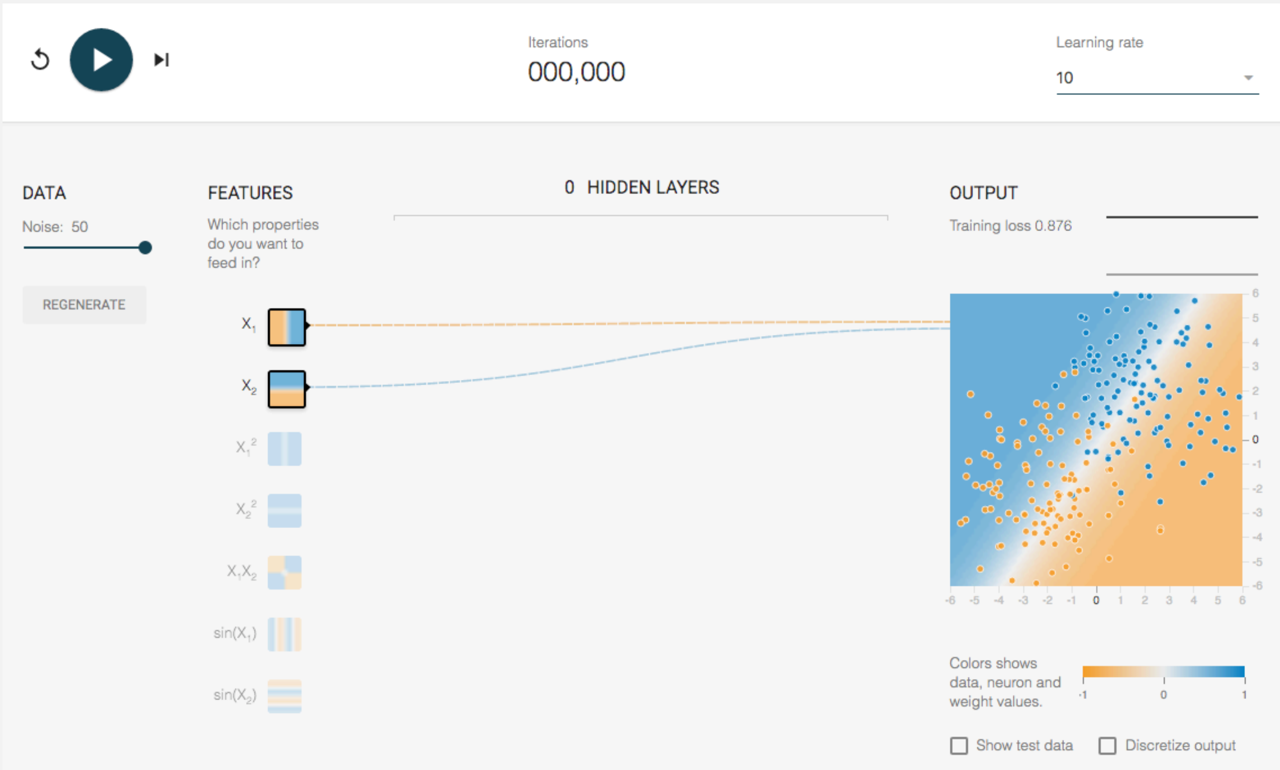
Explain TensorFlow playground



Logistic Regression [10 minutes]

Open the tensorflow playground and

a) Manually adjust the the weights to find best visual separation
b) Start learning with a learning rate 10 what happens?
c) Change learning rate to sensible values.



Notebook [30 minutes]

Please have a look at the logistic regression notebook:

A simple example for logistic regression

This notebook calculates a logistic regression using TensorFlow. It's basically meant to show the principles of TensorFlow.

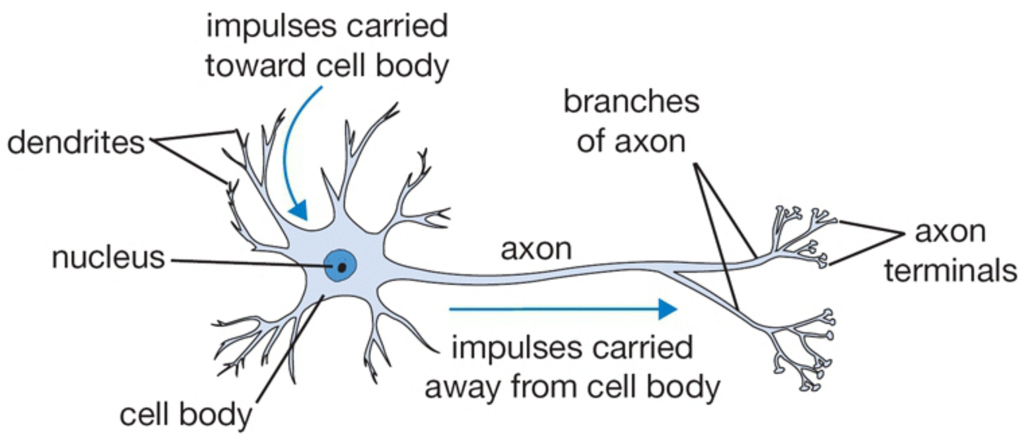
Datset

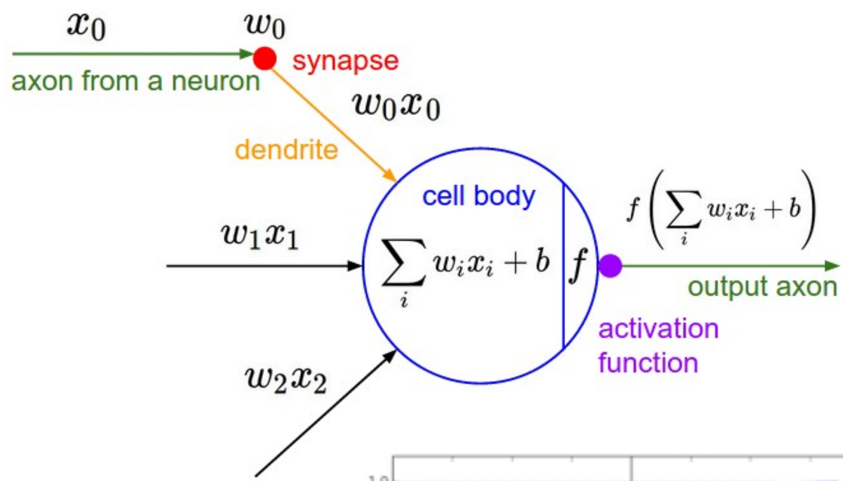
We investigate the data set of the challenger flight with broken O-rings (Y=1) vs start temperature.

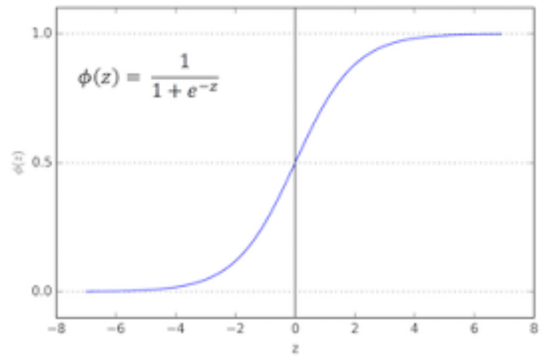
In [1]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as imgplot
import numpy as np
import pandas as pd
import tempfile
data = np.asarray(pd.read_csv('challenger.txt', sep=','), dtype='float32')
plt.plot(data[:,0], data[:,1], 'o')
plt.axis([40, 85, -0.1, 1.2])
plt.xlabel('Temperature [F]')
plt.ylabel('Broken O-rings')
```

Biological Interpretation

- In popular media neural networks are often described as a computer model of the human brain.



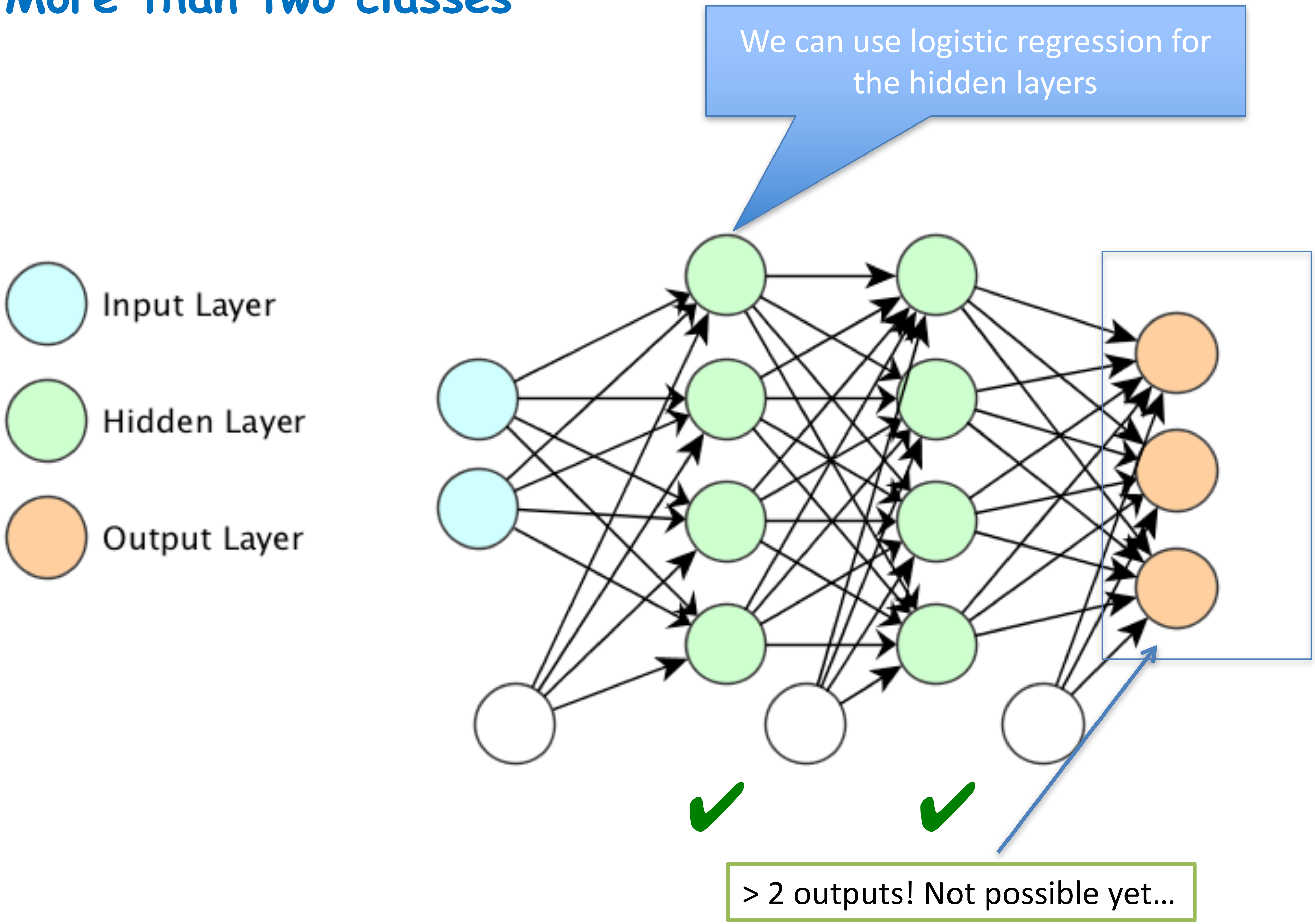




DL *loosely inspired* by how the brain works. Biological neurons are much more complicated.

Images from: http://cs231n.github.io/neural-networks-1/

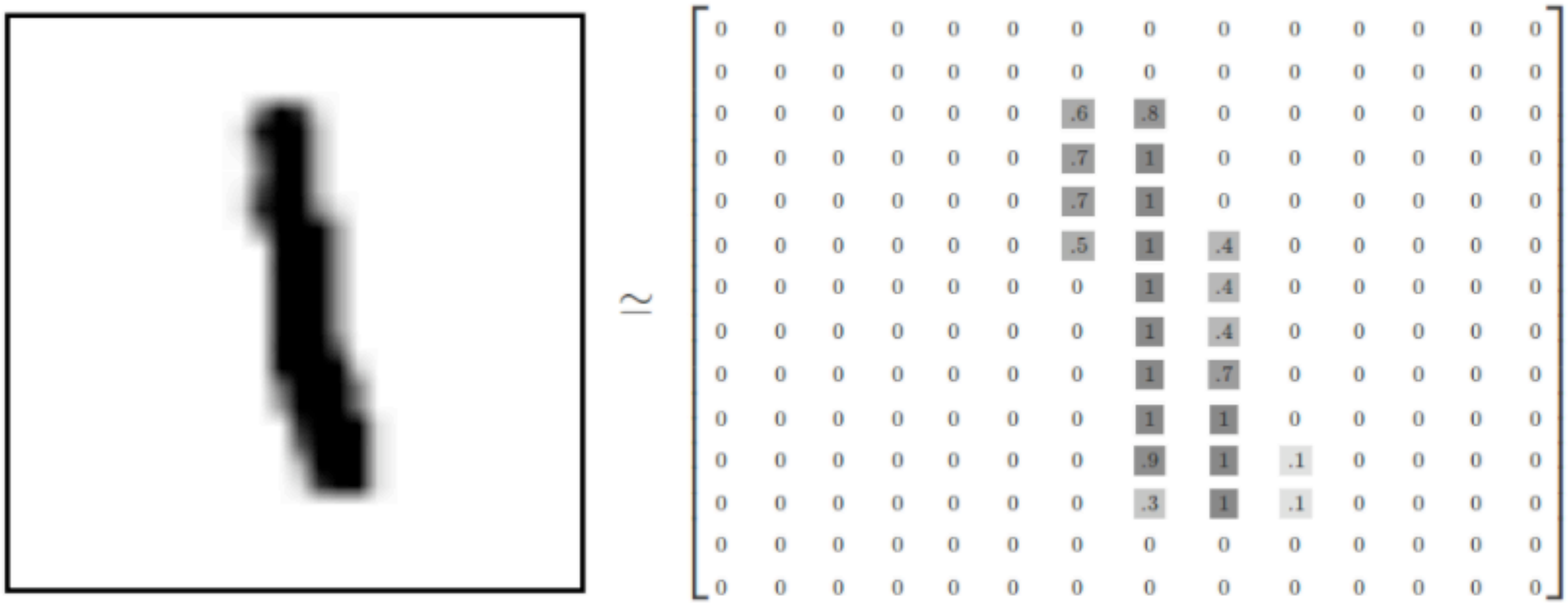
More than two classes



Multinomial Logistic Regression

Exercise: The MNIST Data Set

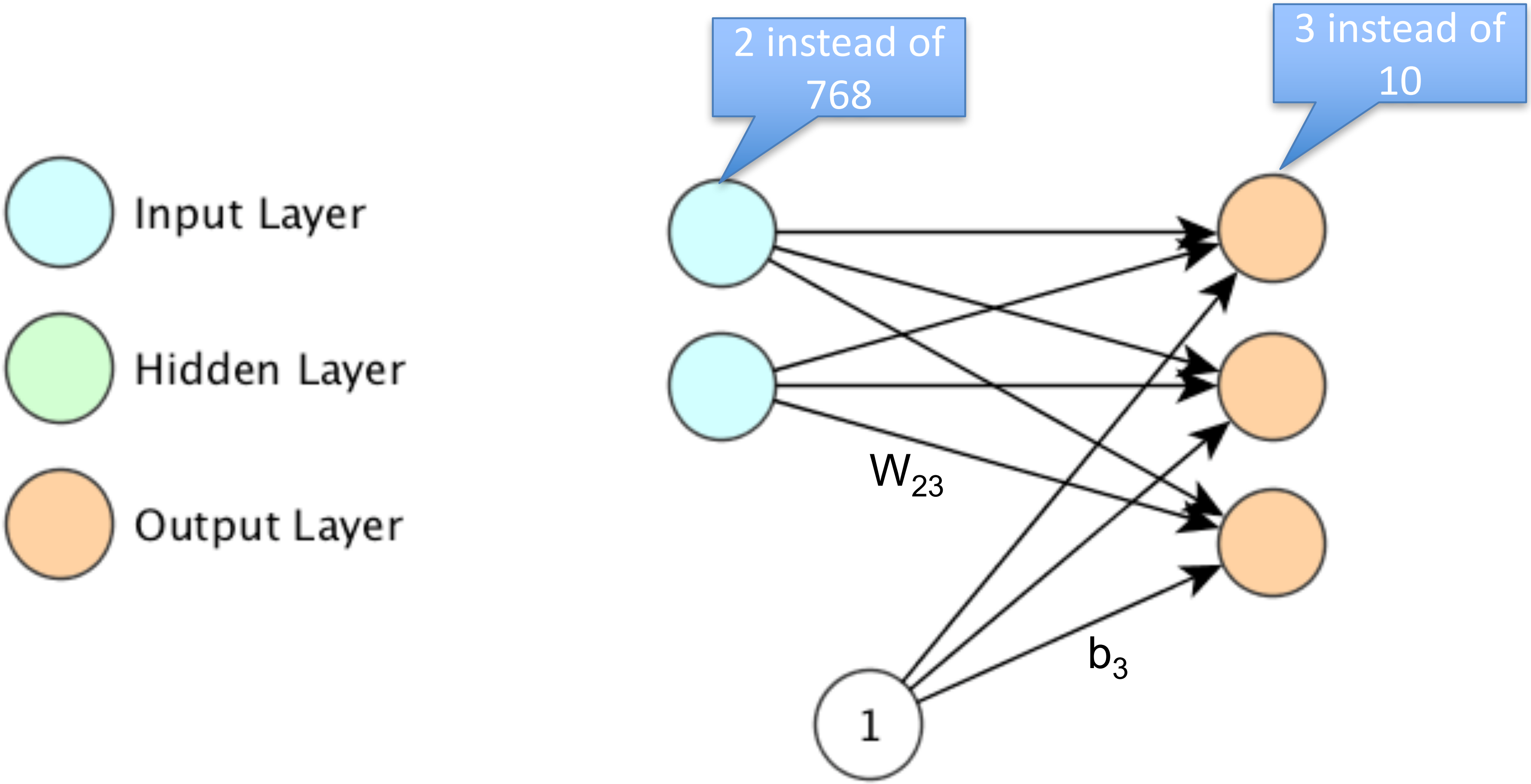
- MNIST the drosophila of all DL-Data sets
  - 50000 handwritten digits to be classified into 10 classes (0-9)



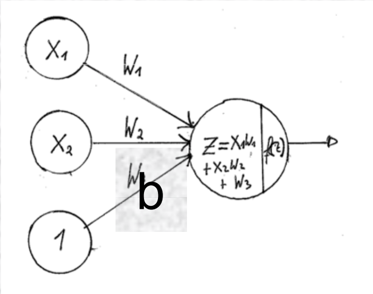
Input tensors: are flattened to 28*28=768 pixels

Image credit: https://www.tensorflow.org/versions/r0.10/images/MNIST-Matrix.png

Multinomial Logistic Regression



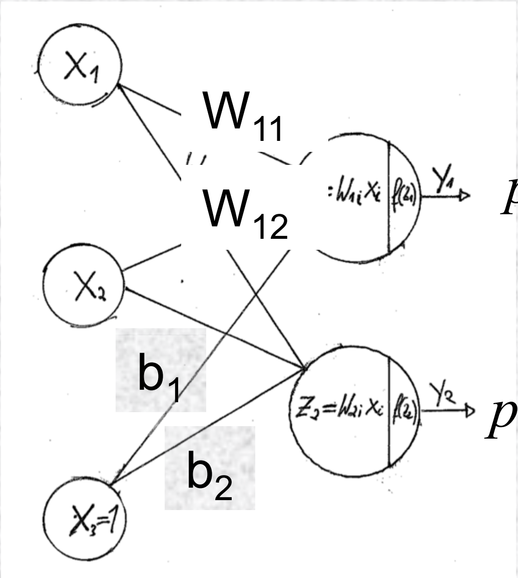
Multinominal Regression



Binary Case

$$P(Y=1\mid X=x)=\frac{1}{1+\exp(-z)}=\frac{\exp(\sum_i x_i W_i)}{1+\exp(\sum_i x_i W_i)}\propto \exp(\sum_i x_i W_i)$$

W_{12} = reads „from node 2 to 1“



More than one class

called logit

$$p_1 = P(Y_1=1\mid X=x)\propto \exp(\sum_i x_i W_{i1}+b_1) \qquad p_1=\frac{\exp(\sum_i x_i W_{i1}+b_1)}{\sum_j \exp(\sum_i x_i W_{ij}+b_j)}$$

$$p_2 = P(Y_2=1\mid X=x)\propto \exp(\sum_i x_i W_{i2}+b_2)$$

Normalisation

$$\sum_{i=1} p_i = 1$$

Multinomial case: just another **non-linearity *softmax***

$$p_1 = P(Y_1=1\mid X=x)=\frac{\exp(\sum_i x_i W_{i1}+b_1)}{\sum_j \exp(\sum_i x_i W_{ij}+b_j)}=\mathrm{softmax}(\sum_i x_i W_{i1}+b_1)$$

Recap: Matrix Multiplication aka dot-product of matrices

We can only multiply matrices if their dimensions are compatible.

A $\times$ **B** = **C**

(m $\times$ **n**) $\times$ (**n** $\times$ p) = (m $\times$ p)

$$\mathbf{A}_{3x3} \times \mathbf{B}_{3x2} = \mathbf{C}_{3x2}$$

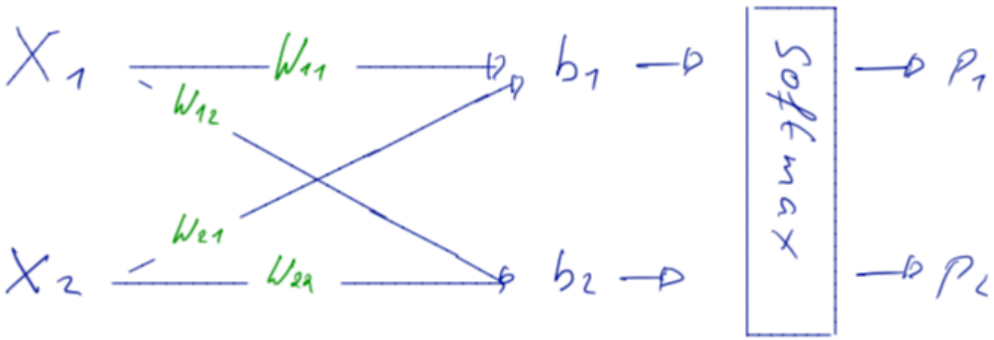
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \boxed{a_{31} \quad a_{32} \quad a_{33}} \end{bmatrix} x \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ \boxed{c_{31}} & c_{32} \end{bmatrix}$$

$$c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31}$$
$$c_{12} = a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32}$$
$$c_{21} = a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31}$$
$$c_{22} = a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32}$$
$$\boxed{c_{31} = a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31}}$$
$$c_{32} = a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32}$$

Example:

$$\mathbf{A}_{1x2} = \begin{pmatrix} \boxed{0 \quad 3} \end{pmatrix} \qquad \mathbf{B}_{2x3} = \begin{pmatrix} 3 & 1 & 7 \\ 8 & 2 & 4 \end{pmatrix} \qquad \mathbf{C}_{1x3} = \mathbf{A}_{1x2} \cdot \mathbf{B}_{2x3} = \begin{pmatrix} 24 & 6 & 12 \end{pmatrix}$$

GPUs love matrices (or tensors)



$$(p_1, p_2) = \mathrm{softmax}(x_1 W_{11} + x_2 W_{21} + b_1, x_1 W_{12} + x_2 W_{22} + b_2)$$

$$= \mathrm{softmax}\left((x_1, x_2)\begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix} + (b_1, b_2)\right)$$

$$p = \mathrm{softmax}(xW + b)$$

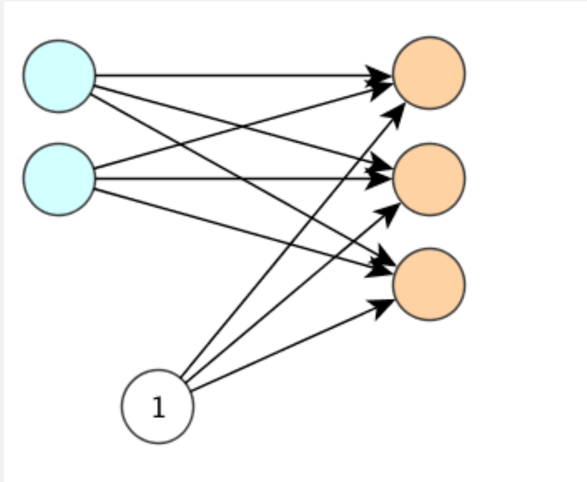
$$p_1 = P(Y_1 = 1 \mid X = x) = \frac{\exp(\sum_i x_i W_{i1} + b_1)}{\sum_j \exp(\sum_i x_i W_{ij} + b_j)} = \mathrm{softmax}(\sum_i x_i W_{i1} + b_1)$$

Your turn

Input x = (1,2)

W = $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

b = (1,2,3)



Calculate the output using numpy:

Hints:

```
x = np.asarray([[1,2]]) #
np.matmul(.,.) # Matrix multiplication
np.exp(.) # Exponential
np.sum(.) # Sum
```

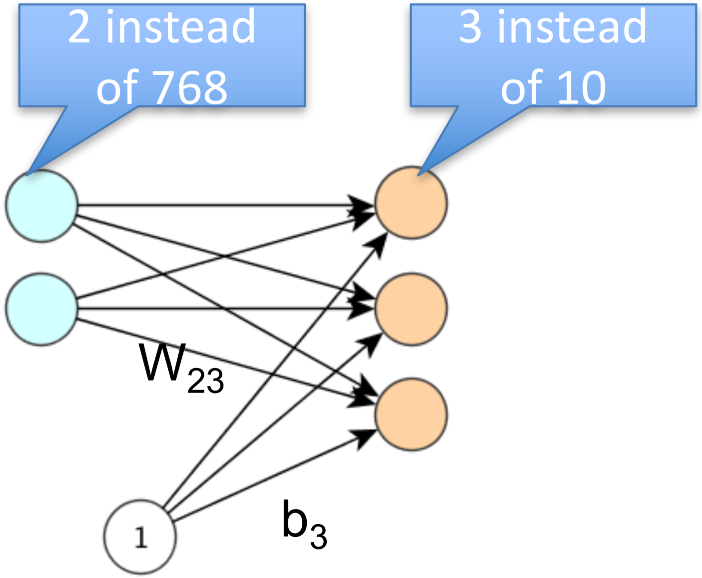
`#Result:` array([[3.29320439e-04, 1.79802867e-02, 9.81690393e-01]])

GPUs love matrices: Use the source luke

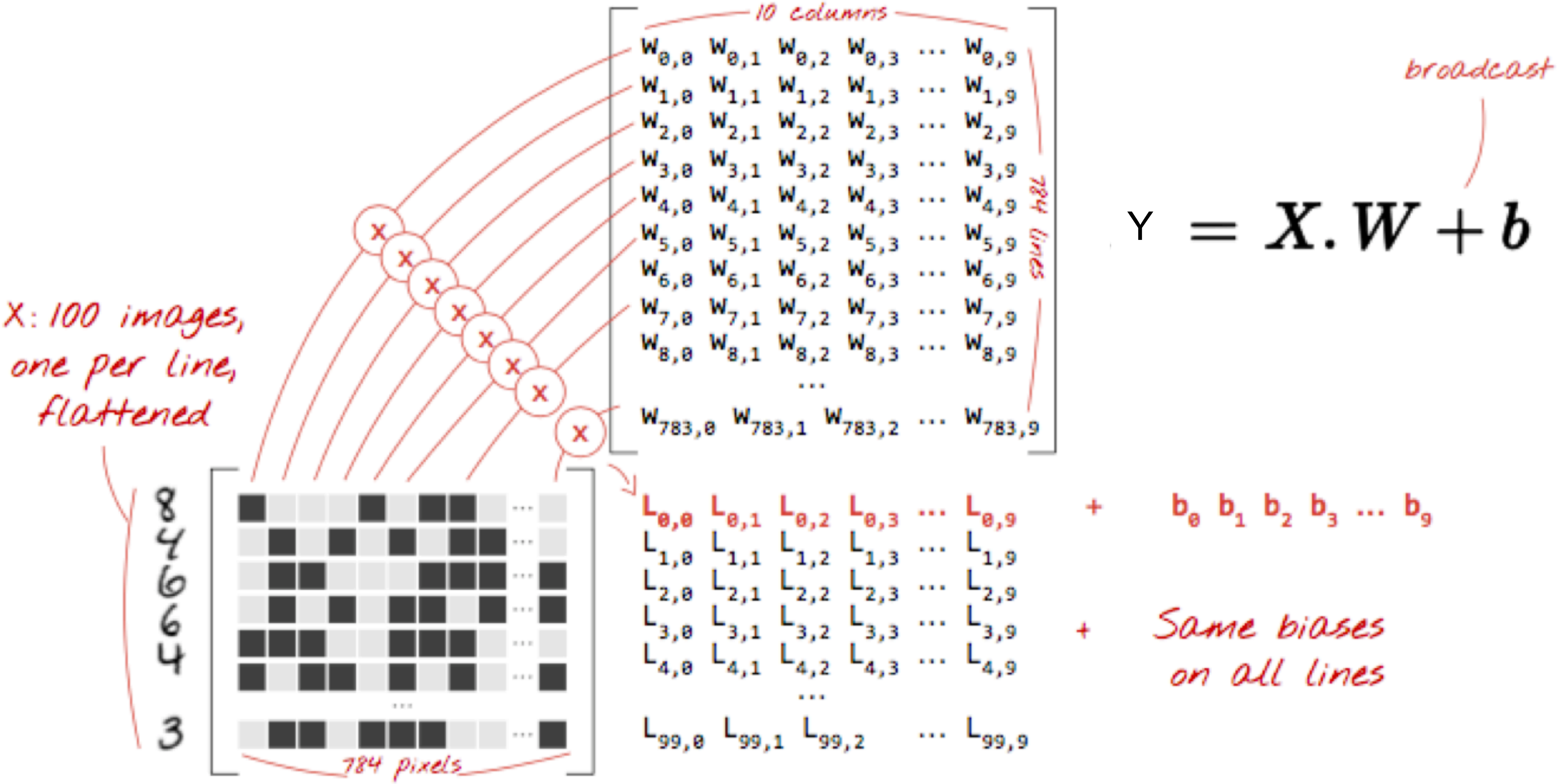
Mini batch size at runtime

```
…
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)
```

Data is usually processed in (mini-) batches. Instead of X being a 28*28=784 long vector, we use a batch (e.g. size 100)



GPUs love matrices:



$$Y = X.W + b$$

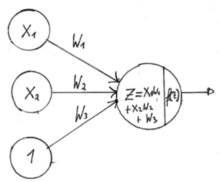
```
y = tf.nn.softmax(tf.matmul(x, W) + b)
```

Slide Credit: Martin Gröner TensorFlow and DL without a PhD https://docs.google.com/presentation/d/1TVixw6ltiZ8igjp6U17tcgoFrLSaHWQmMOwjlgQY9co/pub?slide=id.g110257a6da_0_431

Loss for multinomial regression

This is the prob. the model evaluates for the true class $y^{(i)}$ of training example $x^{(i)}$

Training Examples Y=1 or Y=0



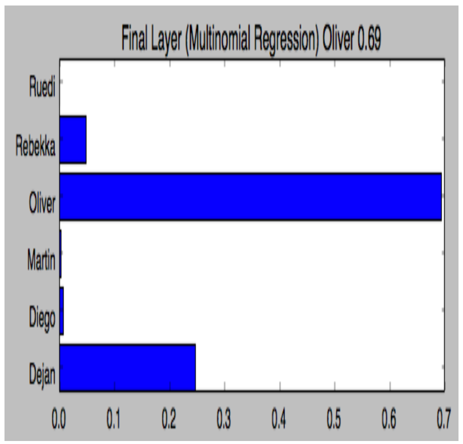
$$\text{loss} = -\frac{1}{N}\sum_{n=1}^{N}\log(p_{\text{model}}(y^{(i)} \mid x^{(i)};\theta))$$

$$\text{loss} = -\frac{1}{N}\sum_{n=1}^{N}\log(p_{\text{model}}(y^{(i)} \mid x^{(i)};\theta)) = -\frac{1}{N}\left(\sum_{i \in All\ ones}\log(p_1(x^{(i)})) + \sum_{i \in All\ zeros}\log(p_0(x^{(i)}))\right)$$

N Training Examples classes (1,2,3,…,K)

$$\text{loss} = -\frac{1}{N}\sum_{n=1}^{N}\log(p_{\text{model}}(y^{(i)} \mid x^{(i)};\theta)) = -\frac{1}{N}\left(\sum_{i \in y_j=1}\log(p_1(x^{(i)})) + \sum_{i \in y_j=2}\log(p_2(x^{(i)})) + \ldots + \sum_{i \in y_j=K}\log(p_K(x^{(i)}))\right)$$

p_i



Output of last layer

Example: Look at class of single training example. Say it's Dejan, if classified correctly p_dejan = 1 ➔ Loss = 0. Real bad classifier put's p_dejan=0 ➔ Loss = Inf.

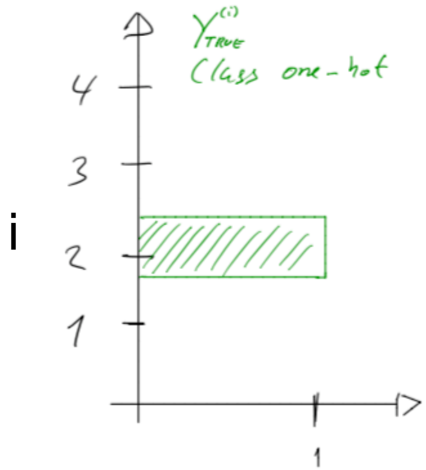
One more Trick: Loss function with indicator function

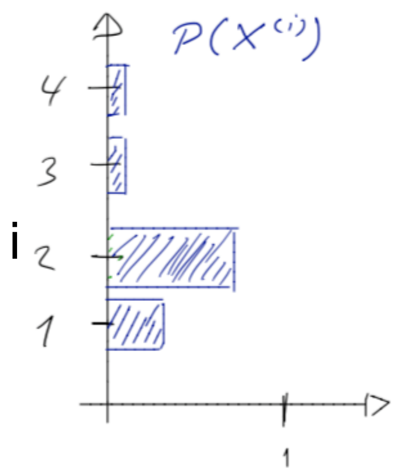
A one-hot-encoded y picks the right class, form all of the K different classes.

For MNIST K=10, so why calculate, 9 logs and through them away?
(Parallel executions)

$$-N\cdot \text{loss} = \sum_{i\in y_j=1}\log(p_1(x^{(i)})) \quad + \sum_{i\in y_j=2}\log(p_2(x^{(i)})) + \ldots + \sum_{i\in y_j=K}\log(p_K(x^{(i)})) = \sum_{i=1}^{N} y^{(i)}{}_{\text{true}}\log(p(x^{(i)})) = \sum_{i=1}^{N} y^{(i)}{}_{\text{true}}\log(y_i)$$

one-hot-encoded

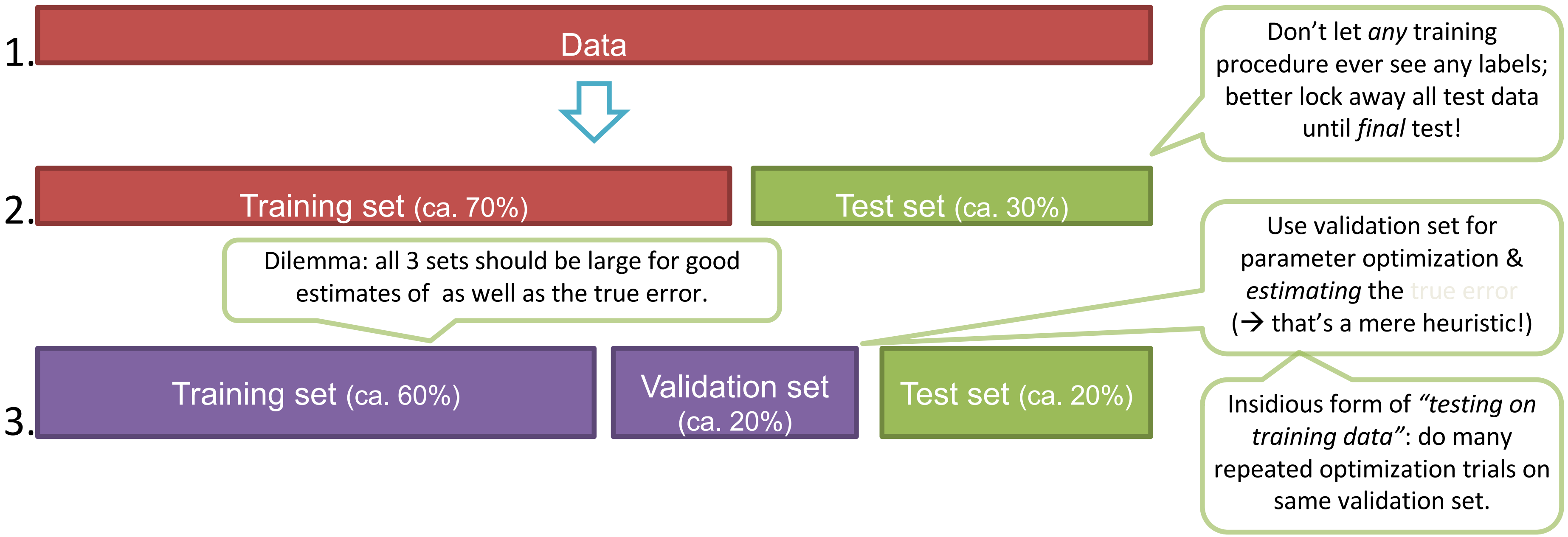




$$Loss = \frac{-1}{N}\sum_i Y^{(i)}_{TRUE} \ln P(X^{(i)})$$

See later crossentropy and KL-Distance between y_i and $p(x^{(i)})$

Training Neural Networks: Split of the data

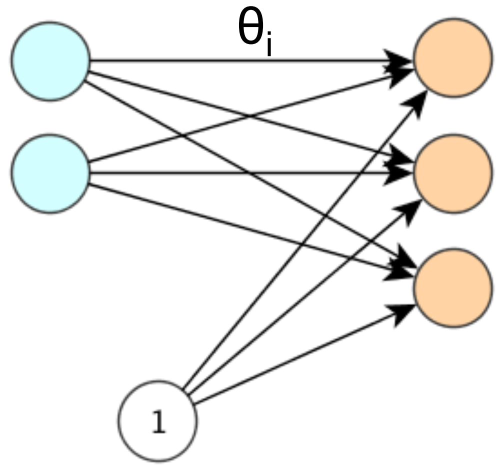


Taken from V03 ModelAssessment. For neural networks no cross-validation is done (long learning times).

For our use case (4000 images)

- Training set 3000, Test set 1000
- 20% of the Training set is taken as Validation Set

Stochastic gradient descent



- The loss function

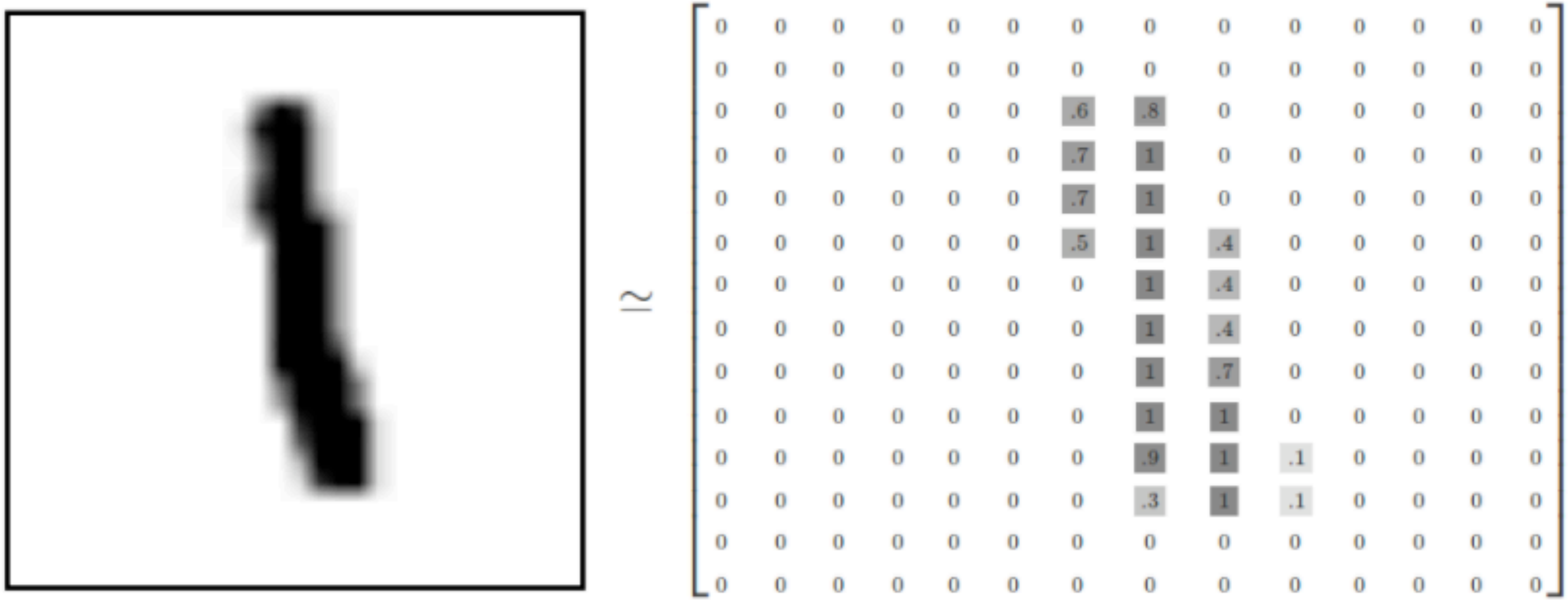
$$\text{loss} = -\frac{1}{N}\sum_{n=1}^{N}\log(p_{\text{model}}(y^{(i)} \mid x^{(i)};\theta))$$

- A particular weight is calculated by the partial derivative of the loss function w.r.t θ_i
- The sum is taken over the whole training set of size N. Often the training set is split into mini-batches size of e.g. b=128 (*)
- These mini-batches are processed one after another
- When all examples have been processed once, we speak of one epoch being finished
- For a new epoch one often reshuffles the data

- The batch size is chosen so that input tensor fits on the GPU.

(*) For some purists only when b=1 is called stochastic gradient descent

Exercise: The MNIST Data Set



Input tensors

One minibatch has dimension (128, 28, 28, 1) (batch, x,y, color)

or (128, 784) flattened

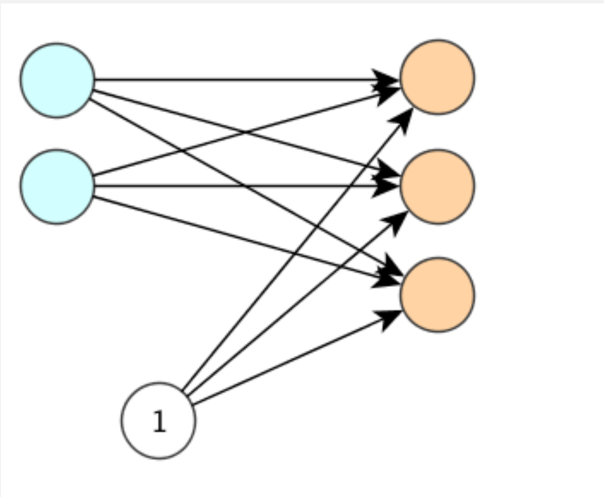
Image credit: https://www.tensorflow.org/versions/r0.10/images/MNIST-Matrix.png

Exercise: Implement multinomial logistic regression

Finish the code in the notebook: **Multinomial Logistic Regression**

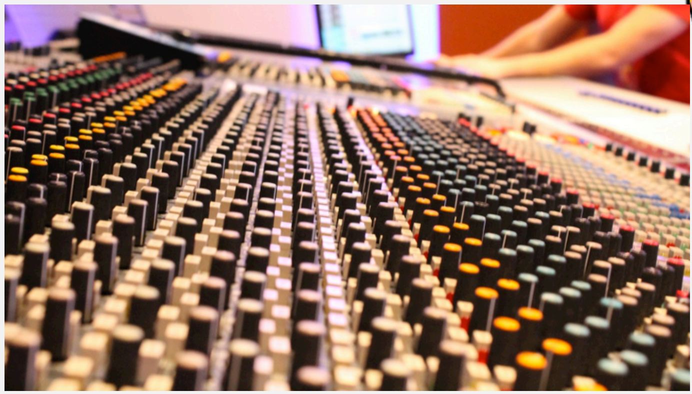
- Think about the trick how the loss is calculated!
- Compare the loss and accuracy in the validation set with the loss in the training set. Why is there such a difference?
- Question: How many parameters do we have?

Hints:



$$p_j = \frac{\exp(\sum_i x_i W_{ij} + b_j)}{\sum_j \exp(\sum_i x_i W_{ij})} = \left(\text{softmax(xW+b)}\right)_j$$

SOLUTION



- We have
  - For W 28*28*10 = 7840 Parameter
  - For b 10 Parameter
  - Together 7850 Parameters

- Trick with the loss function
  - `loss = tf.reduce_mean(-tf.reduce_sum(y_true * tf.log(y_pred), reduction_indices=[1]))`

- See: https://github.com/tensorchiefs/dl_course/blob/master/notebooks/05_Multinomial_Logistic_Regression_solution.ipynb

- https://github.com/tensorchiefs/dl_course/blob/master/notebooks_misc/Explanation_of_loss.ipynb

Alternative solution

```
w = tf.Variable(tf.random_normal([784, 10], stddev=0.01))
b = tf.Variable(tf.zeros([10]))
z = tf.matmul(x,w)+b #aka logits
loss = tf.reduce_mean(
       tf.nn.softmax_cross_entropy_with_logits(labels=y_true,logits=z)
)

#Old Solution
prob = tf.nn.softmax(z)
loss_old = tf.reduce_mean(-tf.reduce_sum(y_true * tf.log(prob),
reduction_indices=[1]))
```

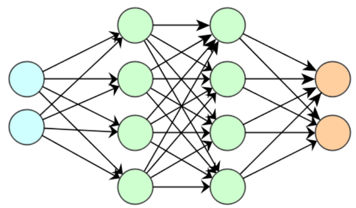
For numerical stability, one should use

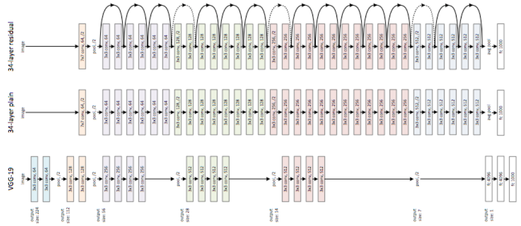
`tf.nn.softmax_cross_entropy_with_logits`

There is also a sparse version (no one hot encoded needed)

`tf.nn.sparse_softmax_cross_entropy_with_logits`

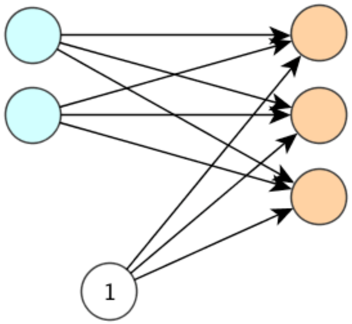
Learning Objectives





- Increase our knowledge in TF

- Foundations of DL
  - **Loss Function (what to minimize)**
    - Mean Squared Error
    - Loss Function for classification
      - Binary cross entropy loss for logistic regression
      - Cross entropy loss for multinomial logistic regression
    - Two principles to construct loss functions
      - Maximum Likelihood Principle
      - Cross Entropy
  - Gradient Descent
    - How to minimize



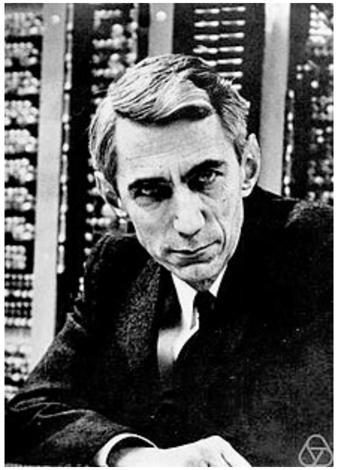
We use networks with no hidden layers to explain basics. Loss function and gradient descent stay the same.

Now we have the tools. Next time, we go deeper, promised.
Make yourself aware of the chain rule.

Backup

Why the hack they call it cross entropy?

Entropy and Cross Entropy



- The central loss function for classification is called cross entropy, why?
- This is a different viewpoint to the max-likelihood approach.
- See also
  - https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/
  - https://www.quora.com/Whats-an-intuitive-way-to-think-of-cross-entropy
  - https://www.khanacademy.org/computing/computer-science/informationtheory/moderninfotheory/v/information-entropy
  - https://medium.com/swlh/shannon-entropy-in-the-context-of-machine-learning-and-ai-24aee2709e32

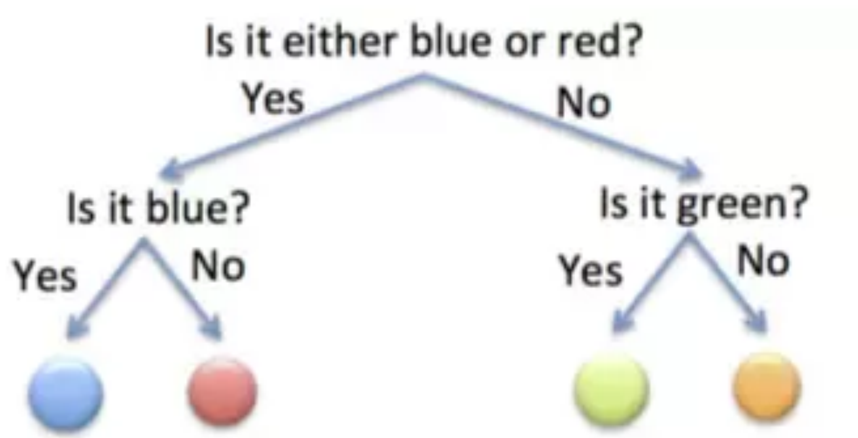
- Let's start by defining the (information) entropy
  - It's somewhat like the amount of surprise you get from a sample.
  - Let's first do an simple example

Image: Wikipedia

Information Content of a single outcome

- 4 Balls each with same probability 25%

- How can your friend ask you which ball you picked, with minimum number of questions?



Let's say we have a red ball.
Two questions need to be ask.

Coding for red ball (yes=1)
10 // Information content 2 bits

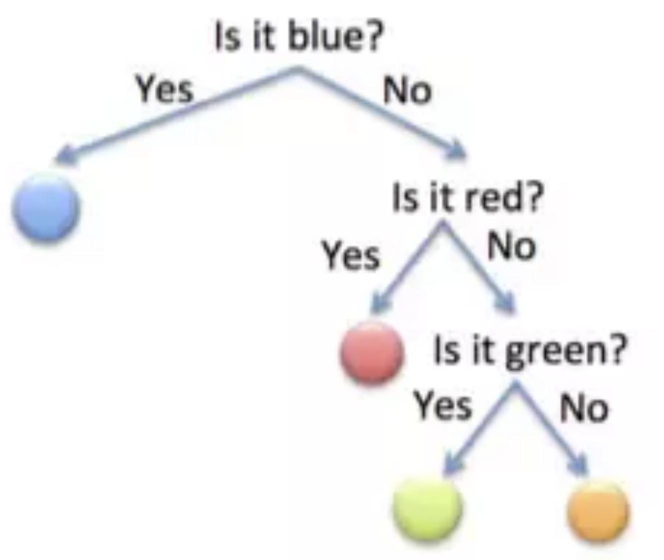
Coding for orange (your turn)
00 // Information content 2 bits

Example form https://www.quora.com/Whats-an-intuitive-way-to-think-of-cross-entropy

Information Content of a single outcome

- 4 Balls each with different probability 50%, 25%, 12.5%, 12.5%

- How can your friend ask you which ball you picked, with minimum number of questions (on average)?



Let's say we have a blue ball.
One questions need to be ask.

Coding for blue ball (yes=1)
1 // Information content 1 bit

Coding for red (2 questions)
01 // Information content 2 bit

Coding for green (your turn)
001 // Information content 3 bit

Example form https://www.quora.com/Whats-an-intuitive-way-to-think-of-cross-entropy

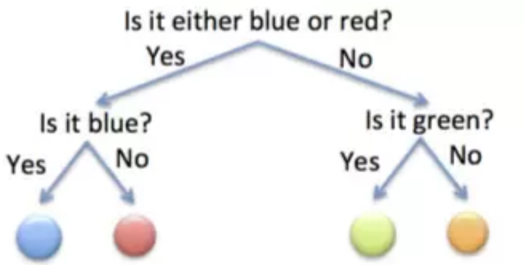
Information Content

- For that easy example, we found the best coding by hand.
- Let's define the (self-) information (Turns out to be the minimal coding length "Shannon's source coding theorem")

- Requirement for Information (or surpirse)
  - p_i the probability of event i (or prob. that symbol i occurs)
  - Seldom examples should have more surprise.
    - $I(p_i)$ should be monotonic decreasing function
  - Information should be non-negative
    - $I(p_i) \geq 0$
  - Uninformative, or sure events should have no Information
    - $I(p_i) = 0$
  - Information of independent events i, j should add up
    - $I\left(p_{(i,j)}\right) = I\left(p_i p_j\right) = I(p_i) + I(p_j)$
- ➔ $\boldsymbol{I(p) = -\log_2(p)}$
  - (defined up to basis), 2 is often chosen

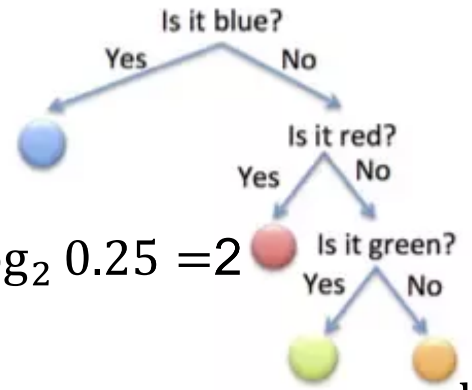
Information Content → Entropy

- Entropy (average Information Content)
  - $H(p) = \sum p_i I(p_i) = -\sum p_i \log_2(p_i)$



$-\log_2 0.25 = 2$

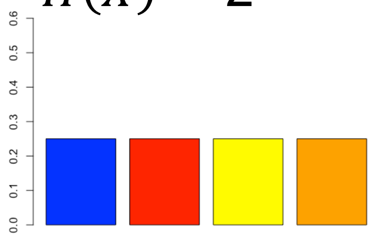
$-\log_2 0.5 = 1$



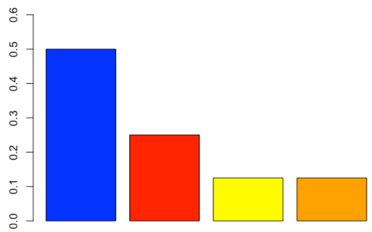
$-\log_2 0.25 = 2$

$-\log_2 0.125 = 3$

$H(X) = 2$



$H(X) = 0.5 + 0.25*2+0.25*3=1.75$



In general: Maximal Entropy if uniform, minimal if peaked (see also in physical Systems)

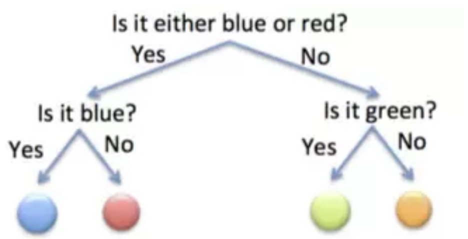
Cross Entropy

- If we know the distribution p, we can find the best coding and need H bits on average

- If we have a “wrong” distribution q how many bits do we need on average

  - $H(p,q) = -\sum p_i log2(q_i) \geq H(p)$

- Example, we think symbols come uniform distributed q. But they come (0.5,0.25,0.125,0.125)



Optimal Coding Scheme for Uniform q

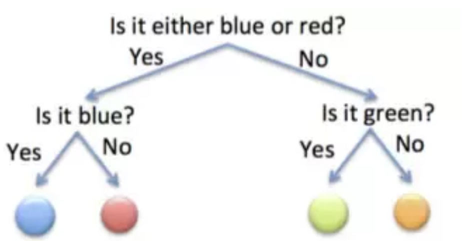
$$H(p,q) = 0.5\ 2 + 0.25\ 2 + 0.125\ * 2 + 0.125\ 2 = 2 > 1.75$$

KL-Divergence

- If we have a “wrong” distribution q how many bits do we have more than the minimal possible amount $H(p)$

$$-\ D_{KL}(p||q) = H(p,q) - H(p) \geq 0$$

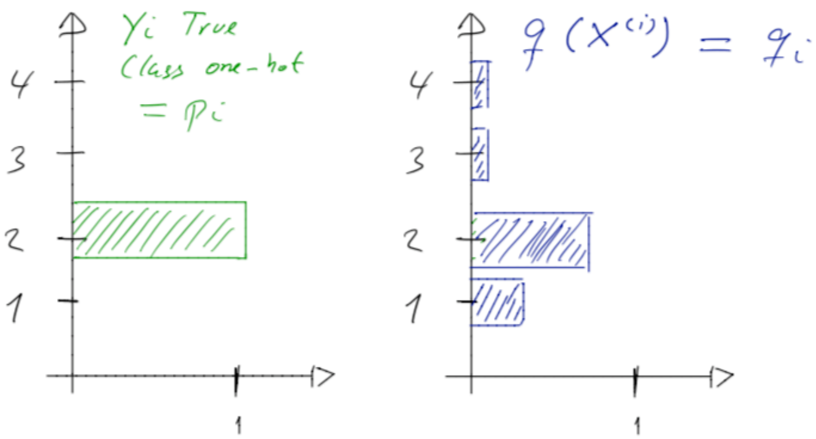
- Example, we think symbols come uniform distributed q. But they come (0.5,0.25,0.125,0.125)



Optimal Coding Scheme for Uniform q

$$D(p,q) = H(p,q) - H(p) = 2 - 1.75 = 0.25$$

Cross Entropy in DL



$H(p,q) = -\sum p_i ln q_i$ (for one example of the training set)

$H(p,q) = -\sum\sum p_i^{(j)} ln q_i^{j}$ (for the training set)

We minimize the cross entropy by changing q, the minimum is reached when q is identical to distribution of real labels p

Alternatively we could also minimize the KL-Divergence

Just for curiosity