Machine Intelligence:: Deep Learning
Week 2

Oliver Dürr

Institut für Datenanalyse und Prozessdesign

Zürcher Hochschule für Angewandte Wissenschaften

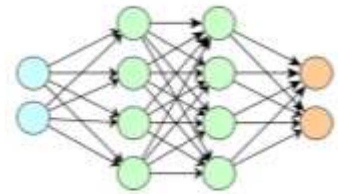
Winterthur, 27. Feb. 2018

Organizational Issues: Times

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

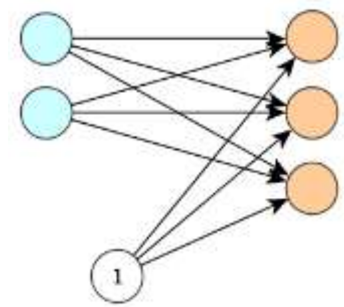
- Please interrupt us if something is unclear!

Learning Objectives



- Increase our knowledge in TF

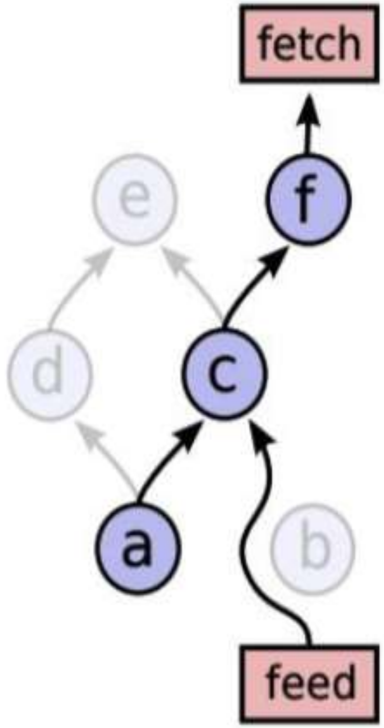
- Foundations of DL
  - **Loss Function (what to minimize)**
    - Loss Function for Regression
      - Mean Squared Error
    - Loss Function for classification
      - Binary cross entropy loss for logistic regression
      - Cross entropy loss for multinomial logistic regression
    - Two principles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



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

Recap from last week

Recap: Feed and Fetch




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

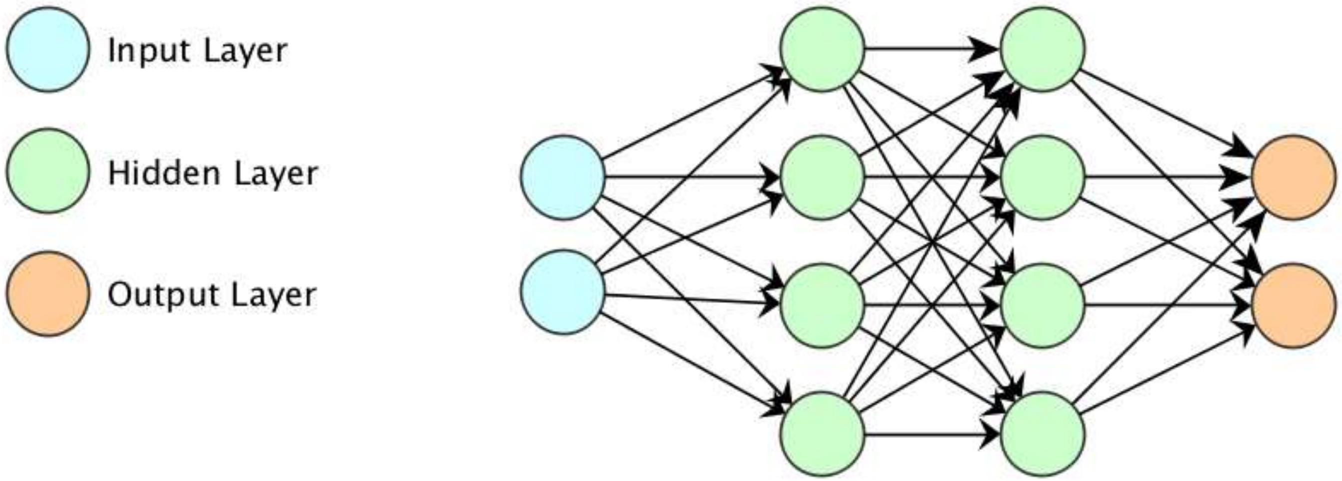
fetch
(the numeric value)

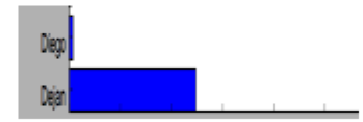
Fetch
f (symbolic)

symbolic

values

Preview: The first network

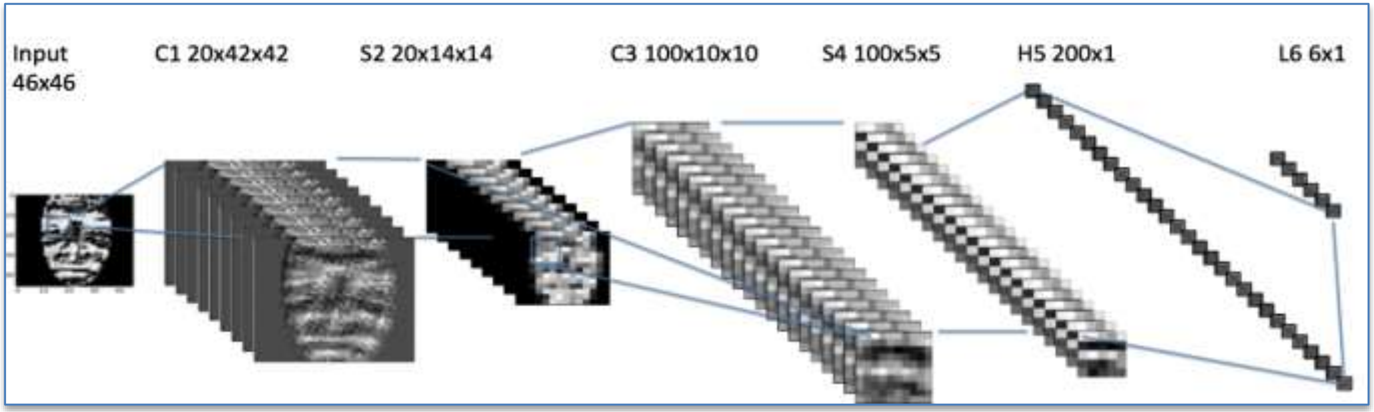


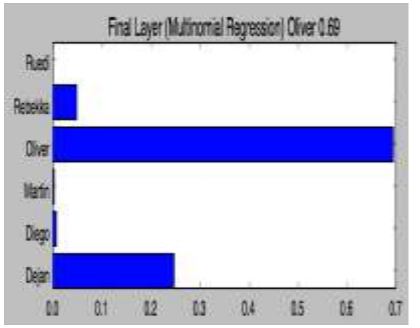


- The input: e.g. intensity values of pixels of an image
- Information is processed layer by layer
- Output: probability that image belongs to certain person

- Arrows are weights (these need to be learned)

Preview: Convolutional Neural Network (CNN)





- The input: e.g. intensity values are arrays (x,y)
- Inner layers: (x,y,z)
- Output: probability that image belongs to certain person

Tuning a neural network: a loss function

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

$\boxed{x^{(i)} \rightarrow \text{model parametrized with weights} \rightarrow \hat{y}^{(i)}}$

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

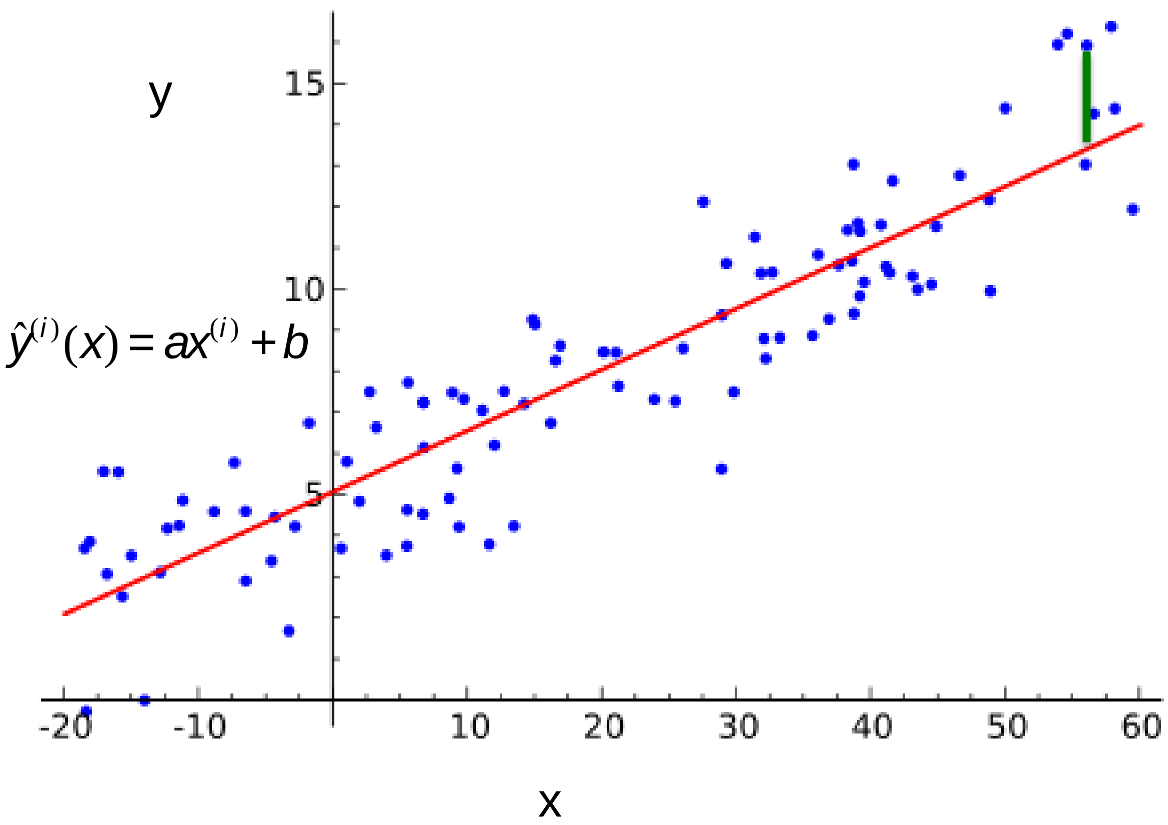
- Examples (your task what are the x's what are the y'?)
  - Facerec.: Faces and Names
  - Age Prediction: Faces and Age (numerical problem)

- 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

Loss for linear regression: sums of squared error

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



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)

Optimization

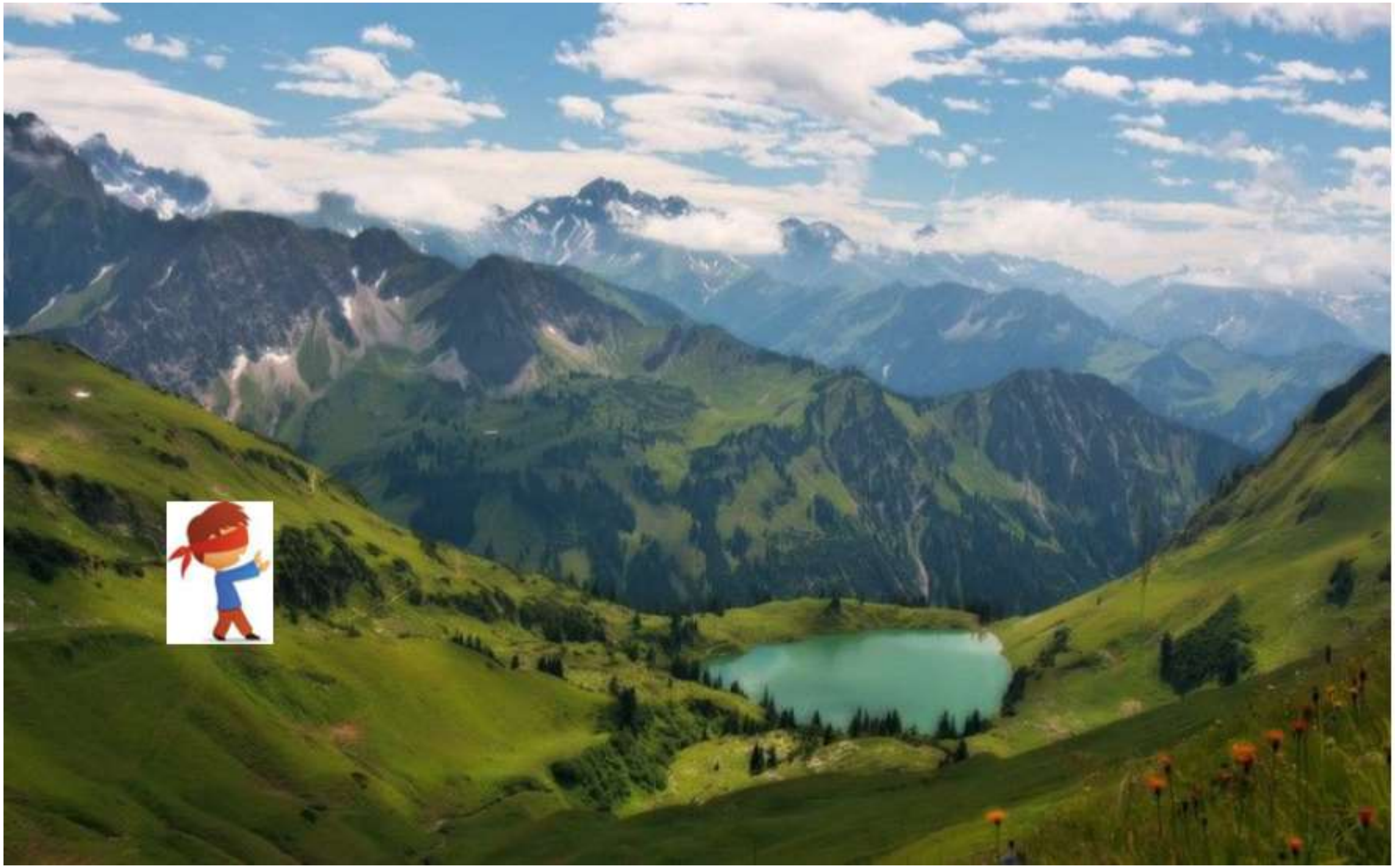
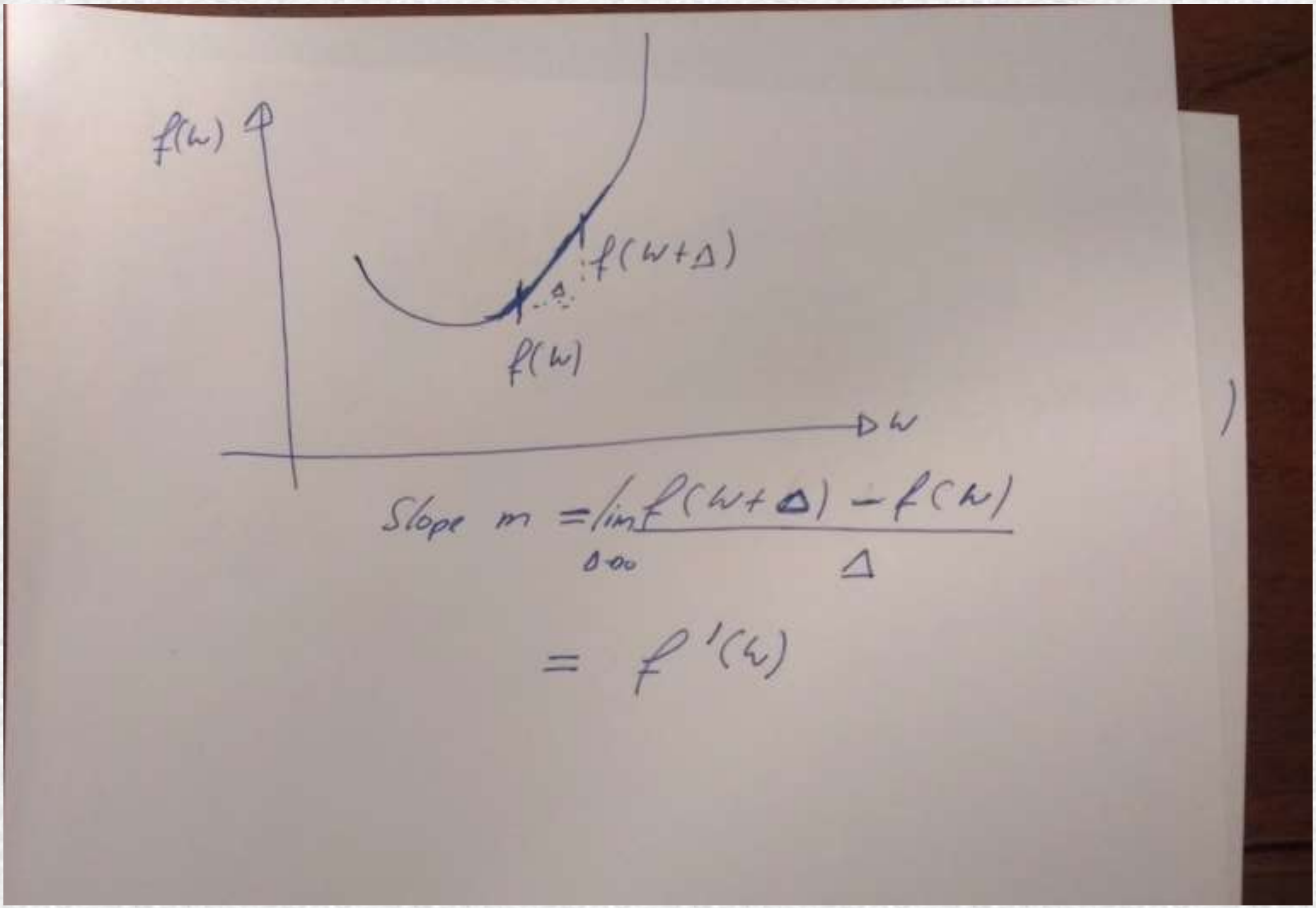


Figure shows a 2 dimensional loss function. In DL Millions!
We just know the current value (blind)

Slide from cs229

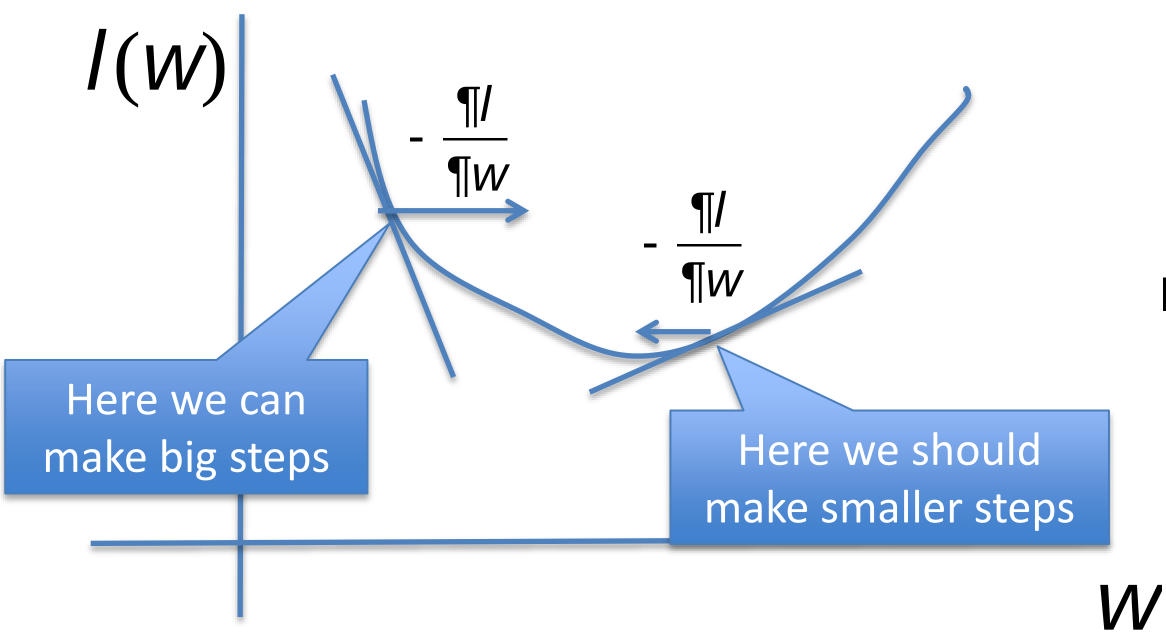
Gradient Descent: Gradient of 1-D function

- Draw



Optimization

- Intuition of Gradient Descent
  - We just know the value of the current loss function
  - The gradient gives us the direction and slope of the steepest descent
  - We just take a single step in direction of the steepest descent
  - The steeper the curve the larger the possible step
- Gradient in 1d (derivative)



$$-\frac{\partial l}{\partial w}$$

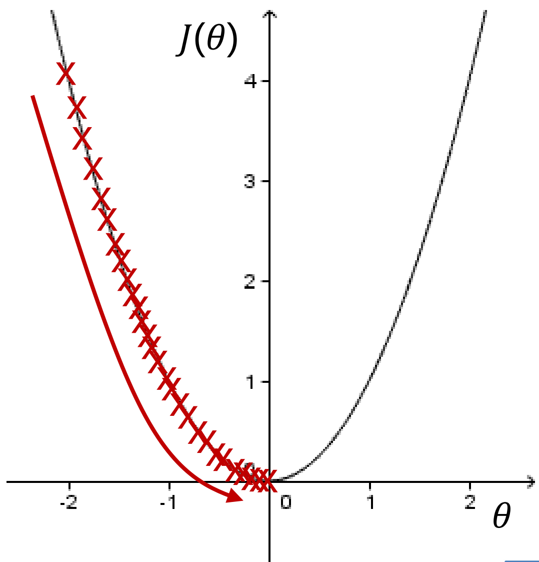
- Points to the downward direction.
- The magnitude is proportional to the slope

Iterative update

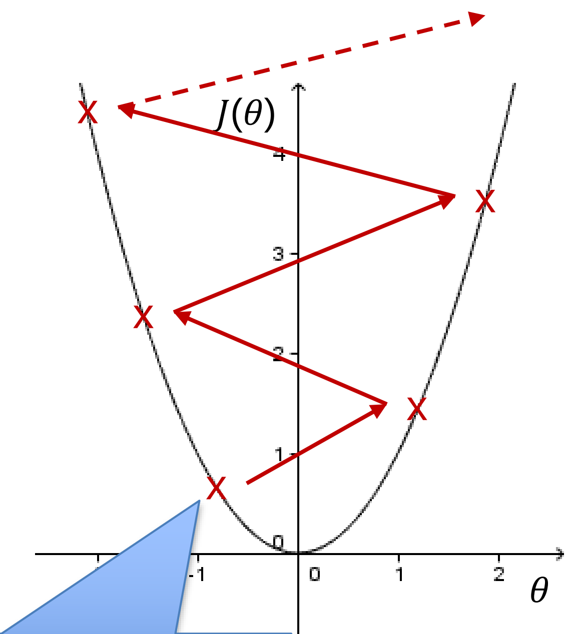
$$w^{t+1} = w^t - \epsilon \left.\frac{\partial l}{\partial w}\right|_{w^t}$$

Problem with step size ε

- too small
  ➔ gradient descent is slow



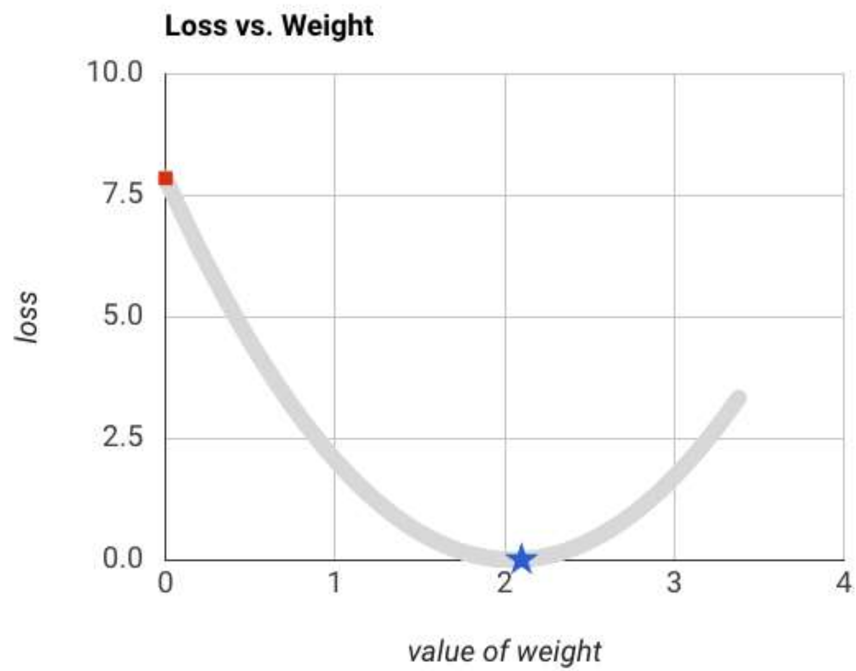
- too large
  ➔ gradient descent overshoots minimum
  ➔ no convergence or divergence!



Note that the points refer to θ.
You could also drawn them below at the axis.

Slide credit: Thilo

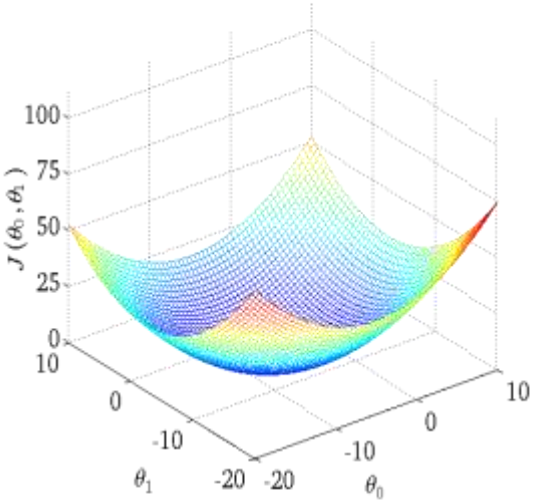
Für’s nächste Mal

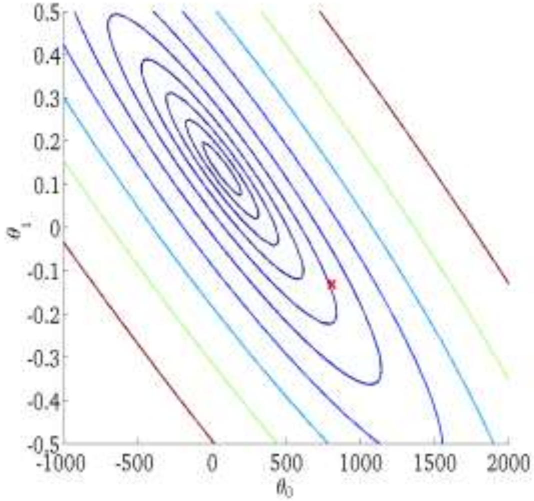


https://developers.google.com/machine-learning/crash-course/fitter/graph

Optimization

- 2 equivalent representations





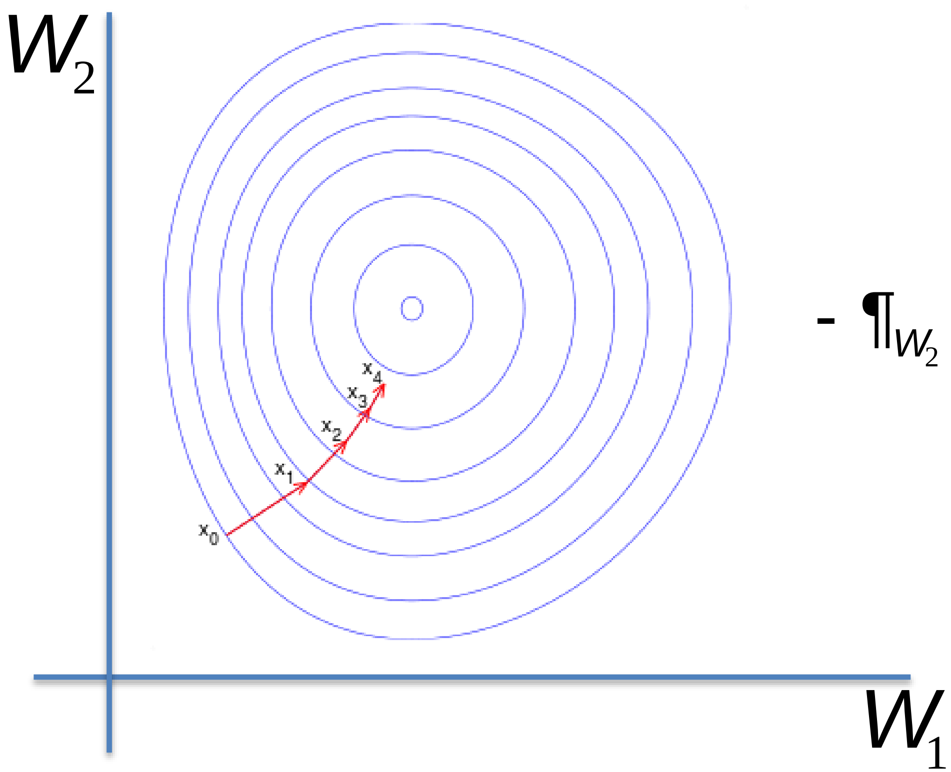
Slide credit: Thilo, wikipedia

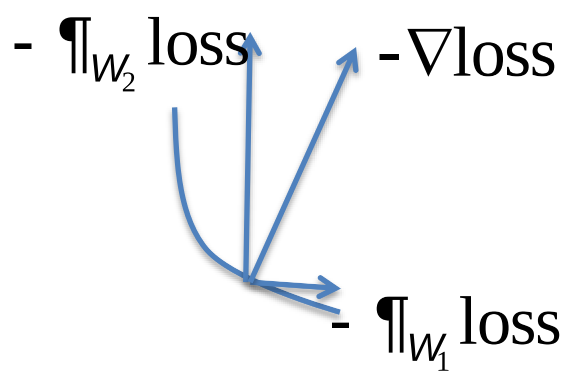
Optimization

- Gradient Descent

  Gradient is perpendicular to levels

$$W_i^{t+1} = W_i^t - e\P_{W_i} \text{loss}$$



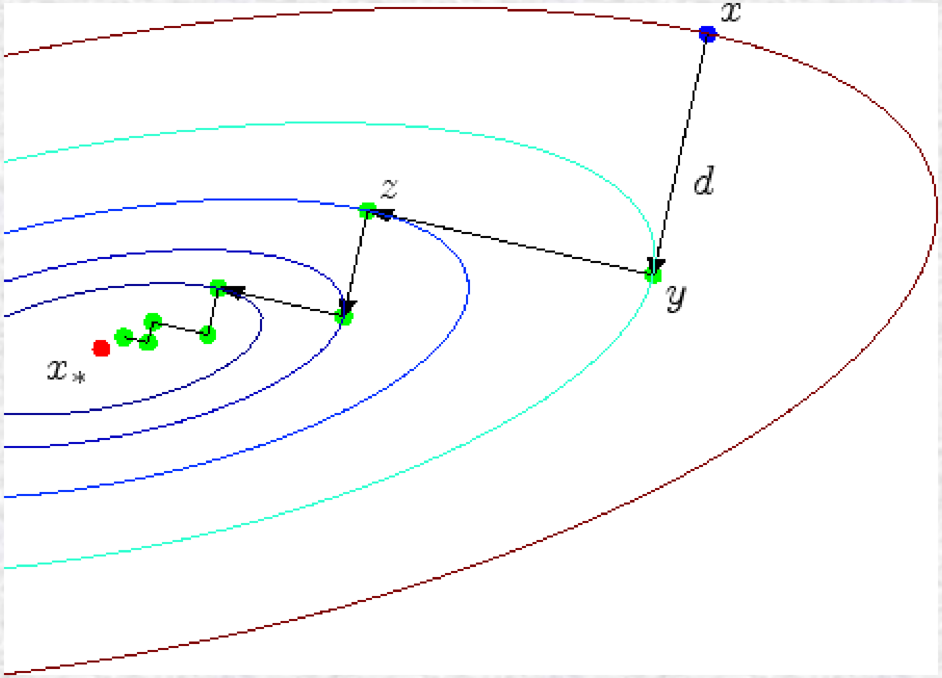


Slide credit: wikipedia

Tafel

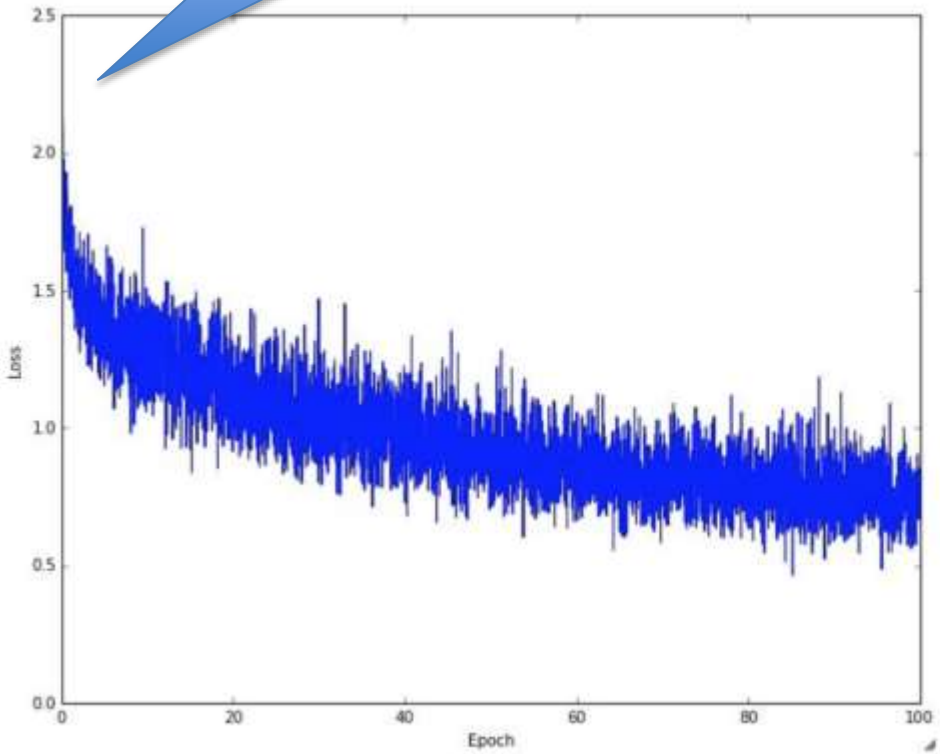
- Höhenlinien einzeichenen und runterhüpfen
  - Gradient senkrecht zu Höhenlinien

- Make clear that this is an iterative

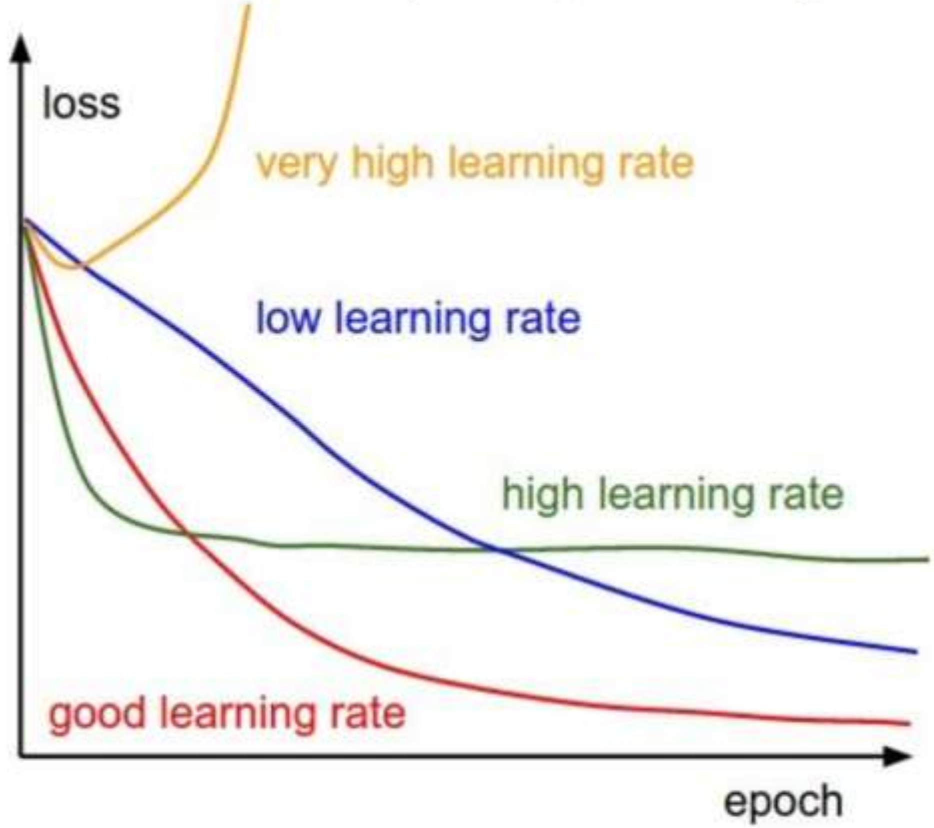


Gradient Descent in DL

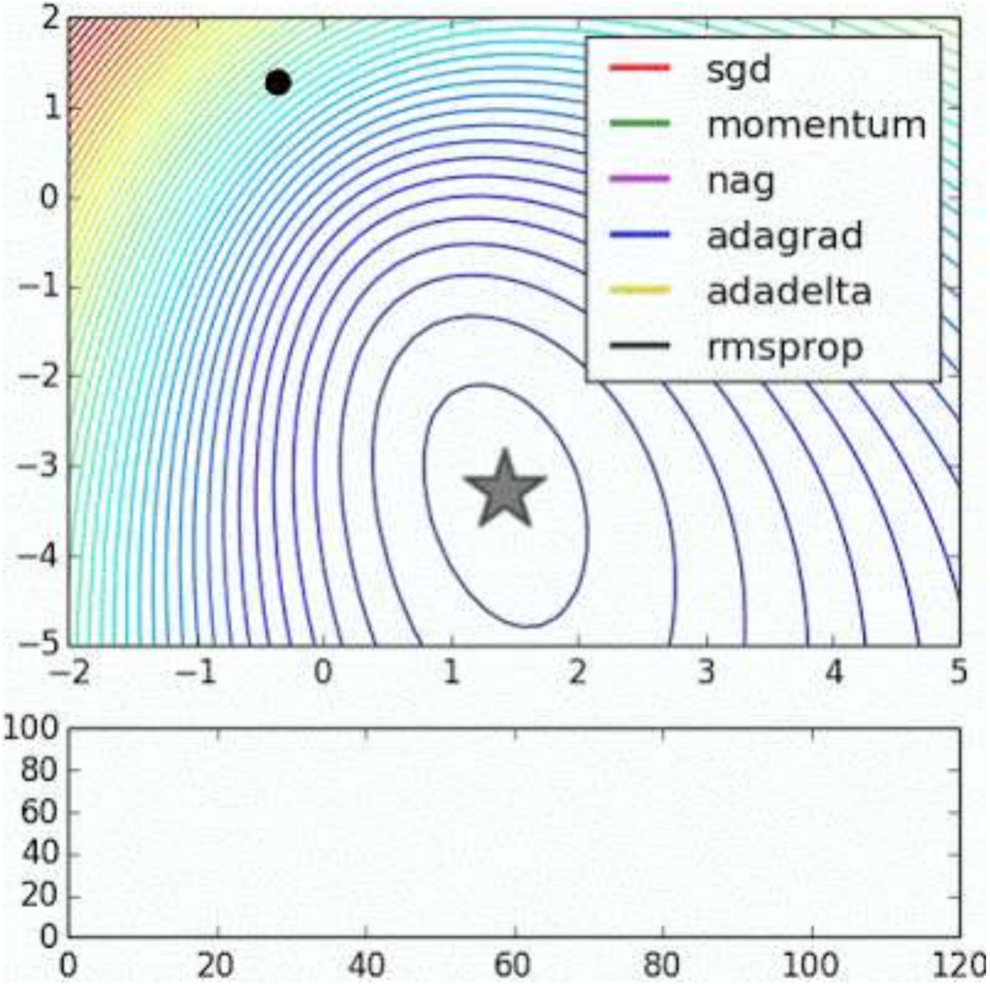
Always have a look at the learning curves



The effects of step size (or "learning rate")



Other Optimization



There are advanced optimization methods like adagrad used in DL.
Animation: http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html

Gradient Descent in TensorFlow

- In Theano and TensorFlow the Framework does the calculation of the gradient for you (autodiff)
- You just have to provide a graph

```
# loss has to be defined symbolically
train_op = tf.train.GradientDescentOptimizer(0.0001).minimize(loss)

…
for e in range(epochs): #Fitting the data for some epochs
  _, res = sess.run([train_op, loss], feed_dict={x:x_data, y:y_data})
```

Example: linear regression with Tensorflow

Finish: Linear regression with TensorFlow, optimization

c) Now let TensorFlow optimize the parameters in cell 7. Modify cell 7 with the right feeding data.

Hint: Look at the learning rate

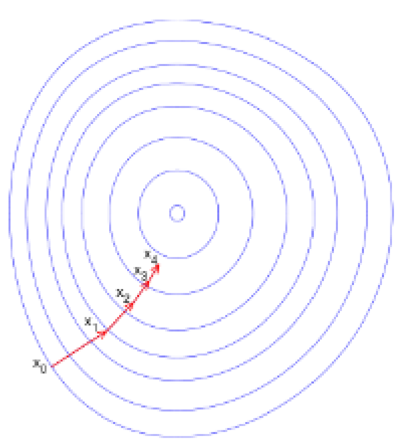
d) Draw the linreg graph and compare your graph with the Tensorboard graph.

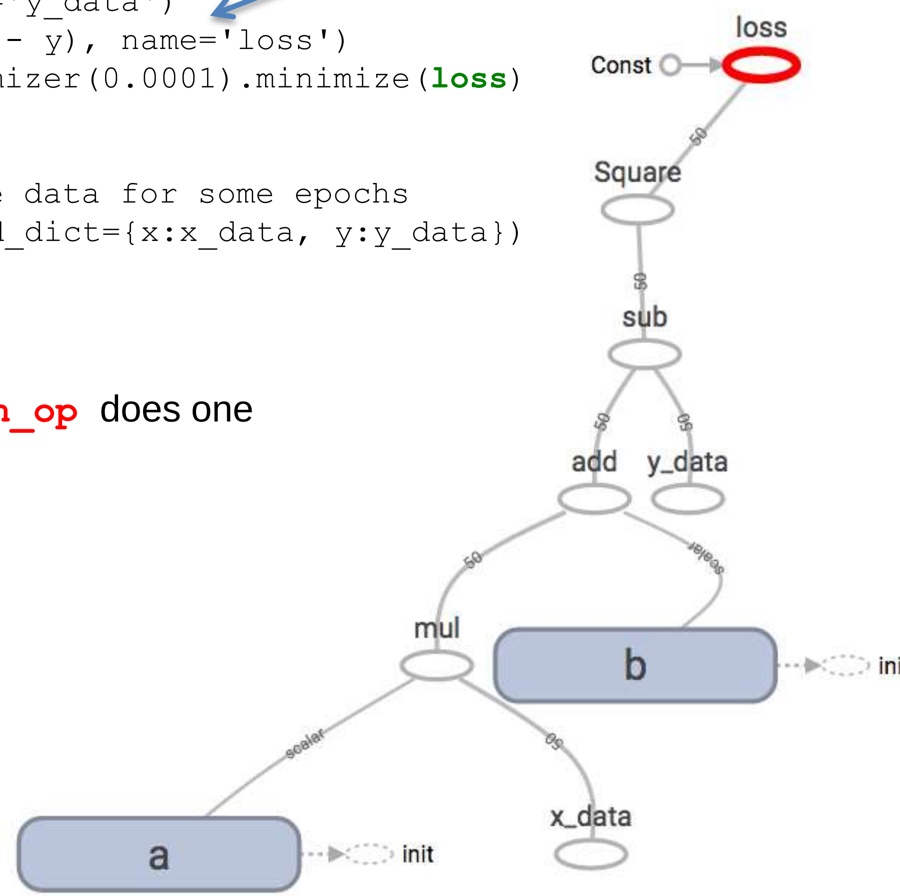
Computation done with a graph

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.





Excurse: Linear Regression

- Loss Function is convex quadratic in a and b

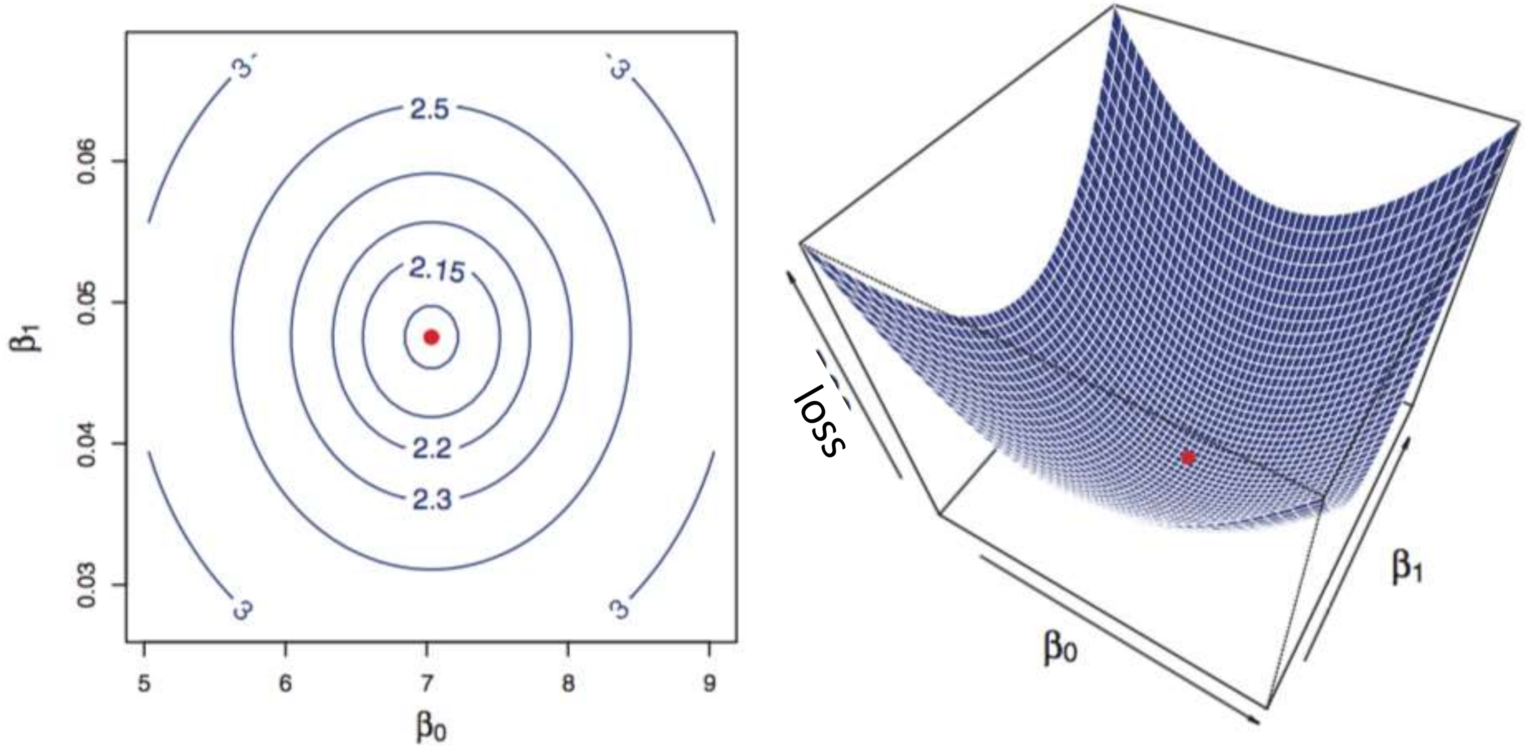
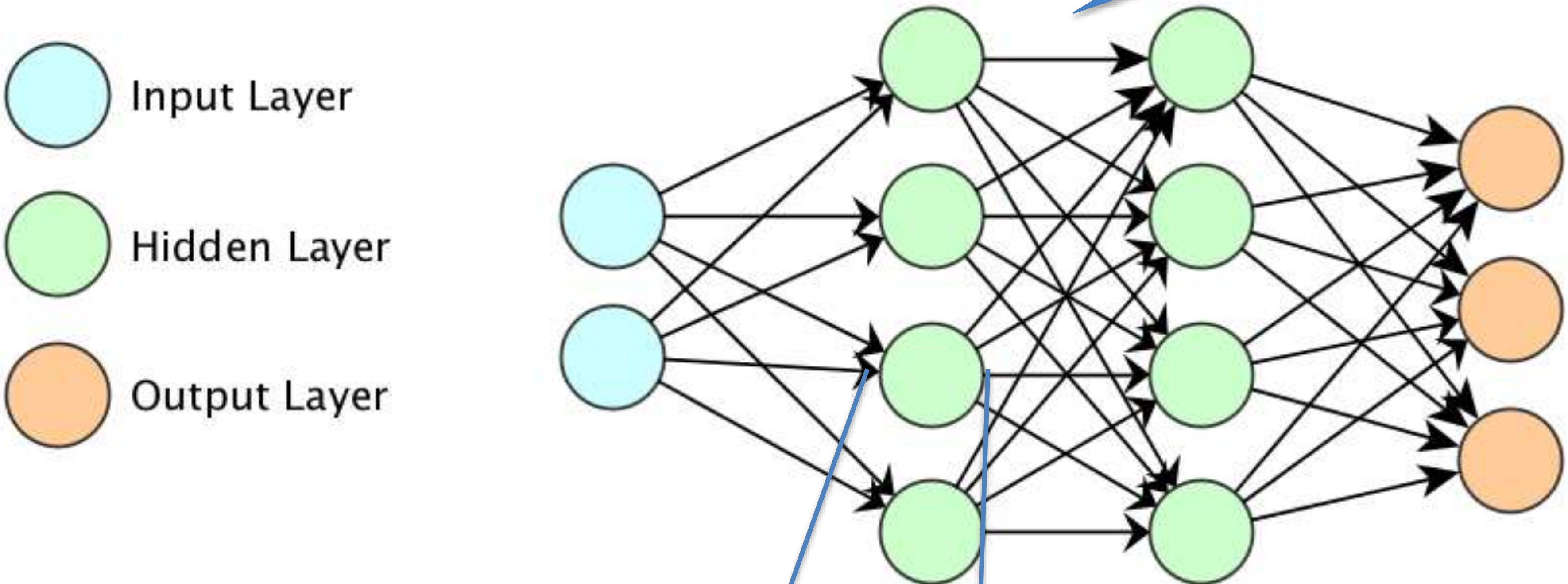


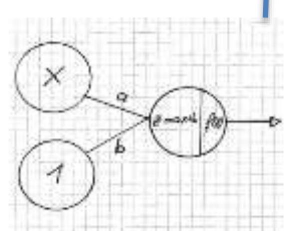
Figure from ISLR $\beta_0 = b$ $\beta_1 = a$

For linear regression convergence is guaranteed.
Closed form exists, these are often used instead of gradient descent.

Fully Connected Networks

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





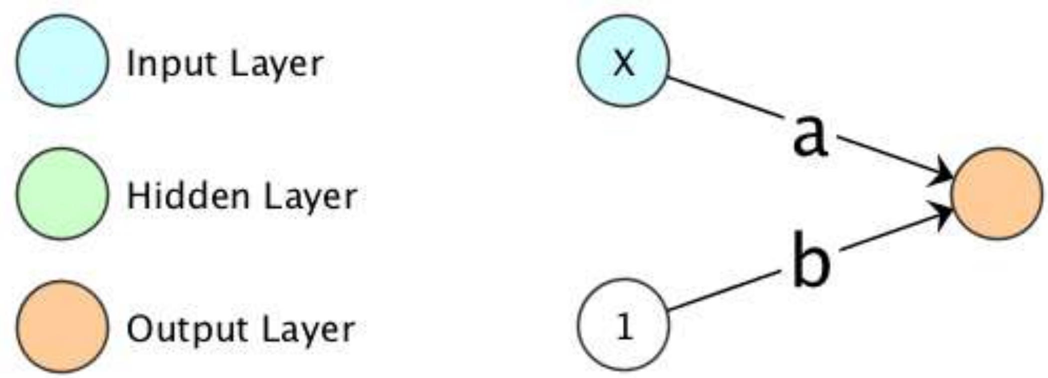
We start with....

1-D Logistic Regression

Logistic Regression

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

- The first building block

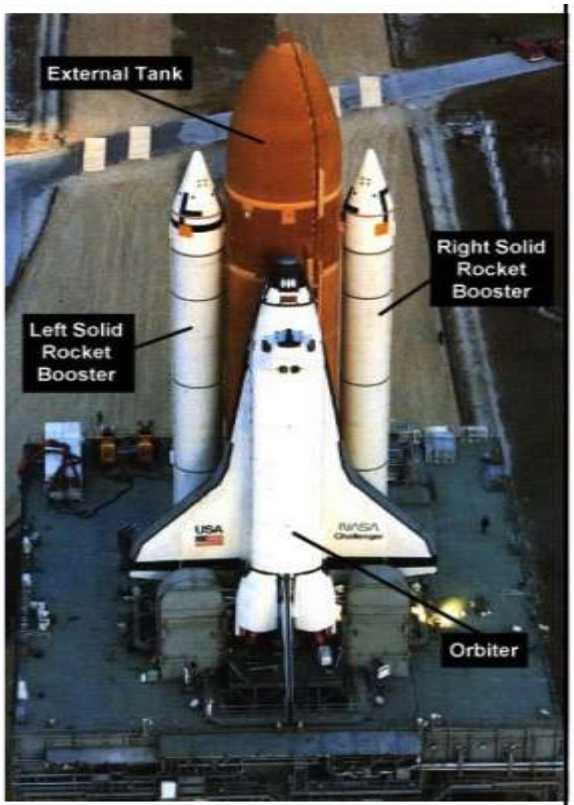


- In the following, have a look at logistic regression and derive the cost-function (log-likelihood) which we maximise.

- 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

- See also: introduction to statistical learning chapter 4.3
- Example

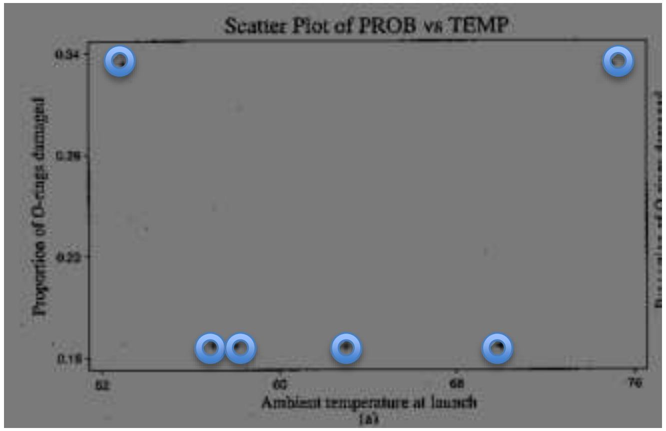


The space shuttle challenger exploded shortly after the start in 1986. One of bearings in the booster has been broken.

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

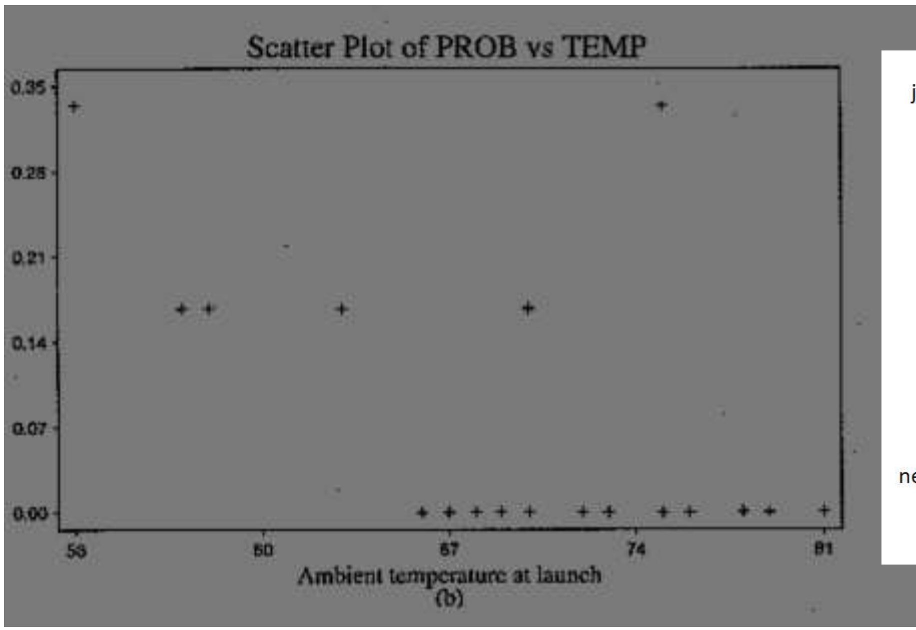
Ambient temperature	Number of O–rings damaged	$\hat{p}$
53°	2	.333
57°	1	.167
58°	1	.167
63°	1	.167
70°	1	.167
70°	1	.167
75°	2	.333

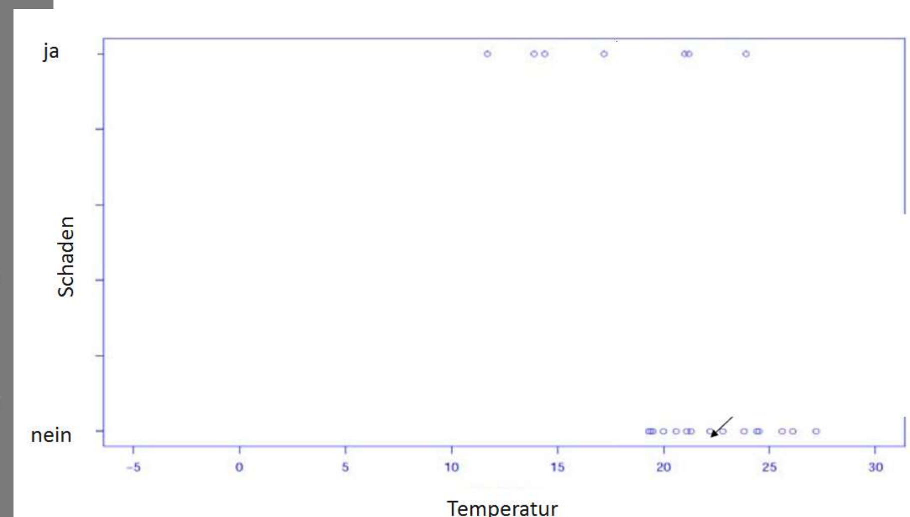


- Is there an increased risk of failure at low temperatures?
- **Would you launch (give reasons)?**

Statistik & Challenger Desaster

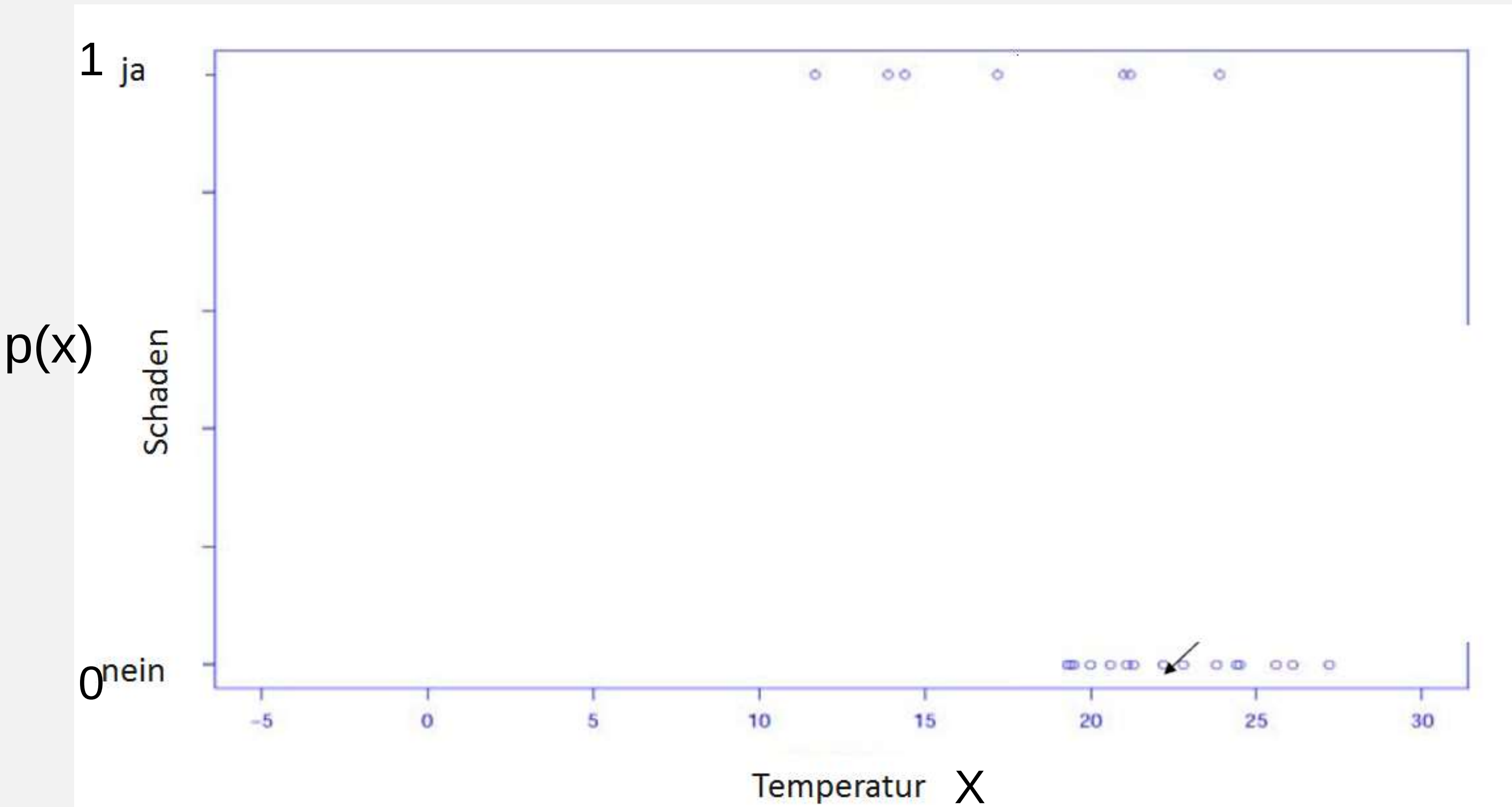
- There is information in the successful flights





Modelling logistic regression

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



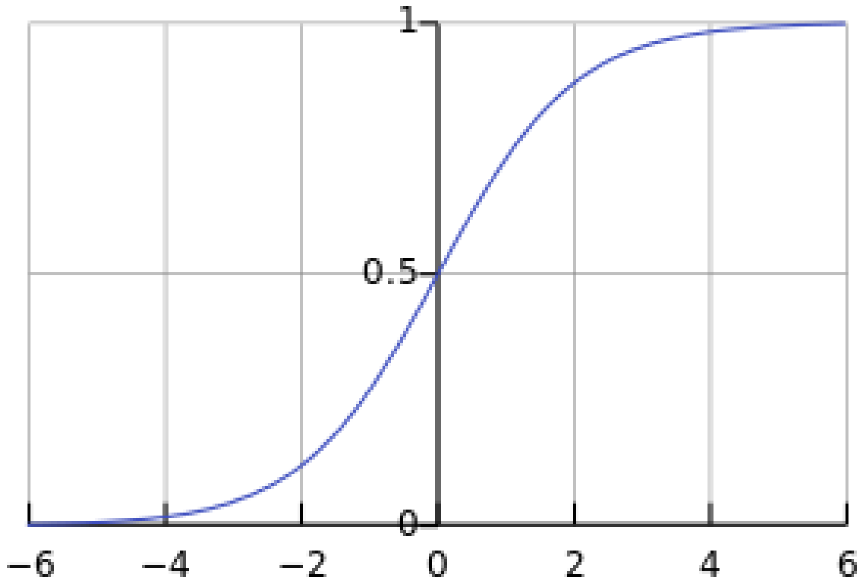
1. Draw a line p(X) = a X + b which fits data best (linear regression)
2. Question: Why is linear regression wrong?

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)

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

f(z)

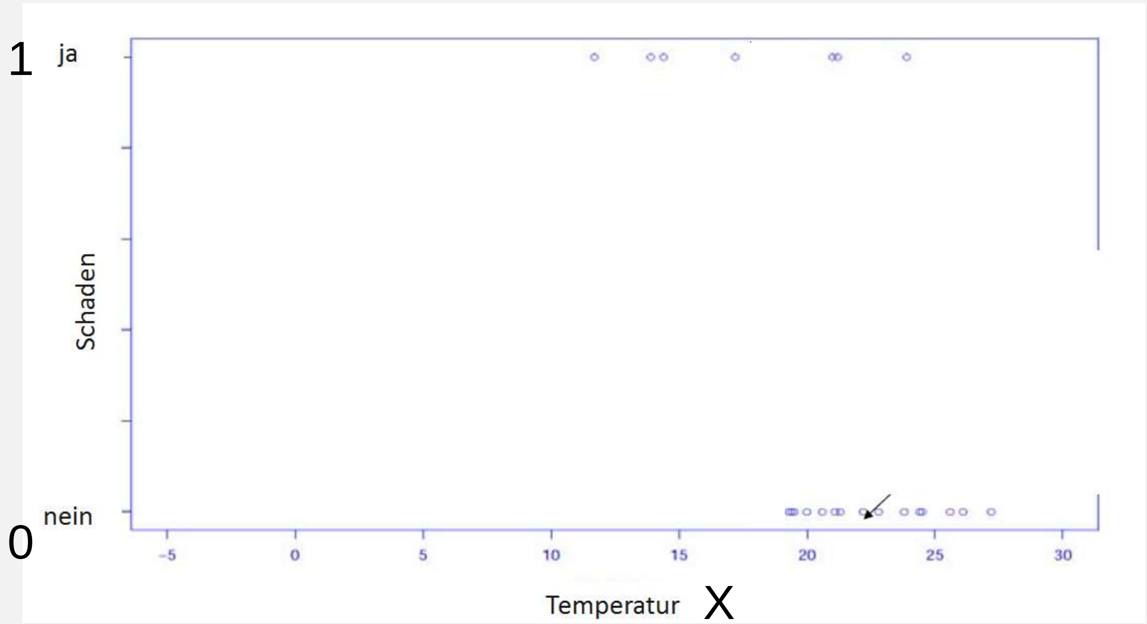


z

Modelling logistic regression

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

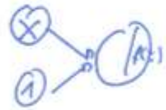
p(x)

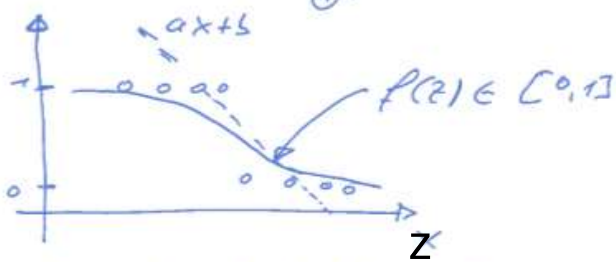


Task: Use the sigmoid function to bend your results of the linear regression.

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 für b'est [illegible] daten punkte (x_i, y_i).

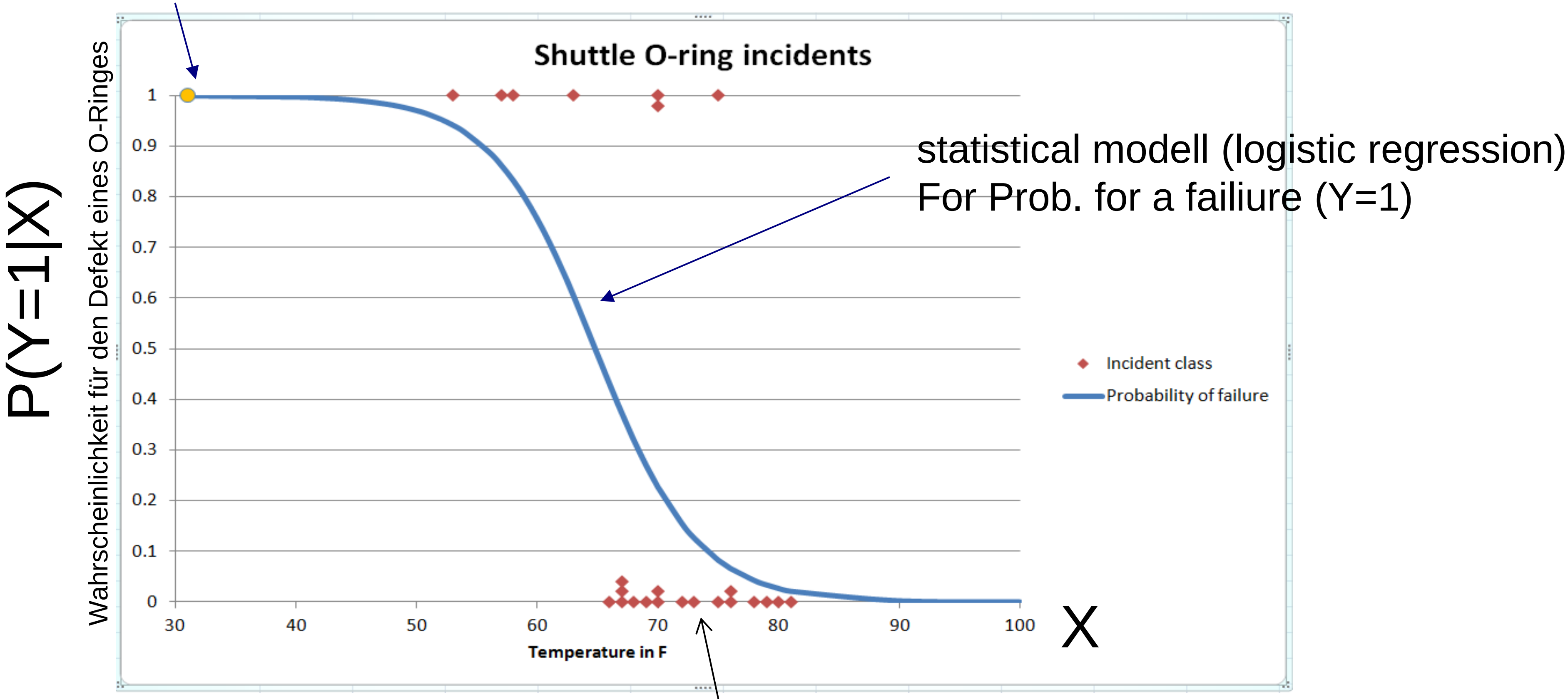
Nur das erklaeren

Logistic Regression: Example challenger O-rings

Predict if O-Ring is broken, depending on temperature

Challenger launch @31 F

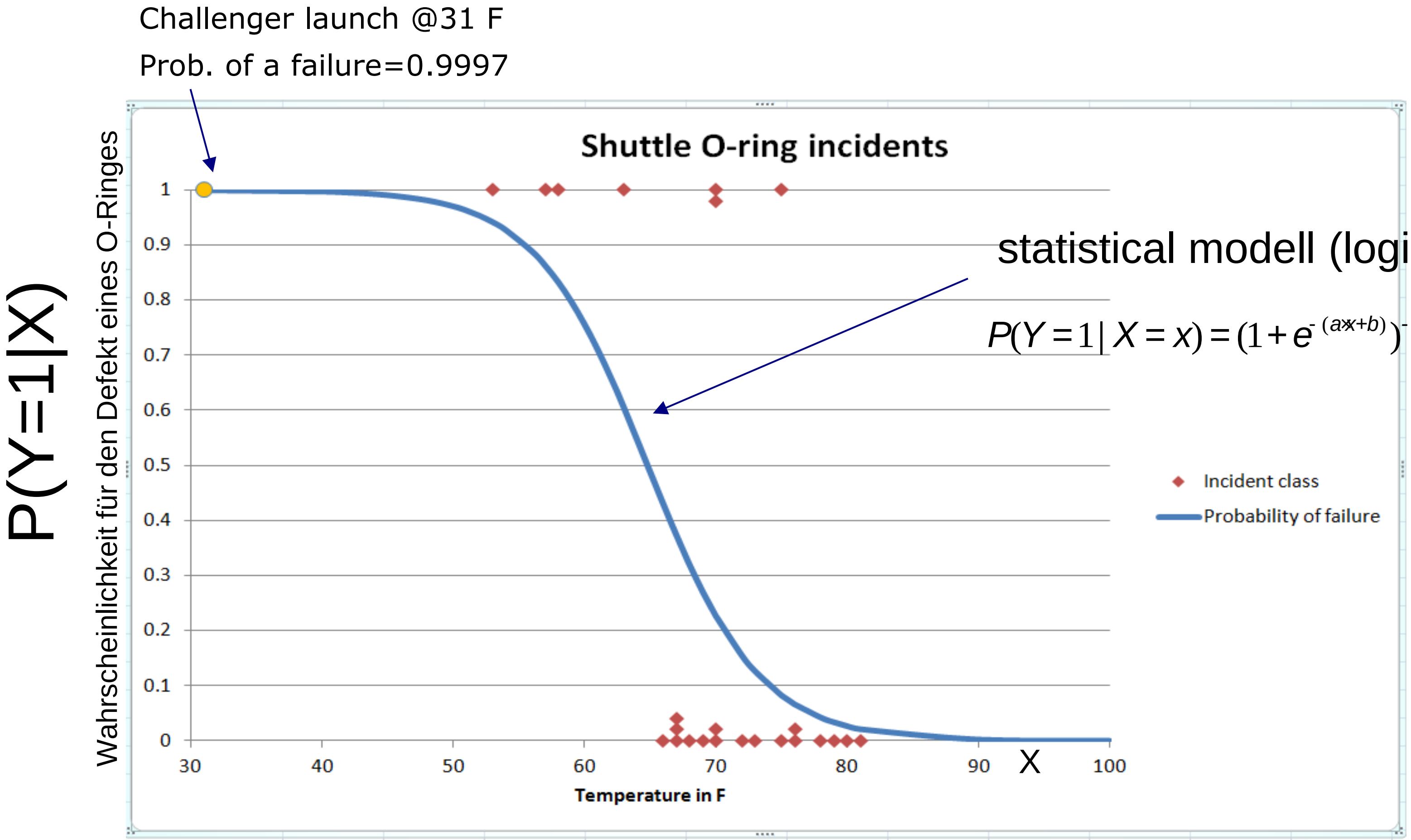
Prob. of a failure=0.9997



No failiure Y=0

Logistic Regression

Predict if O-Ring is broken, depending on temperature



statistical modell (logistic regression)

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

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

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)

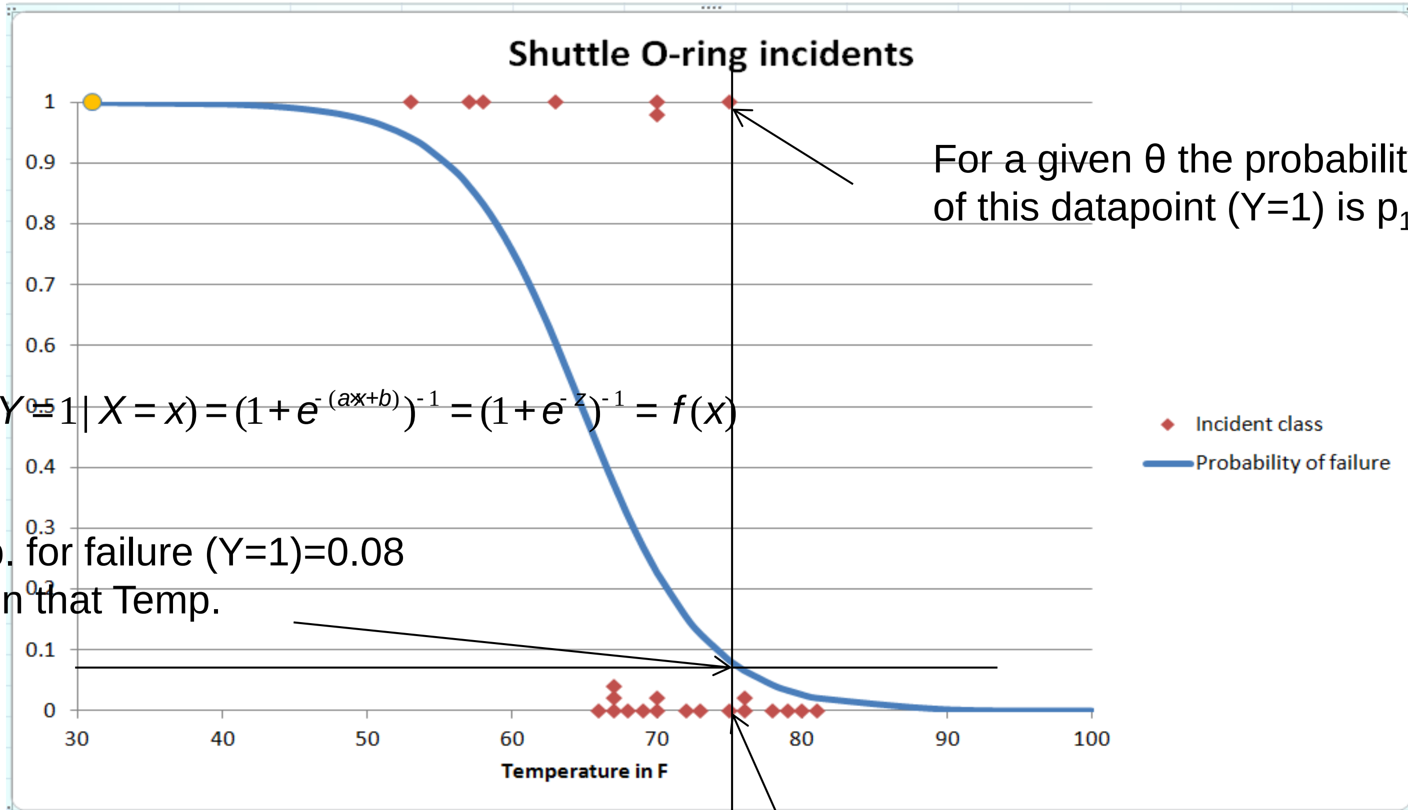
$$M(\theta) \longrightarrow \text{Data}$$

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...

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

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

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

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

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

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)

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

Let's maximize this probability

Maximizing the Likelihood

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

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

Taking log (maximum of log is at same position)

This is like a if-then

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

Same as cross-entropy loss used already for linear regression

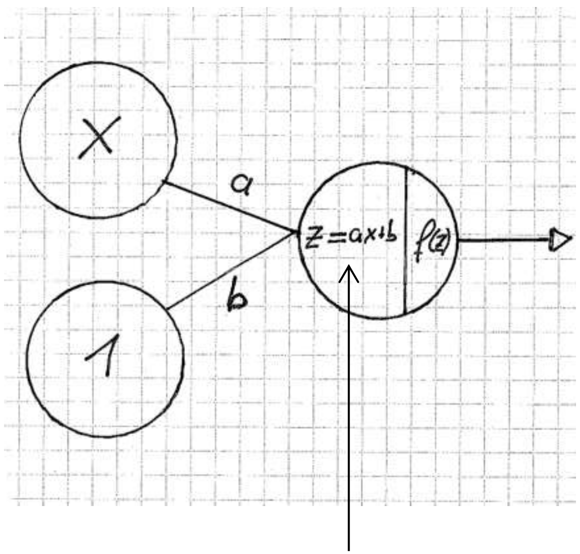
$$\text{loss} = -\frac{1}{N}\sum_{n=1}^{N} \log(p_{\text{model}}(y^{(i)} \mid x^{(i)}; \theta)) = -\frac{1}{N}\left(\sum_{i \in \text{All ones}} \log(p_1(x^{(i)})) + \sum_{i \in \text{All zeros}} \log(p_0(x^{(i)})) \right)$$

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

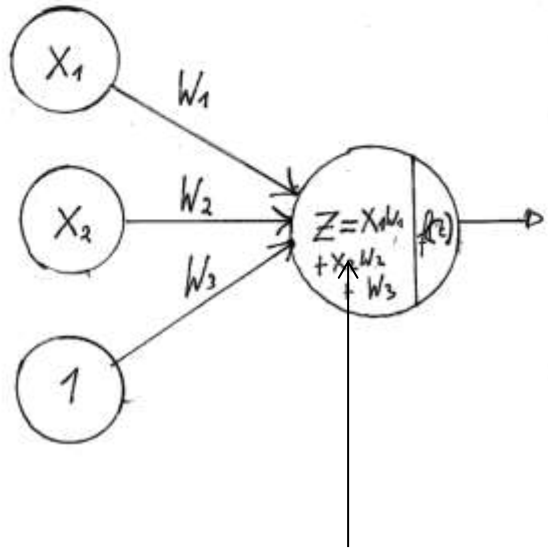
Logistic Regression in the neural net speak

1-D log Regression



$$z = ax + b$$

N-D log Regression



GPUs love linear algebra!

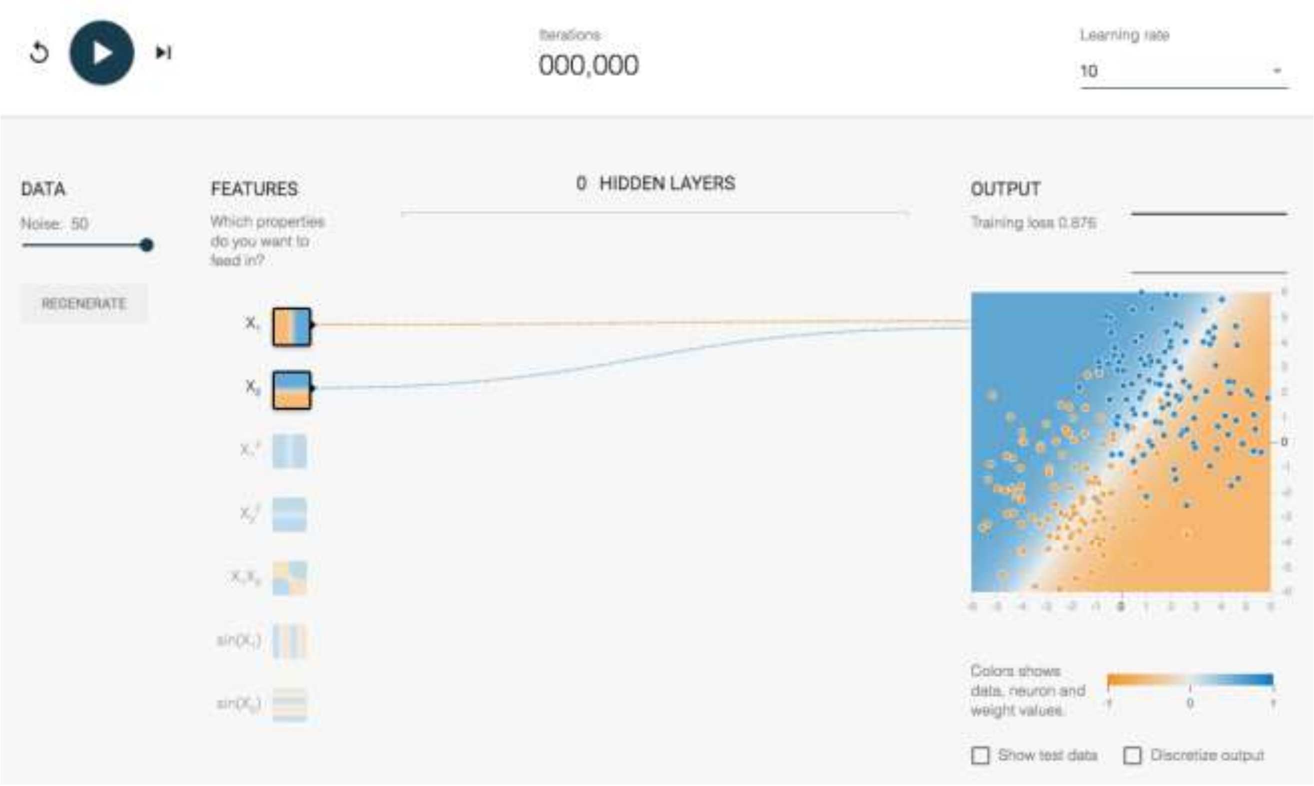
$$z = x_1 W_1 + x_2 W_2 + 1 \cdot W_3 = \vec{x}\,\boldsymbol{W}$$

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

f called
Logit non-linearity

Logistic Regression

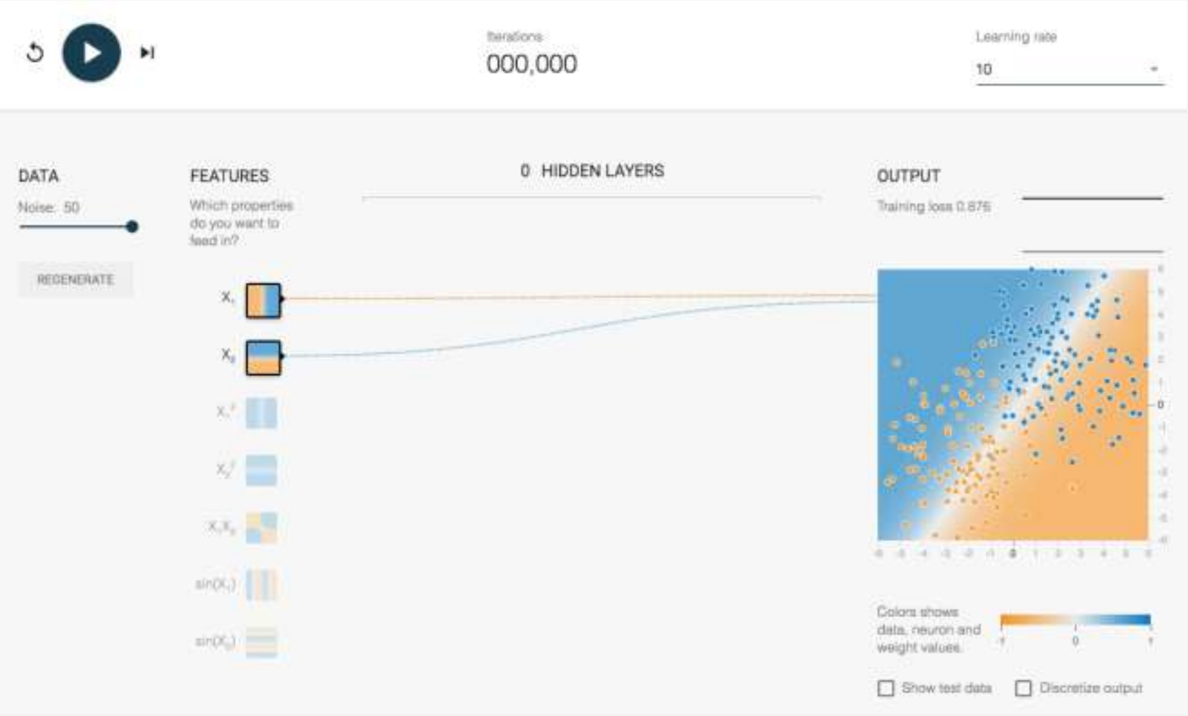
Explain TensorFlow playground



Logistic Regression [10 minutes]

Open the [tensorflow playground](https://playground.tensorflow.org)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.

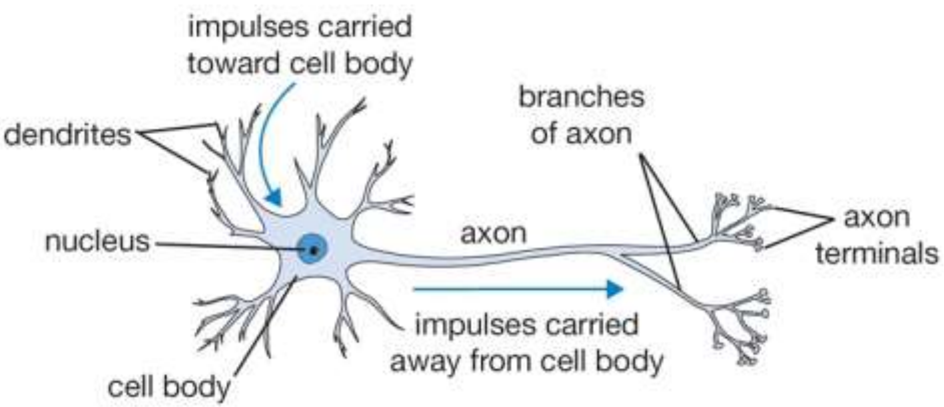
Datset

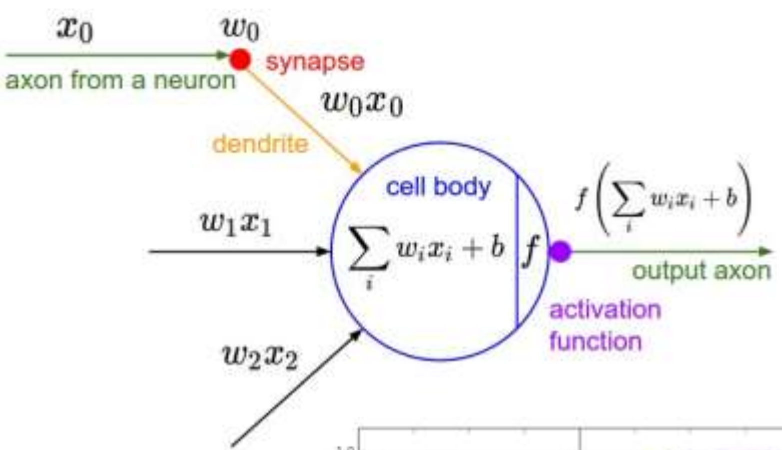
We investigate the data set of the challenger flight with broken O-rings (Y=1) vs start temperature.

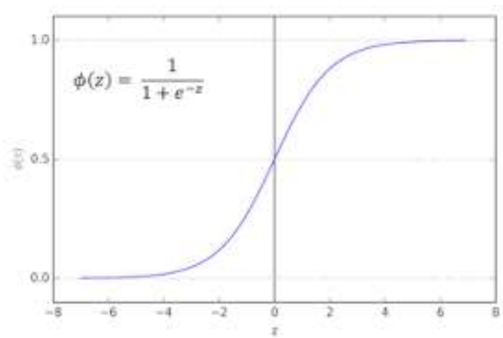
In [1]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as imgplot
import numpy as np
import pandas as pd
import tempfile
data = np.asarray(pd.read_csv('challenger.txt', sep=','), dtype='float32')
plt.plot(data[:,0], data[:,1], 'o')
plt.axis([40, 85,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





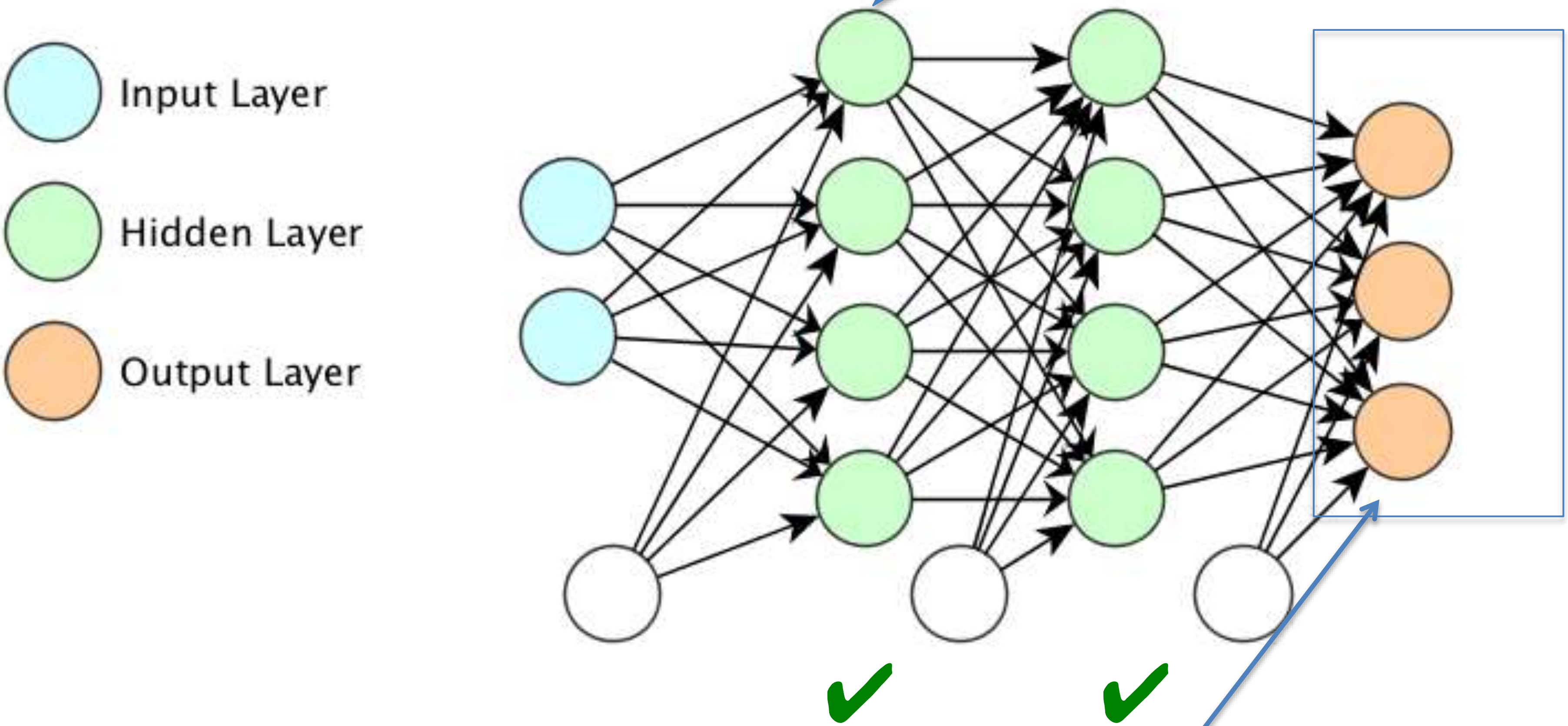


DL *loosely inspired* by how the brain works. Biological neurons are much more complicated.

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

More than two classes

We can use logistic regression for the hidden layers

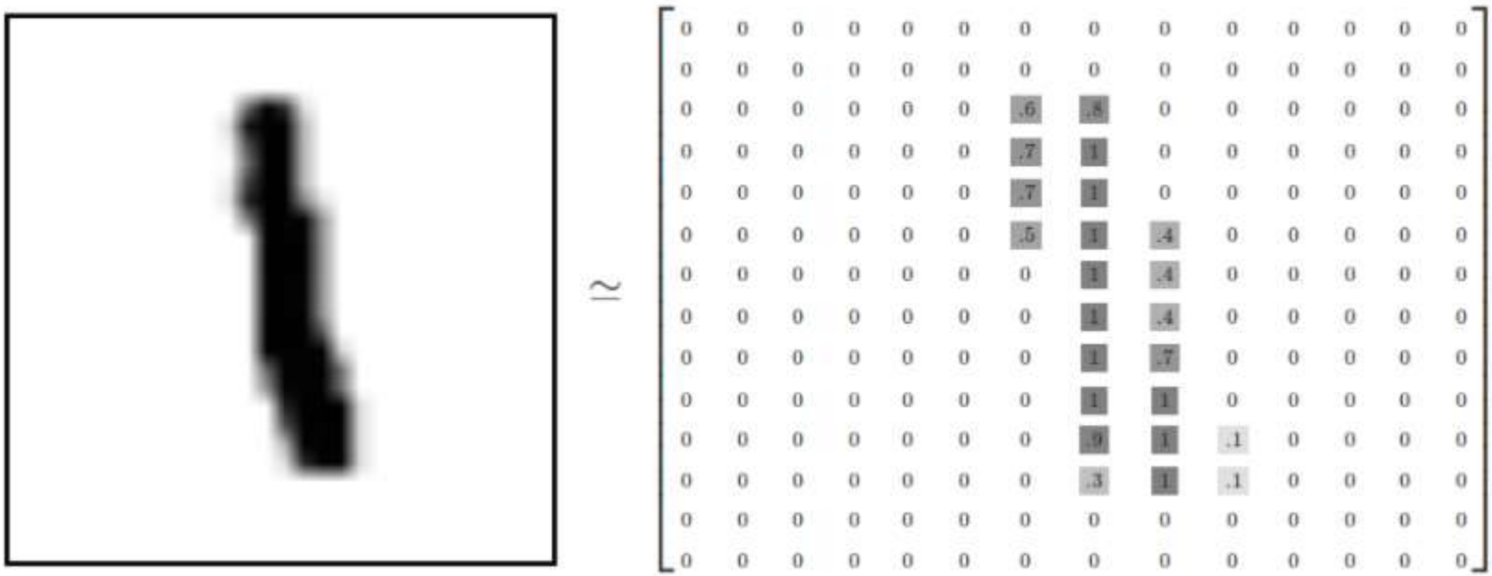


> 2 outputs! Not possible yet…

Multinomial Logistic Regression

Exercise: The MNIST Data Set

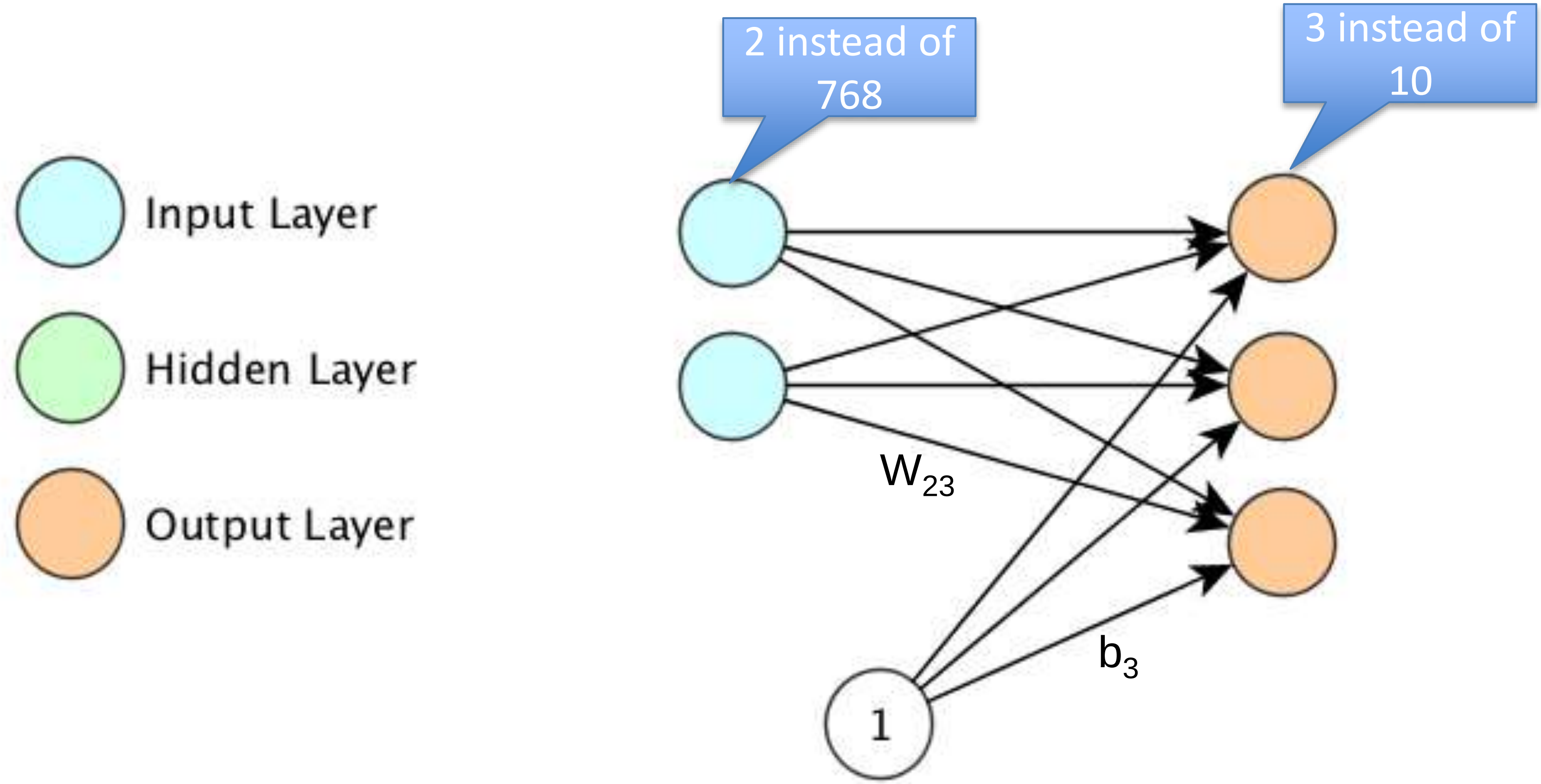
- MNIST the drosophila of all DL-Data sets
  - 50000 handwritten digits to be classified into 10 classes (0-9)



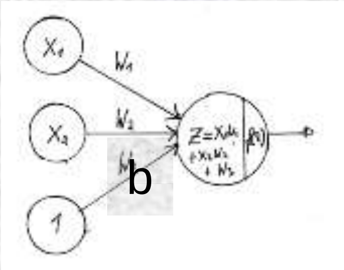
Input tensors: are flattened to 28*28=768 pixels

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

Multinomial Logistic Regression



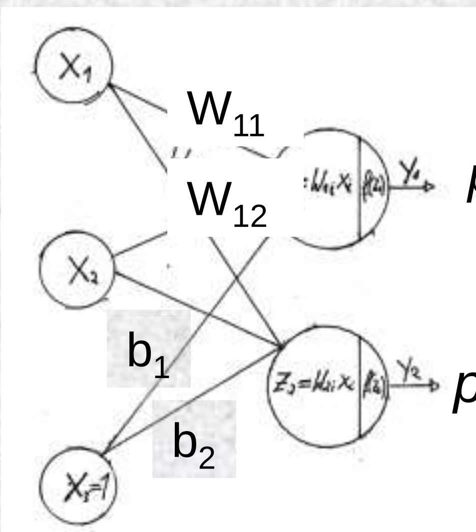
Multinominal Regression



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 × **B** = **C**

(m × **n**) × (**n** × p) = (m × p)

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

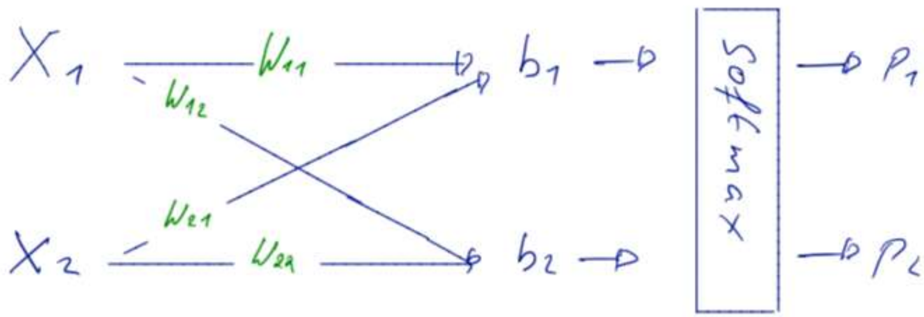
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \boxed{a_{31} \quad a_{32} \quad a_{33}} \end{bmatrix} x \begin{bmatrix} \boxed{\begin{matrix} b_{11} \\ b_{21} \\ b_{31} \end{matrix}} & \begin{matrix} b_{12} \\ b_{22} \\ b_{32} \end{matrix} \end{bmatrix} = \begin{bmatrix} c_{11}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

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

Example:

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

GPUs love matrices (or tensors)



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

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

$$p = \text{softmax}(x W + b)$$

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