# Machine Learning:: Linear Methods

Dozenten: Prof. Dr. M. O. Franz, Prof. Dr. O. Dürr

Remark: Much of the material have been developed together with Beate Sick (ZHAW/Uni Zurich).

# Topics

- Simple (1D) Linear Regression
- Gradient Descent for fitting
- Multiple Linear Regression and Design Matrix
  - Analytical Solution
- Ridge Regression
- Perceptron
- Probabilistic Treatment
  - Maximum likelihood principle
    - Linear regression MSE as maximum likelihood method
  - Logistic Regression

# Supervised: Regression vs Classification

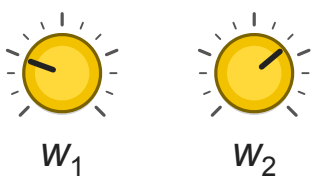
- Supervised learning problems can be divided into:
- **Regression** covers situations where Y is **continuous/numerical**. e.g.
  - Predicting the value of the Dow in 6 months.
  - Predicting the value of a given house based on various inputs.
- **Classification** covers situations where Y is categorical e.g.
  - Will the Dow be up (U) or down (D) in 6 months?
  - Is this email a SPAM or not?
- Some methods work well on both types of problem
  - Linear Methods
    - Linear Regression is a regression method
    - Logistic Regression (is a classification method)
    - Perceptron (is a classification model)
  - Boosting
  - Neural Networks (in the DL Part)

# Supervised: Parametric Models

Almost all supervised learning is done with parametric Models

- Models which have parameters (a.k.a. weights)
- We have (training $i = 1 \dots N_{training}$) data in pairs $x^{(i)}$ and $y^{(i)}$

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

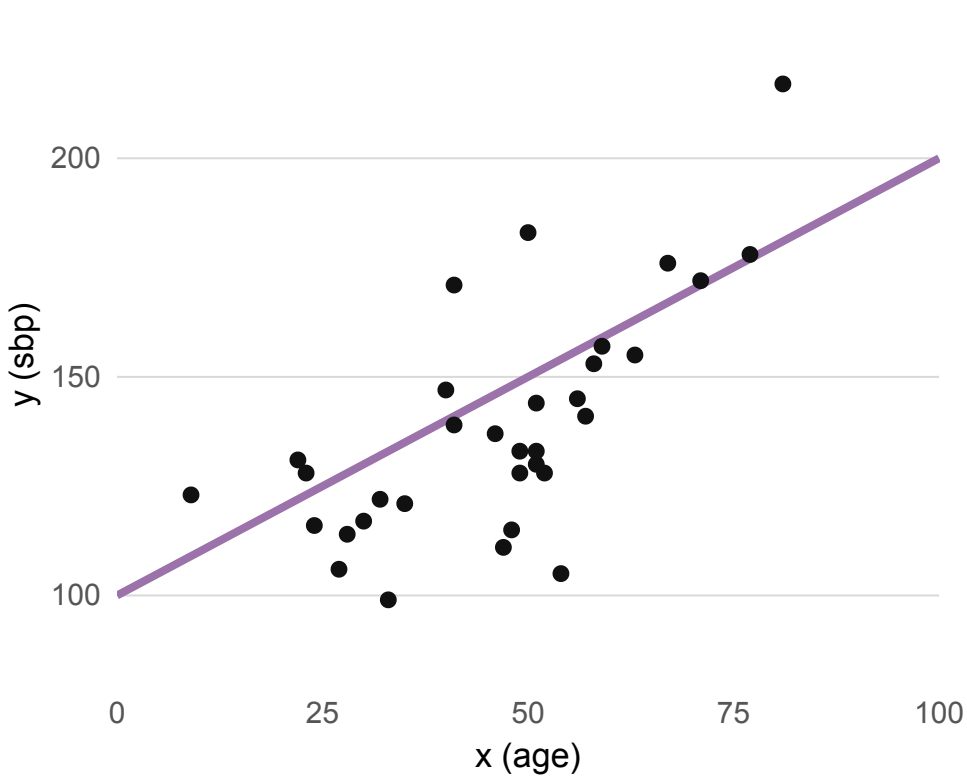



Depending on the weight the model produces a different $\hat{y}^{(i)}$

- After the training, we can discard the training data (it's all in the parameters).
- **Training is tuning the weights / knobs**
  - How to tune the knobs that the output of the model $\hat{y}^{(i)}$ matches the observed / "true" value $y^{(i)}$? We optimize a loss telling us how good we are.

# 1-D Linear Regression (as curve fitting)

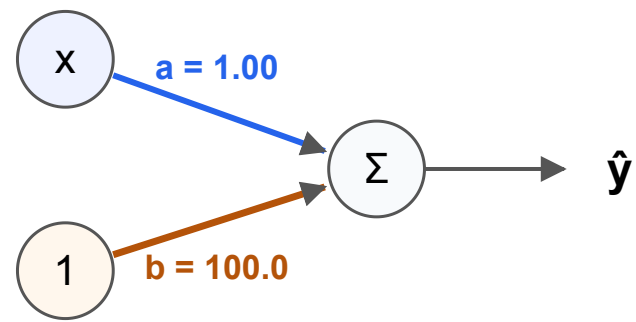
# Example: blood pressure given age [Demo]




$\hat{y} = 1.00 \cdot x + 100.0$

*a* 1.00

*b* 100.0



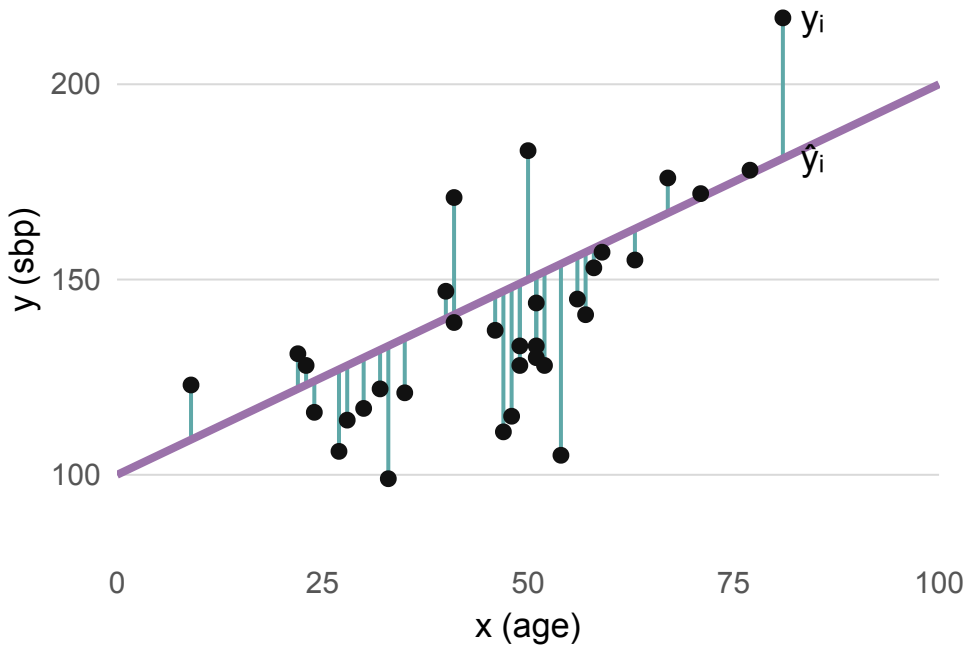

Like a neural net: one neuron, no activation

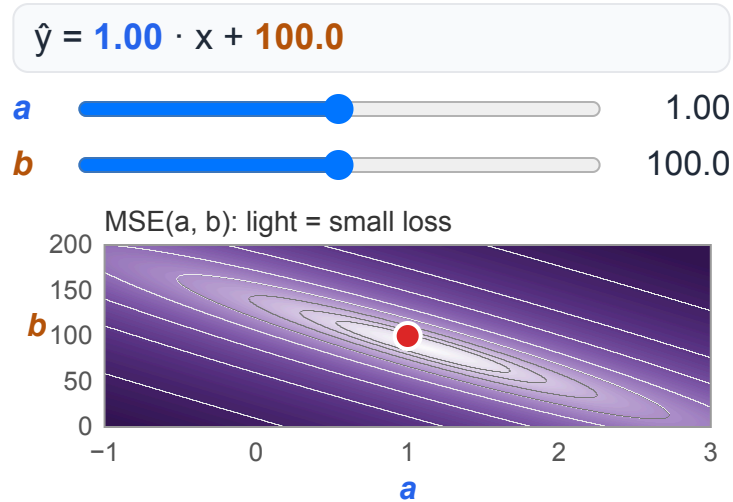
a: change of ŷ per year of age · b: ŷ at age 0

Data: Chapter 3, Probabilistic Deep Learning Book · Demo: oduerr.github.io/stat_demos/linear_regression_1d.html

# Definition of the loss: MSE [Demo]

Demo: oduerr.github.io/stat_demos/linear_regression_1d.html




marked point: $(y_i - \hat{y}_i)^2 = (217 - 181.0)^2 = 1296$

MSE = $\frac{1}{n} \Sigma (y_i - \hat{y}_i)^2$ = **408.2** (n = 33)

least squares: a = 1.11, b = 87.7, MSE = 349.2

$$\text{Loss} = \text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}\big(y_i - (a \cdot x_i + b)\big)^2$$

# Optimizing the loss

$$\text{Loss} = \text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}\big(y_i - (a \cdot x_i + b)\big)^2$$

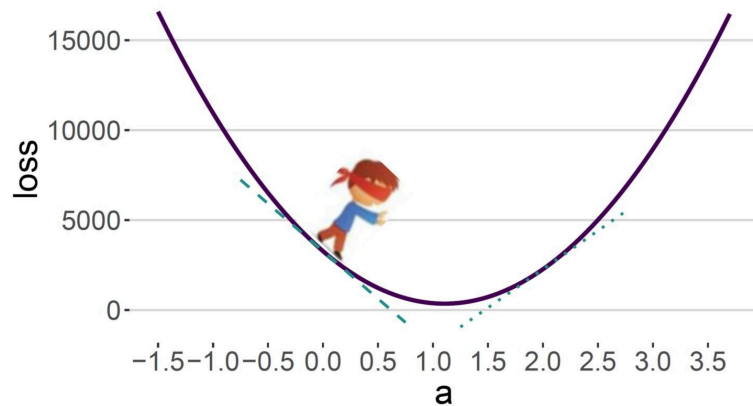
- $\text{loss}(\theta, D)$ is a function from parameters $\theta$ and data $D$
- We choose parameters $\theta$ to minimize the loss
  - $\hat{\theta} = \underset{\theta}{\text{argmin}}\ \text{loss}(\theta, D)$

Minimization

1. $\frac{\partial\, \text{loss}(\theta, D)}{\partial \theta} = 0$ and solve for $\theta$
   - That is the usual way for linear regression!
2. Numerical methods like Gradient Descent
   - That is very versatile works for many methods and is done in Deep Learning all the time

# Optimizing the loss: Gradient Descent

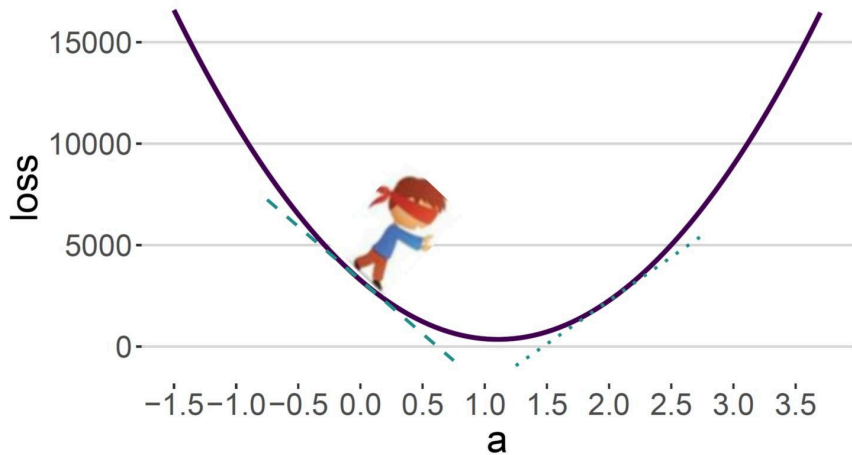
- Note for LR there is a closed form solution (see later). We use gradient descent since it's a very general methods and works for many ML and DL algorithms.
- For the figure only: Fix b and only vary a




- How would you choose the size of the step [you are blind!]

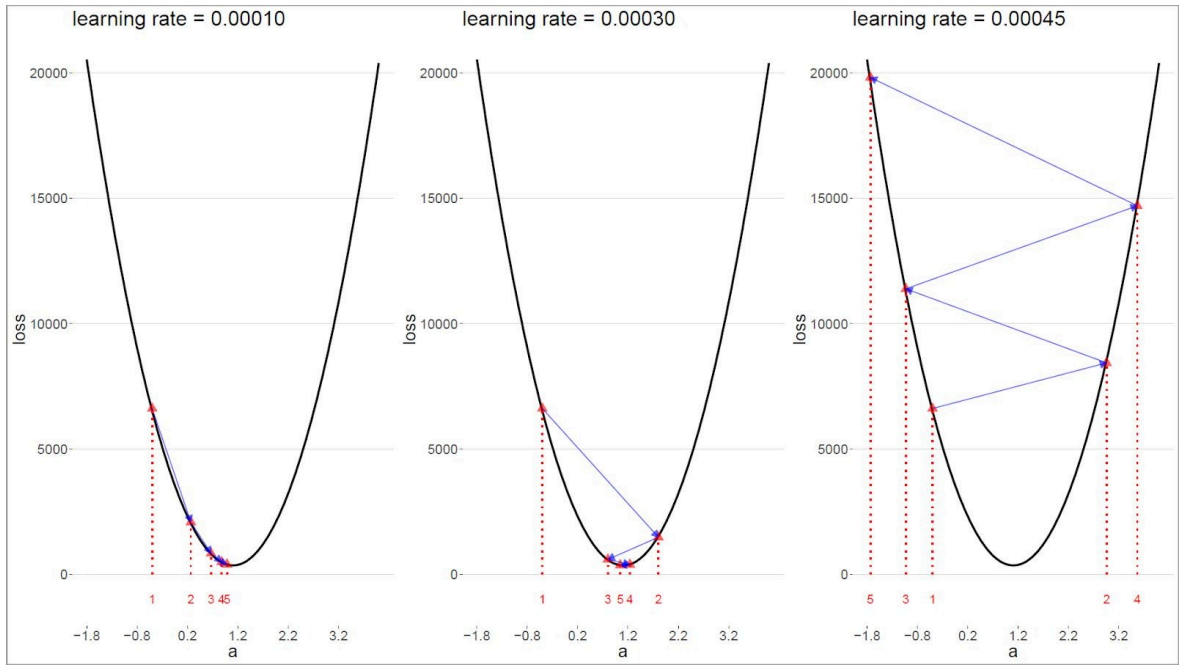
# Gradient update formula

$$a_{t+1} = a_t - \eta \cdot \text{grad}_a(\text{Loss}), \qquad \text{grad}_a(\text{Loss}) = \frac{\partial \,\text{Loss}}{\partial a}$$



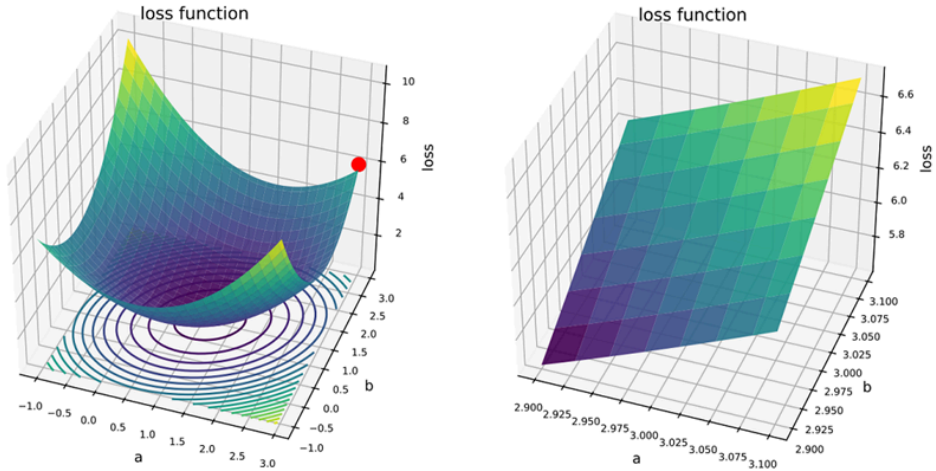

How to choose the parameter $\eta$ (Learning Rate)?

# Proper learning rate




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

# In two dimensions




$$a_{t+1} = a_t - \eta \cdot grad_a$$

$$b_{t+1} = b_t - \eta \cdot grad_b$$

In vector notation: $W^{t+1} = W^t - \eta \cdot \nabla \mathbf{W}$

# In two dimensions gradient

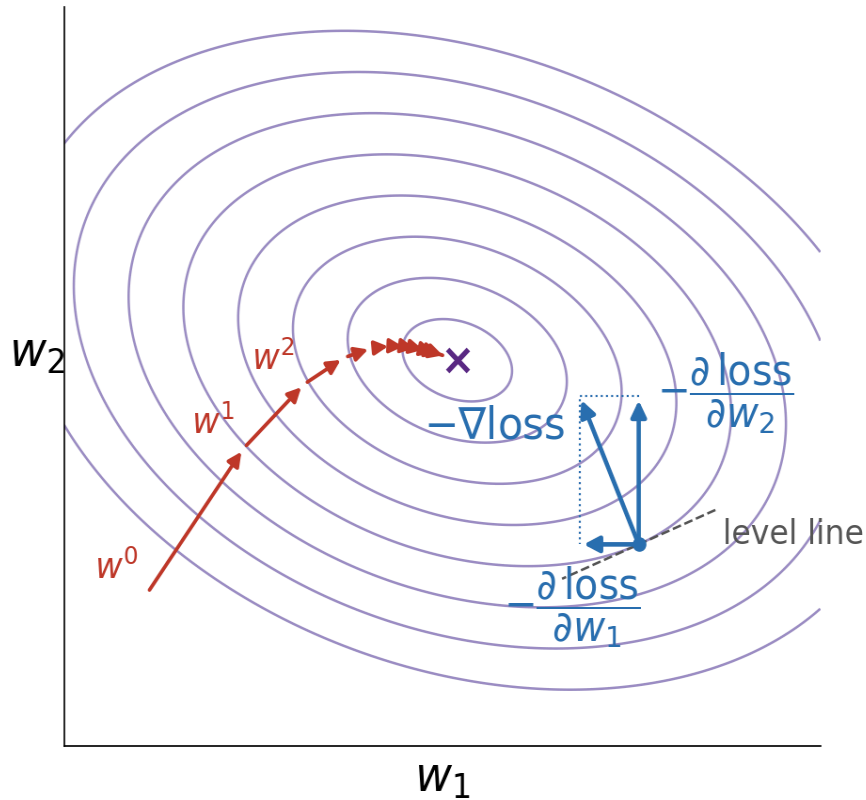
Gradient descent

$$w_i^{t+1} = w_i^t - \eta \cdot \frac{\partial \text{loss}}{\partial w_i}$$

The gradient

$$\nabla \text{loss} = \left( \frac{\partial \text{loss}}{\partial w_1}, \frac{\partial \text{loss}}{\partial w_2} \right)$$

is perpendicular to the level lines.



# Dangers of Gradient Descent in 2D



Many Problems in ML are convex. DL magically also works for non-convex problems

Chapter 3: Probabilistic Deep Learning Book

# Multivariate Linear Regression (as curve fitting)

See notebook linear_methods/LinearRegression

# Example

| | TV | Radio | Newspaper | Sales |
|---|---|---|---|---|
| 0 | 230.1 | 37.8 | 69.2 | 22.1 |
| 1 | 44.5 | 39.3 | 45.1 | 10.4 |
| 2 | 17.2 | 45.9 | 69.3 | 9.3 |
| 3 | 151.5 | 41.3 | 58.5 | 18.5 |
| 4 | 180.8 | 10.8 | 58.4 | 12.9 |
| ... | ... | ... | ... | ... |
| 195 | 38.2 | 3.7 | 13.8 | 7.6 |
| 196 | 94.2 | 4.9 | 8.1 | 9.7 |
| 197 | 177.0 | 9.3 | 6.4 | 12.8 |
| 198 | 283.6 | 42.0 | 66.2 | 25.5 |
| 199 | 232.1 | 8.6 | 8.7 | 13.4 |

200 rows × 4 columns

$$\widehat{sales} = 1 \cdot w_0 + w_1 TV + w_2 Radio + w_3 Newspaper$$

# Matrix notation

Design Matrix

| Intercept | TV | Radio | Newspaper |
|---|---|---|---|
| 1.0 | 230.1 | 37.8 | 69.2 |
| 1.0 | 44.5 | 39.3 | 45.1 |
| 1.0 | 17.2 | 45.9 | 69.3 |
| 1.0 | 151.5 | 41.3 | 58.5 |
| 1.0 | 180.8 | 10.8 | 58.4 |
| ... | ... | ... | ... |
| 1.0 | 38.2 | 3.7 | 13.8 |
| 1.0 | 94.2 | 4.9 | 8.1 |
| 1.0 | 177.0 | 9.3 | 6.4 |
| 1.0 | 283.6 | 42.0 | 66.2 |
| 1.0 | 232.1 | 8.6 | 8.7 |

$$\begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix} =$$

| y_hat | Sales |
|---|---|
| 20.244783 | 22.1 |
| 12.426815 | 10.4 |
| 12.310875 | 9.3 |
| 17.457258 | 18.5 |
| 13.204685 | 12.9 |
| ... | ... |
| 5.815722 | 7.6 |
| 8.534241 | 9.7 |
| 13.000720 | 12.8 |
| 23.386213 | 25.5 |
| 15.297458 | 13.4 |

$$X \cdot w = \hat{y}$$

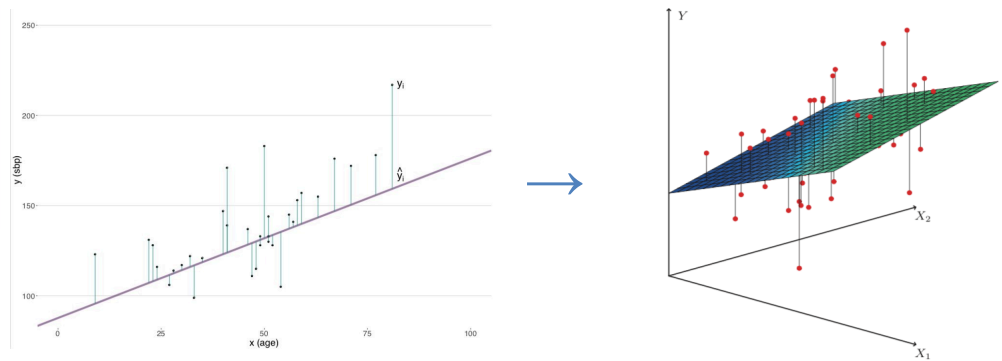
$$\widehat{sales} = 1 \cdot w_0 + w_1 TV + w_2 Radio + w_3 Newspaper$$

# Multivariate Linear regression

- Matrix notation
  - $\hat{y} = f_\theta(x) = 1 \cdot w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d = \sum_{j=1}^{d} w_j x_j + w_0 = \boldsymbol{w^T x} + w_0 =< \boldsymbol{w}, \boldsymbol{x} > + w_0$
- As a NN




- Geometric interpretation $f_\theta(x)$ is a plane



# Multivariate Linear regression

- Simple model
  - $\hat{y} = f_\theta(x) = 1 \cdot w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d = \sum_{j=1}^{d} w_j x_j + w_0 = \vec{w}^{\boldsymbol{t}} \vec{x} + w_0 =< w, x > + w_0$
- A note on notation
  - $w_0$ is often denoted with *b* the bias (extended data matrix)
  - Often the bias is included in $\vec{w}^{\boldsymbol{t}} = (w_0, w_1, w_2, \cdots, w_d)$
    - $\hat{y} = \vec{w} \cdot \vec{\boldsymbol{x}}$
  - In some literature (Tibshirani) $\vec{w} = \vec{\beta}$
  - Often no vector symbols
- Side note Einstein Summation Convention
  - $\hat{y}_i = \sum_{j=0}^{p} X_{ij} w_j = X_{ij} w_j$
  - Numpy: `np.einsum('ij,j->i',X,w)`

# Solving the Regression Problem Step by Step

**Step 1: Loss Function (Squared Error)**

$$L = \sum_i (\hat{y}_i - y_i)^2 = \sum_i (X_{ij} w_j - y_i)^2$$

← Dropped Sum over j, for clarity

The Einstein summation convention implies a sum over repeated indices $j$.

**Step 2: Take the Derivative with Respect to $w_k$**

Vector Notation (Scary and IMO error-prone)

$$\nabla_w L = \nabla_w \sum_i (Xw - y)$$

We apply the chain rule.

$$\frac{\partial L}{\partial w_k} = 2 \sum_i (X_{ij} w_j - y_i) \underbrace{\frac{\partial}{\partial w_k} (X_{ij} w_j - y_i)}_{\delta_{kj} X_{ij}}$$

# Solving the Regression Problem Step by Step

**Step 3: Set the Gradient Equal to Zero**

$$\frac{\partial L}{\partial w_k} = 0 \implies 2(X_{ij}w_j - y_i)X_{ik} = 0$$

$$X_{ik}X_{ij}w_j = X_{ik}y_i$$

$$(X^T X)_{kj}w_j = (X^T y)_k$$

**Step 4: Solve for the Weight Vector**

$$w = (X^T X)^{-1} X^T y$$

This is the final formula for the weight vector $w$, derived by solving the normal equation.

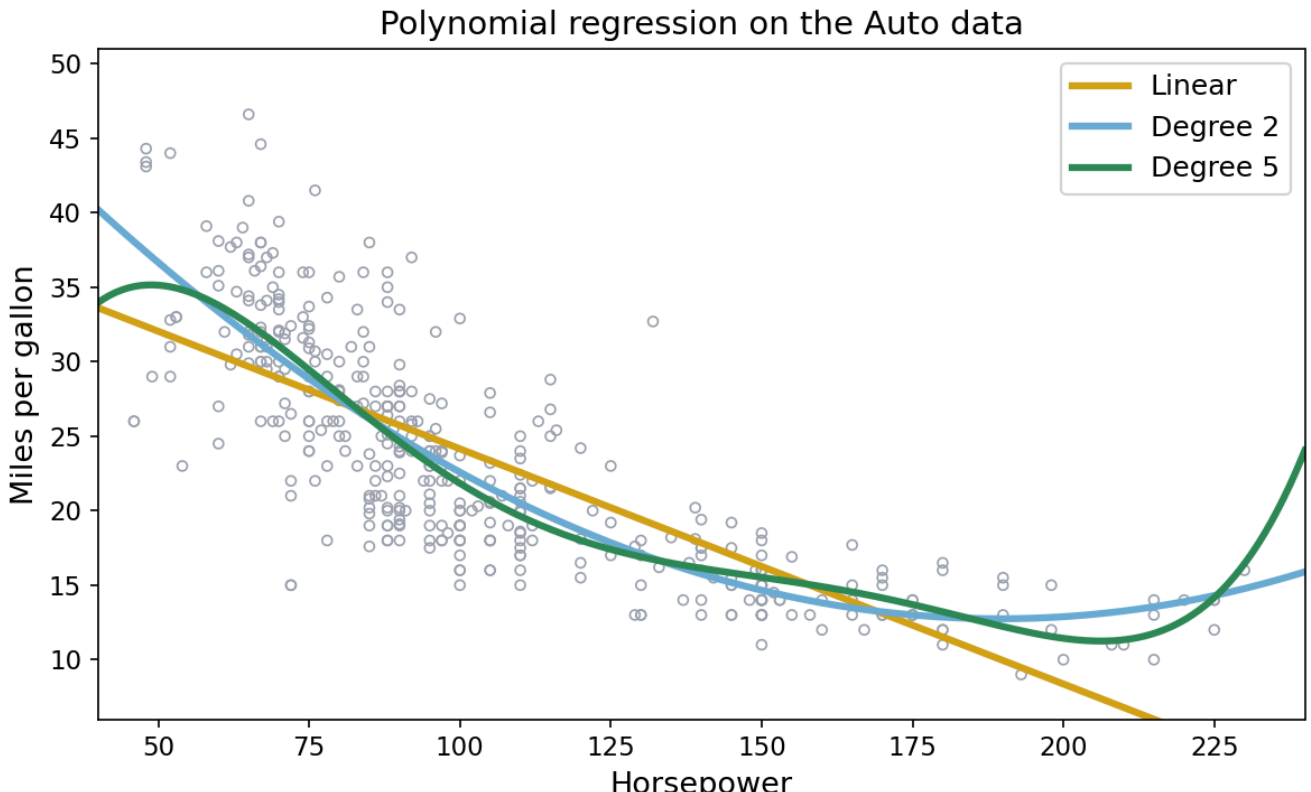
# Pause bis 14:50

# Demo

https://github.com/ioskn/mldl_htwg/blob/master/linear_methods/LinearRegression.ipynb



# Non Linear Effect / Polynomial Data




Data: Auto, ISLR (James et al.), www.statlearning.com

# Non Linear Effect / Polynomial Data

The figure suggests that

$$\text{mpg} = w_0 + w_1 \cdot \text{horsepower} + w_2 \cdot \text{horsepower}^2 + \epsilon$$

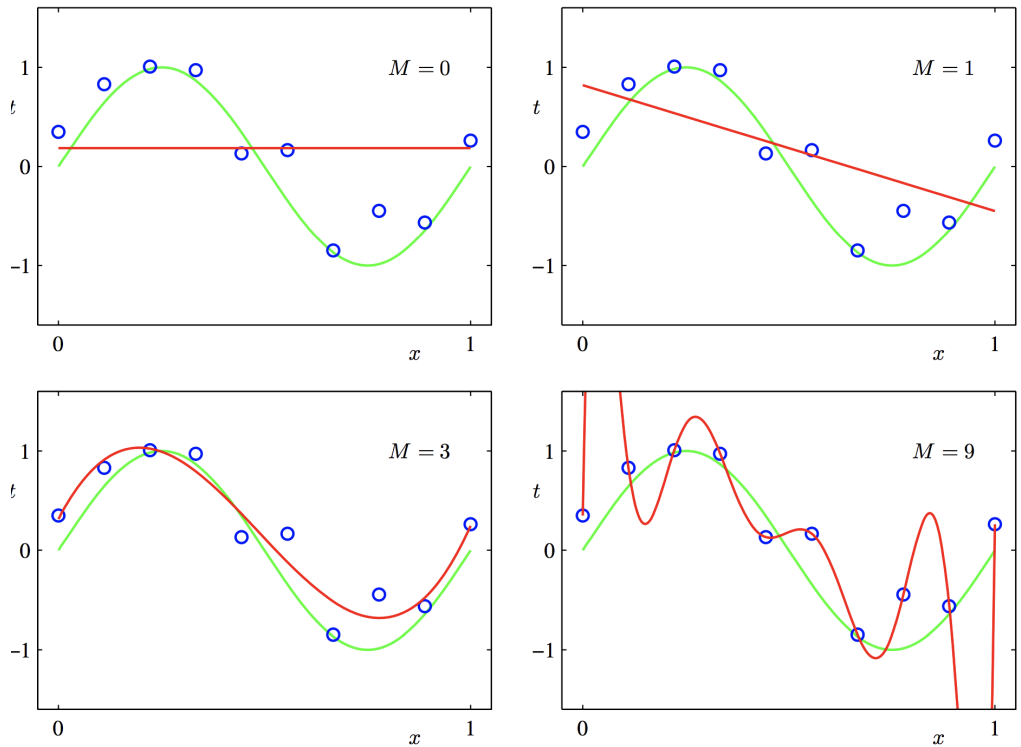
may provide a better fit.

| | Coefficient | Std. Error | t-statistic | p-value |
|---|---|---|---|---|
| Intercept ($w_0$) | 56.9001 | 1.8004 | 31.6 | < 0.0001 |
| horsepower ($w_1$) | −0.4662 | 0.0311 | −15.0 | < 0.0001 |
| horsepower² ($w_2$) | 0.0012 | 0.0001 | 10.1 | < 0.0001 |

Fit on the Auto data (n = 392), ISLR (James et al.); slidev/scripts/auto_poly.py

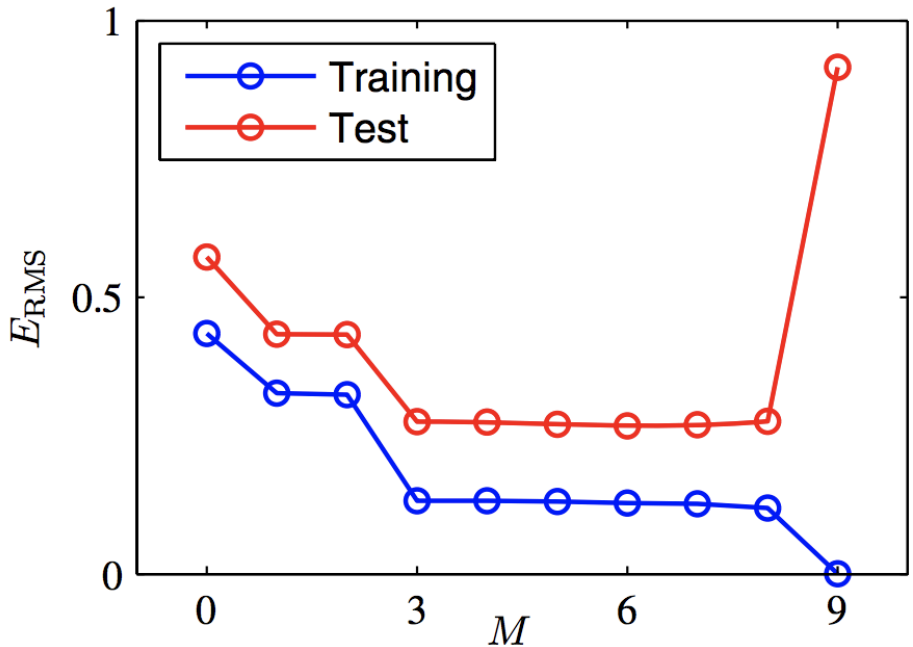
# Ridge Regression

# Overfitting



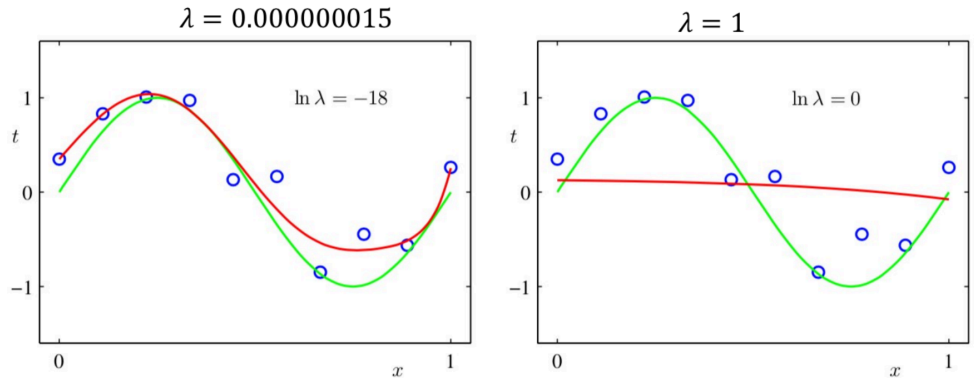

Expand the data matrix to columns $x^0, x^1 \ldots x^M$

# Overfitting




More on overfitting in later lectures …

# Ridge Regression




$$\sum_{i=1}^{n}\left(y_i - b - \sum_{j=1}^{p} w_j x_{ij}\right)^2 + \lambda \sum_{j=1}^{p} w_j^2 = \mathrm{RSS} + \lambda \sum_{j=1}^{p} w_j^2,$$

Additional loss term L2-Regularisation

- It costs to have non-zero parameters.
- Parameters are closer to zero (*shrinkage*)
- Ridge Regression reduces overfitting.
- Allows to fit linear regression to #parameters > # Data problems
- Also popular Lasso L1 regularisation

# Perceptron

- Linear Regression + Sign-Activity

# Perceptron

- The simplest way for a linear machine to do **classification**
  - Other approach is logistic regression
- We have binary classification problem
  - Code $y_i = \pm 1$ for convenience
- Based on **a geometric concept** of hyperplanes
  - Needs classes which are separable by hyperplanes
  - Will be extended in the Support Vector Machine

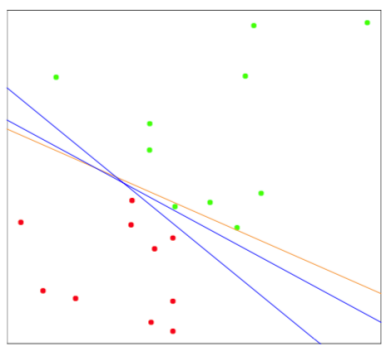
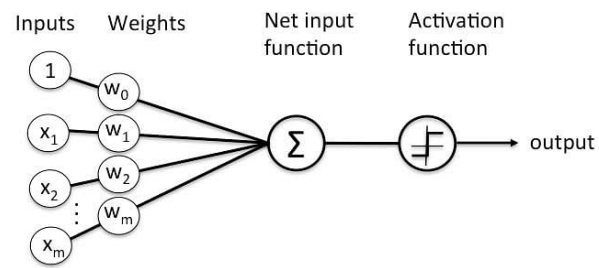
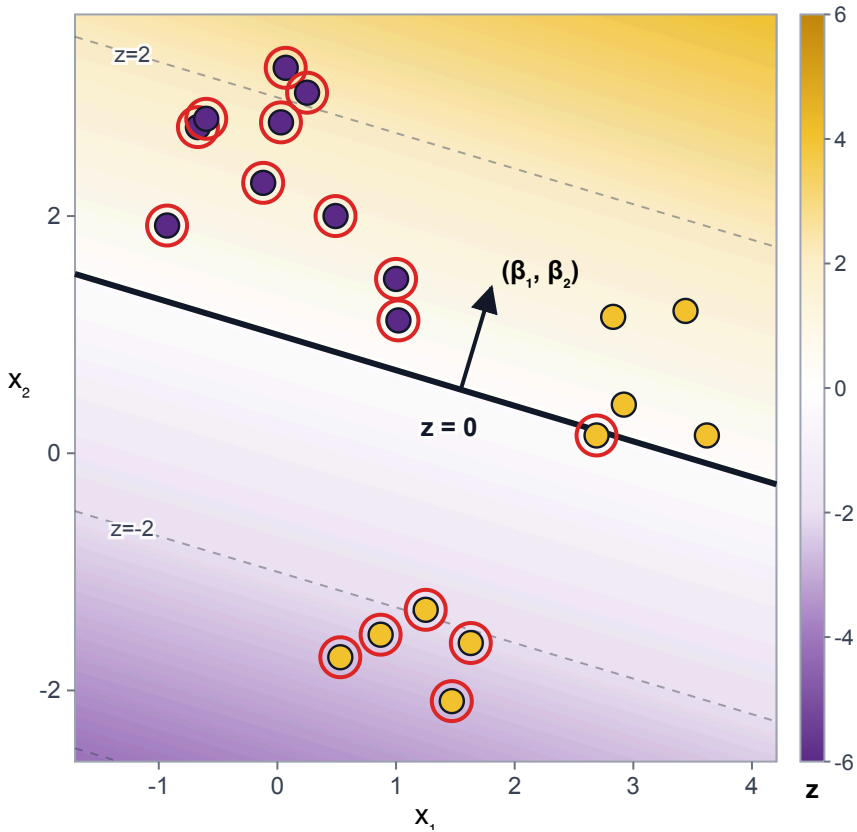


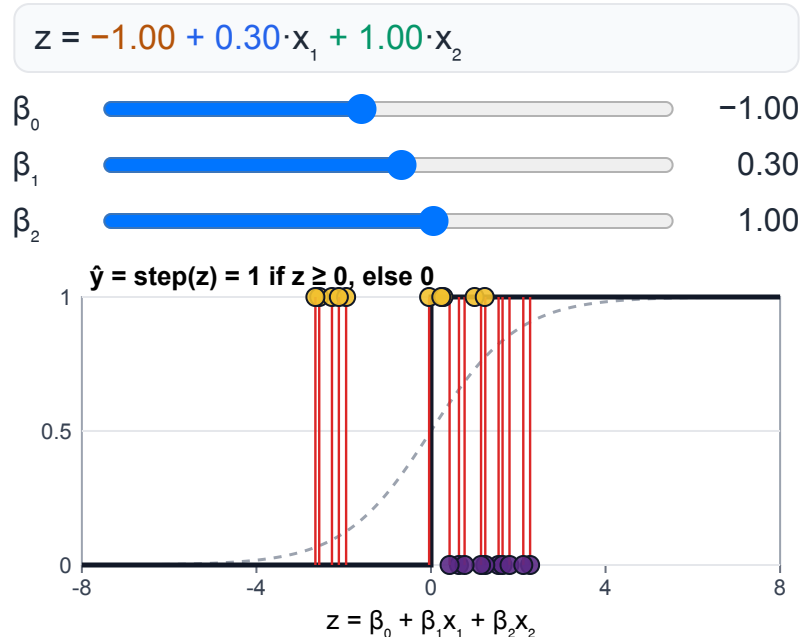
Figure: ESLR chapter 4.5




Orange line is linear regression

# Perceptron in 2D [Demo]




update **0** · misclassified **16**/20 · perceptron loss **25.18** · ‖β‖ **1.04**

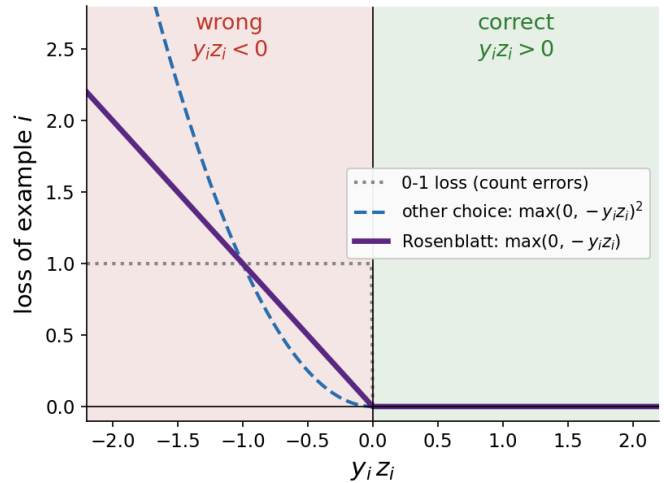
# Definition of a loss function

- $z_i = \sum_{j=1}^{d} w_j x_{ij} + b$ is the **signed distance to the hyperplane**
- For examples i with $y_i = 1$ this should be positive
- For examples i with $y_i = -1$ this should be negative

**Loss function** is not uniquely defined: it should be zero if all examples are correct, $> 0$ otherwise. One possible choice (Rosenblatt):

$$\text{loss} = \sum_i \max(0, -y_i z_i) = \sum_{i \in M} -y_i z_i$$

- Correct point gets zero: loss $= 0$ for separating plane, $> 0$ for non-separating plane
- Taking Gradient ($M$ are the misclassified examples)




$$\frac{\partial D(w,b)}{\partial w_j} = -\sum_{i \in \mathcal{M}} y_i x_{ij}, \qquad \frac{\partial D(w,b)}{\partial b} = -\sum_{i \in \mathcal{M}} y_i.$$

# Learning Rule [in python]

### Calculating the gradient by hand

Only visit wrongly classified points. Take gradient wrt w and b extra.

$$\frac{\partial l}{w} = -y \cdot w$$
$$\frac{\partial l}{b} = -y$$

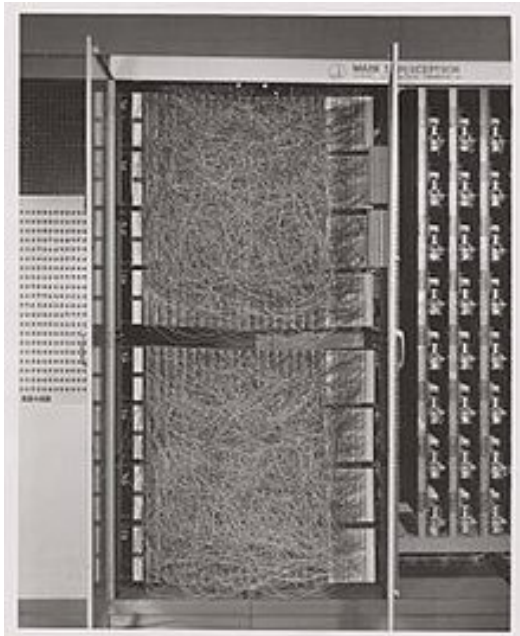
The gradient descent update formula becomes:

```
def update_params(X,y,w,b):
  didUpdate = False
  for ys,xs in zip(y,X): #some python show off
    if (ys * (np.dot(xs, w) + b) < 0):
      w = w + ys*xs#eta = 1
      b = b + ys*1
      didUpdate = True
  return w,b, didUpdate
```

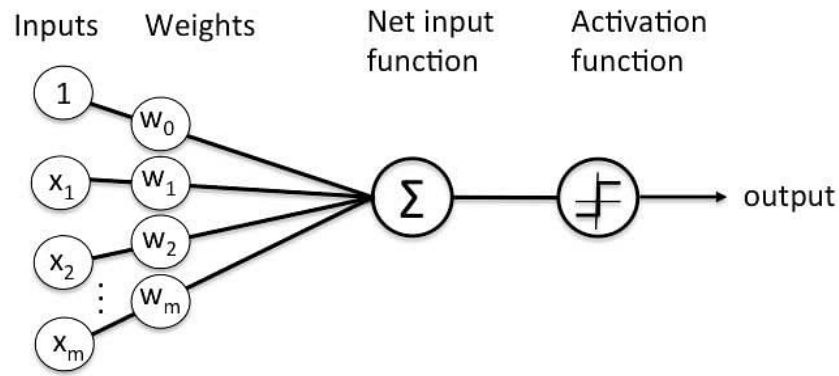
Algorithm converges if problem is separable:

See https://github.com/iosknn/mldl_htwg/tree/master/perceptron

# Perceptron as network (Frank Rosenblatt in 1957.)



Mark 1 Perceptron 1957
Wired to be a perceptron



# WS 2026 Ende erste Vorlesung

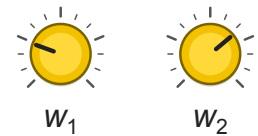
Feedback:

- Einstein Summationszeugs etwas verhaspelt und wahrscheinlich auch nicht soo wichtig
- Animation des Preceptrons dort muss man in der Mirror Mode gehen Striche oder so werden nicht angezeigt

# Probabilistic Treatment

# Linear Models so far

$x \to$ model parametrized with weights $\to \hat{y}$




Training = turn the knobs until a **loss** is small. But which loss?

| | Linear regression | Perceptron |
|---|---|---|
| **Model** | $\hat{y} = a \cdot x + b \in \mathbb{R}$ | $\hat{y} = \text{sign}(z),\ z = w_1 x_1 + w_2 x_2 + b$ |
| **Loss** | $\sum_i (y_i - \hat{y}_i)^2$ (quadratic) | $\sum_{i \in M} -y_i z_i$ (zero if separated) |
| **Idea** | small deviations from the data | just separate the two classes |
| **But** | Why quadratic? Why not $\lvert y_i - \hat{y}_i \rvert$? | Which of the separating lines? How sure is the classifier? |

Both losses make sense, but neither is **compelling**.
→ Probabilistic treatment: the loss follows from the model (maximum likelihood).

# Why probabilistic treatment?

- Inferences from data are intrinsically uncertain.
- Probability theory: model uncertainty instead of ignoring it.
- Unifying framework
  - Maximum Likelihood principle for loss function
  - Many ML Algorithms can be seen as probabilistic models
  - Almost all DL Algorithms can be seen in this framework
- Allows for generalizations
  - Linear Regression → Logistic Reg., Multinomial Regression, Count Data
  - Ada Boost → XGBoost
- Additional reading
  - Bishop Chapter 1.2.5 (with strange notation)
  - Deep Learning Book Chapter 3 (https://www.deeplearningbook.org/)
  - **Probabilistic Machine Learning: An Introduction**

# Probabilistic Models

# Probabilistic model: a die as the simplest example

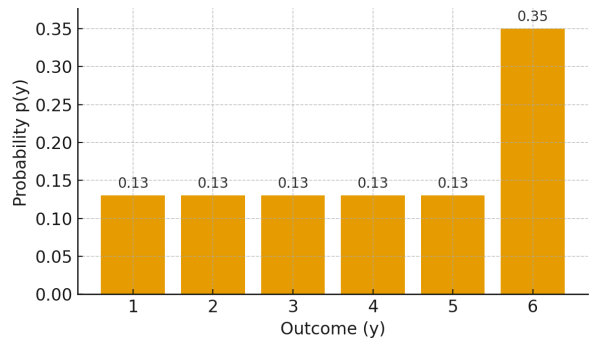
Probability Distribution for Data:

**Unconditional model:** $p_\theta(y)$

Parameters $\theta_1 = \text{prob of } 1,\ \theta_2 = \text{prob of } 2, \ldots$

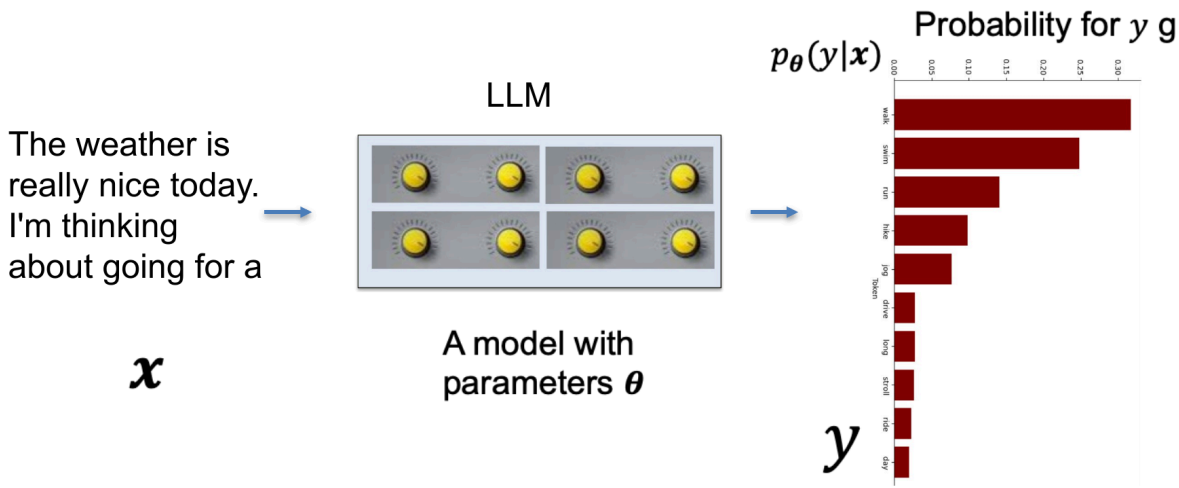
A model with parameters $\boldsymbol{\theta}$ $\longrightarrow$






Quiz: Number of parameters $\theta$ for a dice?

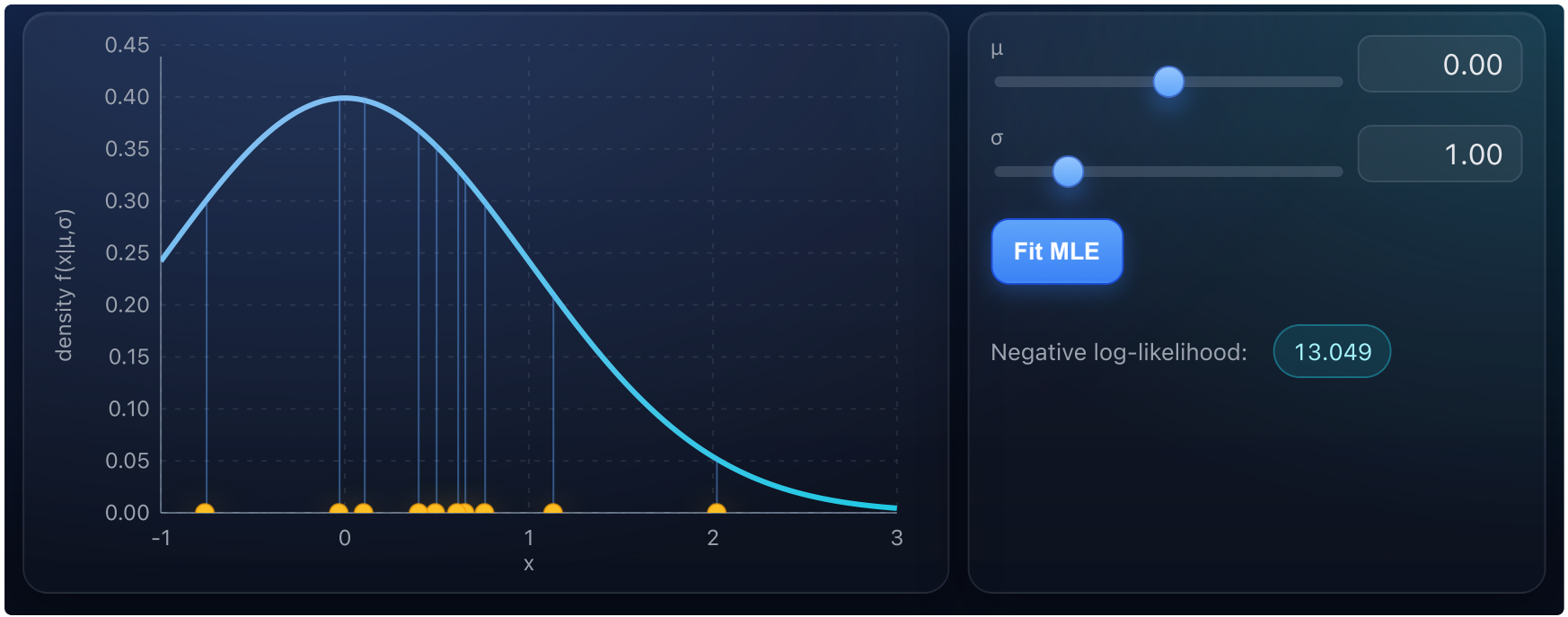
# The most famous probabilistic model




$p_\theta(y|x)$ is the probability for the next token (word) $y$ **given the context** $x$

Quiz: Number of parameters $\theta$ in LLMs?

# Ye Olde Gaussian — the archetype of probabilistic models



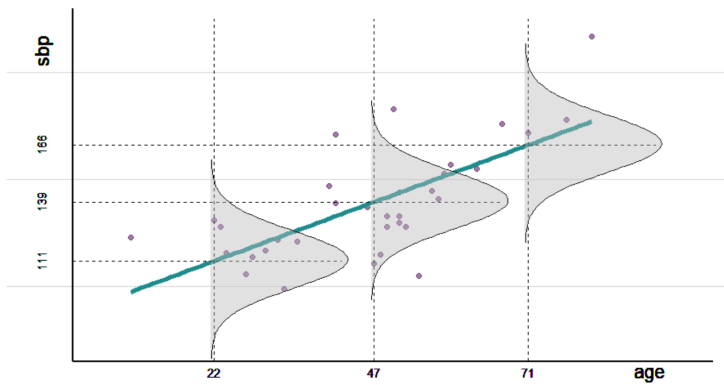

https://oduerr.github.io/stat_demos/normal_mle.html

Quiz: Number of parameters $\theta$ in Gaussian?

# Linear Regression as a probabilistic model

Trick: $\mu$ is different for each x, $\sigma$ is modeled as constant in standard linear regression ("homoscedastic")*

Gaussian with $\mu_x = a \cdot x + b$ and $\sigma = constant$

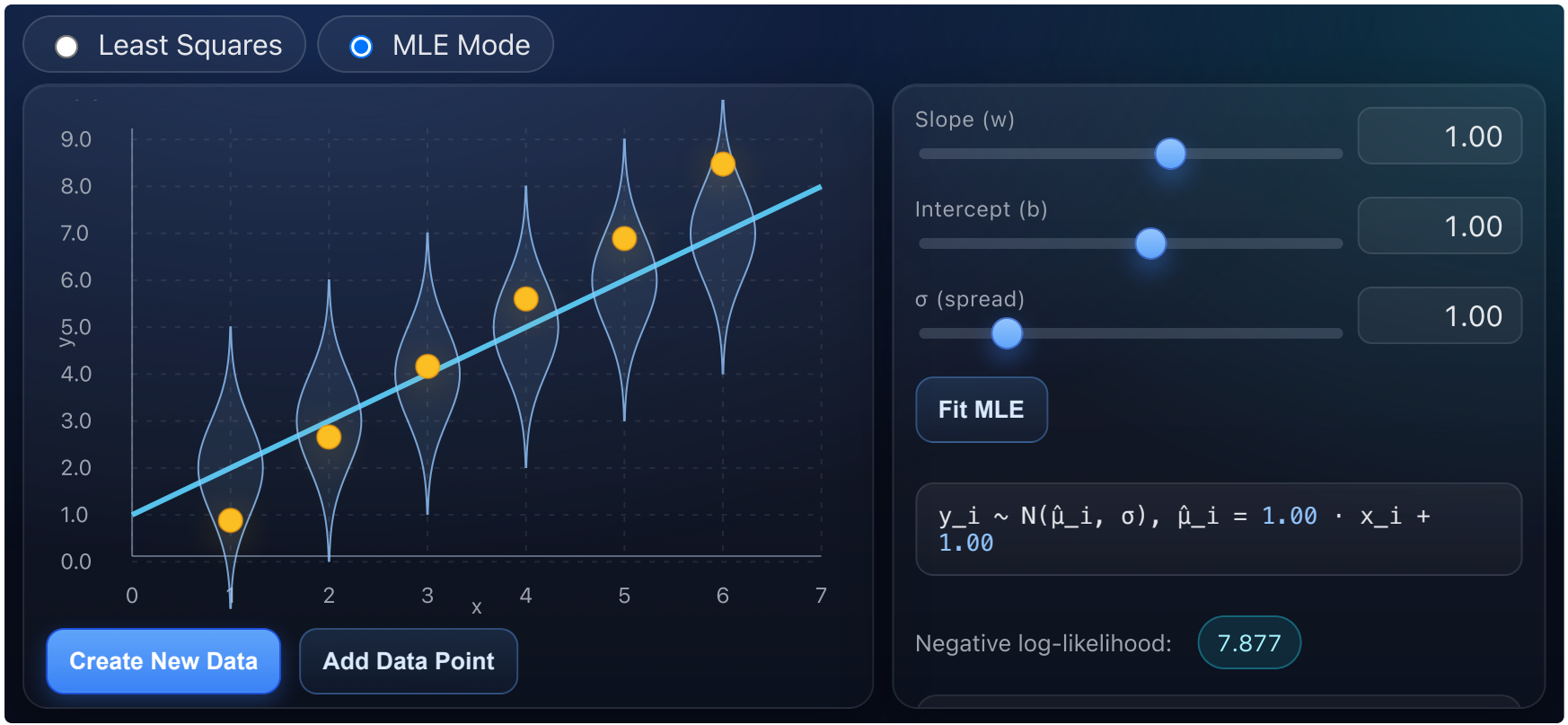



$$p_\theta(y_i|x_i) = N(y_i; \mu_{x_i}, \sigma)$$
$$\mu_{x_i} = a \cdot x_i + b$$

Quiz: Number of parameters $\theta$ in 1 dimensional linear regression?

*It's because of those words nobody likes stats.

# Linear Regression as a probabilistic model (demo)

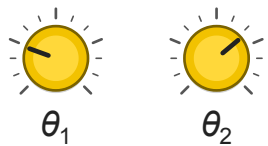



https://oduerr.github.io/stat_demos/linear_regression_nn.html

# Probabilistic Models (Summary)

- We model a random variable $Y_{|x}$

$x \rightarrow$ model parametrized with parameters $\theta \rightarrow$ parameters of distribution

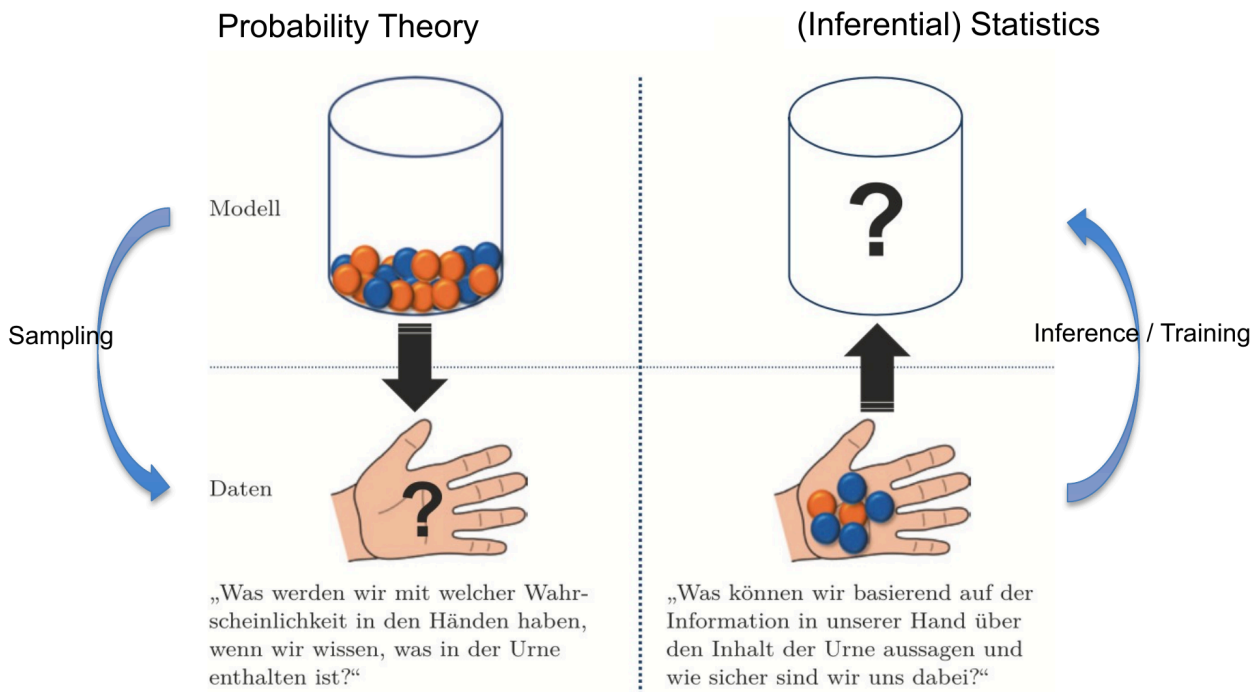



$$p_\theta(y|x) = \text{pdf}(\text{para} = f(x))$$

Examples of probabilistic models:

- Linear regression
  - $p_\theta(y|x) = \mathrm{N}(y; \mu = w \cdot x, \sigma = const.)$
- LLM
  - $p_\theta(y|x) = \mathrm{Categorical}(\mathrm{NN}_\theta(\mathrm{x}))$
- Dice (and other simple
  - $p_\theta(y) = \mathrm{Categorical}((\theta_1, \theta_2, \ldots))$
- Many other data classes like Count Data (exercise), censored data …

# Generative Probabilistic Models




Aus Lukas Meier W‘keit und Statistik, adapted

# Max Likelihood Optimization

**All probabilistic models can be fitted with it**

# On the training of probabilistic models

What ChatGPT says about itself



Describe your technology in one sentence.



I am a Generative Language model based on the transformer architecture that uses self-attention mechanisms and is trained with maximum likelihood principle to predict the next token in a sequence of tokens and generate new text.

# On the training of probabilistic models

If you have a probabilistic model $p_\theta(y|x)$ you have the loss-function using the **maximum likelihood principle**.



Maximum Likelihood Principle
The hammer of

- Classical Statistics
- Machine learning (when probabilistic)
- Deep Learning (99.5% of cases)

⚡ **If you use a different loss function, Thor — or I — will haunt you!** ⚡

# Probability of observed data

- In ML or DL context, we have
  - Observed data (training) data $D = (x_i, y_i)_{i=1}^N$
  - Probabilistic model $p_\theta(y|x)$
- Probability to observe the (training) data $D$
  - We assume independence of the training data and the that all comes from the same mode (independent identically distributed iid)
  - Remember $P(A, B) = P(A) \cdot P(B)$ if A and B are independent
  - $p_\theta(D) = p_\theta(y_1|x_1) \cdot p_\theta(y_2|x_2) \dots p_\theta(y_N|x_N) = \prod_{i=1}^N p_\theta(y_i|x_i)$
  - This quantity is called likelihood, and is sometimes written as
    - $p_\theta(D) = p(D|\theta)$

# Maximum Likelihood

## (one of the most beautiful ideas in statistics)

Tune the parameter(s) θ of the model $p_\theta(D)$ so that (observed) data *D* is most likely

In math

$$\hat{\theta} = \mathrm{argmax}(\mathrm{p}_\theta(D))$$



Ronald Fisher in 1913

Also used before by Gauss, Laplace

# A recipe for training probabilistic models

**Step1 (build probabilistic model)**

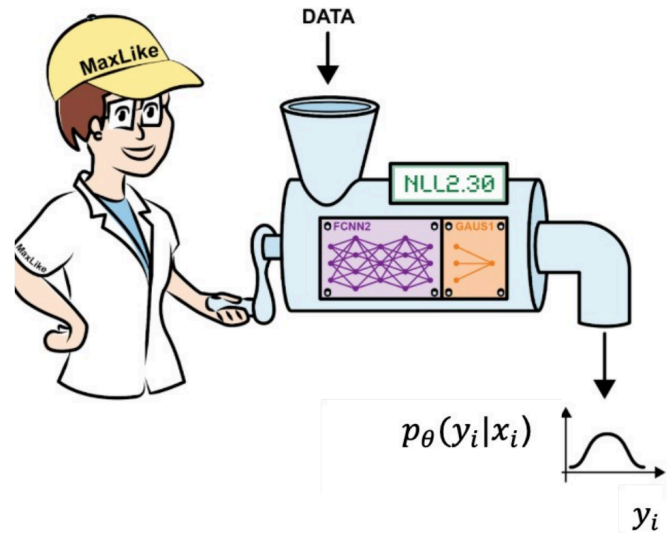
- Construct a Model which has parameters $\theta$ and outputs a probability for observed Data $D$

  $$p_\theta(D)$$

- In Regression the probability of the observed (trainings) data $D = (x_i, y_i)_{i=1}^N$ is given by

  $$p_\theta(D) = \prod_{i=1}^{N} p_\theta(y_i|x_i)$$

- $p_\theta(y_i|x_i)$ is probability (density) of the model for observing the ith data point.




**Step2 (Optimize probability of observed data)**

$$\hat{\theta}_{ML} = \text{argmax}\left(p(D|\theta)\right)$$

# A recipe for training probabilistic models

**Step2 (Optimize probability of observed data)**

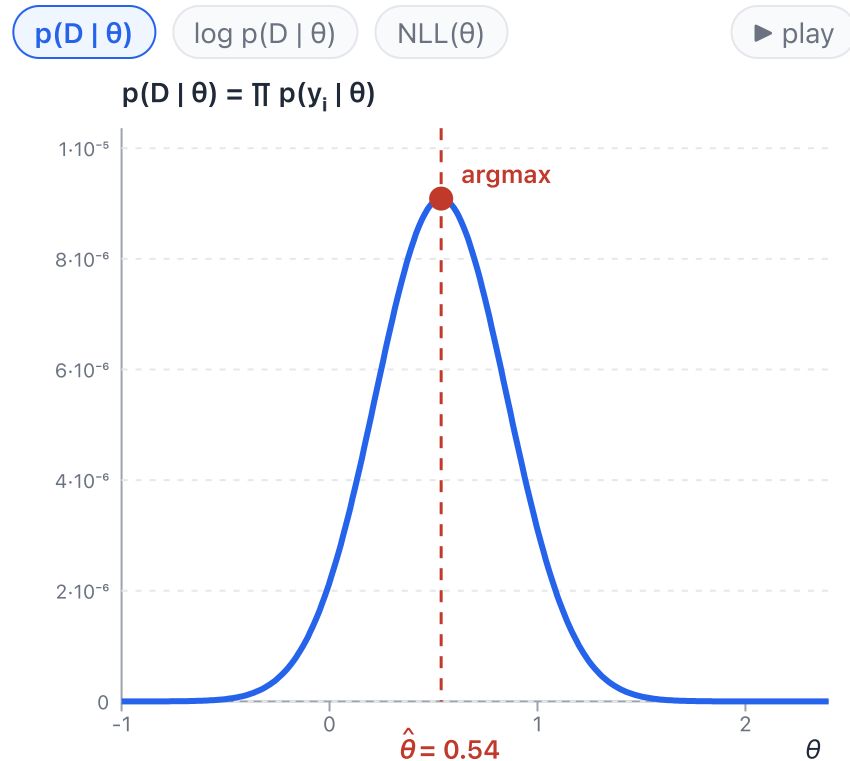
- $\hat{\theta}_{ML} = \text{argmax}\left(p(D|\theta)\right)$
- $p_\theta(D) = \prod_{i=1}^{N} p_\theta(y_i|x_i)$
- $\hat{\theta}_{ML} = \text{argmax}\left(\prod_{i=1}^{N} p_\theta(y_i|x_i)\right)$

Instead often, (yielding same result $\hat{\theta}$)

$$\hat{\theta}_{ML} = \text{argmax}\left(\sum \log(p_\theta(y_i|x_i))\right)$$

$$\hat{\theta}_{ML} = \text{argmin}\left(-\frac{1}{N}\sum_{i=1}^{N} \log(p_\theta(y_i|x_i))\right)$$

NLL=$-\frac{1}{N}\sum_{i=1}^{N} \log(p_\theta(y_i|x_i))$




stat_demos/likelihood_log_nll.html

# Example: Fitting Ye Olde Gaussian by Maximum Likelihood

Given the datapoints $D = (y_1, \ldots, y_N)$ from a Gaussian

$$y_i \sim N(\mu, \sigma)$$

We tune $\theta = (\mu, \sigma)$ so that

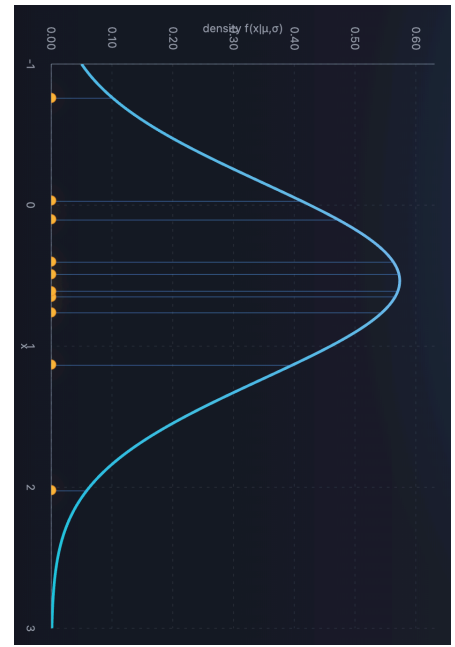
$$p_\theta(D) = \prod_{i=1}^{N} N(y_i;\, \mu, \sigma)$$

is maximal. $N(y;\, \mu, \sigma)$ is the pdf of the Gaussian.

Technically, we minimize the NLL:

$$\mathrm{NLL}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \log N(y_i;\, \mu, \sigma)$$

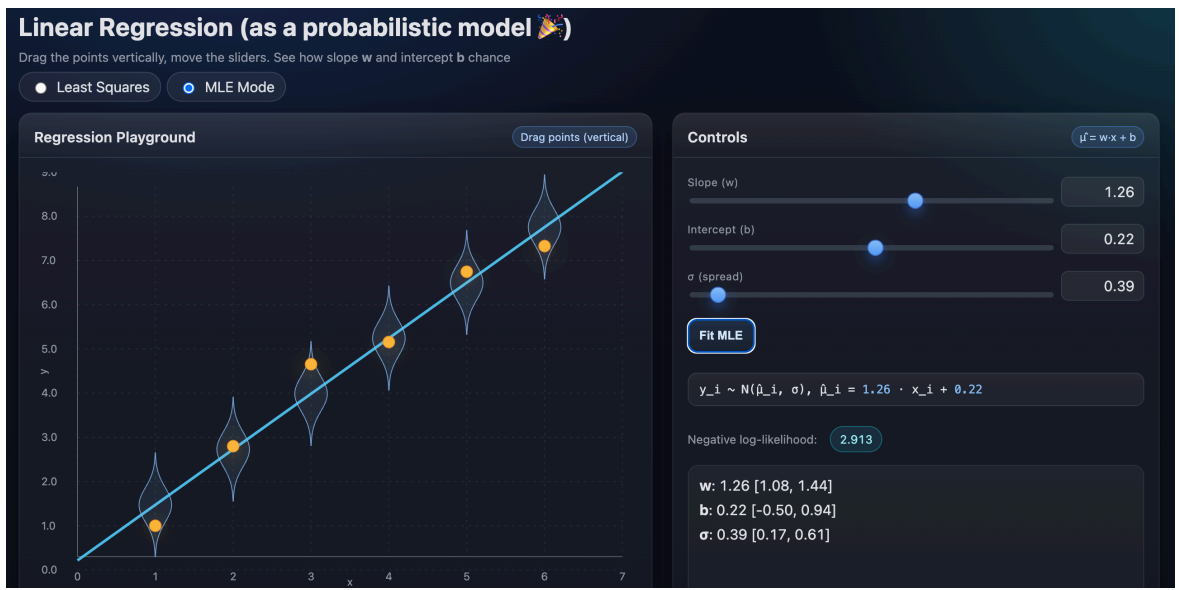
$p_\theta(y)$




https://oduerr.github.io/stat_demos/normal_mle.html

# Example: Linear Regression

- Maximizing
  - $p_\theta(y_i|x_i) = \prod N(y_i; a \cdot x_i + b, \sigma)$
  - For all data points




https://oduerr.github.io/stat_demos/linear_regression_nn.html

# Derivation of the MSE

Model: $y_i \sim N(\mu_i, \sigma)$ with $\mu_i = \sum_{j=1}^{d} w_j x_{ij} + b$, parameters $\theta = (w_1, \ldots, w_d, b)$, $\sigma$ fixed

$$\hat{\theta} = \operatorname*{argmax}_{\theta} \left\{ \prod_{i=1}^{N} p_\theta(y_i \mid x_i) \right\} = \operatorname*{argmax}_{\theta} \left\{ \prod_{i=1}^{N} N(y_i; \sum_j w_j x_{ij} + b, \sigma) \right\} \quad (1)(2)$$

$$= \operatorname*{argmax}_{\theta} \left\{ \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left( -\frac{(\sum_j w_j x_{ij} + b - y_i)^2}{2\sigma^2} \right) \right\} \quad (3)$$

$$= \operatorname*{argmin}_{\theta} \left\{ -\log \left( \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left( -\frac{(\sum_j w_j x_{ij} + b - y_i)^2}{2\sigma^2} \right) \right) \right\} \quad (4)$$

$$= \operatorname*{argmin}_{\theta} \left\{ \sum_{i=1}^{N} -\log\left( \frac{1}{\sqrt{2\pi\sigma^2}} \right) + \frac{1}{2\sigma^2} (\sum_j w_j x_{ij} + b - y_i)^2 \right\} \quad (5)$$

$$= \operatorname*{argmin}_{\theta} \left\{ \frac{1}{2\sigma^2} \sum_{i=1}^{N} (\sum_j w_j x_{ij} + b - y_i)^2 \right\} \quad (6)$$

**(1)** Independent data: likelihood = product of the densities

**(2)** Plug in the model: $y_i \sim N(\mu_i, \sigma)$

**(3)** Write out the Gaussian pdf

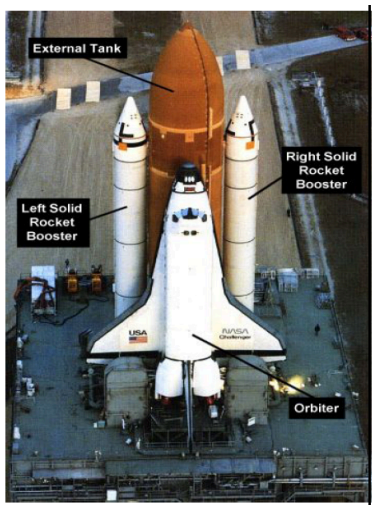
**(4)** $-\log$ is decreasing: max becomes min, same $\hat{\theta}$

**(5)** $\log$ of a product = sum of the logs; $\log(\exp(z)) = z$

**(6)** Drop the constant (no $\theta$ in it). Left: sum of squared errors $= N \cdot \mathrm{MSE}$, times $\frac{1}{2\sigma^2}$: same minimum

# Logistic Regression

# Statistik & Challenger Desaster [motivation of logistic]







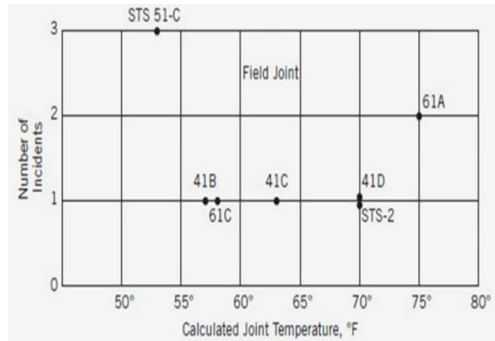

Die bemannte Raumfähre Challenger explodierte 1986 nach dem Start, weil die Dichtungsringe an den Boostern nicht dicht hielten.

Untersuchungskommission (Rogers Commission)
Richard Feynman (Nobel Preisträger in Physik), demonstriert die fehlende Elastizität Dichtungsringes bei kleinen Temperaturen.
**Wie konnte man das übersehen** …

# 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 |

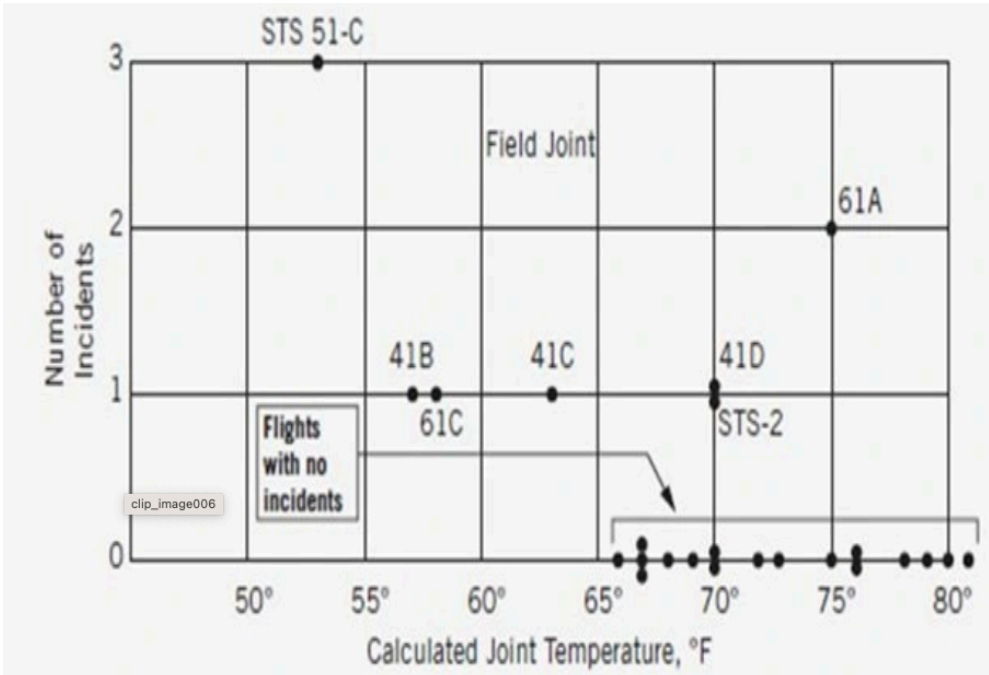



Figures from: PRESIDENTIAL COMMISSION on the Space Shuttle Challenger Accident (https://history.nasa.gov/rogersrep/v4part3.htm)

- 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

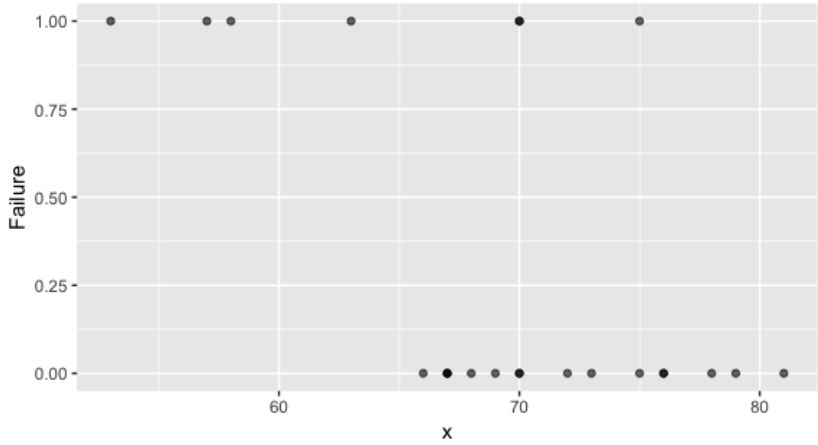


# Modelling with logistic regression

Binarize to a zero / one classification

Want: $p(x) = P(Y = 1|x)$

Prob. for a O-ring to be defect Y=1 at a given temperature X




Notation: statistics books and R / statsmodels write $\beta_0 = b$ and $\beta_j = w_j$

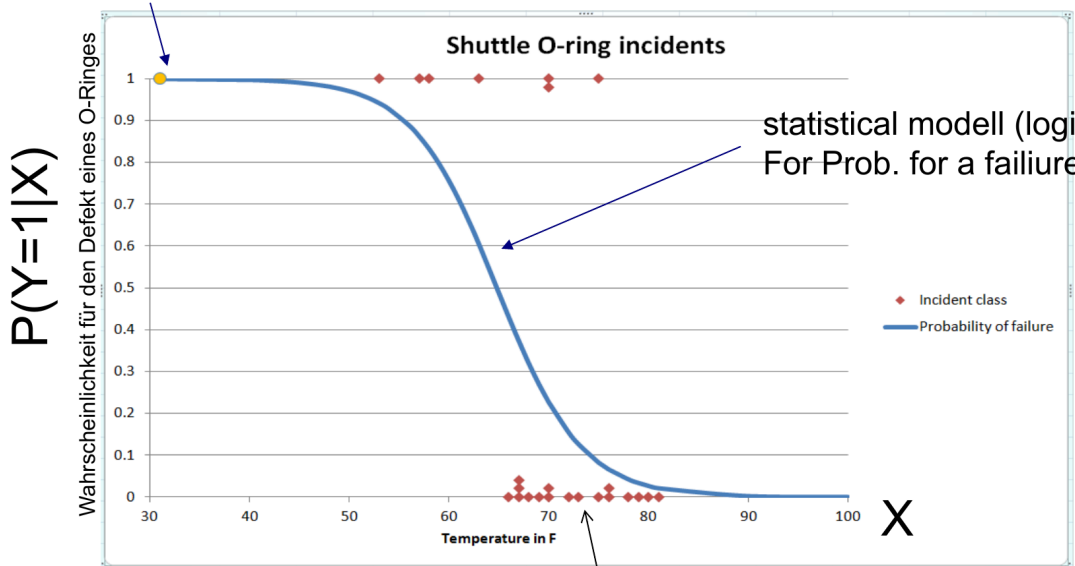
**Question:**

- **Guess curve?**
- **Why is $p(x) = w_1 x + b$ (linear regression) wrong?**

# 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

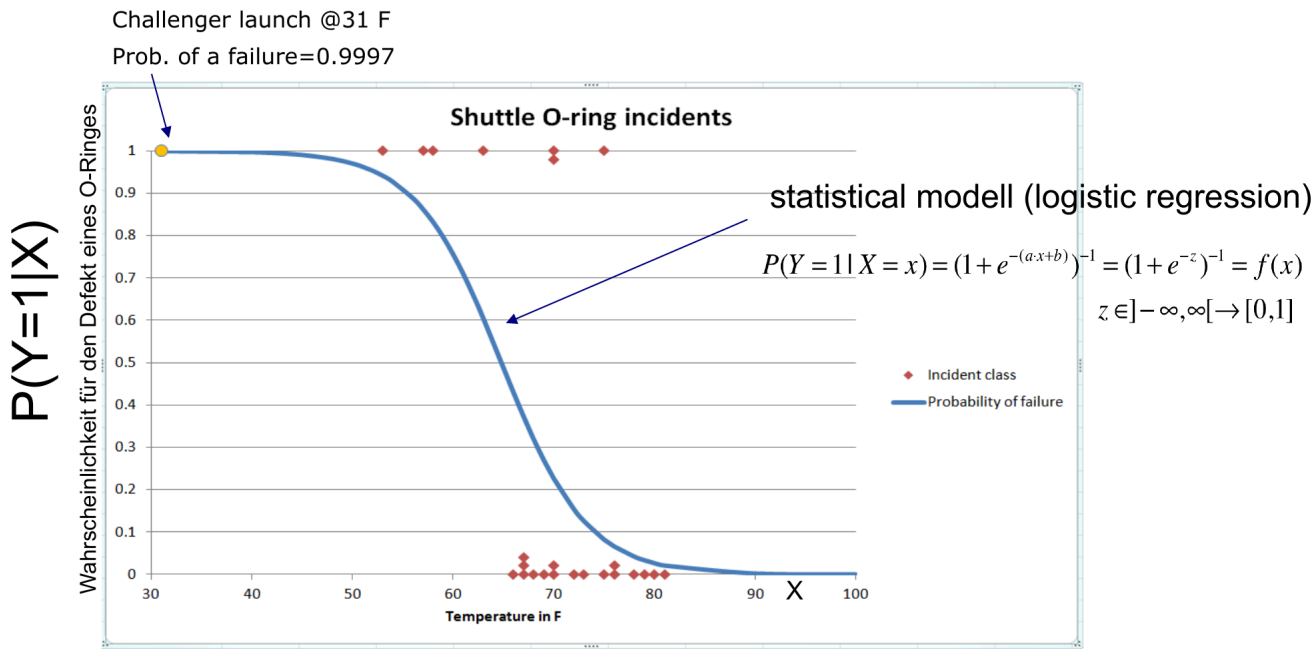



statistical modell (logistic regression)
For Prob. for a failiure (Y=1)

No failiure Y=0

# Logistic Regression

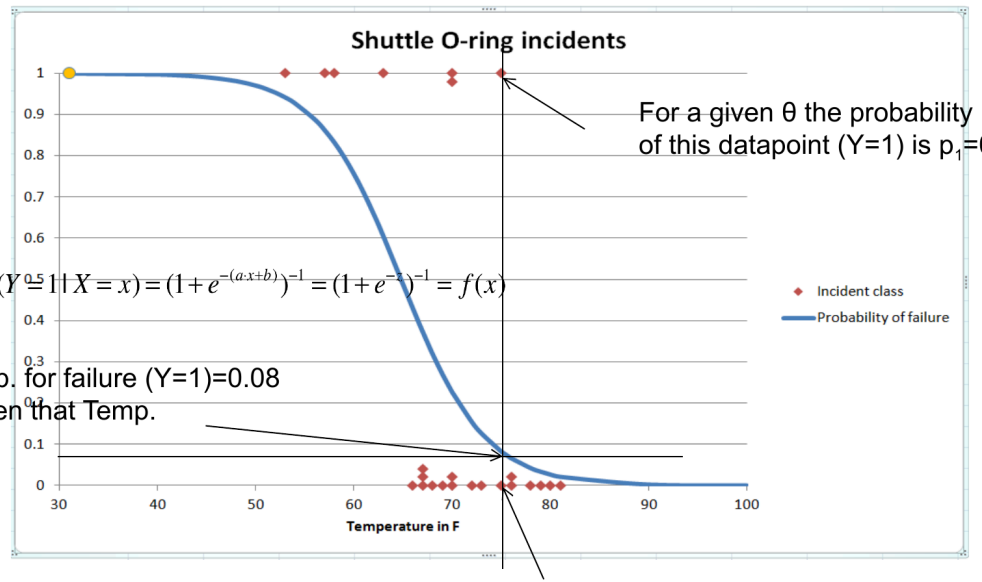
Predict if O-Ring is broken, depending on temperature




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

# Likelihood: Probability of a single observation

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




Y

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

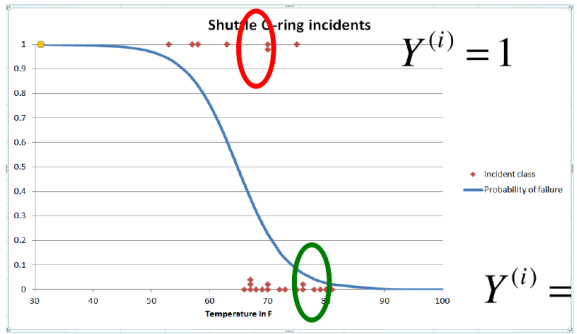
For a given θ the probability of this datapoint (Y=1) is $p_1$=0.08

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

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

Taking log (maximum of log is at same position)

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

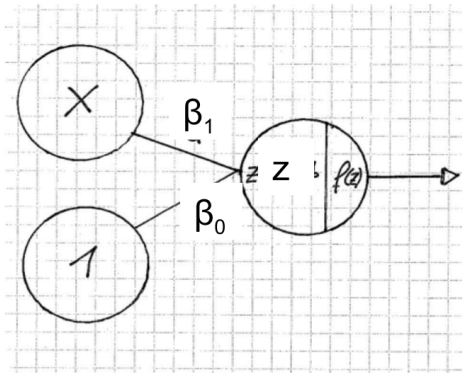
Loss function (negative log-likelihood, cross-entropy)

$$NLL = -\frac{1}{N}\sum_{i=1}^{N} \log(p_\theta(y^{(i)}|x^{(i)})$$

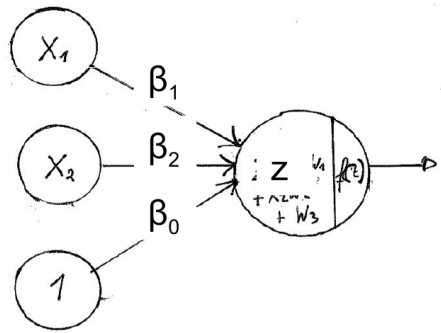
This is the prob. the model evaluates for the true class $y_i = y^{(i)}$ of training example $x_i = x^{(i)}$

# Logistic Regression the mother of all neural networks

1-D log Regression




Multivariate Log.-Regression

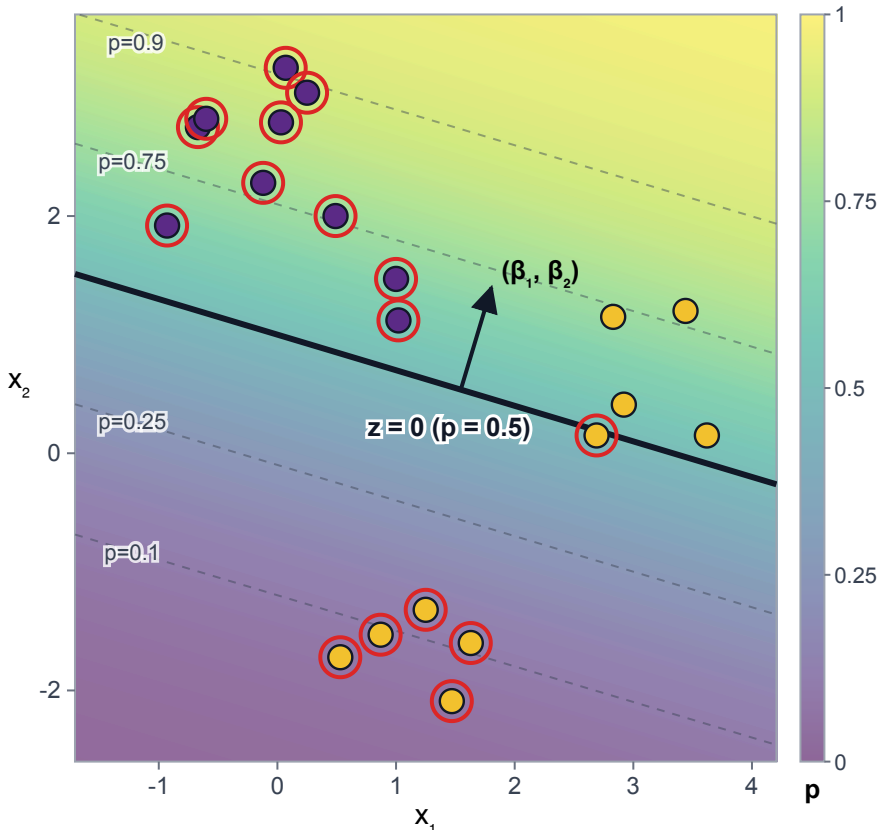



$$z = \beta_0 + \beta_1 x$$

$$z = \beta_0 + x_1\beta_1 + x_2\beta_2 = \beta^T x$$

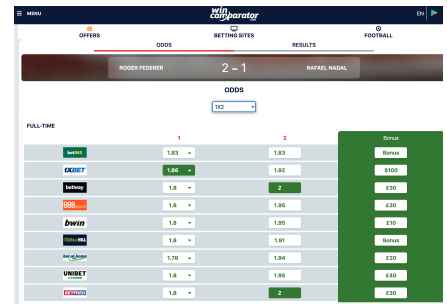
$$p_1(x) = P(Y = 1 \mid X = x) = [1 + \exp(-\beta^T x)]^{-1} = \frac{\exp(\beta^T x)}{1 + \exp(\beta^T x)} = f(\beta^T x)$$

# Logistic Regression in 2D [Demo]



# Interpretation with odds

- Probability for event
  - $p(x) = \frac{e^{w_1 x+b}}{1+e^{w_1 x+b}}$
  - $odds(x) := \frac{p(x)}{1-p(x)} = e^{w_1 x+b}$
  - $\log\left(odds(x)\right) = w_1 x + b$
- Facts
  - Modeling the log odds as linear regression
  - Linear Boundary
  - Odds in range 0 to $+\infty$
  - Log-odds in range $-\infty$ to $+\infty$




https://www.wincomparator.com/roger-federer-id2930-rafael-nadal-id2905/

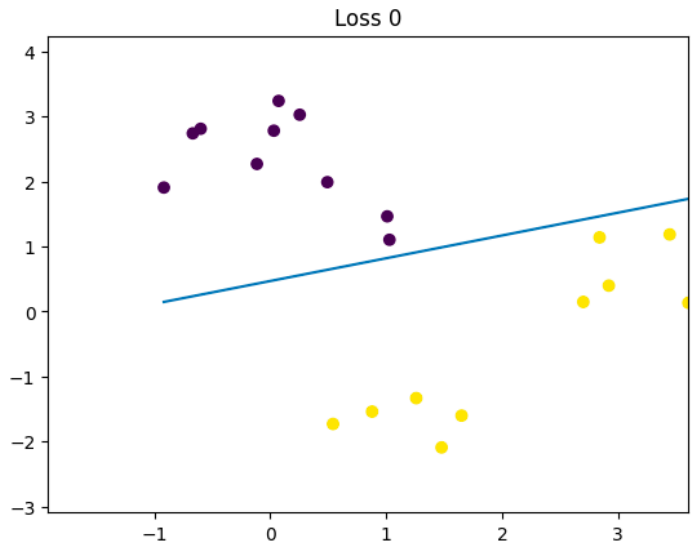
- Interpretation (odds):
  - Examples
    - Odds : „Prob for winning“ / „Prob for not winning“
      - Federer against Nadal 2:1  Two times more likely that Federer wins
      - Horse Clever Hans wins 3:1  Three times more likely that Hans wins (vs. not winning)
    - Odds : „Probability of broken o-ring“ vs. Non-broken
      - 8:1

# Odds ratios

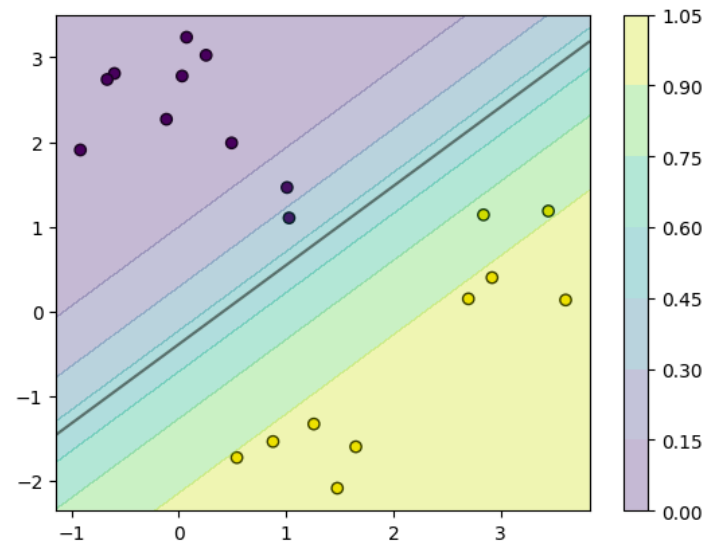
- Q: How does the odds change with $x$
- Example Tennis (Federer against Nadal)
  - $x = 0$ (not grass) $odds(x = 0) = \exp(b) = 1.2$
  - $x = 1$ (grass) $odds(x = 1) = \exp(w_1 \cdot 1 + b) = 2$
  - Odds Ratio: $OR_{x=0 \to x=1} = \frac{odds(x=1)}{odds(x=0)} = \exp(w_1 \cdot 1)$
  - $\log(OR_{x=0 \to x=1}) = w_1$
- Works also for continuous $x$ important is just $\Delta x$
- For more than one variable $x$ rest needs to be constant
- Question
  - Consider challenger example x is temperature in Fahrenheit
    - What is $w_1$ (just the sign)?

# Comparing Logistic Regression vs Perceptron

Perceptron



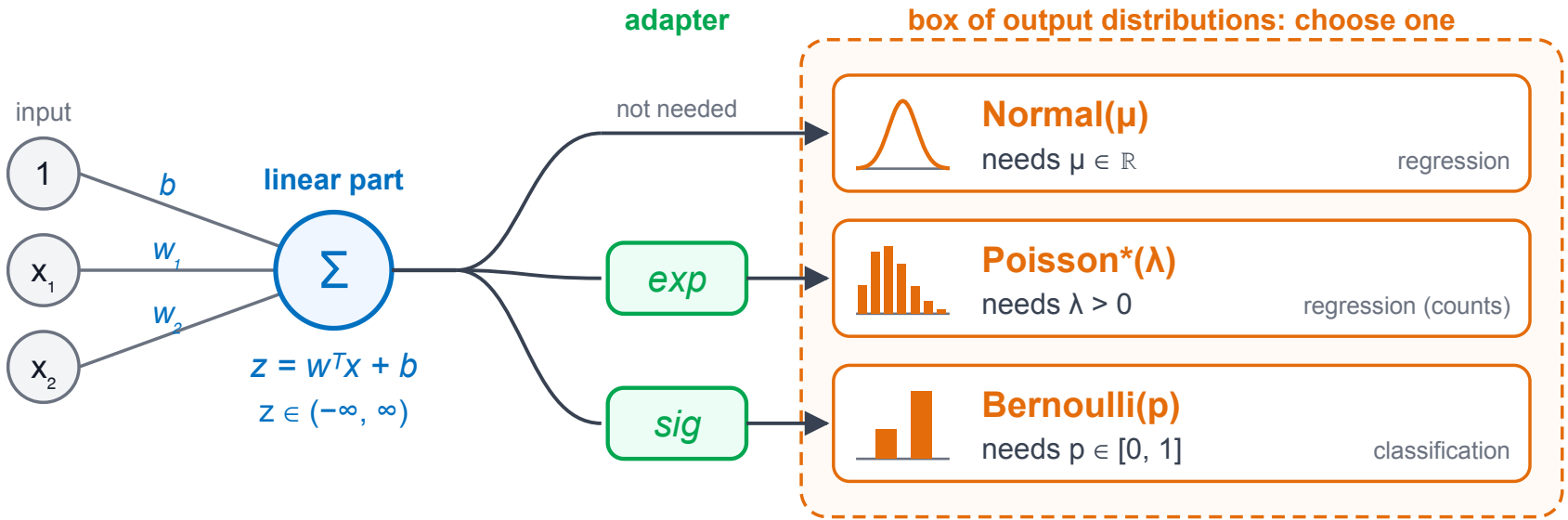

Log-Regression



```
logreg = LogisticRegression()
logreg.fit(X, y)
```

github/ioskn/mldl_htwg/blob/master/perceptron/perceptron.ipynb

# Wrapping up: one linear part, many output distributions




Pick the distribution that fits $Y$. The adapter maps $z \in (-\infty, \infty)$ into the range of its parameter.

**Deep learning:** replace the linear part by a deep network, keep the box and the adapter

* Poisson regression for counts: see exercise (camper data)

# The loss is derived, not invented

**Ad hoc: a new invention for each model**

- Least squares: "take the squared error" — why not $|\cdot|$ or $(\cdot)^4$?
- Perceptron: "penalize wrong sides" — any separating line has loss 0
- New problem (counts, ratings, …)? Invent again.

**Probabilistic: one recipe for all models**

1. Choose the distribution of $Y$ given $x$
2. Minimize $\mathrm{NLL}(w) = -\frac{1}{N}\sum_i \log p_w(y_i \mid x_i)$

Bonus: the model also predicts its uncertainty ($\sigma$, $p$).

| You choose $Y \sim$ | The NLL becomes (up to constants) | Known as |
|---|---|---|
| $\mathrm{Normal}(\mu = w^\top x,\ \sigma)$ | $\frac{1}{2\sigma^2} \cdot \frac{1}{N}\sum_i (y_i - w^\top x_i)^2$ | MSE — least squares, **derived** |
| $\mathrm{Laplace}(\mu = w^\top x,\ b)$ | $\frac{1}{b} \cdot \frac{1}{N}\sum_i \lvert y_i - w^\top x_i \rvert$ | MAE — robust regression |
| $\mathrm{Poisson}(\lambda = \exp(w^\top x))$ | $\frac{1}{N}\sum_i \left(\lambda_i - y_i \log \lambda_i\right)$ | Poisson regression — counts |
| $\mathrm{Bernoulli}(p = \mathrm{sig}(w^\top x))$ | $-\frac{1}{N}\sum_i \left[y_i \log p_i + (1-y_i)\log(1-p_i)\right]$ | cross-entropy — logistic regression |

**Choose the distribution, and the loss follows. The same recipe trains deep networks.**

# Note for Exercise

Arbeitsblatt 2, part "The Fish Data Set":
Linear_Regression_Eigengesichter_Naive_Bayes.ipynb

# The camper example

N=250 groups visiting a national park

**Y=count: number of fishes caught**

X1=persons: number of persons in group

X2=child: number of children in the group

X3=bait: indicates of life bait was used

X4=camper: indicates if camper is brought



Data: https://stats.idre.ucla.edu/r/dae/zip

# Aufgabe Poisson Regression

- Modeling Count Data
- Data Set
  - $x_i$ features (number of people)
  - Want to predict number of fish $k_i$
  - $k_i \sim Pois(\text{rate})$
    - $rate$ must be positive
  - Use similar trick as the $\sigma$ in Logistics
  - $\text{rate} = \mu_i = \exp(\sum_j w_j x_{ij} + b)$
- Observation
  - $k_i$ number of fish
  - ➔ Likelihood

