Hochschule Konstanz
Fakultät Informatik

Prof. Dr. Oliver Dürr

KI Vorlesung: Machine Learning(I)

Status: Kleine Änderungen bis zum Begin der Vorlesung moeglich

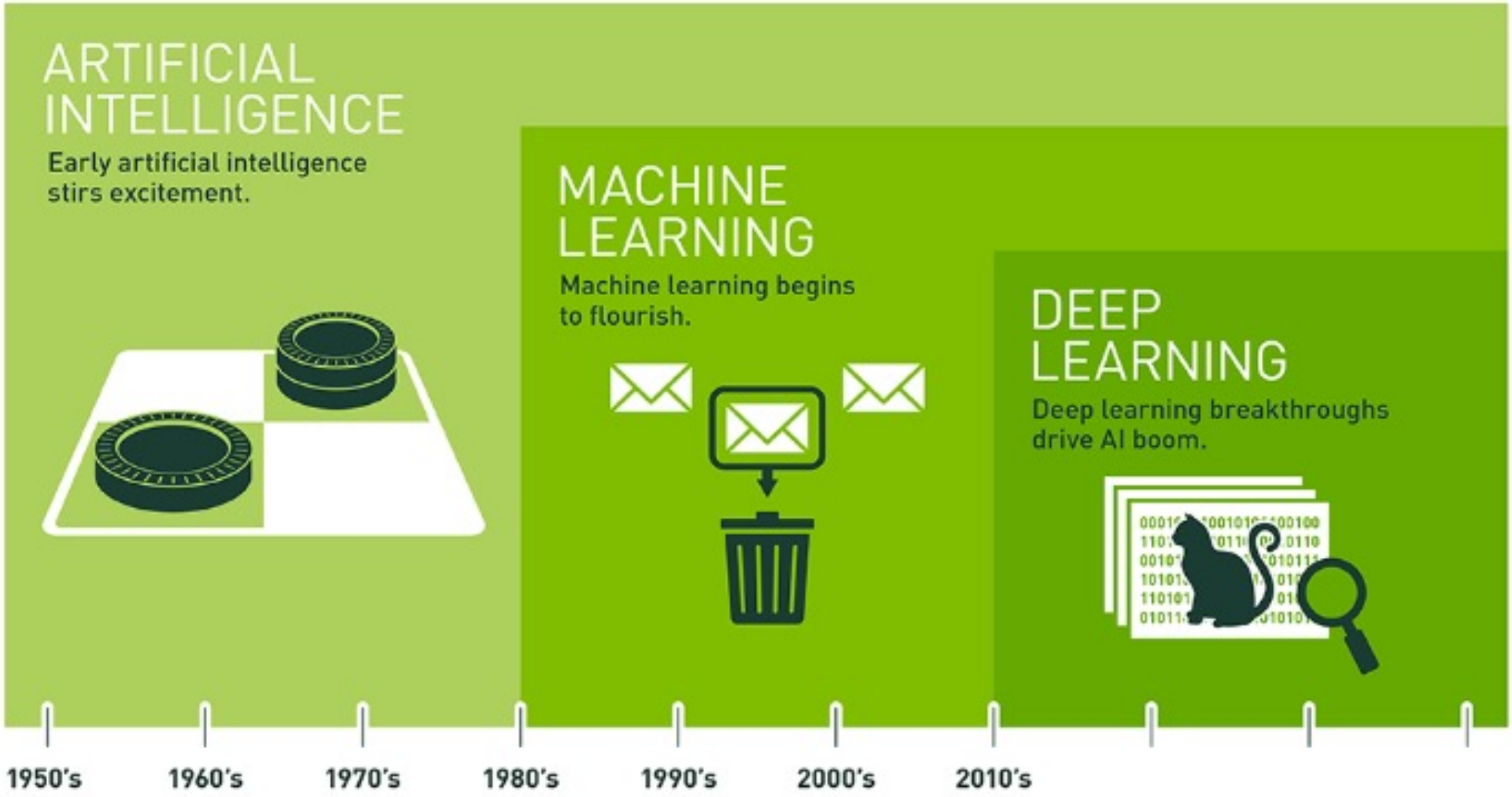
Remark: Much of the material have been developed together with Beate Sick (ZHAW/Uni Zurich).

Learning Objective / Further Resources

- Objectives
  - Know the difference between supervised and unsupervised learning
  - Know the principles of training supervised linear regression
  - Understand gradient descent
  - Understand linear regression

- Further Resources:
  - https://developers.google.com/machine-learning/crash-course/
    - Too? strongly focus on google tool chain
  - Linear regression is taken from beginning of chapter 3 of
    - https://www.manning.com/books/probabilistic-deep-learning

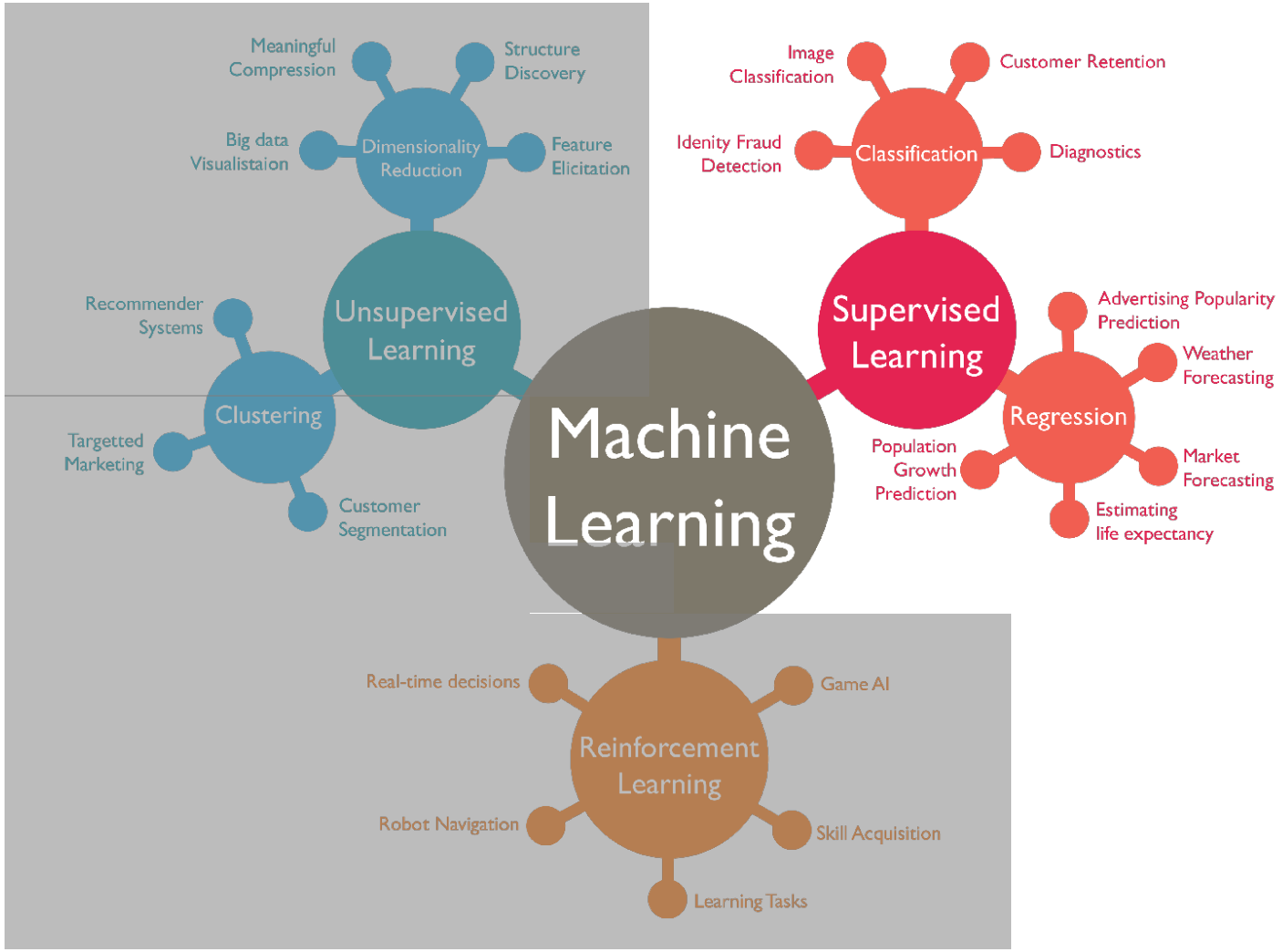
AI, Machine Learning, Deep Learning



Slide credit: https://www.datasciencecentral.com/profiles/blogs/artificial-intelligence-vs-machine-learning-vs-deep-learning

Definition

"ML […] *gives computers the ability to learn* ***without being explicitly programmed***" attributed to A. Samuel, 1959



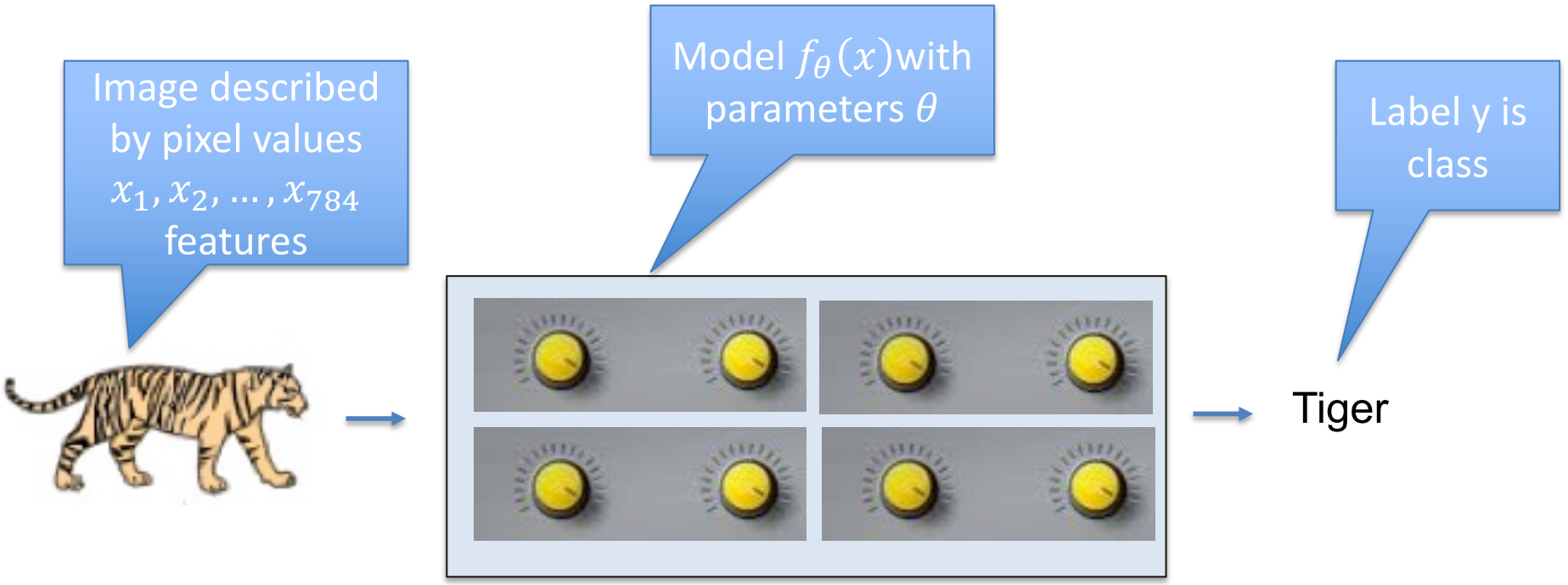
Examples are given

Example are created

Image credit: https://www.wordstream.com/blog/ws/2017/07/28/machine-learning-applications

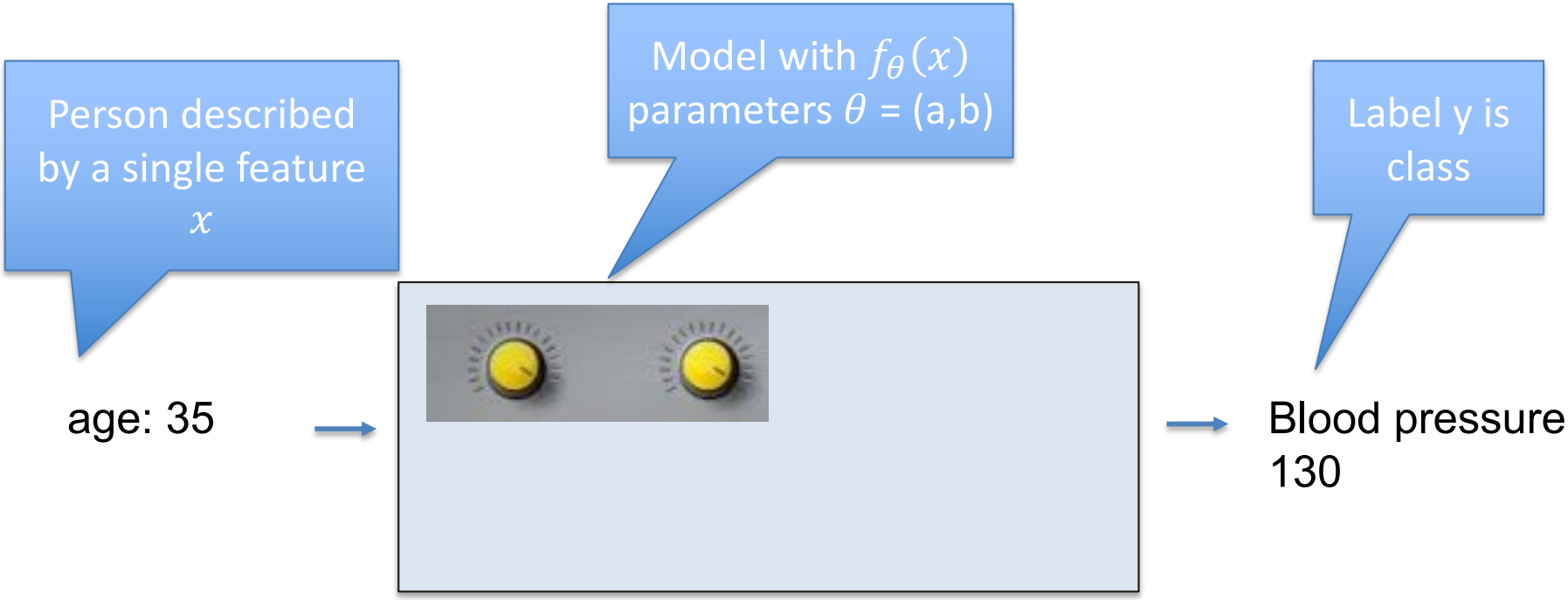
High Level View
Supervised Learning
(with parametric Models)

Supervised Learning: Image Classification



Complex models can have millions / billions of parameters

Supervised Learning: Lineare Regression

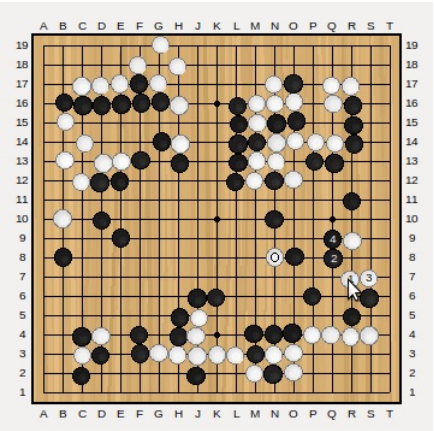


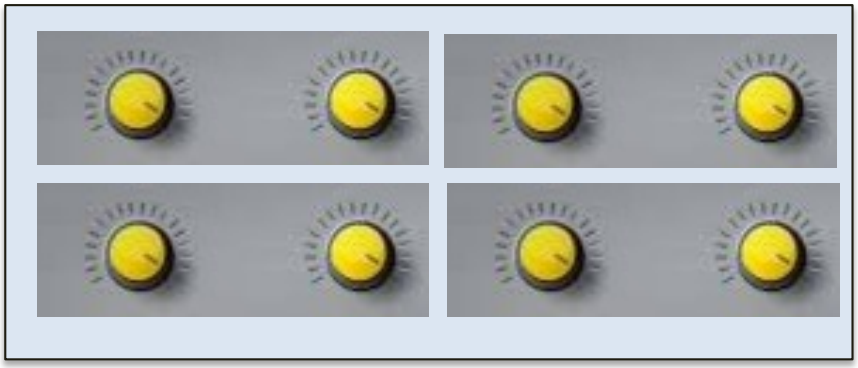
This simple model has two parameters

Prinzipielle Funktionsweise: GO

Input

Output





„A6 “

Deep Learning Modelle mit Mio. von Parametern

Wrap up: parameteric models

- Definition Supervised ML
  - ML systems learn how to combine input x to produce useful predictions $\hat{y}$ on never-before-seen data.

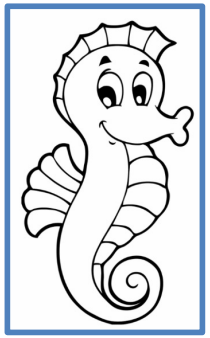
- All examples show fit in the same scheme of parameteric models

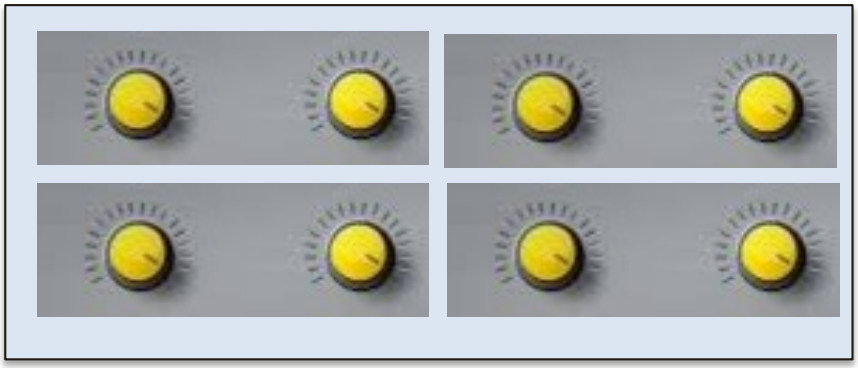
$$\hat{y} = f_\theta(x)$$

The hat indicates a prediction

- The parameters θ are learned from examples…

Supervised Learning: Training (Image Classification)

x	True Class y		Predicted class $\hat{y}$
	Tiger	→	Seal 👎
	Tiger	→	Tiger 👍
	Seah orse	→	Seahorse 👍
...			



Typical 1 Mio. Trainingsdata

Training Principle:

Parameter are adjusted so that training error is minimal.

Training of a Model

x	True Value y	Predicted Value $\hat{y}$
Alter: 55	152	130
Alter: 35	155	151
Alter: 55	140	131
...		

Training Principle:

Parameter are adjusted so that training error is minimal.

Warp up training

- Training Data is given in pairs $\{x^{(i)}, y^{(i)}\}_{i=1,\ldots,N}$
- Training is minimizing the error in the trainingdata, quantified by a loss.
  - Typical losses
    - Mean Squared Error for regression
    - Log-loss for classification

- All machine and deep learning methods in this course follow the principle
  1. Define a model f_θ (e.g. linear regression, deep neural networks)
  2. Train the model on the training data, by minimizing an appropriate loss
  3. The trained model can than be used for predictions

In code

```
from sklearn.linear_model
    import LinearRegression
model = LinearRegression()
```

A specific parametric model

```
model.fit(X_train,y_train)
```

X_train training data e.g. [N,2]
y_train lables [N,1]
Parameters of the model are optimized to the training data

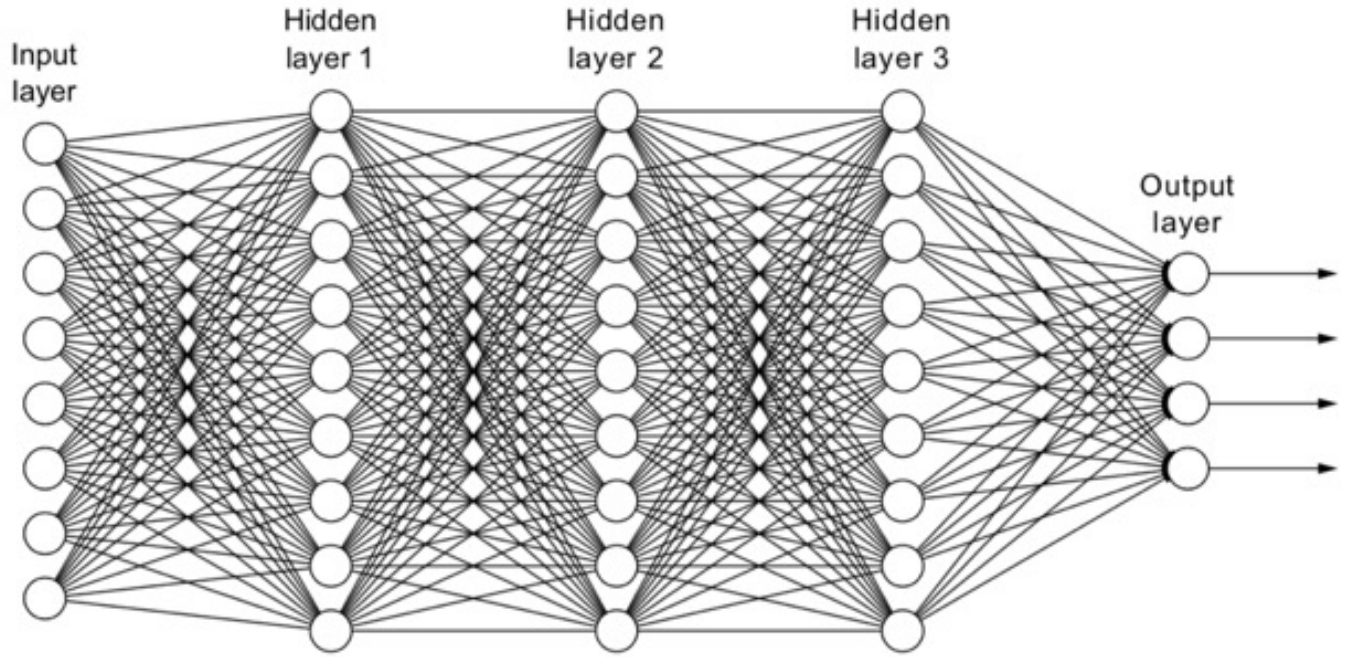
```
X_test = np.array([[3, 5]])
model.predict(X_test)
```

predict on unknown data

In math: $\hat{y} = f_\theta(x)$

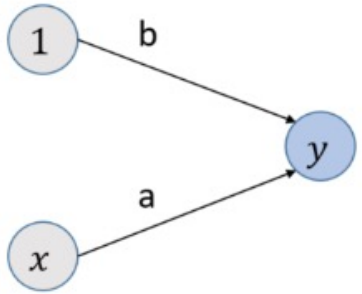
Models in ML

- In this course, we focus on neural network like models.



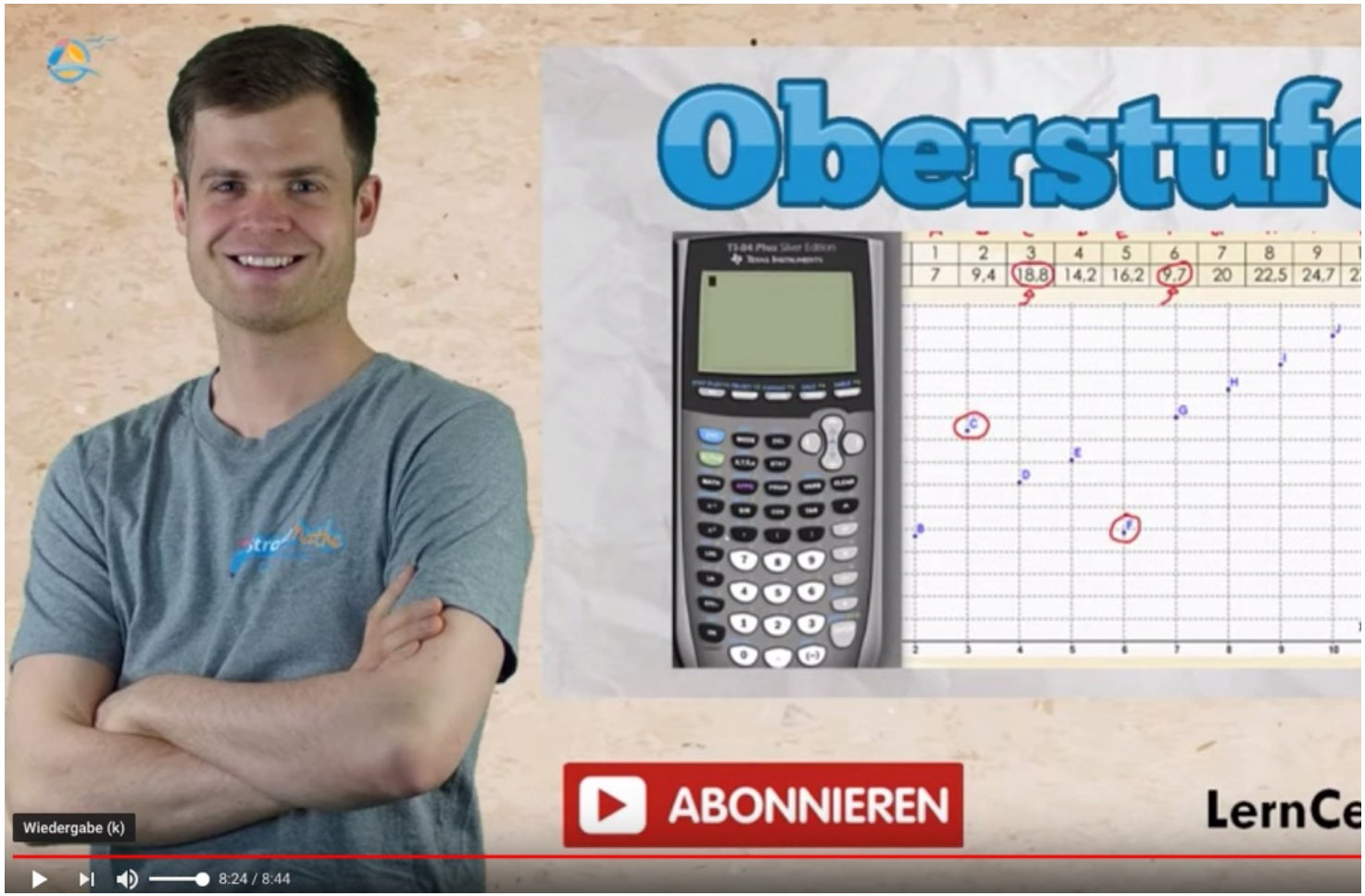
- Not covered in this course are e.g. decision trees, random forests, ensemble methods and support vector machines

Linear regression



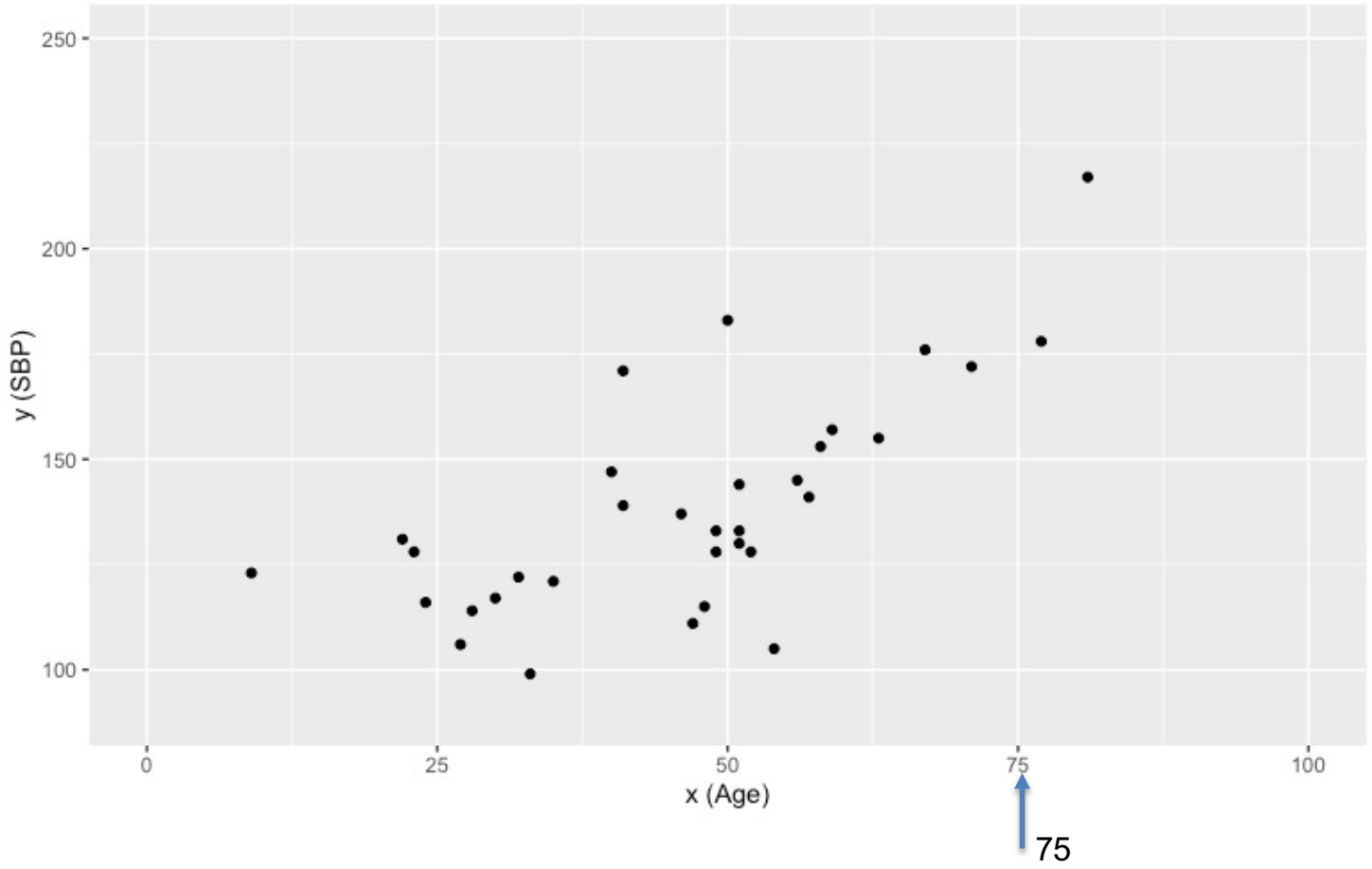
Lineare Regression || Oberstufe:
https://www.youtube.com/watch?v=lehGScwWnpo

- Good old linear regression is mother of all ML and DL



Example Predicting Blood Pressure

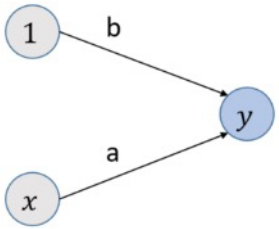
- Training Data
  - Systolic Blood Pressure (SBP) against age of 33 women

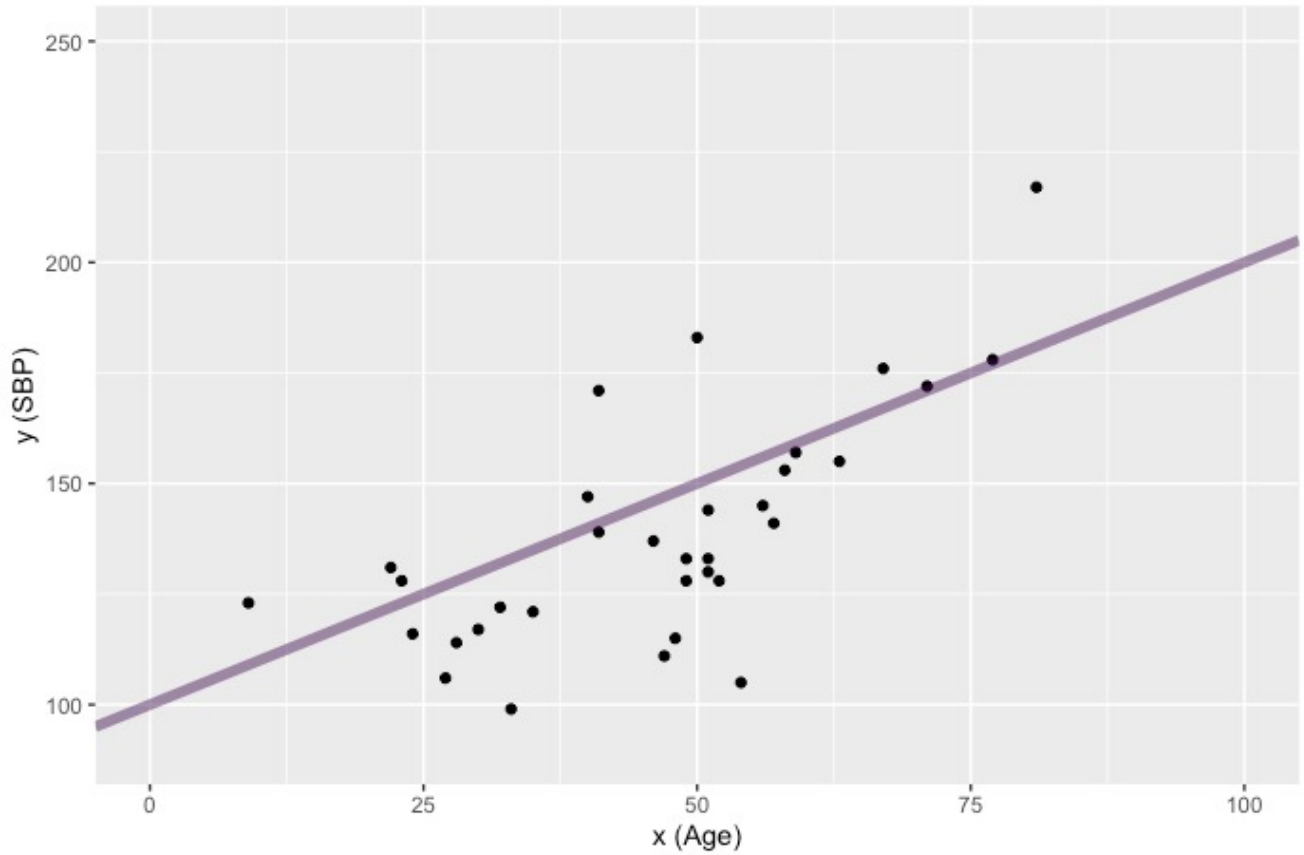


Model should predict blood pressure at e.g. 75 what is a good guess?

Simple Model linear regression

- $\hat{y} = a \cdot x + b$
  - a slope
  - b intercept





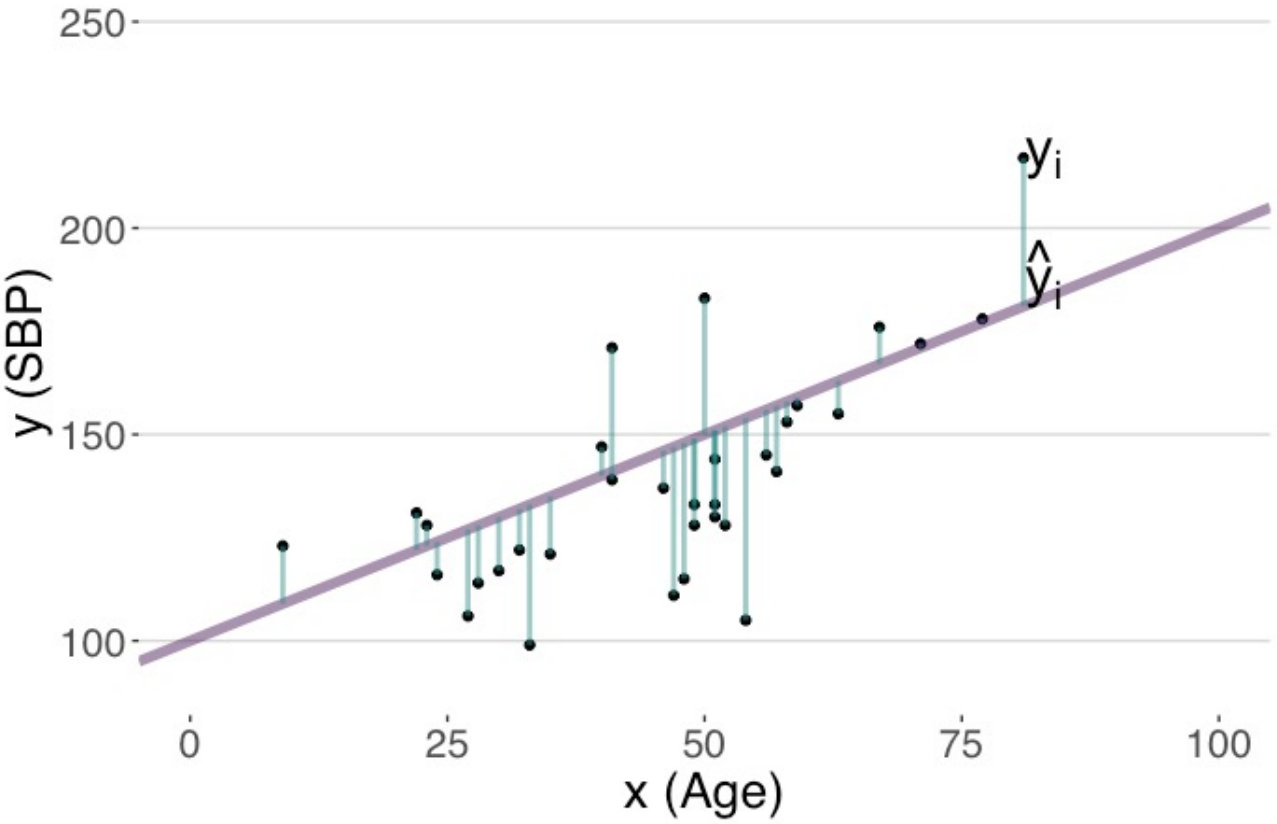
$\hat{y} = 1 \cdot x + 100$

For age=75

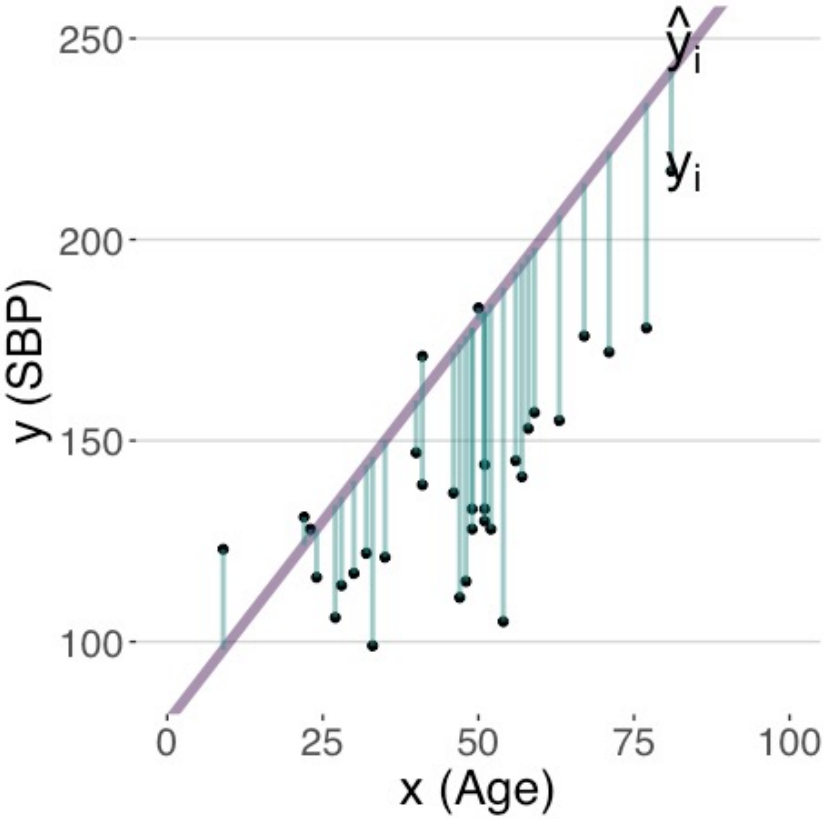
$\hat{y} = 1 \cdot 75 + 100 = 175$

Loss on trainingsdata

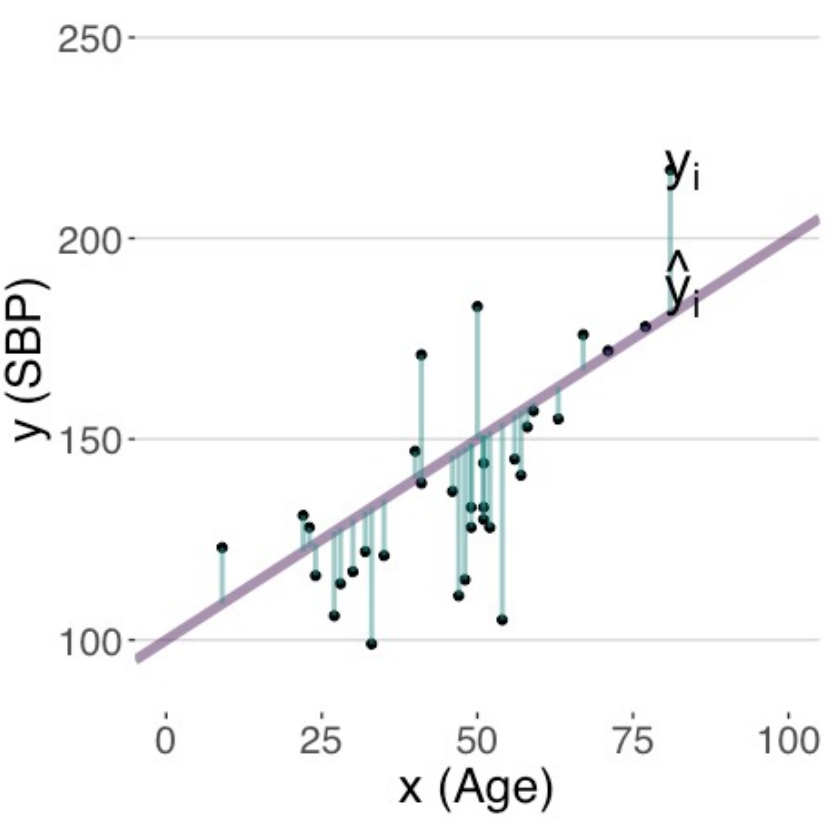
- Trainingdata* in pairs $\{x_i, y_i\}$ i $= 1, \ldots, N$



Two examples with high and low loss



Bad fit, high loss



Good fit, low loss

$$\text{loss} = \text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Loss on trainings data

Table 3.1 Data and derived quantities for the first 5 entries in figure 3.3, calculated with slope = 1 and intercept = 100

x	y	$\hat{y} = a \cdot x + b$	Residual	Squared residual
22	131	122	9	81
41	139	141	–2	4
52	128	152	–24	576
23	128	123	5	25
41	171	141	30	900

Typical loss for regression (mean squared error) MSE

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

Listing 3.1 Calculation of the MSE in Python with NumPy

```
a = 1
b = 100
y_hat = a*x + b
r = (y - y_hat)
MSE = np.sum(np.square(r)) / len(y)
MSE
```

Column 3 in table 3.1 (note the vector notation and broadcasting)

MSE calculated from column 5 in table 3.1 (created, summed, and divided by the number of data points in the data set).

Column 4 in table 3.1

Finding the best values $\hat{a}, \hat{b}$

- Locking for the solution with minimal loss on training data

$$(\hat{a}, \hat{b}) = \text{argmin MSE}(a, b)$$

in general

$$\hat{\theta} = \text{argmin loss}(\theta)$$

- For linear regression, there is an analytic solution

$$\frac{\partial \text{loss}}{\partial a} = 0 \text{ and } \frac{\partial \text{loss}}{\partial b} = 0$$

$$\hat{a} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \text{ and } \hat{b} = \bar{y} - \hat{a} \cdot x$$

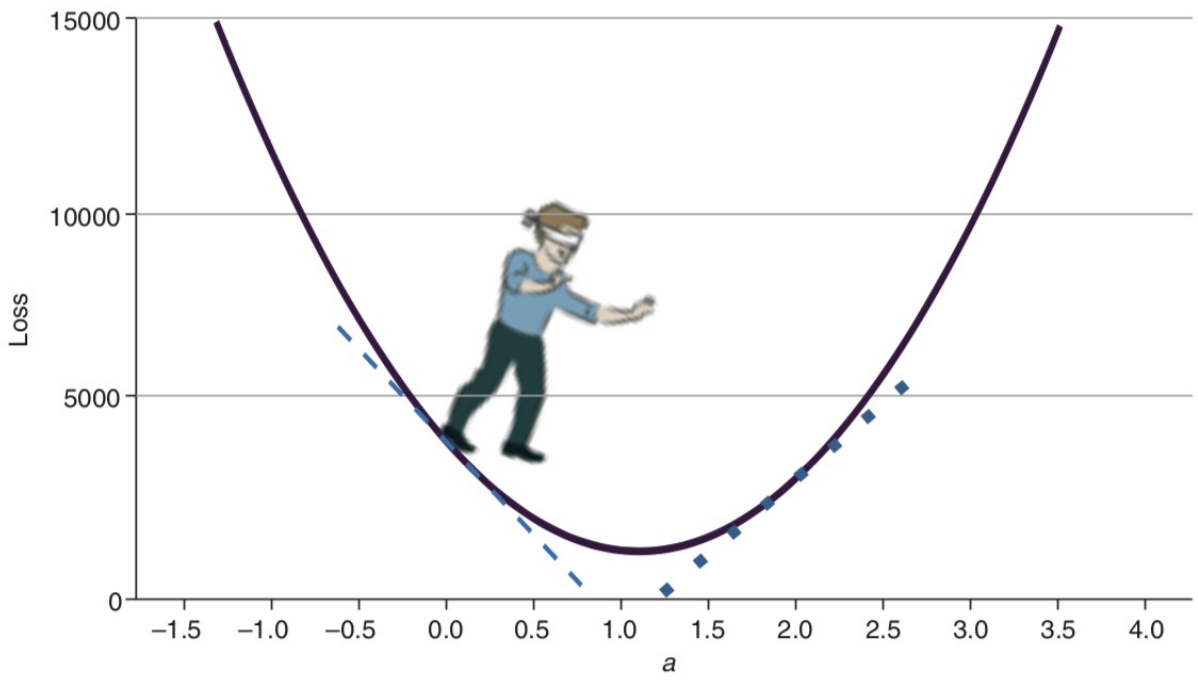
- Gradient descent methods works for most ML methods

Gradient Descent for one variable

- First assume, we know the optimal value for $b = 87.6$.

$$\text{loss} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - (a \cdot x_i + 87.6))^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - a \cdot x_i - 87.6)^2$$

- Intuition for gradient descent



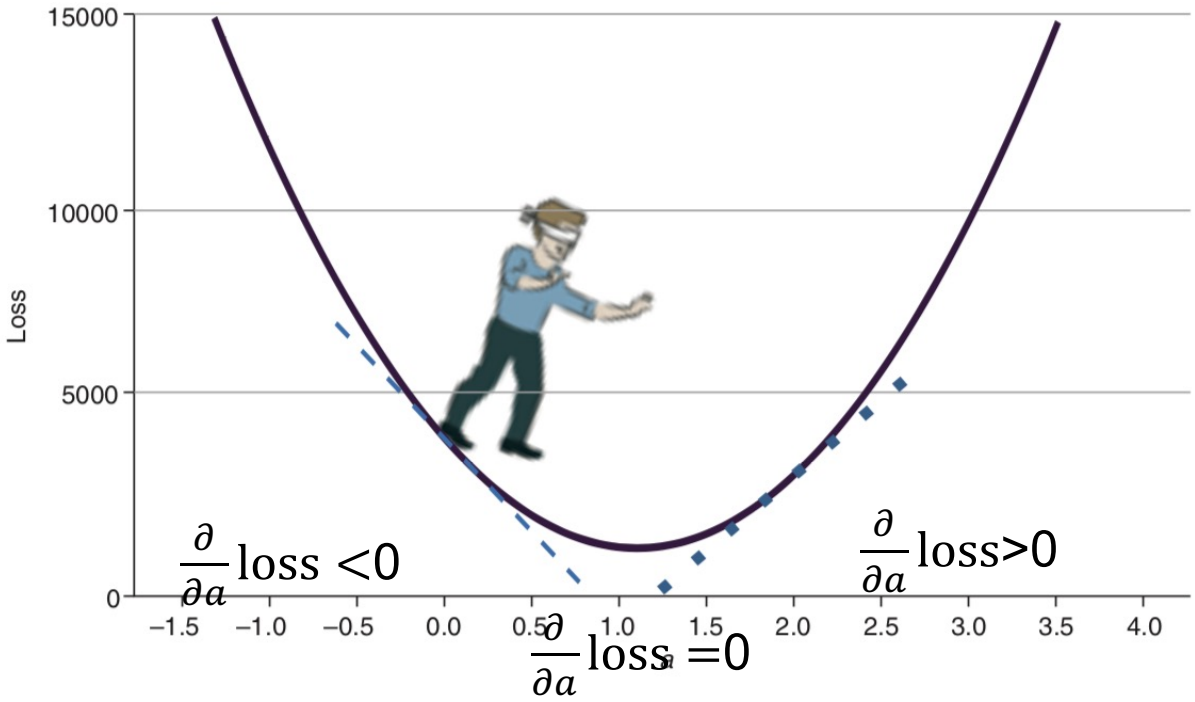
Have value

$\text{loss}(a)$

And gradient

$\frac{\partial \text{loss}(a)}{\partial a}$

Gradient Descent: update formula



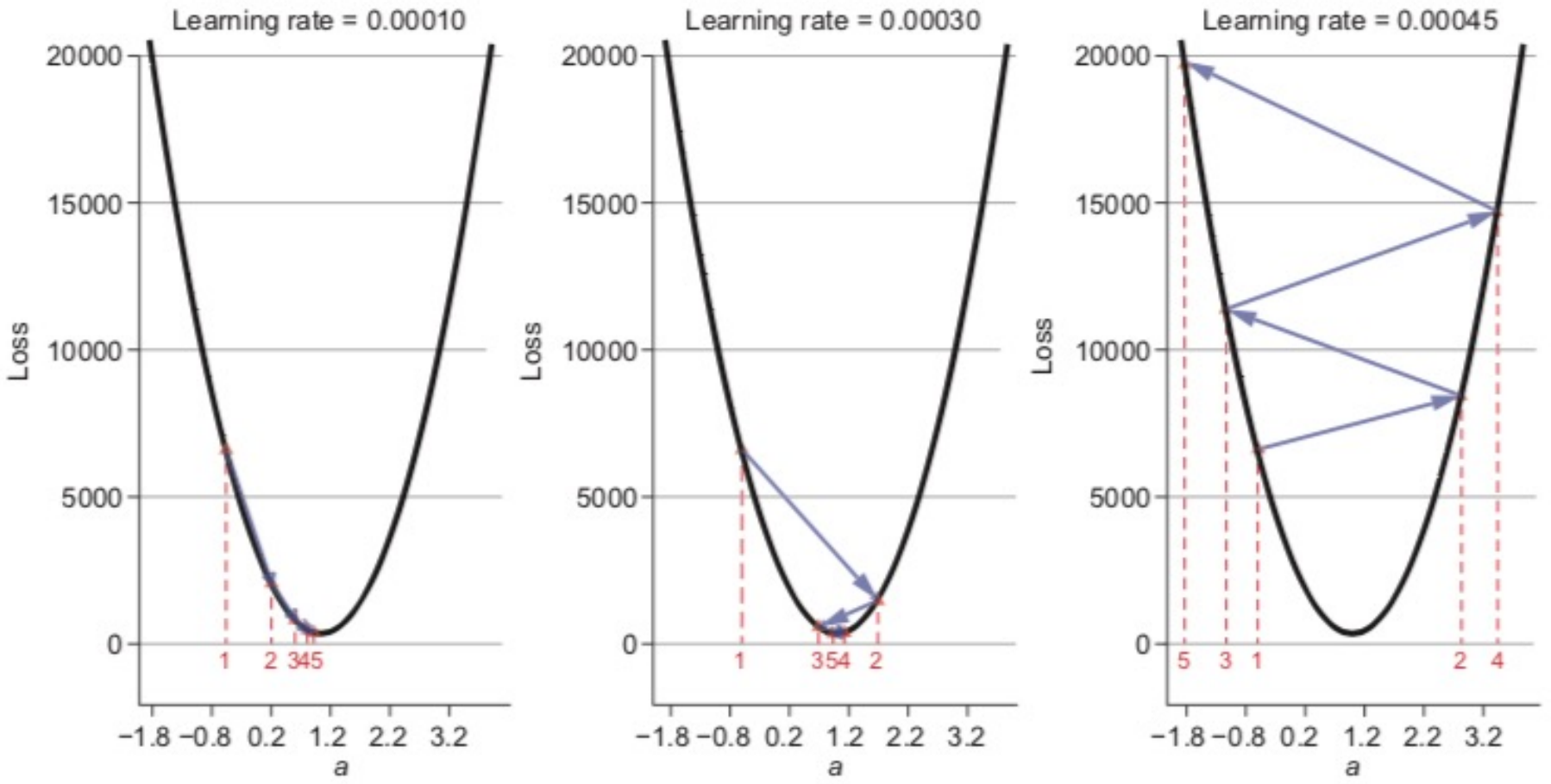
$\frac{\partial}{\partial a}\text{loss}$ is negative slope at *a*.

$$a_{t+1} = a_t - \eta \cdot \frac{\partial}{\partial a}\text{loss}$$

$$\text{loss} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - (a \cdot x_i + 87.6))^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - a \cdot x_i - 87.6)^2$$

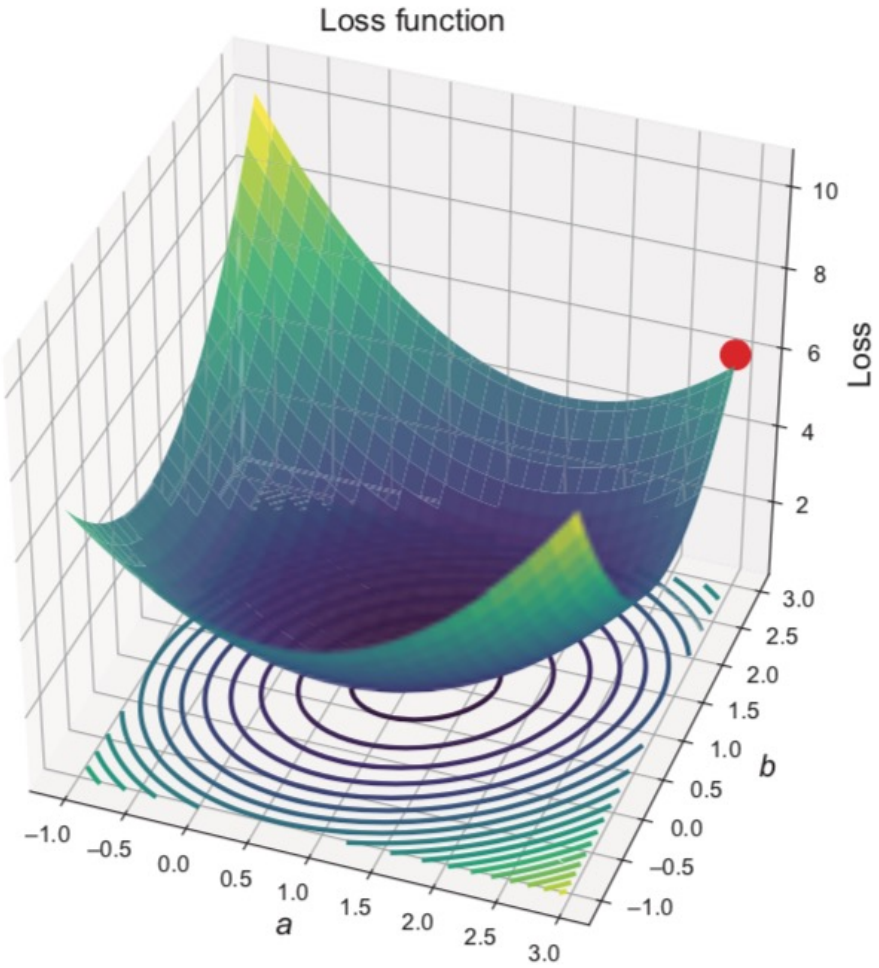
Blackboard $\frac{\partial loss}{\partial a}$

Effect of learning rate η

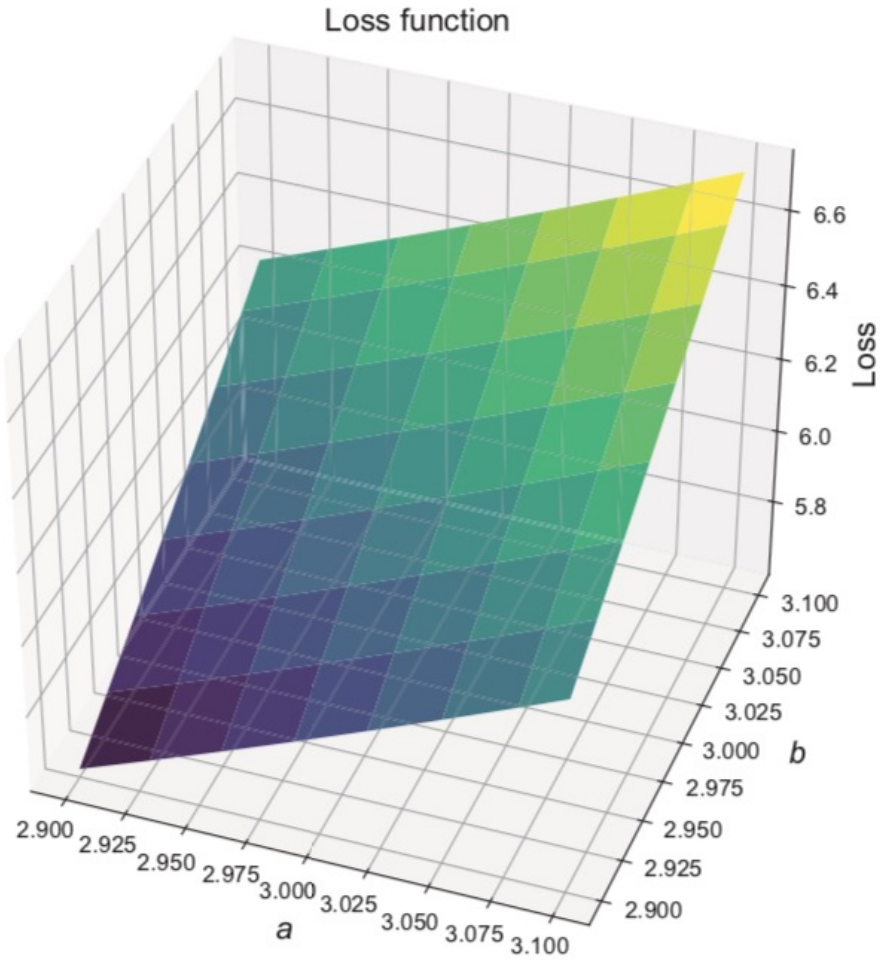


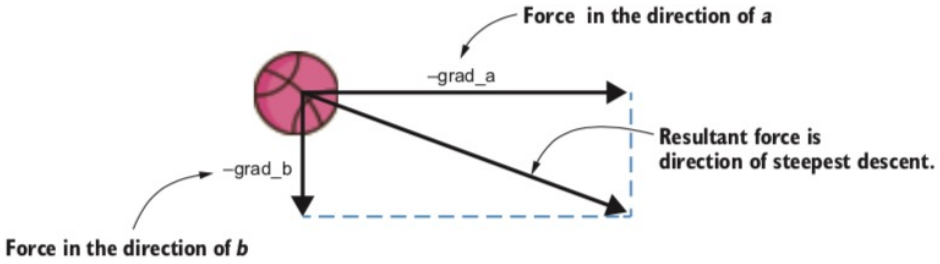
Learning rate is a *hyperparamer.* Later, we learn how to optimally choose the hyperparamerers in crossvalidation.

More than two parameters



Zoomed in

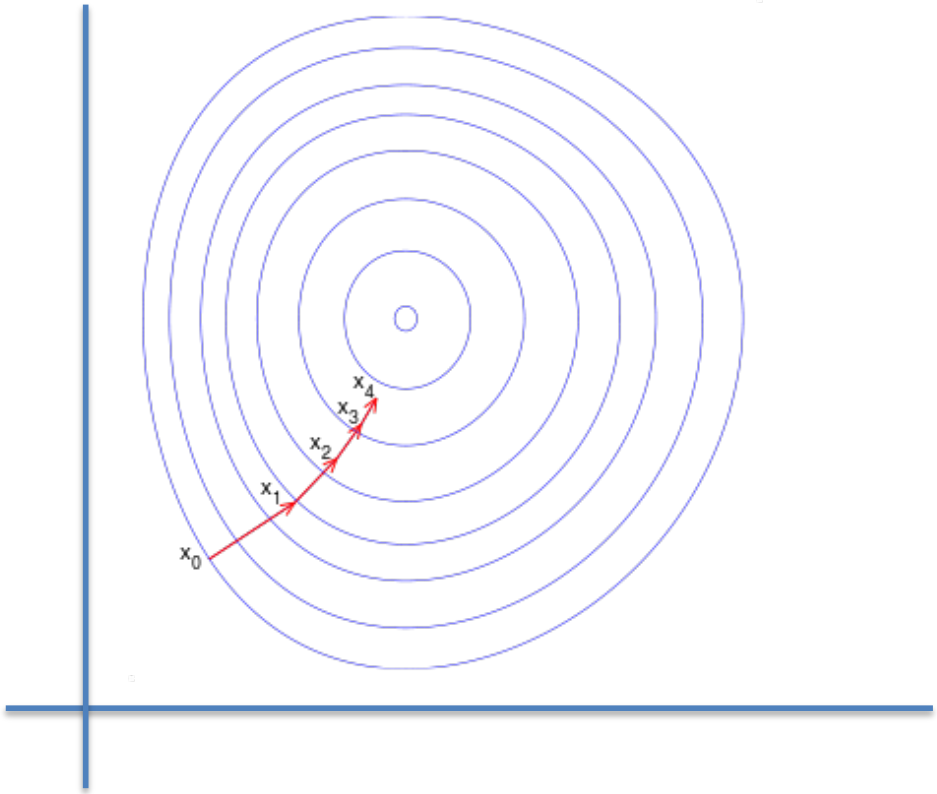


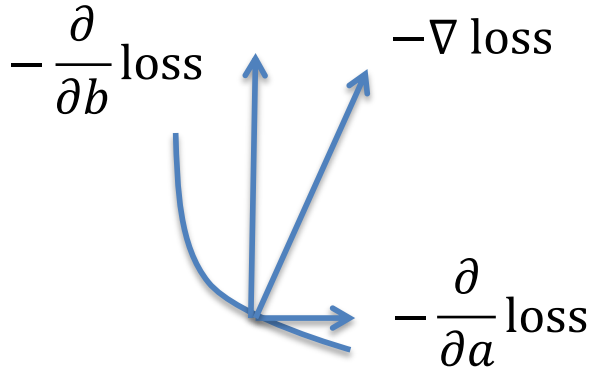


Optimization

- Gradient Descent

  Gradient is perpendicular to levels





$$a_{t+1} = a_t - \eta \frac{\partial}{\partial a}\text{loss}$$

$$b_{t+1} = b_t - \eta \frac{\partial}{\partial b}\text{loss}$$

$$\begin{pmatrix} a_{t+1} \\ b_{t+1} \end{pmatrix} = \begin{pmatrix} a_t \\ b_t \end{pmatrix} - \eta \begin{pmatrix} \frac{\partial}{\partial a} \\ \frac{\partial}{\partial b} \end{pmatrix} \text{loss}$$

In general

$$\overrightarrow{\theta_{t+1}} = \overrightarrow{\theta_t} - \eta \nabla \text{loss}$$

Slide credit: wikipedia

A potential pitfall

In linear regression, loss landscape is like a bowl.
For neural networks, it's not a bowl, but some magic happens, and Gradient Descent is OK.

Code Beispiel

Zur Vertiefung des Stoffs (1-D Regression) ist folgendes Notebook geeignet:

Github:
https://github.com/oduerr/ki/blob/main/linear_regression/lr_gradient_descent.ipynb

In Colab:
https://colab.research.google.com/github/oduerr/ki/blob/main/linear_regression/lr_gradient_descent.ipynb

Multiple Linear Regression

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

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

Multivariate Linear regression

- Simple model
  - $\hat{y} = f_\theta(x) = 1 \cdot w_o + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d = < w, x > + w_0$

- A note on notation
  - w_0 is often denoted with *b* the bias (extended data matrix)
  - Often the bias is included in $\vec{w} = (w_o, w_1, w_2, \cdots, w_d)$
    - $\hat{y} = < w, x > + w_0$
  - In some literature (Tibshirani) $\vec{w} = \vec{\beta}$
  - Often no vector symbols

Linear Regression as NN / Matrix Notation

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
...	...	...	...

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

y_hat
20.244783
12.426815
12.310875
17.457258
13.204685
...

- **Blackboard: Interpretation as Neural Network**

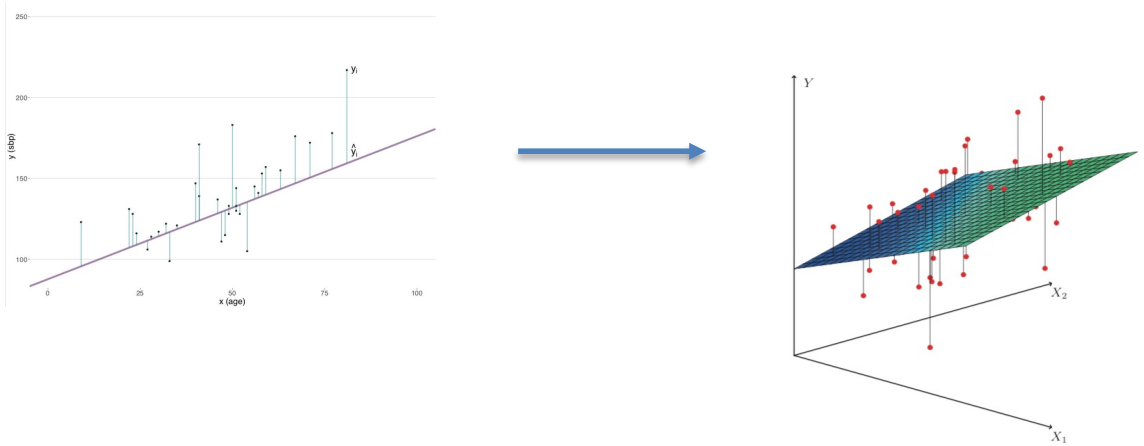
*We use w instead of β (β not common in NN)

Multiple Linear regression

- Simple model
  - $\hat{y} = \hat{f}_\theta(x) = 1 \cdot w_o + w_1 x_1 + w_2 x_2 + \cdots + w_p x_p = \boldsymbol{x}\,\boldsymbol{w^T} + w_0 = < \boldsymbol{w}, \boldsymbol{x} > + w_0$

- As a NN

- Geometric interpretation $\hat{f}_\theta(x)$ is a plane



Loss

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
20.244783
12.426815
12.310875
17.457258
13.204685
...
5.815722
8.534241
13.000720
23.386213
15.297458

Sales
22.1
10.4
9.3
18.5
12.9
...
7.6
9.7
12.8
25.5
13.4

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

loss = np.mean((y - y_hat)**2) #True sales is y

Loss depends now on 4 parameters w_0, w_1, w_2, w_3. Minimization with gradient descent, or analytical solution.