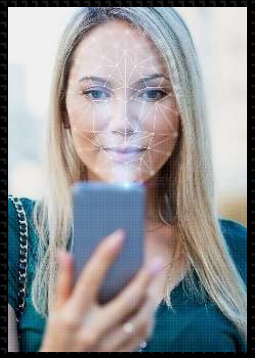
GFT

AI & Machine Learning mit 8. Klasse Mathematik

Javaland 16.03.2021, Tim Schade

Steckbrief

- Tim Schade
- Berliner im Rheinland
- Java Entwickler
- Software Architekt
- DevOps begeistert
- ML vernarrt

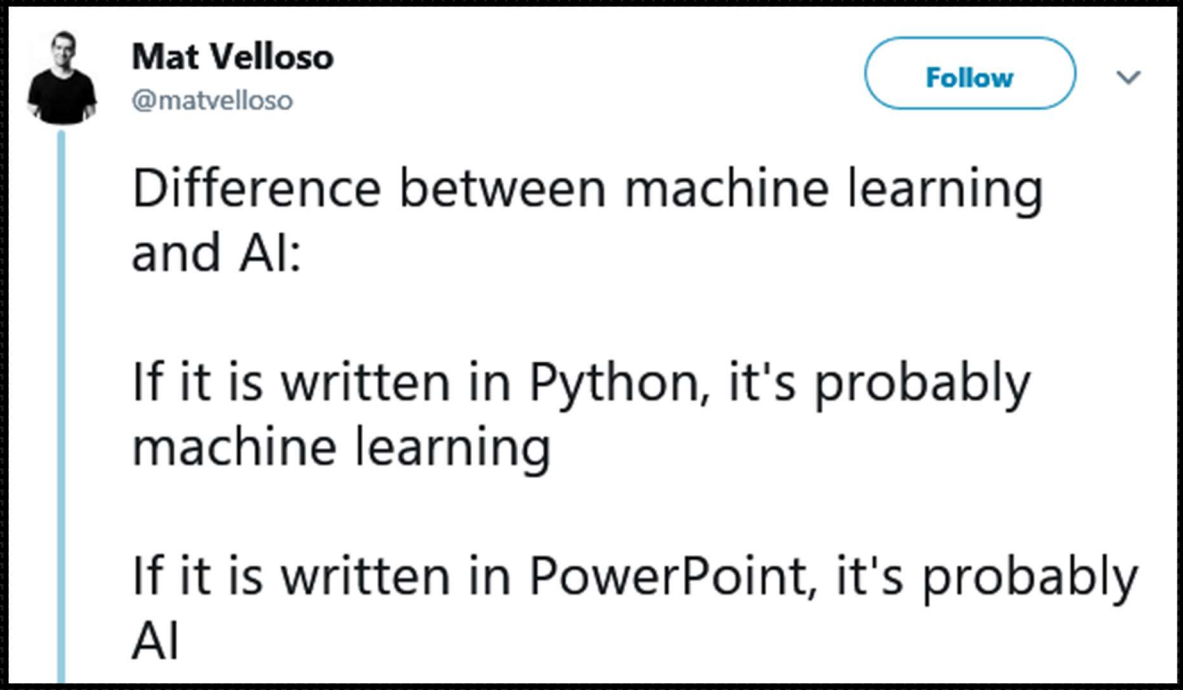


Trainierender Algorithmus
Neuronale Netze
Modelle

https://windowsunited.de/entwickler-bald-ueberfluessig-ki-programmiert-selbststaendig/

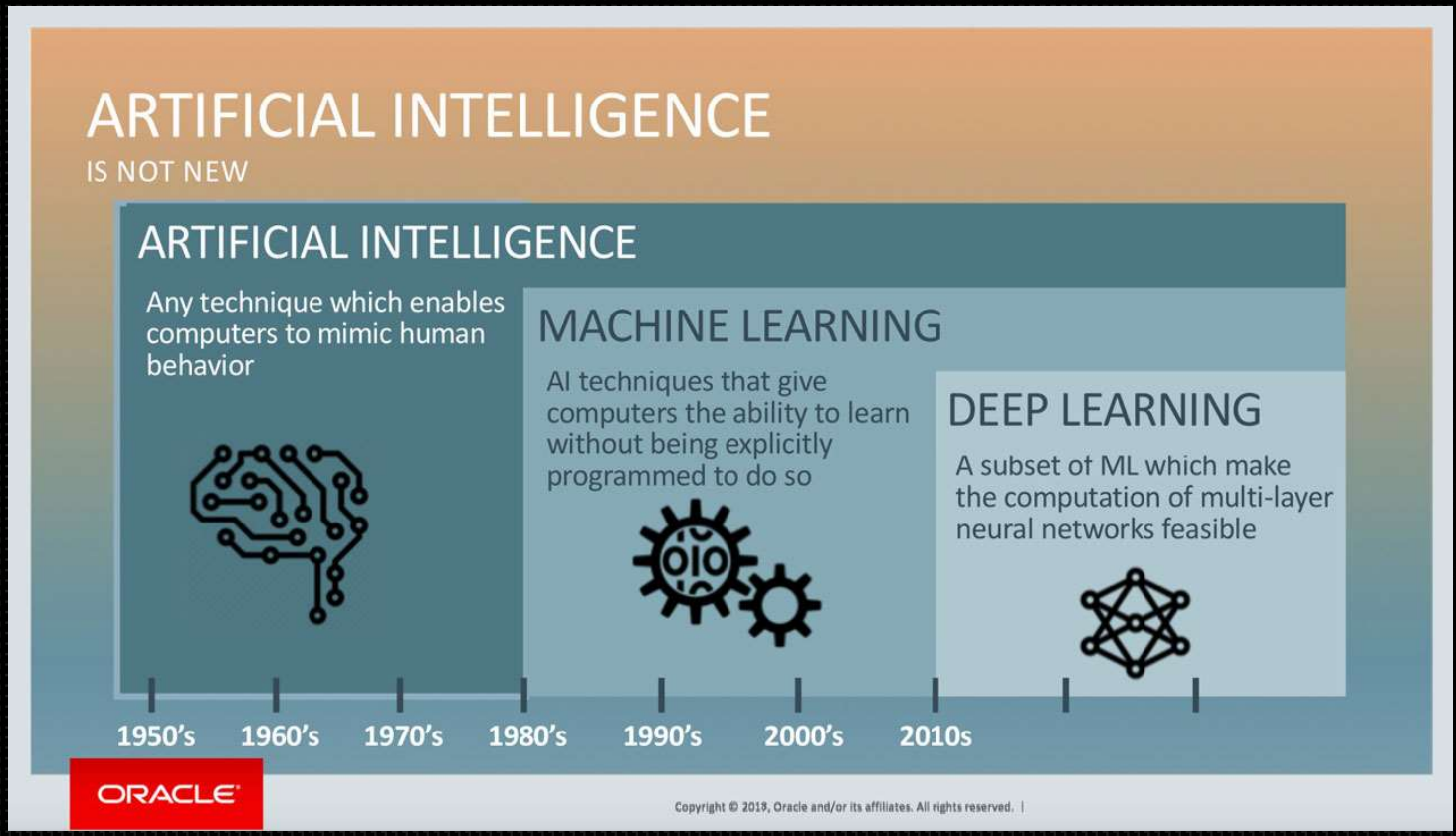
https://www.techbook.de/easylife/ai-pionier-warnt-killer-robotern

Was unterscheidet Artificial Intelligence & Machine Learning?



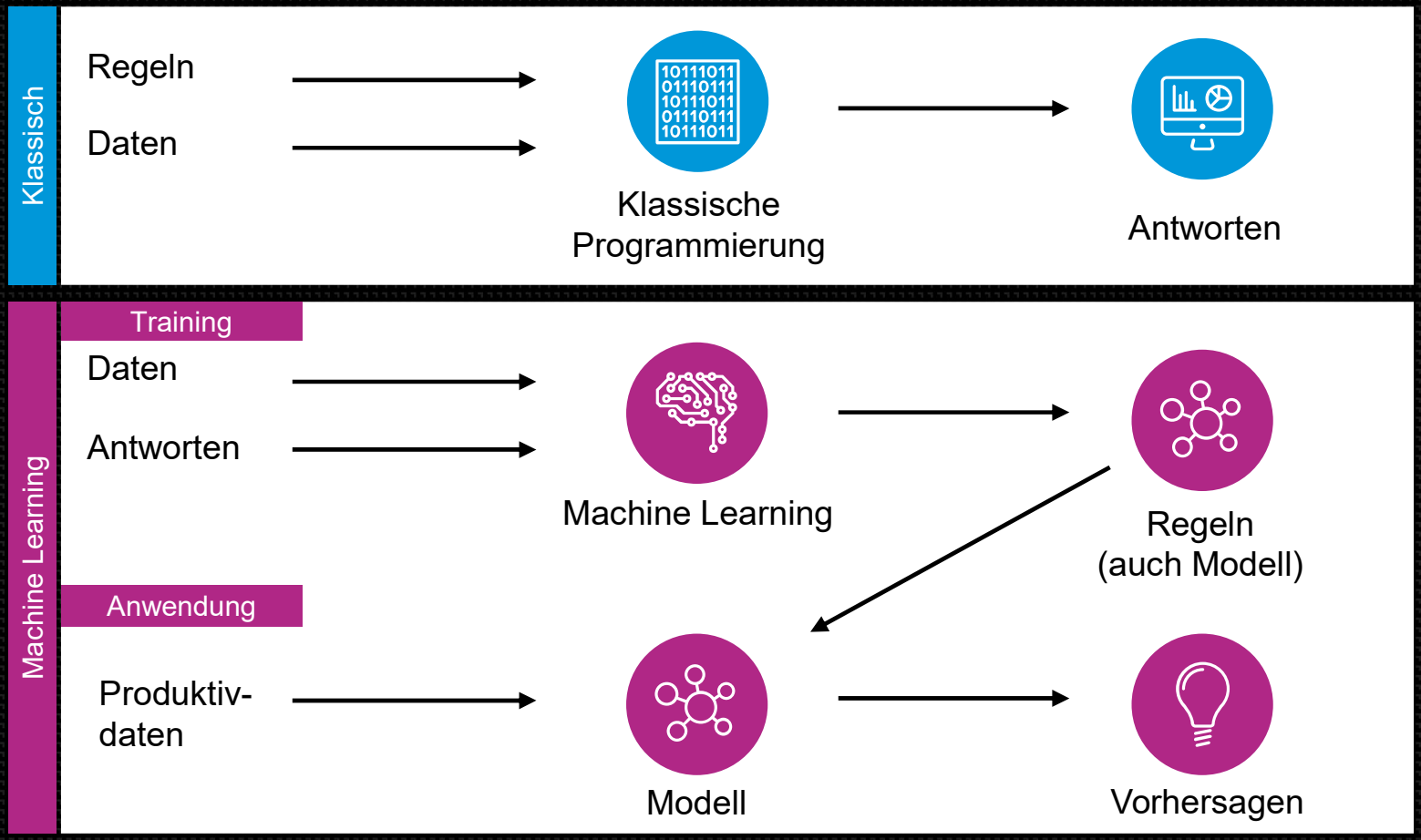
Artificial Intelligence AI =
Künstliche Intelligenz KI

Was unterscheidet Artificial Intelligence & Machine Learning?

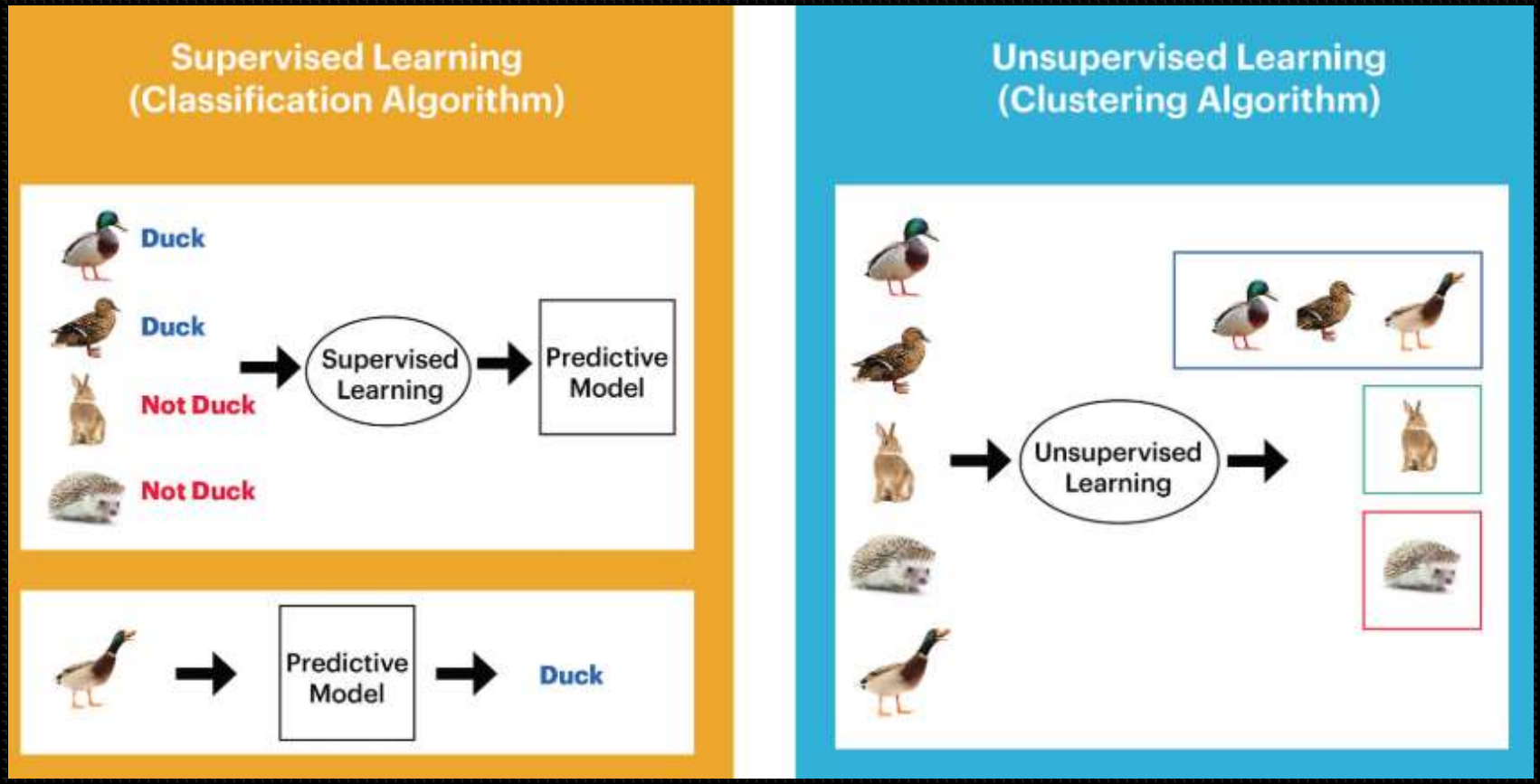


https://blogs.oracle.com/bigdata/difference-ai-machine-learning-deep-learning

Machine Learning vs. klassische Programmierung



Supervised & Unsupervised Learning



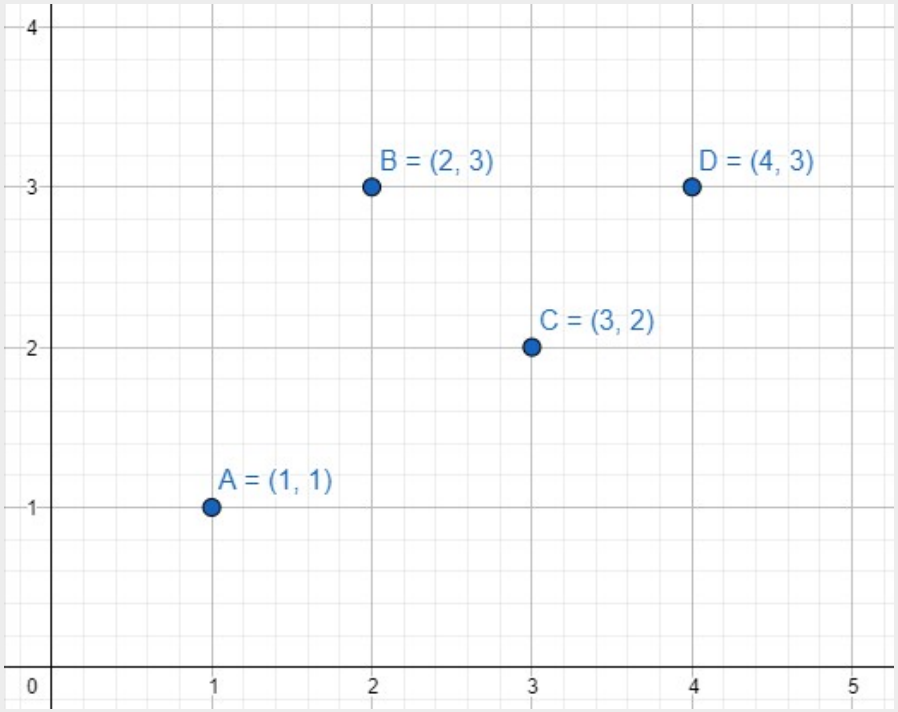
https://www.facebook.com/AIwithDrMalleswar/photos/a.1146377825535526/1171707946335847/?type=3&theater

Prediction mit Linearer Regression (Supervised Learning)

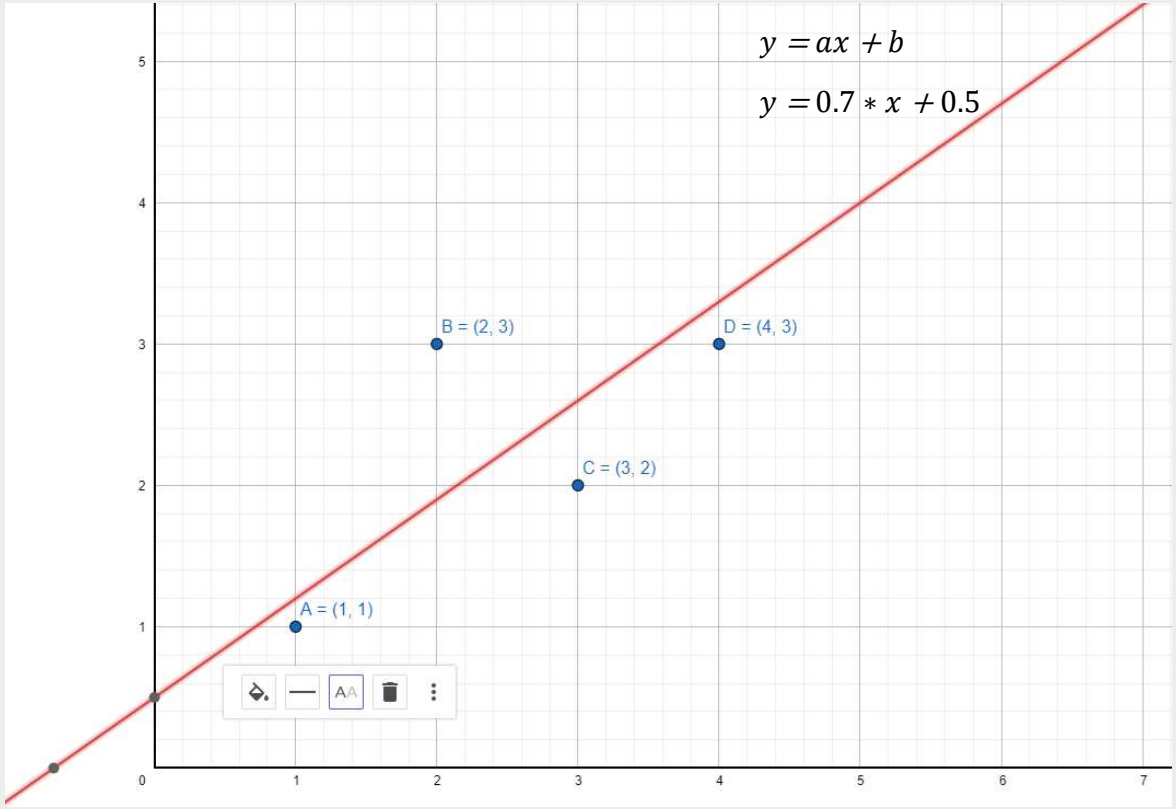
Wie viele Tassen Kaffee werden am Freitag getrunken? „Lagervorrat Vorhersage“

Supervised Learning benötigt Beispiel-Daten und vor allem dazugehörige -Antworten!

	Tag	Mo	Di	Mi	Do	Fr
x	**Anzahl MA**	1	2	3	4	5
y	**Tassen Kaffee**	1	3	2	3	?



Regressionsgerade mit Auge (1)



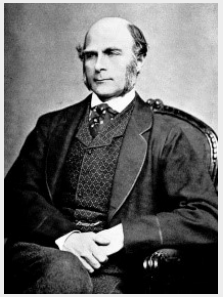
Warum?
Mit Hilfe der Geraden/der Funktion kann zu jedem Punkt x (Mitarbeiter) Die benötige Kaffeemenge vorhergesagt werden

Wie kann der Computer/ ein Algorithmus die
beste Gerade finden?

a=0.7 und b=0.5
Welche Werte für a und b sagen y am besten voraus?

ⓘ Mathem. Funktion = Modell

Regressionsgerade mit Galton (2)



Francis Galton
1822 - 1911

(8. Klasse)

$$y = a * x + b$$

$$a = \sum \frac{(x - \bar{x})(y - \bar{y})}{(x - \bar{x})^2}$$

$$b = \bar{y} - a\bar{x}$$

...

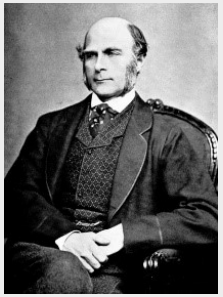
$$y = 0.5x + 1$$

$\bar{x} = 2.5$	x	1	2	3	4
$\bar{y} = 2{,}25$	y	1	3	2	3
	$x - \bar{x}$	-1.5	-0.5	0.5	1.5
	$y - \bar{y}$	-1,25	0.75	-0.25	0.75
Σ=2,5	$(x - \bar{x})\,(y - \bar{y})$	1.875	-0,375	-0.125	1.125
Σ=5	$(x - \bar{x})^2$	2.25	0.25	0.25	2.25

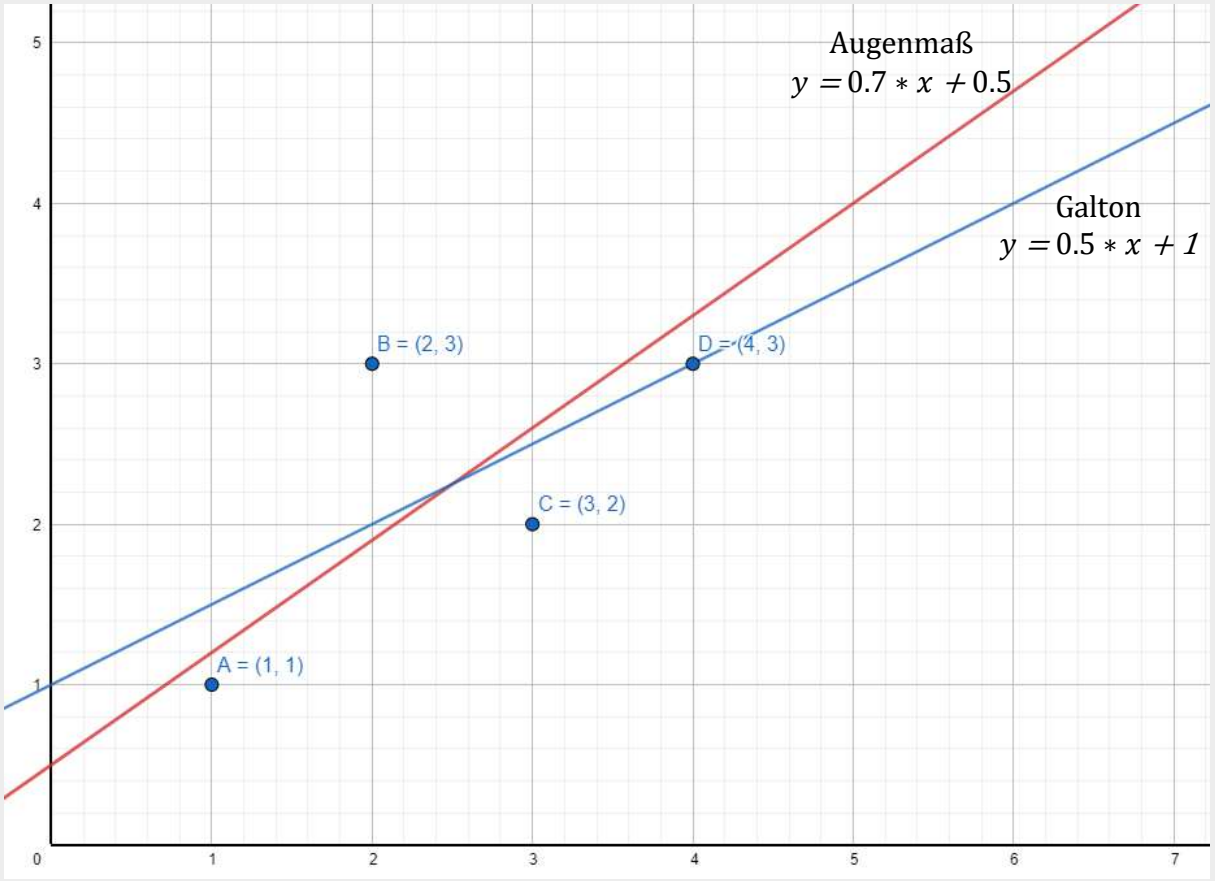
$$a = \frac{2{,}5}{5} = 0.5$$

$$b = \bar{y} - a * \bar{x}$$
$$b = 2.25 - 0.5 * 2.5 = 1$$

Regressionsgerade mit Galton (2)



Francis Galton
1822 - 1911



Regressionsgerade mit Fehlerfunktion und Ausprobieren(3)

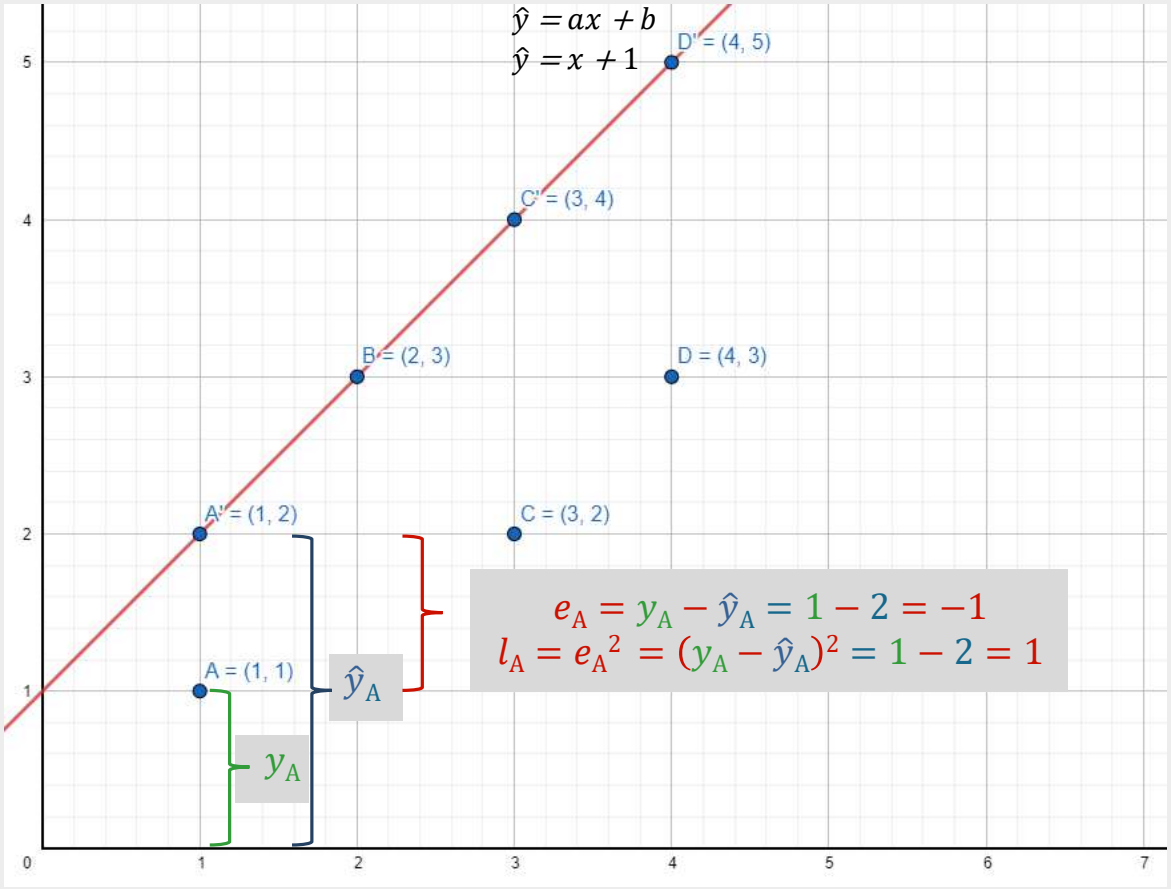
zufällige Geraden also
zufällige Parameter für a und b
z.B.: a =1 und b =1

-> Vergleich der berechneten Werte $\hat{y}$ mit den tatsächlichen Werten y

-> Loss Function zur Bestimmung der Geradenqualität (error)

- $l_{ges} = \frac{1}{n}(l_1 + l_2 + \ldots + l_n)^2$
- $l_{ges} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- $l_{ges} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (ax_i + b))^2$

ⓘ Mean Squared Error



Regressionsgerade mit Fehlerfunktion und Ausprobieren(3)

- $l_{ges}\ (a,b) = \frac{1}{n}(l_1 + l_2 + \ldots + l_n)$
- $l_{ges}(a,b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- $l_{ges}(a,b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (ax_i + b))^2$
- $l_{ges}(a=1, b=1) = 9/4 = 2{,}25$

- $l_{ges} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (ax_i + b))^2 \rightarrow min!$

Folgende Idee:

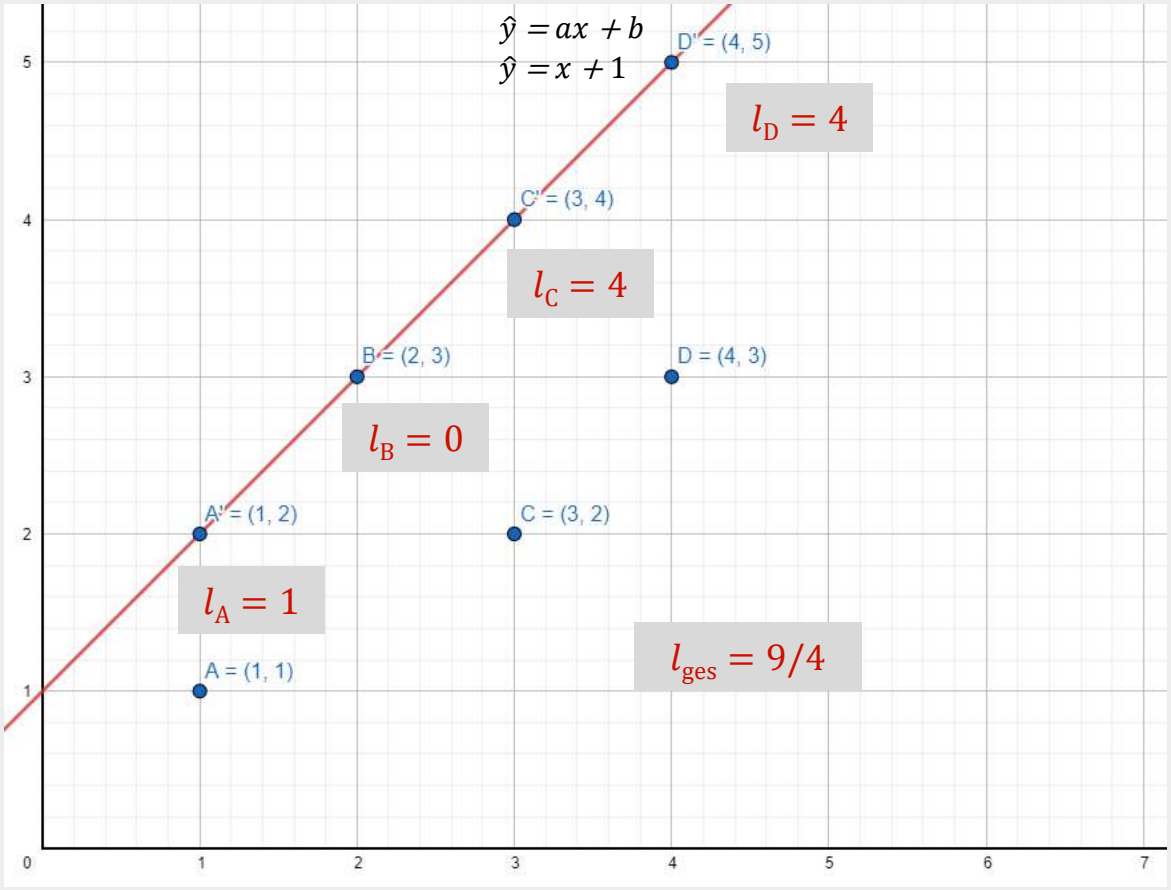
```
forSchleife (Anzahl-Epochen){
  Zufallszahlen wählen für a und b
  Loss Function Berechnen und wenn kleinster Wert,
„merken“
}
```

„Gemerkte“ Werte verwenden für lineare Funktion

$\hat{y} = ax + b$

Evtl. kommt man nahe an das Optimum heran (evtl. auch nicht)

ⓘ Ausprobieren = Training
Anzahl Versuche = Epoch



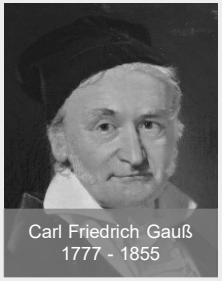
Regressionsgerade mit Fehlerfunktion und Mathematik(4)

Isaac Newton
1642 - 1726

- **Newtonverfahren**

Brook Taylor
1685 - 1731

- **Taylorreihe**



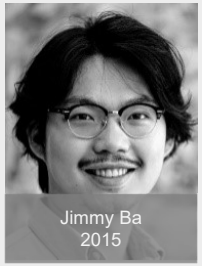
Carl Friedrich Gauß
1777 - 1855

- **Gauß-Newton-Verfahren**

Diverse Andere

- **Gradientenverfahren (gradient descent)**
- **Stochastic gradient descent**
- **Momentum**
- **Adagrad**
- **Nesterov accelerated gradient**
- **…**

Durk Kingma
2015



Jimmy Ba
2015

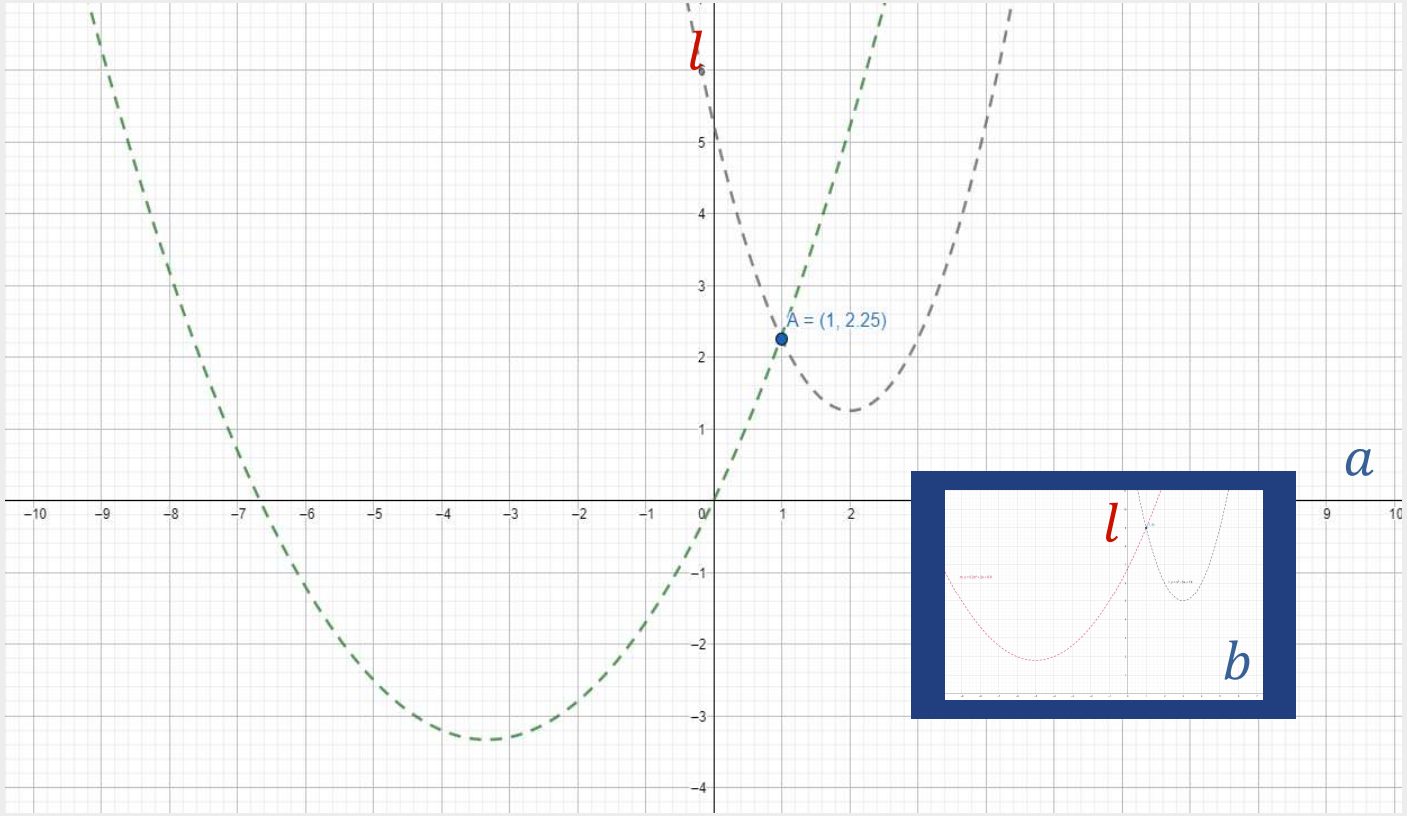
- **Adam**

(derived from adaptive moment estimation)

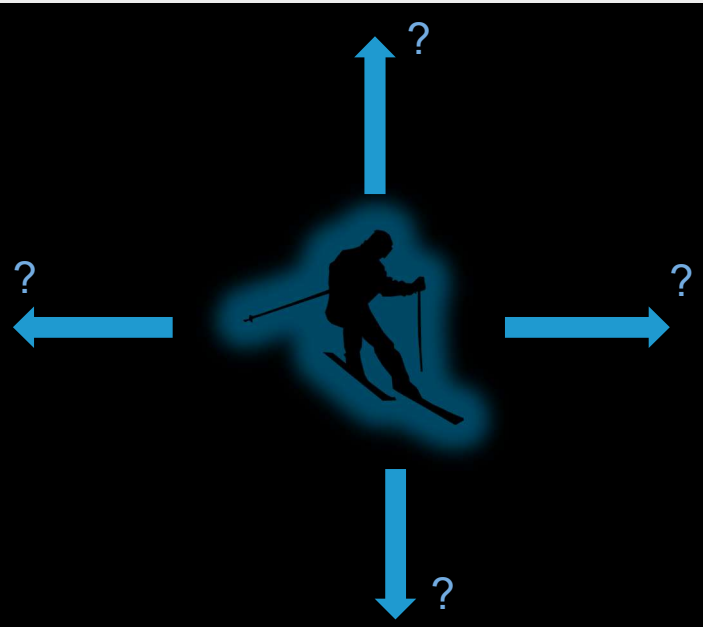
Überblick über „State of the Art“ Algorithmen:
https://ruder.io/optimizing-gradient-descent/index.html

Regressionsgerade mit Gradientenverfahren (Gradient Descent) (5)

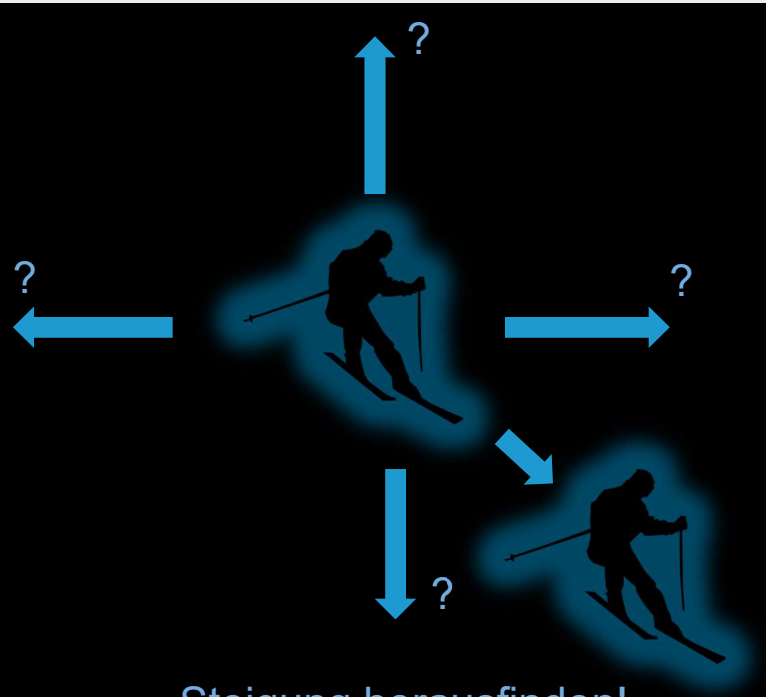
- $l_{ges} = \frac{1}{n} \sum_{i=1}^{n} (y_i - (ax_i + b))^2$
- $l_{ges}(a = 1; b = 1) = 2{,}25$
- $l_{ges} = \frac{1}{n} \sum_{i=1}^{n} (y_i - (ax_i + b))^2 \rightarrow min!$



Schneesturm! Was nun?

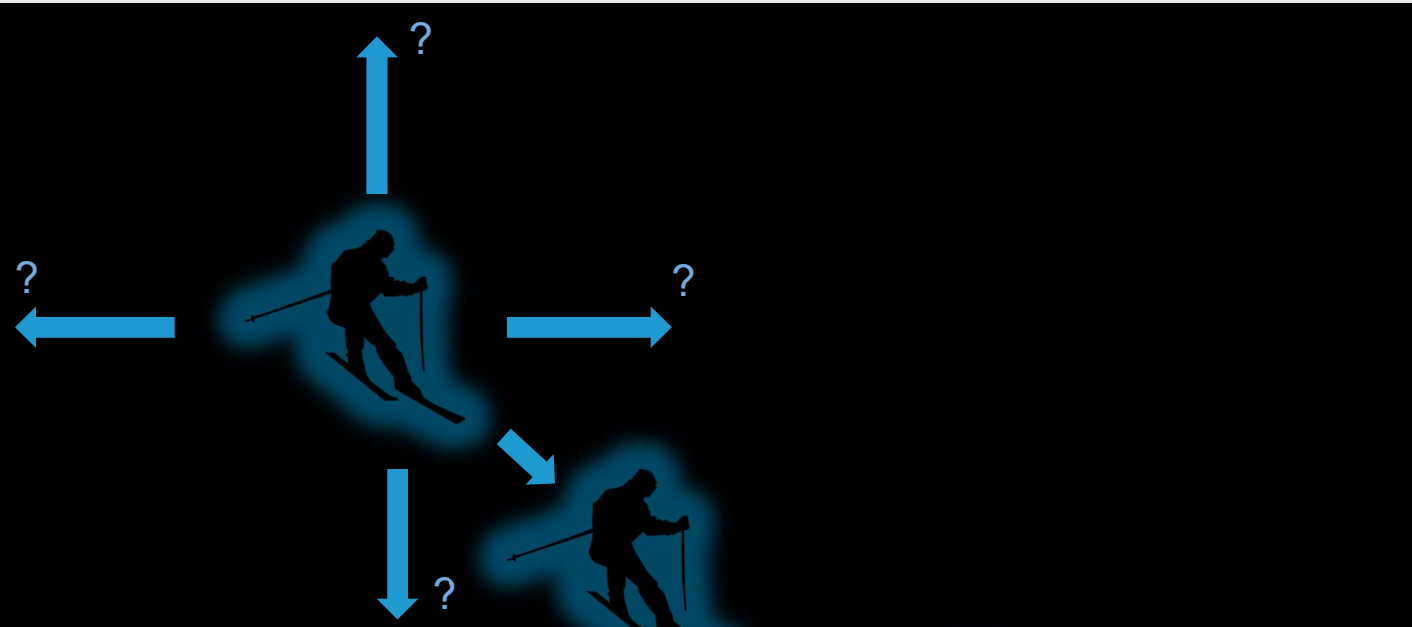


Schneesturm! Was nun?



Steigung herausfinden!

Schneesturm! Was nun?



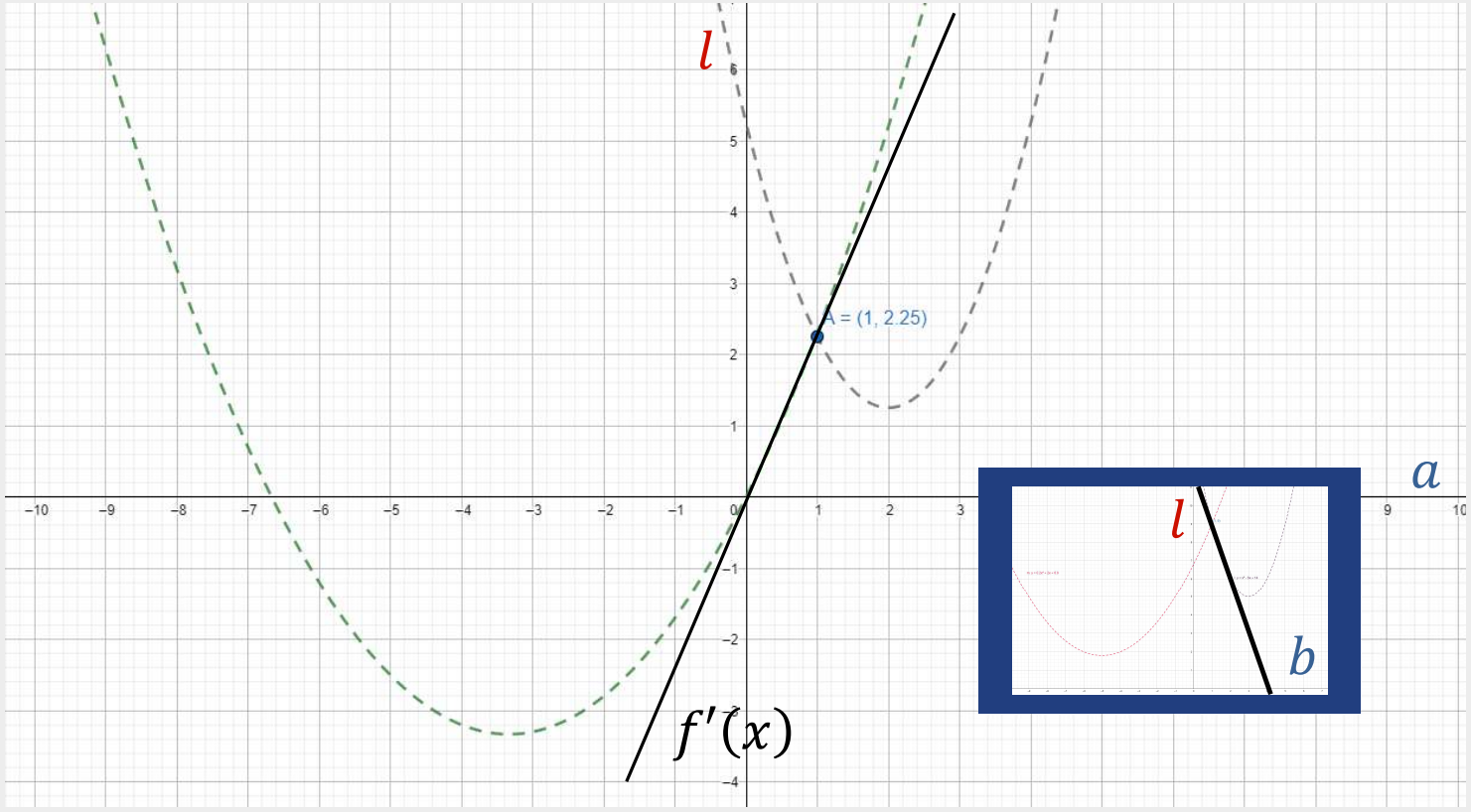
Steigung herausfinden!

Regressionsgerade mit Gradientenverfahren (Gradient Descent) (5)

- Steigung herausfinden!

Differenzialrechnung 11 Klasse:

- Steigung am Punkt (x|y) bestimmbar durch Ableitungsfunktion von $y = f(x)$ auch genannt $y' = f'(x)$



Exkurs: Ableitungen & Differenzenquotienten

Regel	Funktion	1. Ableitung	Beispiel
Potenzregel	$f(a) = a^n$	$\frac{\partial}{\partial a} = f'(a) = n * a^{n-1}$	$f(a) = 2 * a^8$ $f'(a) = 16 * a^7$
Konstantenregel	$f(a) = c$	$\frac{\partial}{\partial a} = f'(a) = 0$	$f(a) = 20$ $f'(a) = 0$
Summen-/Differenzregel	$f(a) = a^n + a^m$	$\frac{\partial}{\partial a} = f'(a) = n * a^{n-1} + m * a^{m-1}$	$f(a) = 2 * a^2 + 3 * a$ $f'(a) = 4 * a^1 + 3$
Kettenregel	$f(a) = u(v(a))$	$\frac{\partial}{\partial a} = f'(a) = u'(v(a)) * v'(a)$	$f(a) = 2 * (2a - 5)^3$ $f'(a) = 6 * (2a - 5)^2 * (2)$
Partielle Ableitungen	$f(a,b) = 3a^2 + 4ab - 2b^2$	Quotienten nach denen nicht abgleitet wird, werden wie Konstanten behandelt (z.B. 1)->(Konstanten Regel)	▪ $f'_a(a,b) = \frac{\partial}{\partial a} = 6a + 4b$ ▪ $f'_b(a,b) = \frac{\partial}{\partial b} = 4a - 4b$

Exkurs: Ableitungen & Differenzenquotienten

Regel	Funktion	1. Ableitung	Beispiel
Potenzregel	$f(a) = a^n$	$\frac{\partial}{\partial a} = f'(a) = n * a^{n-1}$	$f(a) = 2 * a^8$ $f'(a) = 16 * a^7$
Konstantenregel	$f(a) = c$	$\frac{\partial}{\partial a} = f'(a) = 0$	$f(a) = 20$ $f'(a) = 0$
Summen-/Differenzregel	$f(a) = a^n + a^m$	$\frac{\partial}{\partial a} = f'(a) = n * a^{n-1} + m * a^{m-1}$	$f(a) = 2 * a^2 + 3 * a$ $f'(a) = 4 * a^1 + 3$
Kettenregel	$f(a) = u(v(a))$	$\frac{\partial}{\partial a} = f'(a) = u'(v(a)) * v'(a)$	$f(a) = 2 * (2a - 5)^3$ $f'(a) = 6 * (2a - 5)^2 * (2)$
Partielle Ableitungen	$f(a,b) = 3a^2 + 4ab - 2b^2$	Quotienten nach denen nicht abgeleitet wird, werden wie Konstanten behandelt (z.B. 1)->(Konstanten Regel)	▪ $f'_a(a,b) = \frac{\partial}{\partial a} = 6a + 4b$ ▪ $f'_b(a,b) = \frac{\partial}{\partial b} = 4a - 4b$

Partielle Ableitungen

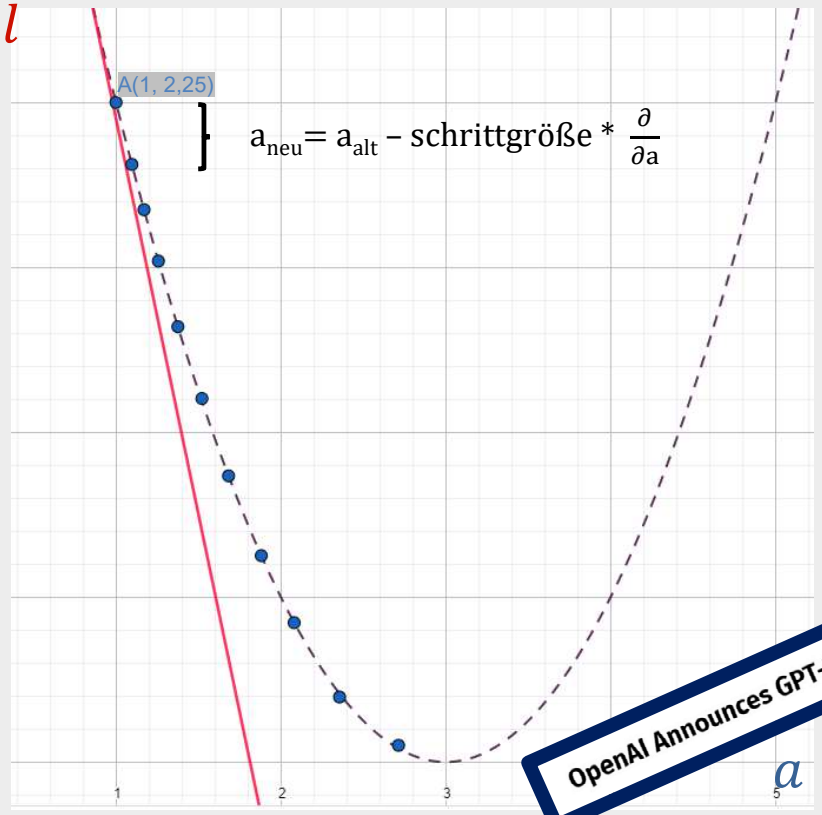
- $l_{ges} = \frac{1}{n} \sum_{i=1}^{n} (y_i - (\overbrace{a}x_i + \overbrace{b}))^2$

Kettenregel

- $\frac{\partial}{\partial a} = \frac{1}{n} \sum_{i=1}^{n} \overbrace{2 * (y_i - (ax_i + b))}^{u'(v(a))} * \overbrace{(-x_i)}^{v'(a)}$
- $\frac{\partial}{\partial b} = \frac{1}{n} \sum_{i=1}^{n} 2 * (y_i - (axi + b)) * (-1)$

Regressionsgerade mit Gradientenverfahren (Gradient Descent) (5)

- $\frac{\partial}{\partial a} = \frac{1}{n} \sum_{i=1}^{n} 2 * (y_i - (ax_i + b)) * (-x_i)$
- $\frac{\partial}{\partial b} = \frac{1}{n} \sum_{i=1}^{n} 2 * (y_i - (axi + b)) * (-1)$



Schrittgröße = Learning Rate

OpenAI Announces GPT-3 AI Language Model with 175 Billion Parameters

Let‘s see some Code

https://github.com/schaDev/javaland-ml8

```java
public static void main(String[] args) {
    int[] xData = {1, 2, 3, 4};
    int[] yData = {1, 3, 2, 3};
    gradientDescent(xData, yData);
}

private static void gradientDescent(int[] xData, int[] yData) {
    List<Dto> data = IntStream.range(0, xData.length).boxed().map(i -> new Dto(xData[i], yData[i])).collect(Collectors.toList());
    double a = 1;
    double b = 1;
    long epochs = 1000;
    double learningRate = 0.01;
    long n = data.size();

    for (int epoch = 1; epoch < epochs; epoch++) {
        final Dto p = new Dto(a, b);
        ToDoubleFunction<Double> yPredicted = x -> p.a * x + p.b;

        double cost = (1d / n) * data.stream().mapToDouble(d -> d.y - Math.pow(d.y - yPredicted.applyAsDouble(d.x), 2)).sum();

        double da = (1d / n) * data.stream().mapToDouble(d -> 2 * (d.y - yPredicted.applyAsDouble(d.x)) * -d.x).sum();
        double db = (1d / n) * data.stream().mapToDouble(d -> 2 * (d.y - yPredicted.applyAsDouble(d.x)) * -1).sum();

        a = a - learningRate * da;
        b = b - learningRate * db;

        System.out.format("a %f, b %f cost %f epoch %d%n", a, b, cost, epoch);
    }
}
```

	Tag	Mo	Di	Mi	Do	Fr
x	Anzahl MA	1	2	3	4	5
y	Tassen Kaffee	1	3	2	3	?

zufällige Parameter für a und b
z.B.: a =1 und b =1

ⓘ Schrittgröße = learning rate

ⓘ Ausprobieren = Training
Anzahl Versuche = Epoch

$$\hat{y} = ax + b$$

ⓘ Mean Squared Error

$$l_{ges} = \frac{1}{n}\sum_{i=1}^{n}(y_i - (ax_i + b))^2$$

$$\frac{\partial}{\partial a} = \frac{1}{n}\sum_{i=1}^{n} 2 * (y_i - (ax_i + b)) * (-x_i)$$

$$\frac{\partial}{\partial b} = \frac{1}{n}\sum_{i=1}^{n} 2 * (y_i - (ax_i + b)) * (-1)$$

$$a_{neu} = a_{alt} - \text{schrittgröße} * \frac{\partial}{\partial a}$$

```python
def gradientDescent(x, y):
    a = 1
    b = 1
    epochs = 1000
    learning_rate = 0.01
    n = len(x)

    for epoch in range(epochs):
        yPredicted = a * x + b

        da = (1 / n) * sum(2 * (y - yPredicted) * -x)
        db = (1 / n) * sum(2 * (y - yPredicted) * -1)

        a = a - learning_rate * da
        b = b - learning_rate * db

        cost = (1 / n) * sum([val ** 2 for val in (y - yPredicted)])

        print("a {} b {} cost {} epoch {}".format(a, b, cost, epoch))


# main application
xData = np.array([1, 2, 3, 4])
yData = np.array([1, 3, 2, 3])

gradientDescent(xData, yData)
```

zufällige Parameter für a und b
z.B.: a =1 und b =1

ⓘ Ausprobieren = Training
Anzahl Versuche = Epoch

ⓘ Schrittgröße = learning rate

$$\hat{y} = ax + b$$

$$\frac{\partial}{\partial a} = \frac{1}{n} \sum_{i=1}^{n} 2 * (y_i - (ax_i + b)) * (-x_i)$$

$$\frac{\partial}{\partial b} = \frac{1}{n} \sum_{i=1}^{n} 2 * (y_i - (ax_i + b)) * (-1)$$

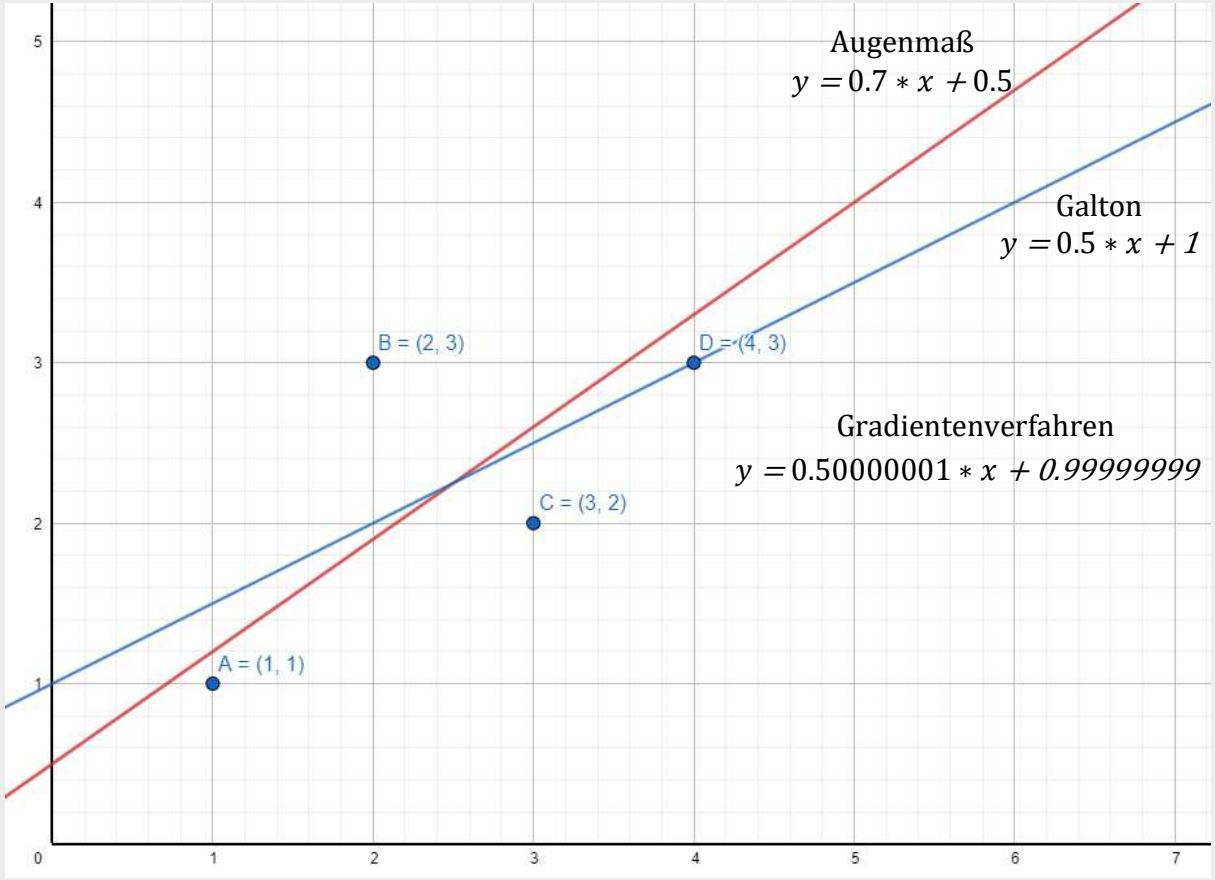
$$a_{neu} = a_{alt} - \text{schrittgröße} * \frac{\partial}{\partial a}$$

$$l_{ges} = \frac{1}{n} \sum_{i=1}^{n} (y_i - (ax_i + b))^2$$

ⓘ Mean Squared Error

	Tag	Mo	Di	Mi	Do	Fr
x	Anzahl MA	1	2	3	4	5
y	Tassen Kaffee	1	3	2	3	?

Regressionsgeraden zur Vorhersage



Kaffeverbrauch bei
5 Mitarbeitern?

$y = 0.5 * x + 1$
$y = 0.5 * 5 + 1$
$y = 3{,}5$

= 3,5 Tassen

Mathem. Funktion = Modell
Ausprobieren = Training
Anzahl Versuche = Epoch
Error = Loss Funktion
(Mean Squared Error)
Schrittgröße = Learning Rate

Danke! Fragen?

GFT Deutschland
Tim Schade
IT Architect

Joseph-Schumpeter-Allee 1
53227 Bonn

T +49 228 2071-3225

Tim.Schade@gft.com

GFT

Shaping the future of digital business

GFT Deutschland
Tim Schade
IT Architect

Joseph-Schumpeter-Allee 1
53227 Bonn

T +49 228 2071-3225

Tim.Schade@gft.com