Praktisches Maschinelles Lernen am Beispiel des Gradientenverfahrens

Billy Joe Franks

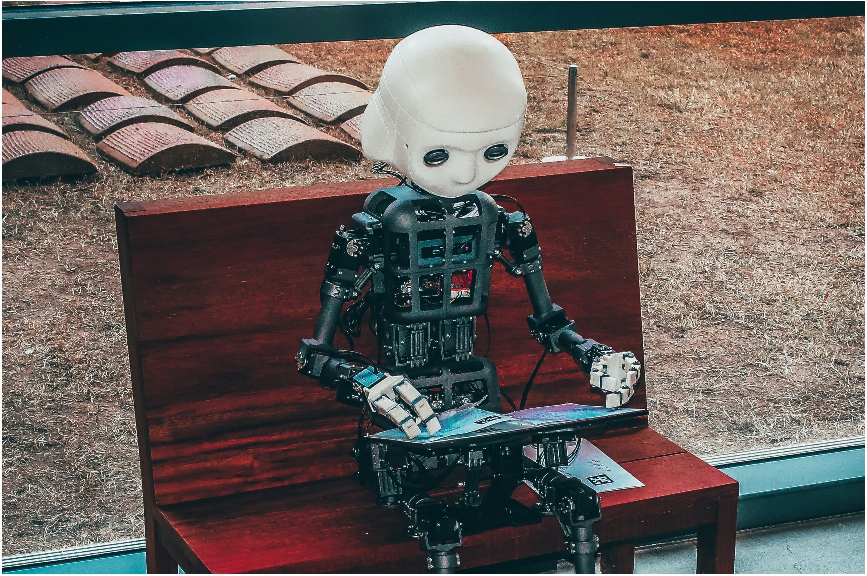
Neustadt, 07.06.2024

Über mich.

Karriere

- Bachelor Informatik (Schwerpunkt Maschinelles Lernen)
- Master Informatik (Schwerpunkt Maschinelles Lernen)
- PhD Student (Schwerpunkt Maschinelles Lernen)

Machinelles **Lernen**



Scikit Learn

scikit learn Install User Guide API Examples Community More

scikit-learn...

Section Navigation

scikit-learn Tutorials

Machine Learning

Linear Soft-Margin Support Vector Machine

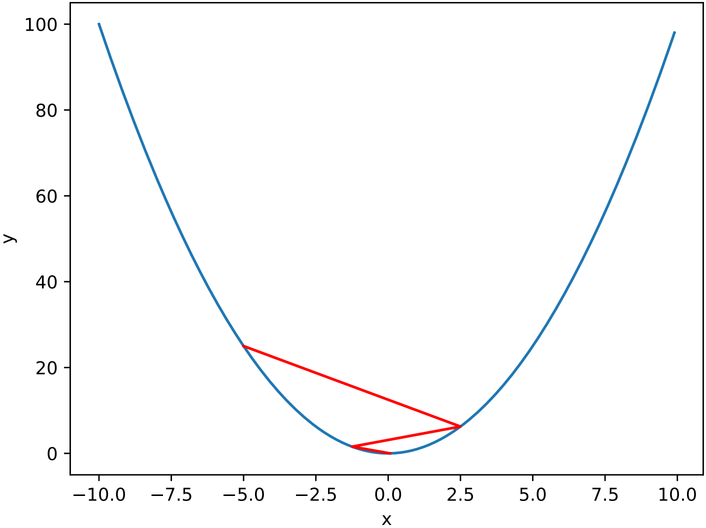
$$\max_{\gamma, b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d \setminus \{0\}, \xi_1, \ldots, \xi_n \geq 0} \quad \gamma - C \sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \qquad y_i \frac{(\mathbf{w}^\top \mathbf{x}_i + b)}{\|\mathbf{w}\|} \geq \gamma - \xi_i, \quad \forall i = 1, \ldots, n$$

Kernel Soft-Margin Support Vector Machine

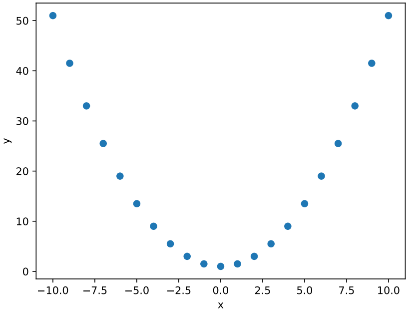
$$\min_{b \in \mathbb{R}, \alpha \in \mathbb{R}^n} \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j k(x_i, x_j) + C \sum_{i=1}^{n} \max(0, 1 - y_i(\sum_{j=1}^{n} \alpha_j k(x_i, x_j) + b))$$

Gradientenverfahren

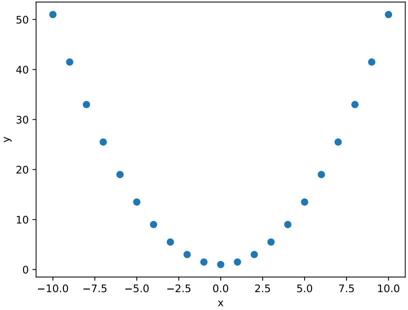


Fehlerminimierung

Fehlerminimierung



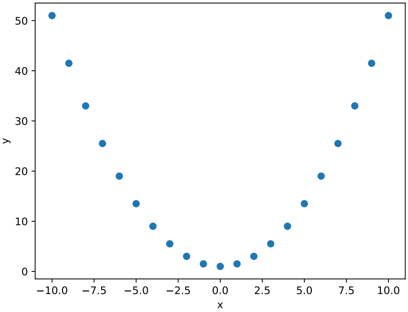
Fehlerminimierung



Methode der kleinsten Quadrate (Least Squares)

Angenommen $D := \{(x_1, y_1), \ldots, (x_n, y_n)\}$ mit $\forall i \in [n] : x_i, y_i \in \mathbb{R}$ und $g_\theta : \mathbb{R} \to \mathbb{R}$ mit Parametern θ, $g_\theta(x) := \theta_1 x^2 + \theta_2$.

Fehlerminimierung

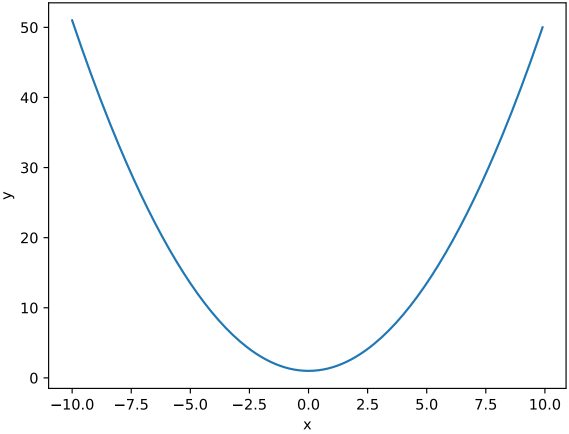


Methode der kleinsten Quadrate (Least Squares)

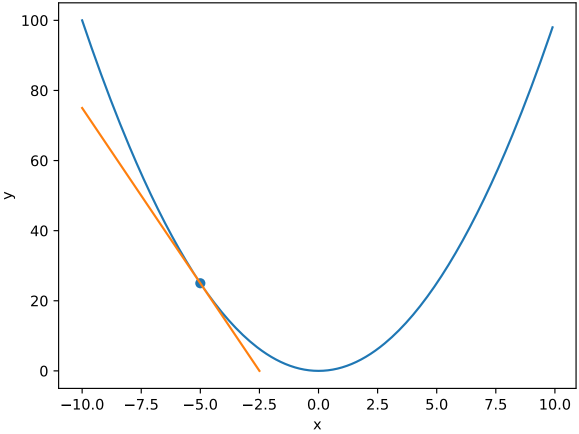
Angenommen $D := \{(x_1, y_1), \ldots, (x_n, y_n)\}$ mit $\forall i \in [n] : x_i, y_i \in \mathbb{R}$ und $g_\theta : \mathbb{R} \to \mathbb{R}$ mit Parametern θ, $g_\theta(x) := \theta_1 x^2 + \theta_2$.

$$\theta^* := \arg\min_\theta \sum_{(x,y)\in D} (g_\theta(x) - y)^2$$

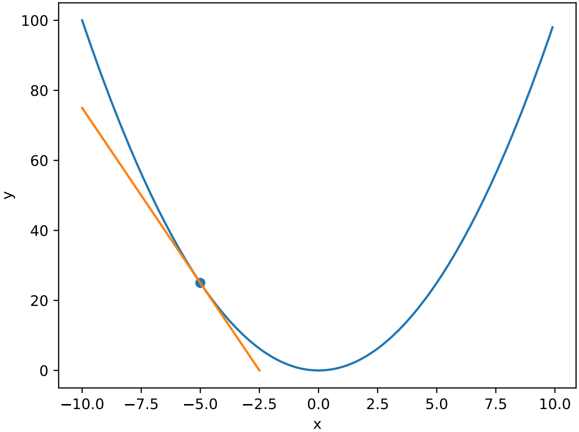
Gradientenverfahren



Gradientenverfahren

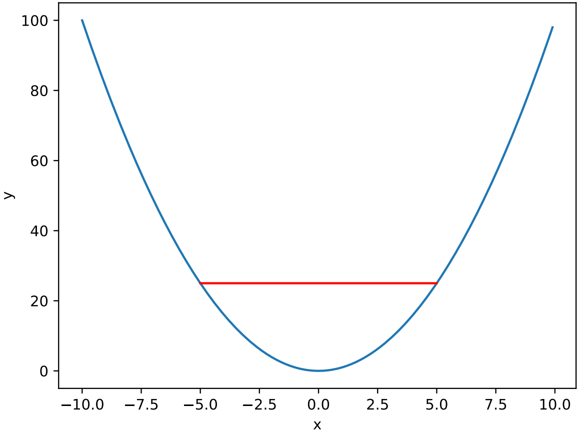


Gradientenverfahren



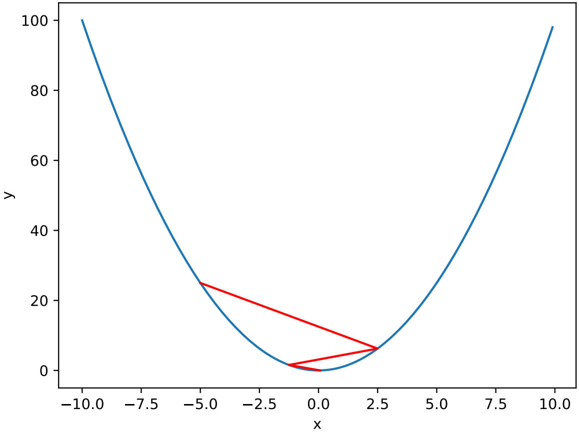
$$x \leftarrow x - \frac{\partial x^2}{\partial x}$$

Gradientenverfahren



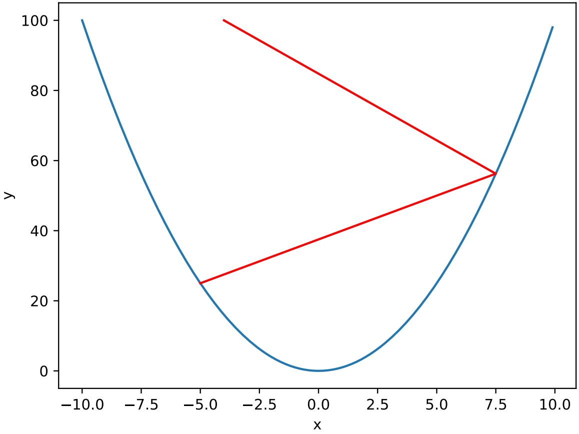
$$x \leftarrow x - \frac{\partial x^2}{\partial x}$$

Gradientenverfahren



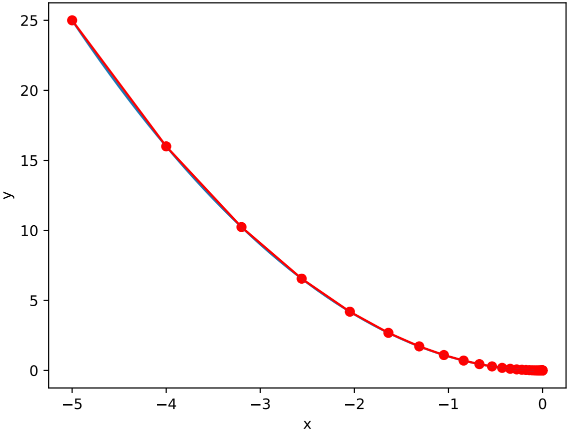
$$x \leftarrow x - \frac{3}{4}\frac{\partial x^2}{\partial x}$$

Gradientenverfahren



$$x \leftarrow x - \frac{5}{4} \frac{\partial x^2}{\partial x}$$

Gradientenverfahren



$$x \leftarrow x - \lambda \frac{\partial x^2}{\partial x}, \lambda = 0.1$$

Gradientenverfahren

Gradient Descent

Input: Initialization θ_0, function f, iterations T, learning rate λ

$\theta \leftarrow \theta_0$
for $i \in [T]$ **do**
  $\theta \leftarrow \theta - \lambda \frac{\partial f(\theta)}{\partial \theta}$
end for
return θ

Gradientenverfahren

Gradient Descent

Input: Initialization θ_0, function f, iterations T, learning rate λ

$\theta \leftarrow \theta_0$
for $i \in [T]$ **do**
  $\theta \leftarrow \theta - \lambda \frac{\partial f(\theta)}{\partial \theta}$
end for
return θ

Least Squares

$$\theta^* := \arg\min_{\theta} \sum_{(x,y) \in D} (g_\theta(x) - y)^2$$

$$\implies f(\theta) := \sum_{(x,y) \in D} (g_\theta(x) - y)^2$$

AdamW

Algorithm 2 Adam with L_2 regularization and Adam with decoupled weight decay (AdamW)

given $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$
initialize time step $t \leftarrow 0$, parameter vector $\boldsymbol{\theta}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, second moment vector $\boldsymbol{v}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
repeat
  $t \leftarrow t + 1$
  $\nabla f_t(\boldsymbol{\theta}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{\theta}_{t-1})$ ▷ select batch and return the corresponding gradient
  $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1})$ $+\lambda\boldsymbol{\theta}_{t-1}$
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t$ ▷ here and below all operations are element-wise
  $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t^2$
  $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t/(1 - \beta_1^t)$ ▷ β_1 is taken to the power of t
  $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t/(1 - \beta_2^t)$ ▷ β_2 is taken to the power of t
  $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$ ▷ can be fixed, decay, or also be used for warm restarts
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \left(\alpha \hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon) +\lambda\boldsymbol{\theta}_{t-1} \right)$
until *stopping criterion is met*
return optimized parameters $\boldsymbol{\theta}_t$

Ihr seid dran!

Gradient Descent

Input: Initialization θ_0, function f, iterations T, learning rate λ

$\theta \leftarrow \theta_0$

for $i \in [T]$ **do**

$\quad \theta \leftarrow \theta - \lambda \frac{\partial f(\theta)}{\partial \theta}$

end for

return θ

Linear Least Squares

$$\theta^* := \arg\min_{\theta} \sum_{(x,y)\in D} (ax - y)^2 \implies f(a) := \sum_{(x,y)\in D} (ax - y)^2$$

$$\frac{\partial f(a)}{\partial a} = \sum_{(x,y)\in D} 2(ax - y)a, \qquad a \leftarrow a - \lambda \sum_{(x,y)\in D} 2(ax - y)a$$