Gradientenverfahren

Wie kann man Minima einer differenzierbaren Abbildung $f : \mathbb{R}^n \to \mathbb{R}$ finden?

Gradient

Sei $f : U \to \mathbb{R}$ differenzierbare Funktion, $a \in U$ und $v := \text{argmax}_{||h||=1}\{\partial_h f(a)\}$. Dann gilt

$$||\nabla f(a)||v = \nabla f(a) \, .$$

Gradient

Der Gradient zeigt in die Richtung des steilsten Anstiegs.

Beweis

Für beliebiges h gilt

$$\partial_h f(a) = df(a)h = \langle \nabla f(a), h \rangle = ||\nabla f(a)|| \cdot ||h|| \cdot \cos(\varphi)$$

wobei φ den Innenwinkel zwischen $\nabla f(a)$ und h bezeichnet. Für $||h|| = 1$ wird somit $\partial_h f(a)$ maximal, wenn $\varphi = 0$ und somit$h = \frac{\nabla f(a)}{||\nabla f(a)||}$ ist.

Extrema

Sei $f : X \subset \mathbb{R}^n \to \mathbb{R}$ eine relle Funktion. Ein Punkt $a \in X$ heißt lokales Maximum bzw. Minimum, falls eine Umgebung U von a existiert, so dass $f(x) \leq f(a)$ bzw. $f(x) \geq f(a)$ für alle $x \in U$ gilt. Liegt einer der beiden Fälle vor, so spricht man von einem lokalen Extremum. Gilt strikt $f(x) < f(a)$ bzw. $f(x) > f(a)$, so nennt man das Extremum isoliert. Ist $U = X$ so nennt man es auch globales Maximum bzw. Minimum.

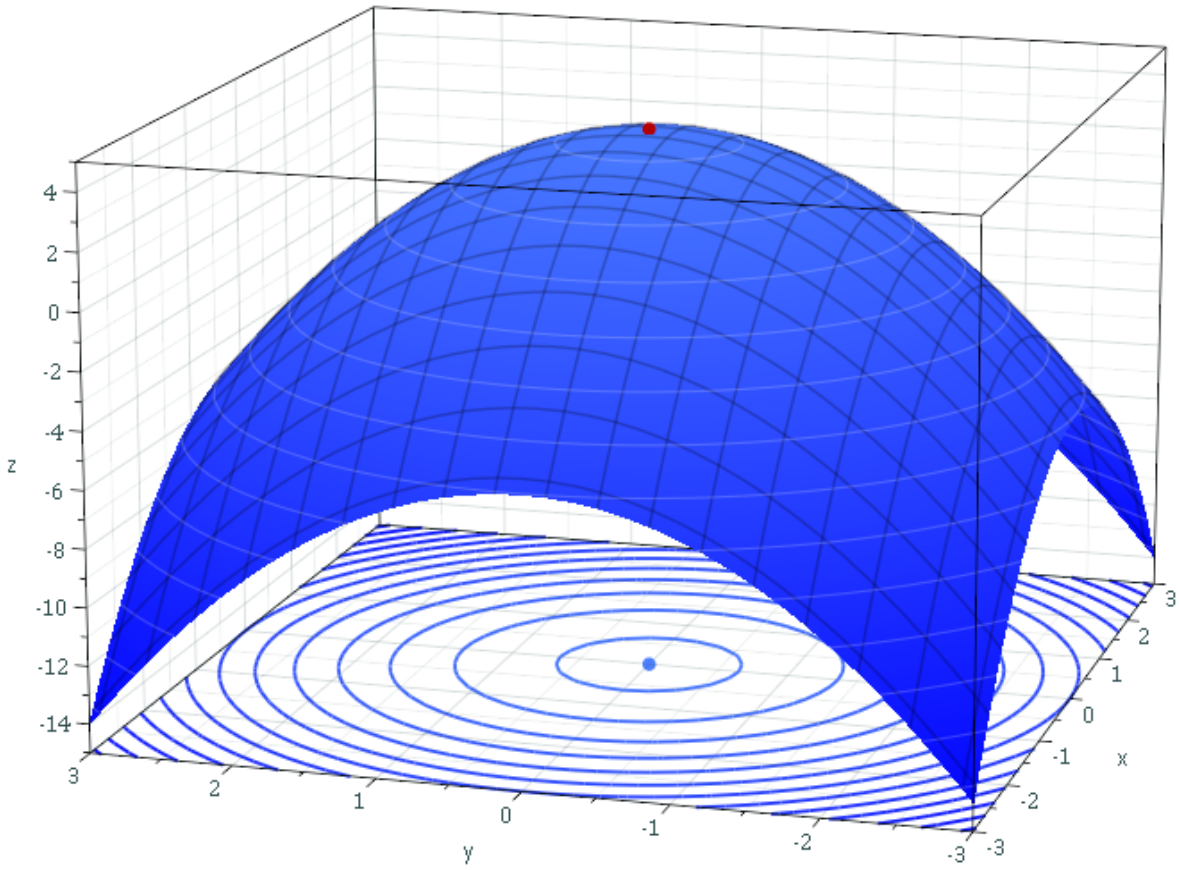


Figure: Quelle: Wikipedia:
https://en.wikipedia.org/wiki/File:MaximumParaboloid.png

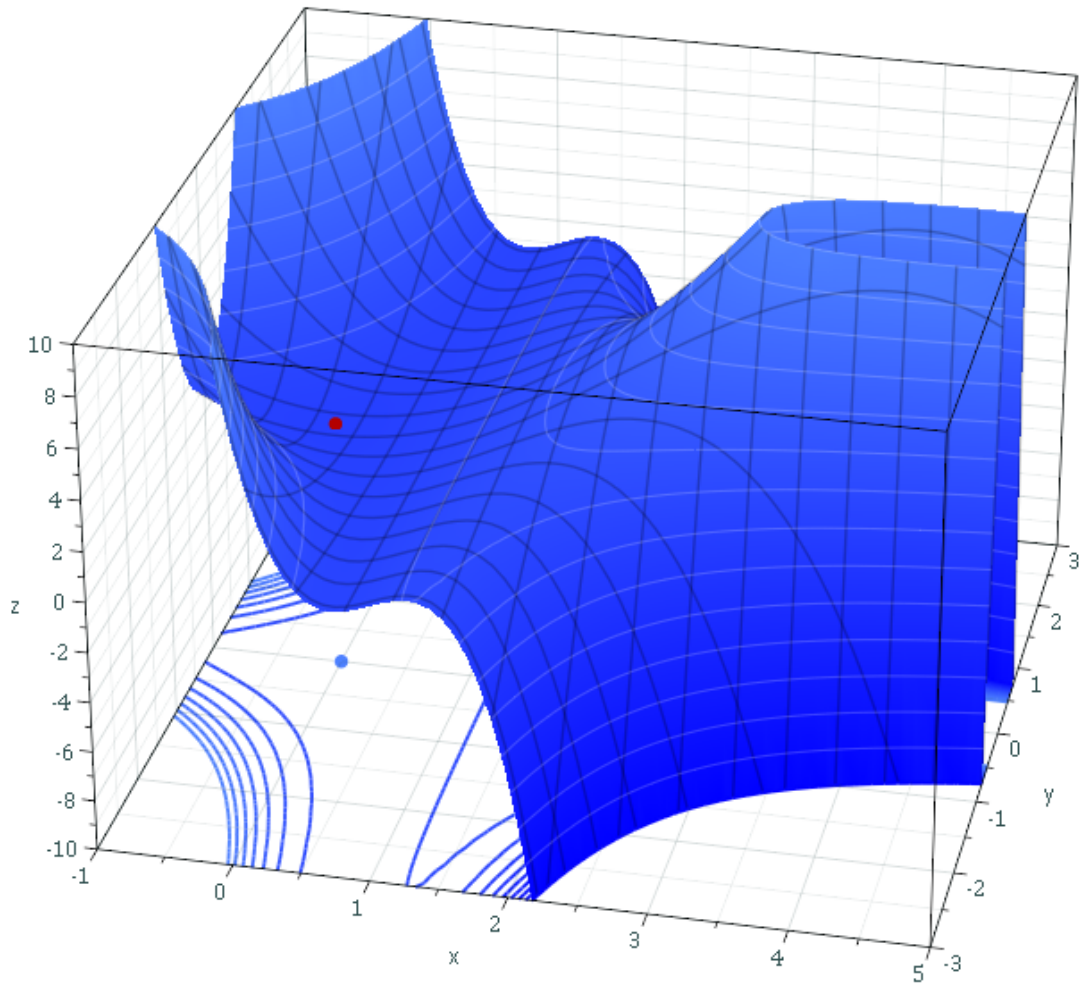


Figure: Quelle: Wikipedia: https://en.wikipedia.org/wiki/File:MaximumCounterexample.png

Extrema Mathlib

Ist $f : U \to \mathbb{R}$ differenzierbar und hat f in $a \in U$ ein lokales Extremum, so gilt

$$\frac{\partial}{\partial x_1} f(a) = \cdots = \frac{\partial}{\partial x_n} f(a) = 0 \,.$$

Sind die partiellen Ableitungen stetig, ist dies gleichbedeutend mit $df(a) = 0$.

Kritischer Punkt

Ein Punkt a mit $df(a) = 0$ wird kritischer Punkt genannt.

Beweis

Setze $F_k(t) := f(a + te_k)$. Da f ein Extremum in a hat, hat F_k in einer hinreichend kleinen Umgebung um 0 ein Extremum. Da F_k eine Funktion einer Veränderlichen ist, gilt $F'(0) = 0$. Da $\frac{\partial}{\partial x_k} f(a) = F_k'(0)$ folgt die Behauptung.

Gradientenverfahren

- An jedem Punkt $x_k \in \mathbb{R}^n$ zeigt der negative Gradient $d_k := -\nabla f(x_k)$ in die steilste Abstiegsrichtung.
- Für hinreichend kleines α_k folgt mit Satz über die lokale Linearisierung:
  $f(x_{k+1}) = f(x_k + \alpha_k d_k) = f(x_k) + \alpha_k df(x_k) d_k + R(\alpha_k dk)$
- Setze $x_{k+1} = x_k + \alpha_k d_k$
- Es gilt $f(x_{k+1}) \leq f(x_k)$, falls $\nabla f(x_k) \neq 0$
- Falls die folge $f(x_k)$ beschränkt ist, so ist dieser Fixpunkt x^* ein Minimum, da $\nabla f(x^*) = 0$ gelten muss.

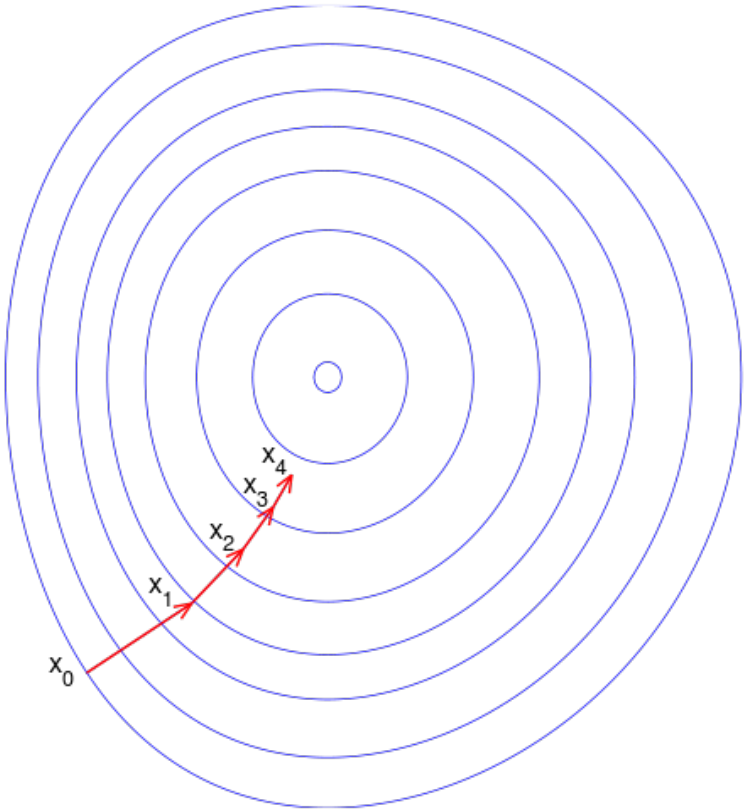


Figure: Quelle: Wikipedia

Höhenlinien

Sei $f : \mathbb{R}^n \to \mathbb{R}$ eine differenzierbare Funktion. Eine Kurve $\gamma : I \to \mathbb{R}^n$, auf der f konstant ist, also $f(\gamma(t)) = c$ für ein festes $c \in \mathbb{R}$ gilt, heißt Höhenlinie.

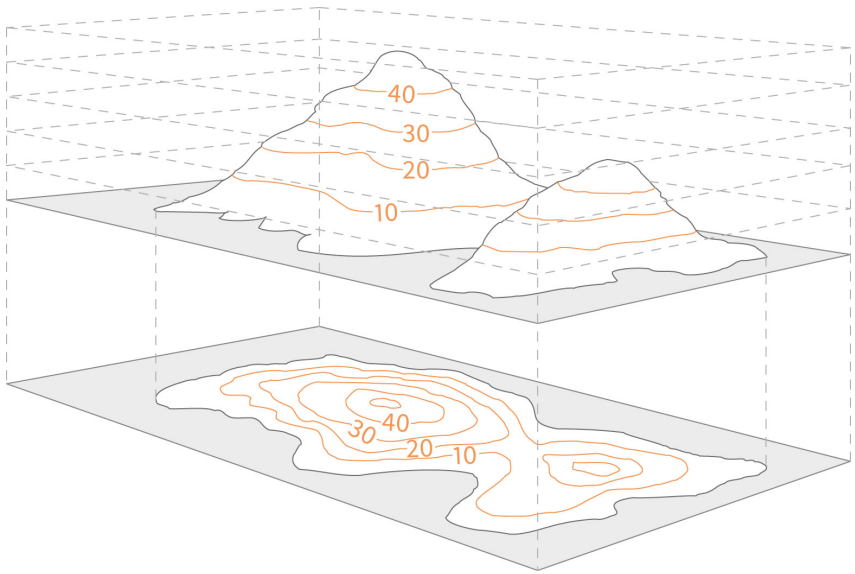


Figure: Quelle: https://getoutside.ordnancesurvey.co.uk/guides/understanding-map-contour-lines-for-beginners/

Höhenlinien

Der Gradient steht senkrecht auf Höhenlinien. Dies bedeutet, dass

$$\langle \nabla f(\gamma(t)), \gamma'(t) \rangle = 0$$

gilt.

Beweis

Aus $f(\gamma(t)) = c$ folgt $\frac{d}{dt} f(\gamma(t)) = 0$. Mit der Kettenregel folgt $\frac{d}{dt} f(\gamma(t)) = df(\gamma(t)) \cdot \gamma'(t) = 0$ und damit $\langle \nabla f(\gamma(t)), \gamma'(t) \rangle = 0$.

Backpropagation

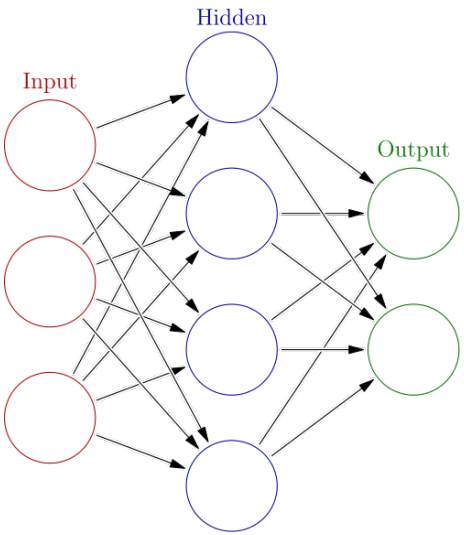
Das Gradientenverfahren angewendet auf eine Lossfunktion eines neuronalen Netzes wird als Backpropagation bezeichnet. Gegeben ist ein neuronales Netz $f : \Omega \times \mathbb{R}^n \to \mathbb{R}^m$, und ein Datensatz $D := \{(x_i, y_i)\}$ mit $x_i \in \mathbb{R}^n, y_i \in \mathbb{R}^m$. Finde Gewichte Omega, so dass Lossfunktion

$$L_D : \Omega \subset \mathbb{R}^n \to \mathbb{R}$$

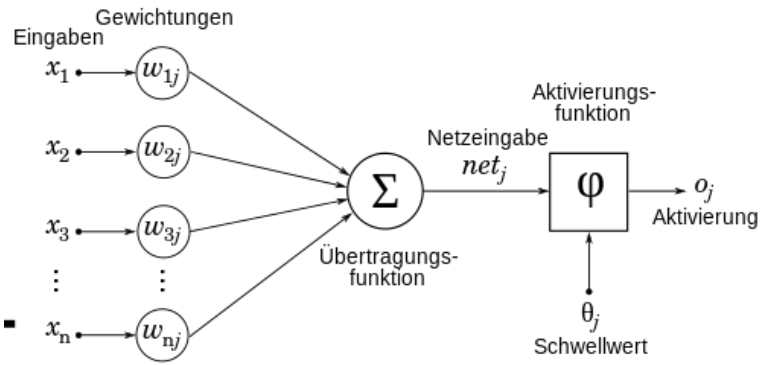
minimal wird. Zum Beispiel

$$L_D(\omega) := \sum_{(x_i,y_i)\in D} (f(\omega, x_i) - y_i)^2$$

.



Figure



Figure

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0.
- Initialisiere Genauigkeit $\epsilon > 0$
- While $||\nabla L_D(\omega)|| > \epsilon$
- Bestimme α_k mit
  $L_D(\omega_k + \alpha d_k) = L_D(\omega_k) + \alpha_k dL_D(\omega_k)d_k + R(\alpha_k dk)$
- Setze $\omega_{k+1} := \omega_k + \alpha_k d_k$.
- $k \leftarrow k + 1$

Mini Batch

- Datensatz D sehr groß (Big Data)
- Berechnung des Gradienten der Lossfunktion entsprechend aufwendig.
- Wende Backpropagation auf Teilräume $D' \subset D$ an (Minibatch).
- $\#D' = 1$ stochastischer Gradientenabstieg.

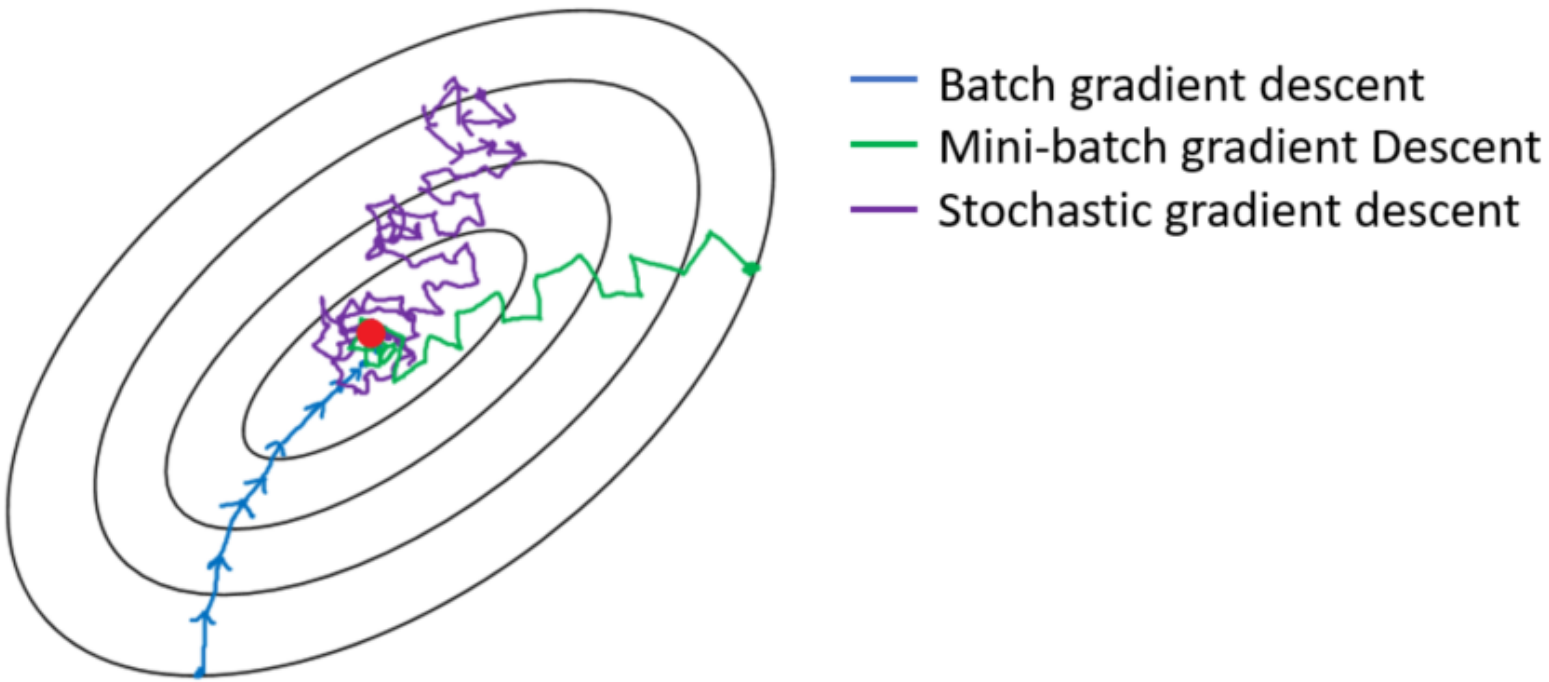


Figure: Quelle: https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0.
- Initialisiere Genauigkeit $\epsilon > 0$
- Wähle Teilmenge $D'_0 \subset D$
- While $||\nabla L_{D'_k}(\omega)|| > \epsilon$
- Bestimme α_k mit
  $L_{D'_k}(\omega_k + \alpha d_k) = L_{D'_k}(\omega_k) + \alpha_k dL_{D'_k}(\omega_k)d_k + R(\alpha_k dk)$
- Setze $\omega_{k+1} := \omega_k + \alpha_k d_k$.
- Wähle neue Teilmenge $D'_{k+1} \subset D$.
- $k \leftarrow k + 1$