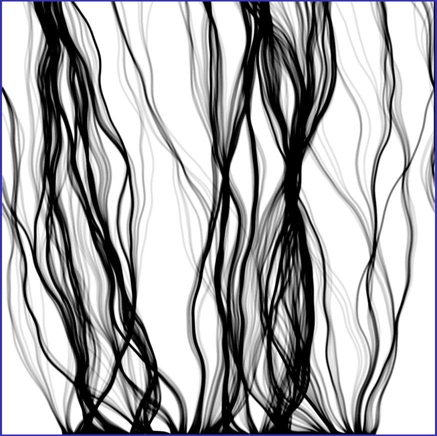
Angewandte Mathematik



Dr. rer. nat. Johannes Riesterer

Gradientenverfahren

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

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.

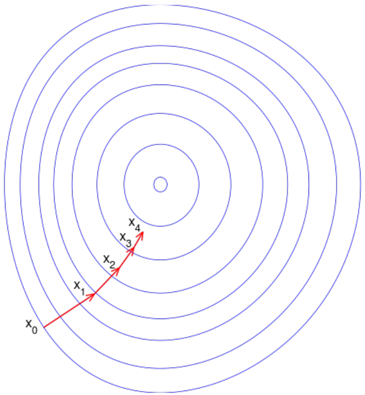


Figure: Quelle: Wikipedia

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 eine festes $c \in \mathbb{R}$ gilt, heißt Höhenlinie.

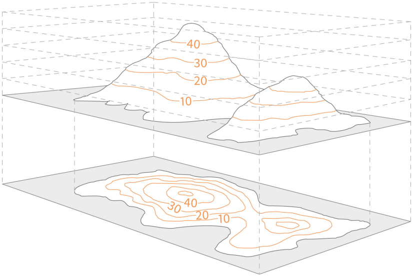


Figure: Quelle: https://getoutside.ordnancesurvey.co.uk/guides/understanding-map-contour-lines-for-beginners/

Höhenlinien

Der Gradient steht senkrecht auf Höhenlinien. Dies bedeutet, dass

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

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

Backpropagation

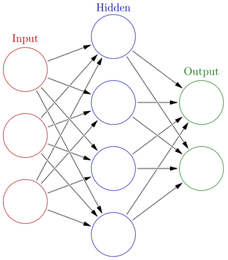
Das Gradientenverfahren angewendet auf eine Lossfunktion eines neuronalen Netzes wird als Backpropagation bezeichnet. Gegeben ist ein neuronales Netz $f : \Omega \times \mathbb{R}^n \to \mathbb{R}^m$, und ein Datensatz $D := \{(x_i, y_i)\}$ mit $x_i \in \mathbb{R}^n, y_i \in \mathbb{R}^m$. Finde Gewichte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

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

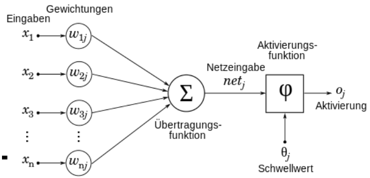
minimal wird. Zum Beispiel

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

.



Figure



Figure

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0.

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0.
- Initialisiere Genauigkeit $\epsilon > 0$

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0.
- Initialisiere Genauigkeit $\epsilon > 0$
- While $||\nabla L_D(\omega)|| > \epsilon$

Backpropagation

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

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 $\omega_{k+1} := \omega_k + \alpha_k d_k$.

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 ist (Big Data)

Mini Batch

- Datensatz D sehr groß ist (Big Data)
- Berechnung des Gradient der Lossfunktion entsprechend aufwendig.

Mini Batch

- Datensatz D sehr groß ist (Big Data)
- Berechnung des Gradient der Lossfunktion entsprechend aufwendig.
- Wende Backpropagation auf Teilräume $D' \subset D$ an. (Minibatch).

Mini Batch

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

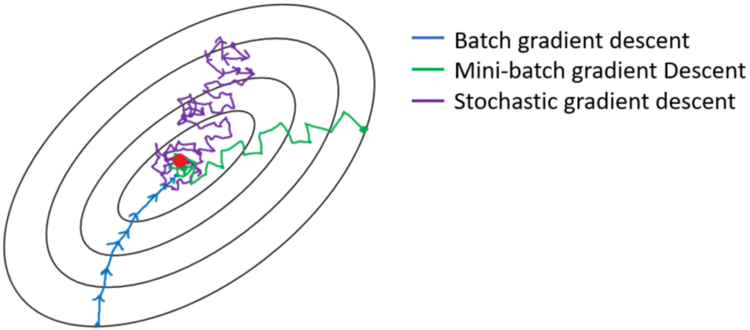


Figure: Quelle: https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0.

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0.
- Initialisiere Genauigkeit $\epsilon > 0$

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0.
- Initialisiere Genauigkeit $\epsilon > 0$
- Wähle Teilmenge $D_0' \subset D$

Backpropagation

- Initialisiere $k := 0$ und zufällige Gewichte w_0.
- Initialisiere Genauigkeit $\epsilon > 0$
- Wähle Teilmenge $D_0' \subset D$
- While $||\nabla L_{D_k'}(\omega)|| > \epsilon$

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 $L_{D_k'}(\omega_k + \alpha d_k) = L_{D_k'}(\omega_k) + \alpha_k dL_{D_k'}(\omega_k) d_k + R(\alpha_k dk)$

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 $\omega_{k+1} := \omega_k + \alpha_k d_k$.

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 $D'_{k+1} \subset D$.

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 $D'_{k+1} \subset D$.
- $k \leftarrow k + 1$

Gradient einer mehrdimensionalen Funktion

Eine Funktion $F : U \to \mathbb{R}^m$ heißt differenzierbar, wenn es eine lineare Abbildung dF gibt, so dass

$$F(a+h) = F(a) + dF(a)h + R(h)$$

mit $\lim_{h\to 0} \frac{R(h)}{||h||} = 0$ gilt für alle $a \in U$ und $h \in \mathbb{R}^n$.

Gradient einer mehrdimensionalen Funktion

Im Fall $n = 1$ stimmt diese Definition mit der alten Definition überein.

Beweis

Nach Satz über die lokale Linearisierung gilt für eine differenzierbare Funktion $f(a + th) = f(a) + dfth + R(th)$ mit $\lim t \to 0 \frac{R(th)}{||th||} = 0$. Umstellen ergibt

$$df(a)h = \lim_{t \to 0} \frac{f(a + th) - f(a)}{t}$$

Gradient einer linearen Funktion

Für $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$ und $x \in \mathbb{R}^n$ ist die Abbildung
$F(x) := Ax + b$ differenzierbar, da
$F(a+h) = A(a+h) + b = Aa + Ah + b = Aa + b + Ah = F(a) + Ah$
und damit für $dF(a) := A$ und $R(h) = 0$ die Definition erfüllt ist.

Differenzierbarkeit von Produktfunktionen

Eine Funktion $F := (F_1, F_2) : U \to \mathbb{R}^m \times \mathbb{R}^k$ ist genau dann differenzierbar, wenn $F_1 : U \to \mathbb{R}^m$ und $F_2 : U \to \mathbb{R}^k$ differenzierbar sind. In diesem Fall ist

$$dF(a) = (dF_1(a), dF_2(a)) \ .$$

Beweis

Sind F_1 und F_2 differenzierbar, so gilt für $i = 1, 2$

$$F_i(a+h) = F_i(a) + dF_i h + R_i(h)$$

Dann gilt mit $dF(a) = (dF_1(a), df_2(a))$ und $R(h) := (R_1(h), R_2(h))$

$$F(a+h) = F(a) + dFh + R(h)$$

mit $\lim_{h \to 0} \frac{R(h):}{||h||} = 0$ und damit ist F differenzierbar. Die Umkehrung folgt analog.

Differenzierbarkeit von Produktfunktionen

Eine Abbildung $F : U \to \mathbb{R}^m$ ist genau dann differenzierbar, wenn ihre Koordinaten-Funktionen $F_1 : U \to \mathbb{R}, \cdots, F_m : U \to \mathbb{R}$ mit $F(a) = \begin{pmatrix} F_1(a) \\ \vdots \\ F_m(a) \end{pmatrix}$ differenzierbar sind. In diesem Fall gilt

$$dF(a) := \begin{pmatrix} \frac{\partial}{\partial x_1} F_1(a) & \cdots & \frac{\partial}{\partial x_n} F_1(a) \\ \vdots & & \vdots \\ \frac{\partial}{\partial x_1} F_m(a) & \cdots & \frac{\partial}{\partial x_n} F_m(a) \end{pmatrix}$$

Differenzierbarkeit von Produktfunktionen

Ein Weg $\gamma = \begin{pmatrix} \gamma_1 \\ \vdots \\ \gamma_m \end{pmatrix} : I \to U$ ist genau dann differenzierbar, wenn γ_i differenzierbar ist für $i = 1, \cdots, m$ und dann gilt

$$\gamma'(t) = \begin{pmatrix} \gamma_1'(t) \\ \vdots \\ \gamma_m'(t) \end{pmatrix} .$$

Kettenregel

Seien $G : U \subset \mathbb{R}^n \to V \subset \mathbb{R}^m$ und $F : V \to Z \subset \mathbb{R}^k$ differenzierbar. Dann ist $F \circ G$ differenzierbar und mit $b := G(a)$ es gilt

$$d(F \circ G)(a) = dF(b) \cdot dG(a)$$

Beweis

Analog zu Baby Kettenregel

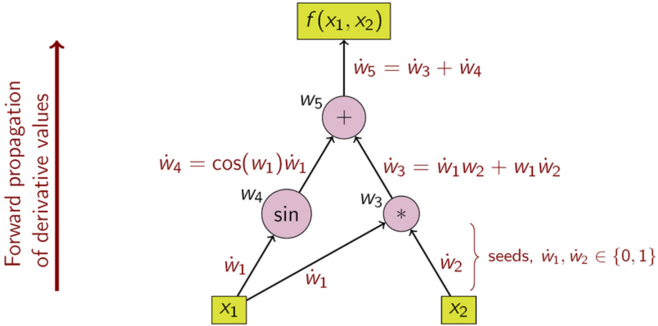


Figure: Quelle: Wikipedia

Automatisches Ableiten in Pytorch