

Training von MLPs

Training von MLPs: Gradientenabstieg

- Problem der logistischen Regression: Funktioniert nur für zweischichtige Perzeptren.
- Allgemeinerer Ansatz: **Gradientenabstieg**.
- Notwendige Bedingung: **differenzierbare Aktivierungs- und Ausgabe-funktionen**.

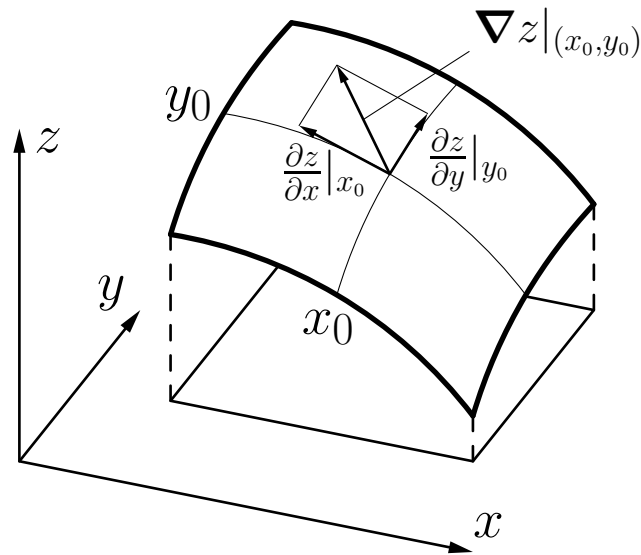


Illustration des Gradienten einer reellwertigen Funktion $z = f(x, y)$ am Punkt (x_0, y_0) .
Dabei ist $\nabla z|_{(x_0, y_0)} = \left(\frac{\partial z}{\partial x}|_{x_0}, \frac{\partial z}{\partial y}|_{y_0} \right)$.

Gradientenabstieg: Formaler Ansatz

Grundidee: Erreiche das Minimum der Fehlerfunktion in kleinen Schritten.

Fehlerfunktion:

$$e = \sum_{l \in L_{\text{fixed}}} e^{(l)} = \sum_{v \in U_{\text{out}}} e_v = \sum_{l \in L_{\text{fixed}}} \sum_{v \in U_{\text{out}}} e_v^{(l)},$$

Erhalte den Gradienten zur Schrittrichtungsbestimmung:

$$\nabla_{\mathbf{w}_u} e = \frac{\partial e}{\partial \mathbf{w}_u} = \left(-\frac{\partial e}{\partial \theta_u}, \frac{\partial e}{\partial w_{up_1}}, \dots, \frac{\partial e}{\partial w_{up_n}} \right).$$

Nutze die Summe über die Trainingsmuster aus:

$$\nabla_{\mathbf{w}_u} e = \frac{\partial e}{\partial \mathbf{w}_u} = \frac{\partial}{\partial \mathbf{w}_u} \sum_{l \in L_{\text{fixed}}} e^{(l)} = \sum_{l \in L_{\text{fixed}}} \frac{\partial e^{(l)}}{\partial \mathbf{w}_u}.$$

Gradientenabstieg: Formaler Ansatz

Einzelmusterfehler hängt nur von Gewichten durch die Netzeingabe ab:

$$\nabla_{\mathbf{w}_u} e^{(l)} = \frac{\partial e^{(l)}}{\partial \mathbf{w}_u} = \frac{\partial e^{(l)}}{\partial \text{net}_u^{(l)}} \frac{\partial \text{net}_u^{(l)}}{\partial \mathbf{w}_u}.$$

Da $\text{net}_u^{(l)} = \mathbf{w}_u \mathbf{in}_u^{(l)}$, bekommen wir für den zweiten Faktor

$$\frac{\partial \text{net}_u^{(l)}}{\partial \mathbf{w}_u} = \mathbf{in}_u^{(l)}.$$

Für den ersten Faktor betrachten wir den Fehler $e^{(l)}$ für das Trainingsmuster $l = (\mathbf{i}^{(l)}, \mathbf{o}^{(l)})$:

$$e^{(l)} = \sum_{v \in U_{\text{out}}} e_u^{(l)} = \sum_{v \in U_{\text{out}}} \left(o_v^{(l)} - \text{out}_v^{(l)} \right)^2,$$

d.h. die Summe der Fehler über alle Ausgabeneuronen.

Gradientenabstieg: Formaler Ansatz

Daher haben wir

$$\frac{\partial e^{(l)}}{\partial \text{net}_u^{(l)}} = \frac{\partial \sum_{v \in U_{\text{out}}} \left(o_v^{(l)} - \text{out}_v^{(l)} \right)^2}{\partial \text{net}_u^{(l)}} = \sum_{v \in U_{\text{out}}} \frac{\partial \left(o_v^{(l)} - \text{out}_v^{(l)} \right)^2}{\partial \text{net}_u^{(l)}}.$$

Da nur die eigentliche Ausgabe $\text{out}_v^{(l)}$ eines Ausgabeneurons v von der Netzeingabe $\text{net}_u^{(l)}$ des Neurons u abhängt, das wir betrachten, ist

$$\frac{\partial e^{(l)}}{\partial \text{net}_u^{(l)}} = -2 \underbrace{\sum_{v \in U_{\text{out}}} \left(o_v^{(l)} - \text{out}_v^{(l)} \right)}_{\delta_u^{(l)}} \frac{\partial \text{out}_v^{(l)}}{\partial \text{net}_u^{(l)}},$$

womit zugleich die Abkürzung $\delta_u^{(l)}$ für die im Folgenden wichtige Summe eingeführt wird.

Gradientenabstieg: Formaler Ansatz

- Unterscheide zwei Fälle:
- Das Neuron u ist ein **Ausgabeneuron**.
 - Das Neuron u ist ein **verstecktes Neuron**.

Im ersten Fall haben wir

$$\forall u \in U_{\text{out}} : \quad \delta_u^{(l)} = \left(o_u^{(l)} - \text{out}_u^{(l)} \right) \frac{\partial \text{out}_u^{(l)}}{\partial \text{net}_u^{(l)}}$$

Damit ergibt sich für den Gradienten

$$\forall u \in U_{\text{out}} : \quad \nabla_{\mathbf{w}_u} e_u^{(l)} = \frac{\partial e_u^{(l)}}{\partial \mathbf{w}_u} = -2 \left(o_u^{(l)} - \text{out}_u^{(l)} \right) \frac{\partial \text{out}_u^{(l)}}{\partial \text{net}_u^{(l)}} \mathbf{in}_u^{(l)}$$

und damit für die Gewichtsänderung

$$\forall u \in U_{\text{out}} : \quad \Delta \mathbf{w}_u^{(l)} = -\frac{\eta}{2} \nabla_{\mathbf{w}_u} e_u^{(l)} = \eta \left(o_u^{(l)} - \text{out}_u^{(l)} \right) \frac{\partial \text{out}_u^{(l)}}{\partial \text{net}_u^{(l)}} \mathbf{in}_u^{(l)}.$$

Gradientenabstieg: Formaler Ansatz

Genaue Formel hängt von der Wahl der Aktivierungs- und Ausgabefunktion ab, da gilt

$$\text{out}_u^{(l)} = f_{\text{out}}(\text{act}_u^{(l)}) = f_{\text{out}}(f_{\text{act}}(\text{net}_u^{(l)})).$$

Betrachte Spezialfall mit

- Ausgabefunktion ist die Identität,
- Aktivierungsfunktion ist logistisch, d.h. $f_{\text{act}}(x) = \frac{1}{1+e^{-x}}$.

Die erste Annahme ergibt

$$\frac{\partial \text{out}_u^{(l)}}{\partial \text{net}_u^{(l)}} = \frac{\partial \text{act}_u^{(l)}}{\partial \text{net}_u^{(l)}} = f'_{\text{act}}(\text{net}_u^{(l)}).$$

Gradientenabstieg: Formaler Ansatz

Für eine logistische Aktivierungsfunktion ergibt sich

$$\begin{aligned} f'_{\text{act}}(x) &= \frac{d}{dx} (1 + e^{-x})^{-1} = - (1 + e^{-x})^{-2} (-e^{-x}) \\ &= \frac{1 + e^{-x} - 1}{(1 + e^{-x})^2} = \frac{1}{1 + e^{-x}} \left(1 - \frac{1}{1 + e^{-x}}\right) \\ &= f_{\text{act}}(x) \cdot (1 - f_{\text{act}}(x)), \end{aligned}$$

und daher

$$f'_{\text{act}}(\text{net}_u^{(l)}) = f_{\text{act}}(\text{net}_u^{(l)}) \cdot (1 - f_{\text{act}}(\text{net}_u^{(l)})) = \text{out}_u^{(l)} (1 - \text{out}_u^{(l)}).$$

Die sich ergebende Gewichtsänderung ist daher

$$\Delta \mathbf{w}_u^{(l)} = \eta (o_u^{(l)} - \text{out}_u^{(l)}) \text{out}_u^{(l)} (1 - \text{out}_u^{(l)}) \mathbf{in}_u^{(l)},$$

womit die Berechnungen sehr einfach werden.

Fehlerrückpropagation engl: error backpropagation

Jetzt: Das Neuron u ist ein **verstecktes Neuron**, d.h. $u \in U_k$, $0 < k < r - 1$.

Die Ausgabe $\text{out}_v^{(l)}$ eines Ausgabeneurons v hängt von der Netzeingabe $\text{net}_u^{(l)}$ nur indirekt durch seine Nachfolgeneuronen $\text{succ}(u) = \{s \in U \mid (u, s) \in C\} = \{s_1, \dots, s_m\} \subseteq U_{k+1}$ ab, insbesondere durch deren Netzeingaben $\text{net}_s^{(l)}$.

Wir wenden die Kettenregel an und erhalten

$$\delta_u^{(l)} = \sum_{v \in U_{\text{out}}} \sum_{s \in \text{succ}(u)} (o_v^{(l)} - \text{out}_v^{(l)}) \frac{\partial \text{out}_v^{(l)}}{\partial \text{net}_s^{(l)}} \frac{\partial \text{net}_s^{(l)}}{\partial \text{net}_u^{(l)}}.$$

Summentausch ergibt

$$\delta_u^{(l)} = \sum_{s \in \text{succ}(u)} \left(\sum_{v \in U_{\text{out}}} (o_v^{(l)} - \text{out}_v^{(l)}) \frac{\partial \text{out}_v^{(l)}}{\partial \text{net}_s^{(l)}} \right) \frac{\partial \text{net}_s^{(l)}}{\partial \text{net}_u^{(l)}} = \sum_{s \in \text{succ}(u)} \delta_s^{(l)} \frac{\partial \text{net}_s^{(l)}}{\partial \text{net}_u^{(l)}}.$$

Fehlerrückpropagation

Betrachte die Netzeingabe

$$\text{net}_s^{(l)} = \mathbf{w}_s \mathbf{in}_s^{(l)} = \left(\sum_{p \in \text{pred}(s)} w_{sp} \text{out}_p^{(l)} \right) - \theta_s,$$

wobei ein Element von $\mathbf{in}_s^{(l)}$ die Ausgabe $\text{out}_u^{(l)}$ des Neurons u ist. Daher ist

$$\frac{\partial \text{net}_s^{(l)}}{\partial \text{net}_u^{(l)}} = \left(\sum_{p \in \text{pred}(s)} w_{sp} \frac{\partial \text{out}_p^{(l)}}{\partial \text{net}_u^{(l)}} \right) - \frac{\partial \theta_s}{\partial \text{net}_u^{(l)}} = w_{su} \frac{\partial \text{out}_u^{(l)}}{\partial \text{net}_u^{(l)}},$$

Das Ergebnis ist die rekursive Gleichung (Fehlerrückpropagation)

$$\delta_u^{(l)} = \left(\sum_{s \in \text{succ}(u)} \delta_s^{(l)} w_{su} \right) \frac{\partial \text{out}_u^{(l)}}{\partial \text{net}_u^{(l)}}.$$

Fehlerrückpropagation

Die sich ergebende Formel für die Gewichtsänderung ist

$$\Delta \mathbf{w}_u^{(l)} = -\frac{\eta}{2} \nabla_{\mathbf{w}_u} e^{(l)} = \eta \delta_u^{(l)} \mathbf{in}_u^{(l)} = \eta \left(\sum_{s \in \text{succ}(u)} \delta_s^{(l)} w_{su} \right) \frac{\partial \text{out}_u^{(l)}}{\partial \text{net}_u^{(l)}} \mathbf{in}_u^{(l)}.$$

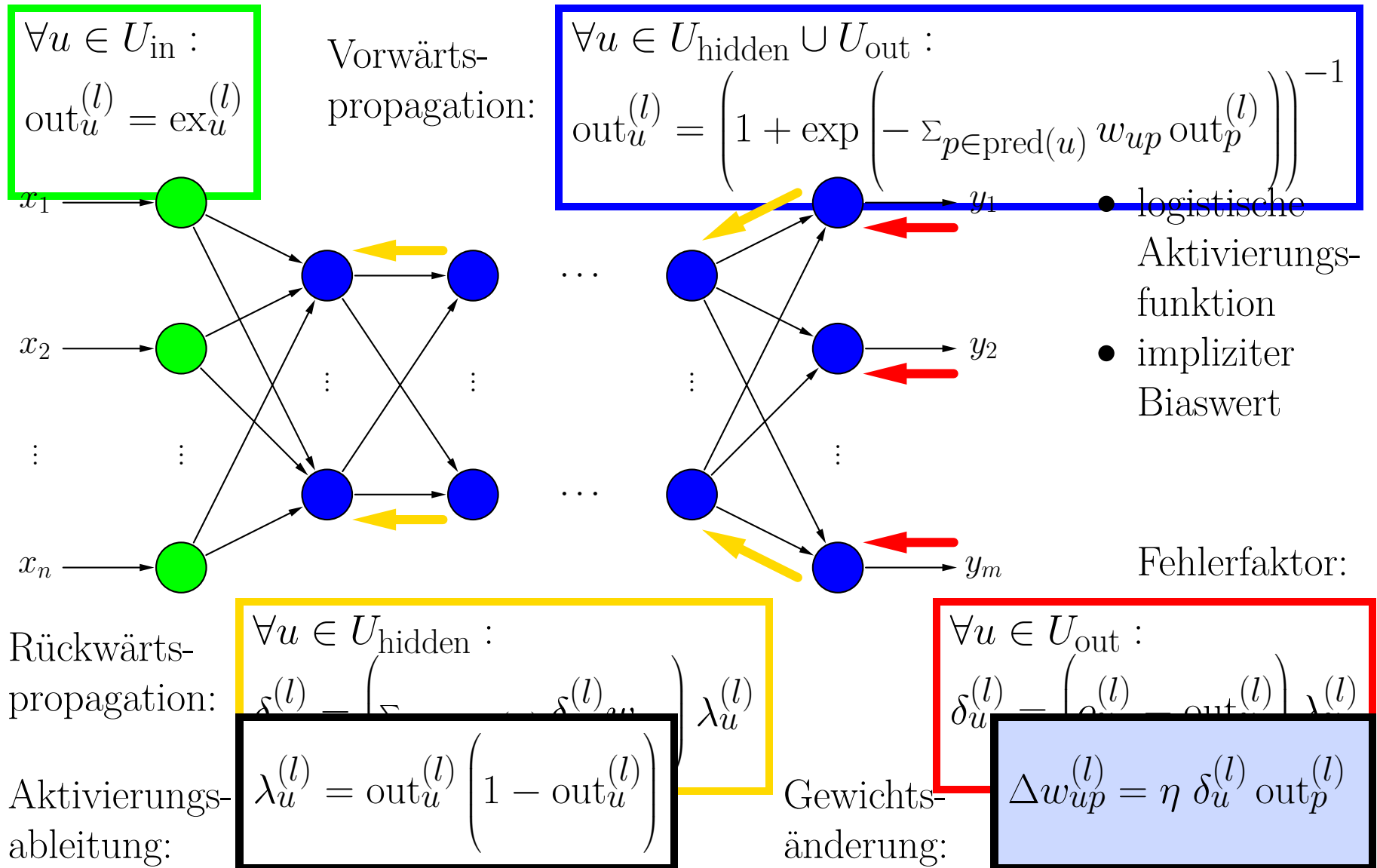
Betrachte erneut den Spezialfall mit

- Ausgabefunktion: Identität,
- Aktivierungsfunktion: logistisch.

Die sich ergebende Formel für die Gewichtsänderung ist damit

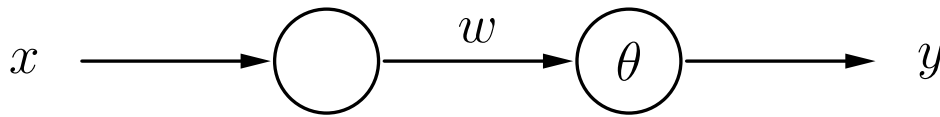
$$\Delta \mathbf{w}_u^{(l)} = \eta \left(\sum_{s \in \text{succ}(u)} \delta_s^{(l)} w_{su} \right) \text{out}_u^{(l)} (1 - \text{out}_u^{(l)}) \mathbf{in}_u^{(l)}.$$

Fehlerrückpropagation: Vorgehensweise

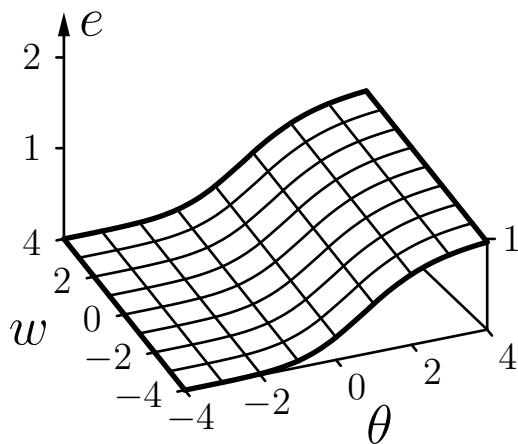


Gradientenabstieg: Beispiele

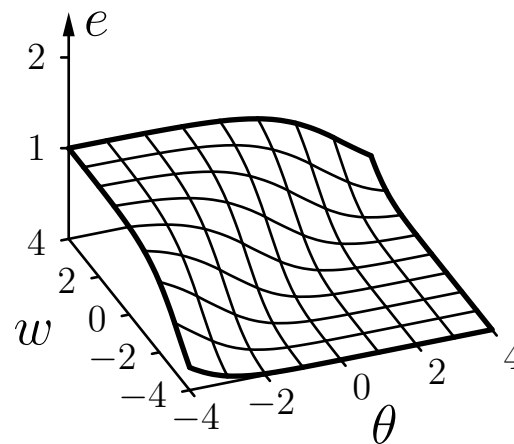
Gradientenabstieg für die Negation $\neg x$



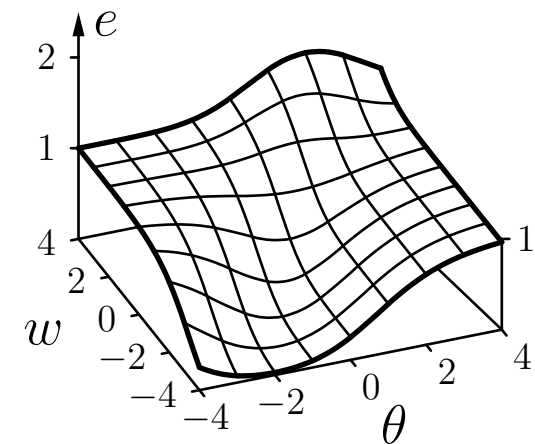
x	y
0	1
1	0



Fehler für $x = 0$



Fehler für $x = 1$



Summe der Fehler

Gradientenabstieg: Beispiele

Epoche	θ	w	Fehler
0	3.00	3.50	1.307
20	3.77	2.19	0.986
40	3.71	1.81	0.970
60	3.50	1.53	0.958
80	3.15	1.24	0.937
100	2.57	0.88	0.890
120	1.48	0.25	0.725
140	-0.06	-0.98	0.331
160	-0.80	-2.07	0.149
180	-1.19	-2.74	0.087
200	-1.44	-3.20	0.059
220	-1.62	-3.54	0.044

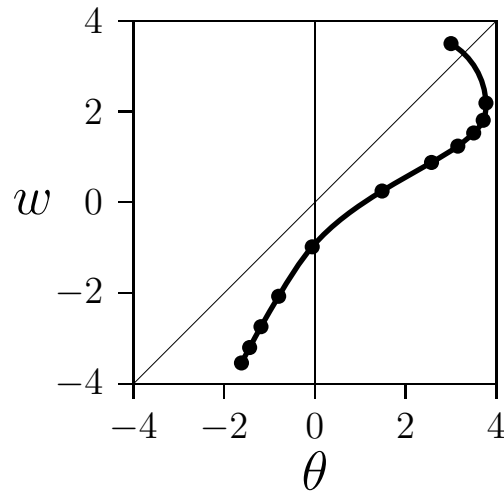
Online-Training

Epoche	θ	w	Fehler
0	3.00	3.50	1.295
20	3.76	2.20	0.985
40	3.70	1.82	0.970
60	3.48	1.53	0.957
80	3.11	1.25	0.934
100	2.49	0.88	0.880
120	1.27	0.22	0.676
140	-0.21	-1.04	0.292
160	-0.86	-2.08	0.140
180	-1.21	-2.74	0.084
200	-1.45	-3.19	0.058
220	-1.63	-3.53	0.044

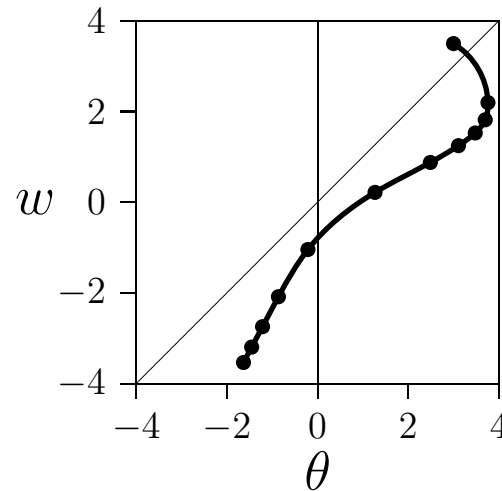
Batch-Training

Gradientenabstieg: Beispiele

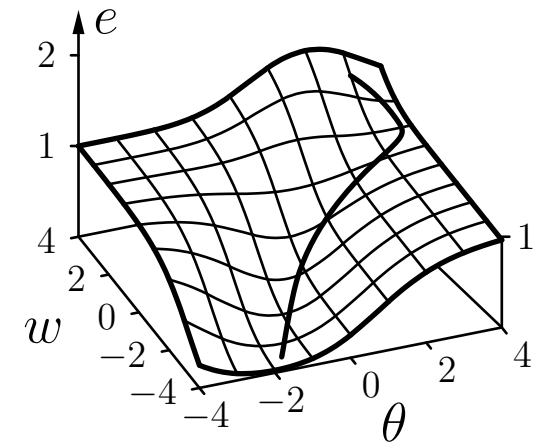
Visualisierung des Gradientenabstiegs für die Negation $\neg x$



Online-Training



Batch-Training



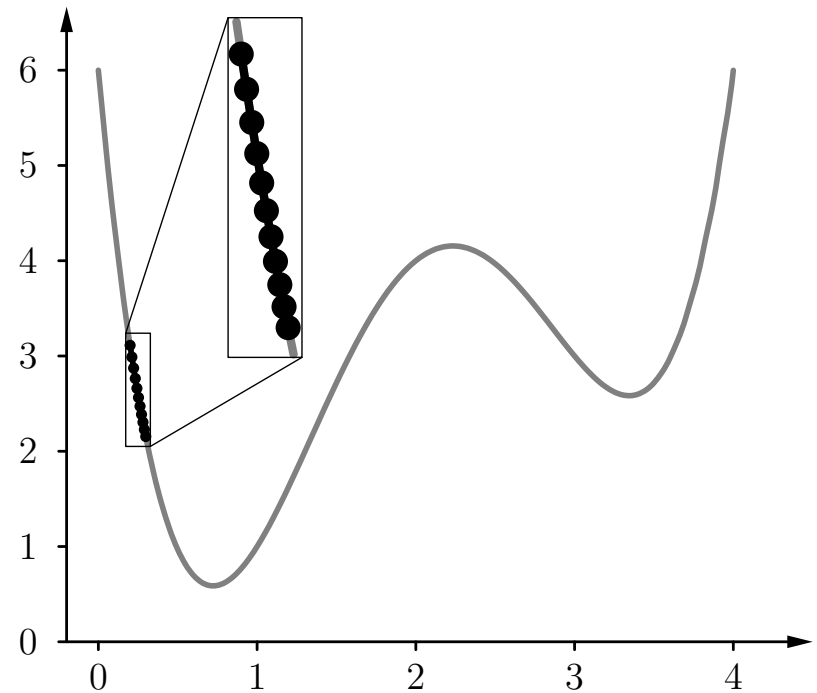
Batch-Training

- Das Training ist offensichtlich erfolgreich.
- Der Fehler kann nicht vollständig verschwinden, bedingt durch die Eigenschaften der logistischen Funktion.

Gradientenabstieg: Beispiele

Beispielfunktion: $f(x) = \frac{5}{6}x^4 - 7x^3 + \frac{115}{6}x^2 - 18x + 6,$

i	x_i	$f(x_i)$	$f'(x_i)$	Δx_i
0	0.200	3.112	-11.147	0.011
1	0.211	2.990	-10.811	0.011
2	0.222	2.874	-10.490	0.010
3	0.232	2.766	-10.182	0.010
4	0.243	2.664	-9.888	0.010
5	0.253	2.568	-9.606	0.010
6	0.262	2.477	-9.335	0.009
7	0.271	2.391	-9.075	0.009
8	0.281	2.309	-8.825	0.009
9	0.289	2.233	-8.585	0.009
10	0.298	2.160		



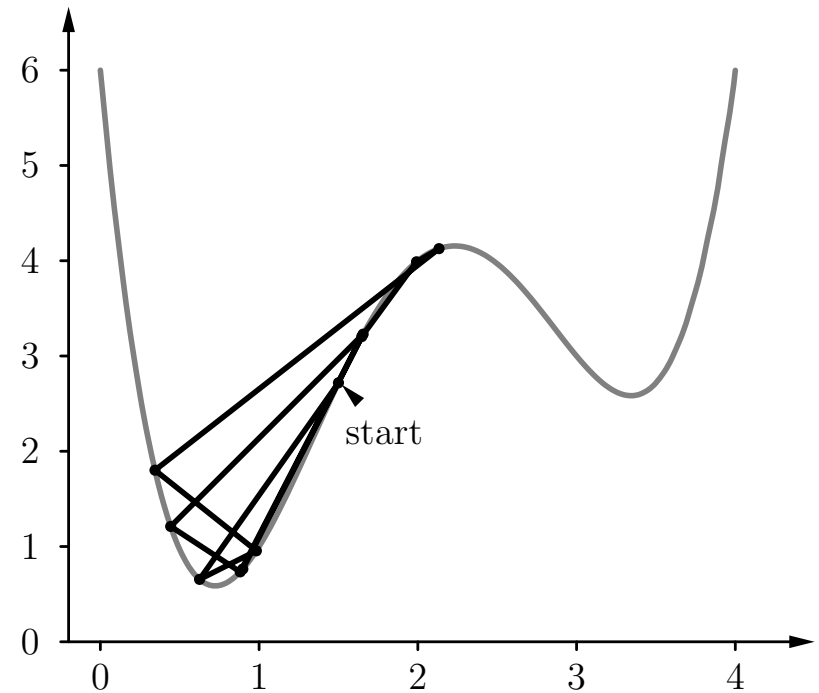
Gradientenabstieg mit Startwert 0.2 und Lernrate 0.001.

Gradientenabstieg: Beispiele

Beispielfunktion:

$$f(x) = \frac{5}{6}x^4 - 7x^3 + \frac{115}{6}x^2 - 18x + 6,$$

i	x_i	$f(x_i)$	$f'(x_i)$	Δx_i
0	1.500	2.719	3.500	-0.875
1	0.625	0.655	-1.431	0.358
2	0.983	0.955	2.554	-0.639
3	0.344	1.801	-7.157	1.789
4	2.134	4.127	0.567	-0.142
5	1.992	3.989	1.380	-0.345
6	1.647	3.203	3.063	-0.766
7	0.881	0.734	1.753	-0.438
8	0.443	1.211	-4.851	1.213
9	1.656	3.231	3.029	-0.757
10	0.898	0.766		



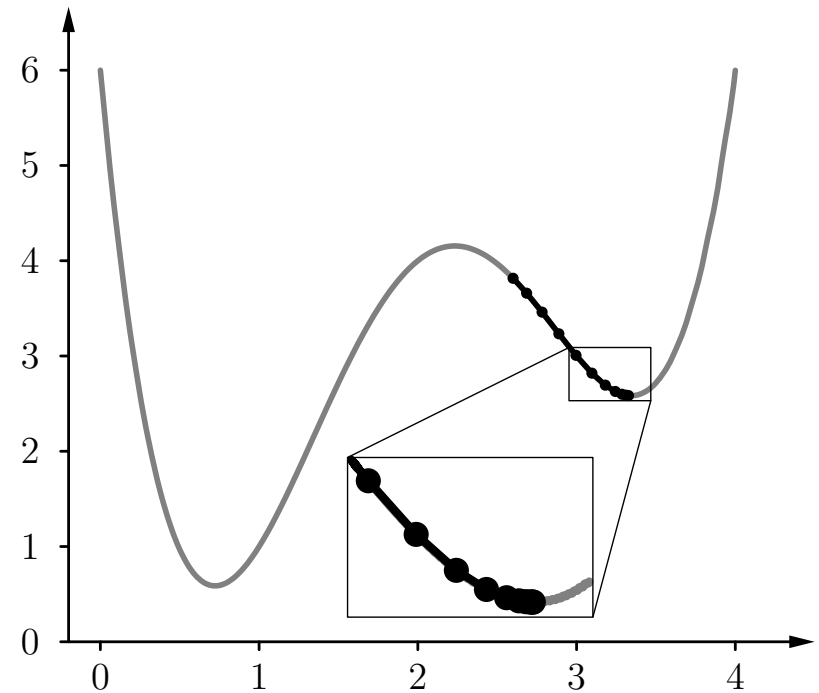
Gradientenabstieg mit Startwert 1.5 und Lernrate 0.25.

Gradientenabstieg: Beispiele

Beispielfunktion:

$$f(x) = \frac{5}{6}x^4 - 7x^3 + \frac{115}{6}x^2 - 18x + 6,$$

i	x_i	$f(x_i)$	$f'(x_i)$	Δx_i
0	2.600	3.816	-1.707	0.085
1	2.685	3.660	-1.947	0.097
2	2.783	3.461	-2.116	0.106
3	2.888	3.233	-2.153	0.108
4	2.996	3.008	-2.009	0.100
5	3.097	2.820	-1.688	0.084
6	3.181	2.695	-1.263	0.063
7	3.244	2.628	-0.845	0.042
8	3.286	2.599	-0.515	0.026
9	3.312	2.589	-0.293	0.015
10	3.327	2.585		



Gradientenabstieg mit Startwert 2.6 und Lernrate 0.05.

Gradientenabstieg: Varianten

Gewichts-Updateregel:

$$w(t + 1) = w(t) + \Delta w(t)$$

Standard-Backpropagation:

$$\Delta w(t) = -\frac{\eta}{2} \nabla_w e(t)$$

Manhattan-Training:

$$\Delta w(t) = -\eta \operatorname{sgn}(\nabla_w e(t))$$

d.h. es wird nur die Richtung (Vorzeichen) der Änderung beachtet und eine feste Schrittweite gewählt

Moment-Term:

$$\Delta w(t) = -\frac{\eta}{2} \nabla_w e(t) + \beta \Delta w(t - 1),$$

d.h. bei jedem Schritt wird noch ein gewisser Anteil des vorherigen Änderungsschritts mit berücksichtigt, was zu einer Beschleunigung führen kann

Selbstadaptive Fehlerrückpropagation:

$$\eta_w(t) = \begin{cases} c^- \cdot \eta_w(t-1), & \text{falls } \nabla_w e(t) \cdot \nabla_w e(t-1) < 0, \\ c^+ \cdot \eta_w(t-1), & \text{falls } \nabla_w e(t) \cdot \nabla_w e(t-1) > 0 \\ & \wedge \nabla_w e(t-1) \cdot \nabla_w e(t-2) \geq 0, \\ \eta_w(t-1), & \text{sonst.} \end{cases}$$

Elastische Fehlerrückpropagation:

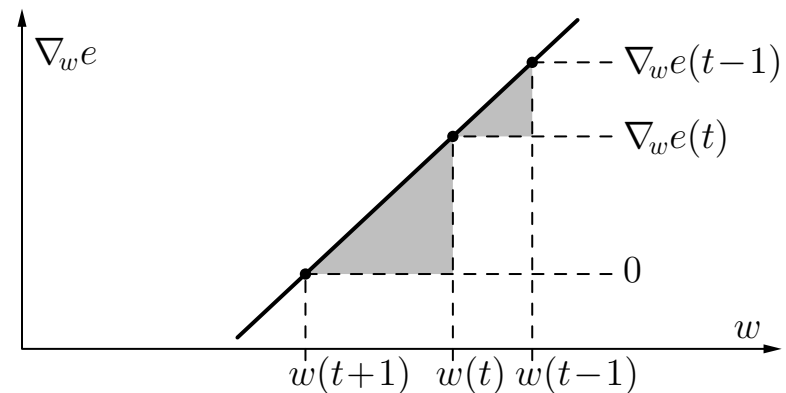
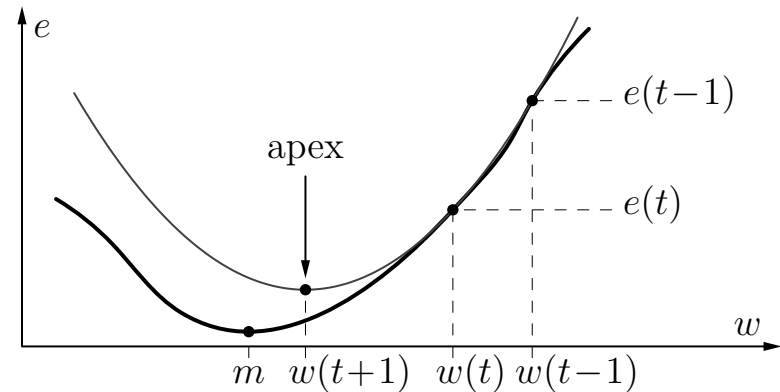
$$\Delta w(t) = \begin{cases} c^- \cdot \Delta w(t-1), & \text{falls } \nabla_w e(t) \cdot \nabla_w e(t-1) < 0, \\ c^+ \cdot \Delta w(t-1), & \text{falls } \nabla_w e(t) \cdot \nabla_w e(t-1) > 0 \\ & \wedge \nabla_w e(t-1) \cdot \nabla_w e(t-2) \geq 0, \\ \Delta w(t-1), & \text{sonst.} \end{cases}$$

Typische Werte: $c^- \in [0.5, 0.7]$ und $c^+ \in [1.05, 1.2]$.

Quickpropagation

Die Gewichts-Updaterregel kann aus den Dreiecken abgeleitet werden:

$$\Delta w(t) = \frac{\nabla_w e(t)}{\nabla_w e(t-1) - \nabla_w e(t)} \cdot \Delta w(t-1).$$



Gradientenabstieg: Beispiele

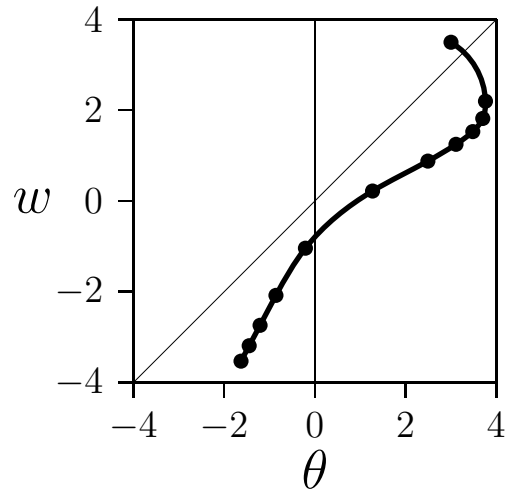
Epoche	θ	w	Fehler
0	3.00	3.50	1.295
20	3.76	2.20	0.985
40	3.70	1.82	0.970
60	3.48	1.53	0.957
80	3.11	1.25	0.934
100	2.49	0.88	0.880
120	1.27	0.22	0.676
140	-0.21	-1.04	0.292
160	-0.86	-2.08	0.140
180	-1.21	-2.74	0.084
200	-1.45	-3.19	0.058
220	-1.63	-3.53	0.044

ohne Momentterm

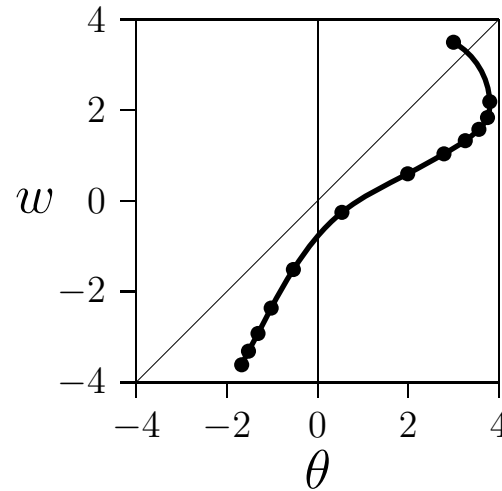
Epoche	θ	w	Fehler
0	3.00	3.50	1.295
10	3.80	2.19	0.984
20	3.75	1.84	0.971
30	3.56	1.58	0.960
40	3.26	1.33	0.943
50	2.79	1.04	0.910
60	1.99	0.60	0.814
70	0.54	-0.25	0.497
80	-0.53	-1.51	0.211
90	-1.02	-2.36	0.113
100	-1.31	-2.92	0.073
110	-1.52	-3.31	0.053
120	-1.67	-3.61	0.041

mit Momentterm

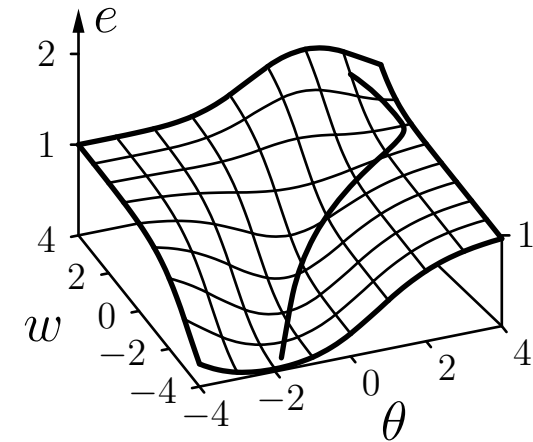
Gradientenabstieg: Beispiele



ohne Momentterm



mit Momentterm



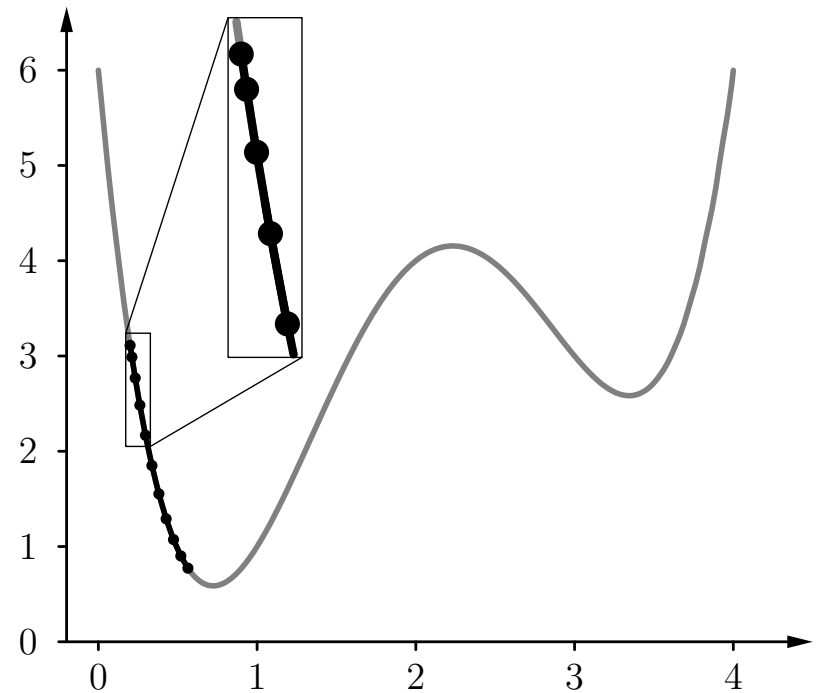
mit Momentterm

- Punkte zeigen die Position alle 20 (ohne Momentterm) oder alle zehn Epochen (mit Momentterm).
- Lernen mit Momentterm ist ungefähr doppelt so schnell.

Gradientenabstieg: Beispiele

Beispielfunktion:
$$f(x) = \frac{5}{6}x^4 - 7x^3 + \frac{115}{6}x^2 - 18x + 6,$$

i	x_i	$f(x_i)$	$f'(x_i)$	Δx_i
0	0.200	3.112	-11.147	0.011
1	0.211	2.990	-10.811	0.021
2	0.232	2.771	-10.196	0.029
3	0.261	2.488	-9.368	0.035
4	0.296	2.173	-8.397	0.040
5	0.337	1.856	-7.348	0.044
6	0.380	1.559	-6.277	0.046
7	0.426	1.298	-5.228	0.046
8	0.472	1.079	-4.235	0.046
9	0.518	0.907	-3.319	0.045
10	0.562	0.777		



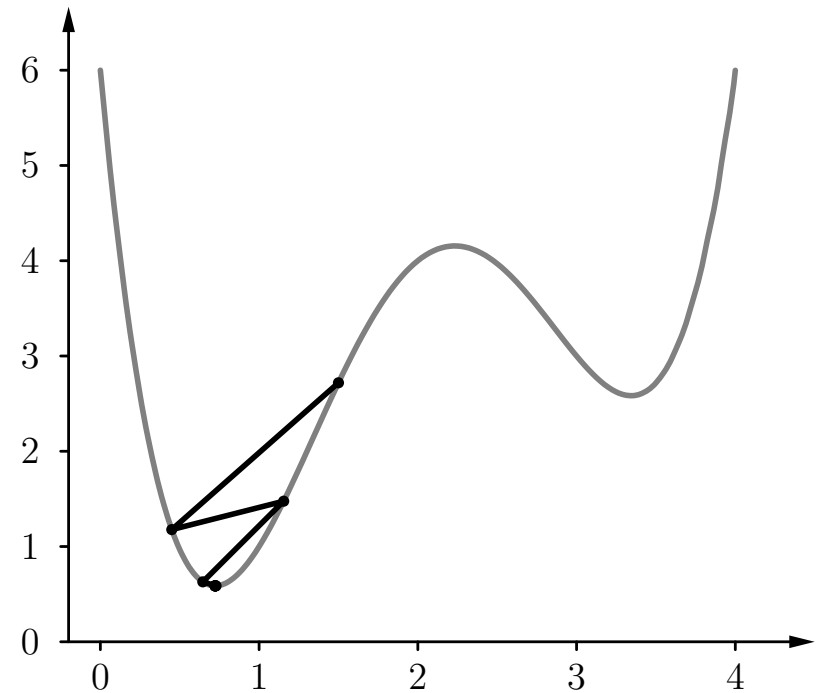
Gradientenabstieg mit Momentterm ($\beta = 0.9$)

Gradientenabstieg: Beispiele

Beispielfunktion:

$$f(x) = \frac{5}{6}x^4 - 7x^3 + \frac{115}{6}x^2 - 18x + 6,$$

i	x_i	$f(x_i)$	$f'(x_i)$	Δx_i
0	1.500	2.719	3.500	-1.050
1	0.450	1.178	-4.699	0.705
2	1.155	1.476	3.396	-0.509
3	0.645	0.629	-1.110	0.083
4	0.729	0.587	0.072	-0.005
5	0.723	0.587	0.001	0.000
6	0.723	0.587	0.000	0.000
7	0.723	0.587	0.000	0.000
8	0.723	0.587	0.000	0.000
9	0.723	0.587	0.000	0.000
10	0.723	0.587		



Gradientenabstieg mit selbstadaptierender Lernrate ($c^+ = 1.2$, $c^- = 0.5$).

Flat Spot Elimination:

$$\Delta w(t) = -\frac{\eta}{2} \nabla_w e(t) + \zeta$$

- Eliminiert langsames Lernen in der Sättigungsregion der logistischen Funktion.
- Wirkt dem Verfall der Fehlersignale über die Schichten entgegen.

Gewichtsverfall: (engl. weight decay)

$$\Delta w(t) = -\frac{\eta}{2} \nabla_w e(t) - \xi w(t),$$

- Kann helfen, die Robustheit der Trainingsergebnisse zu verbessern.
- Kann aus einer erweiterten Fehlerfunktion abgeleitet werden, die große Gewichte bestraft:

$$e^* = e + \frac{\xi}{2} \sum_{u \in U_{\text{out}} \cup U_{\text{hidden}}} \left(\theta_u^2 + \sum_{p \in \text{pred}(u)} w_{up}^2 \right).$$

Sensitivitätsanalyse

Problem: schwer verständliches Wissen, das in einem gelernten neuronalen Netz gespeichert ist:

- Geometrische oder anderweitig anschauliche Deutung gelingt nur bei einfachen Netzen, versagt aber bei komplexen praktischen Problemen
- Versagen des Vorstellungsvermögens insbesondere bei hochdimensionalen Räumen
- Das neuronale Netz wird zu einer *black box*, die auf unergründliche Weise aus den Eingaben die Ausgaben berechnet.

Idee: Bestimmung des Einflusses einzelner Eingaben auf die Ausgabe des Netzes.

→ Sensitivitätsanalyse

Sensitivitätsanalyse

Frage: Wie wichtig sind einzelne Eingaben für das Netzwerk?

Idee: Bestimme die Änderung der Ausgabe relativ zur Änderung der Eingabe.

$$\forall u \in U_{\text{in}} : \quad s(u) = \frac{1}{|L_{\text{fixed}}|} \sum_{l \in L_{\text{fixed}}} \sum_{v \in U_{\text{out}}} \frac{\partial \text{out}_v^{(l)}}{\partial \text{ex}_u^{(l)}}.$$

Formale Herleitung: Wende Kettenregel an.

$$\frac{\partial \text{out}_v}{\partial \text{ex}_u} = \frac{\partial \text{out}_v}{\partial \text{out}_u} \frac{\partial \text{out}_u}{\partial \text{ex}_u} = \frac{\partial \text{out}_v}{\partial \text{net}_v} \frac{\partial \text{net}_v}{\partial \text{out}_u} \frac{\partial \text{out}_u}{\partial \text{ex}_u}.$$

Vereinfachung: Nimm an, dass die Ausgabefunktion die Identität ist.

$$\frac{\partial \text{out}_u}{\partial \text{ex}_u} = 1.$$

Sensitivitätsanalyse

Für den zweiten Faktor bekommen wir das allgemeine Ergebnis:

$$\frac{\partial \text{net}_v}{\partial \text{out}_u} = \frac{\partial}{\partial \text{out}_u} \sum_{p \in \text{pred}(v)} w_{vp} \text{out}_p = \sum_{p \in \text{pred}(v)} w_{vp} \frac{\partial \text{out}_p}{\partial \text{out}_u}.$$

Das führt zur Rekursionsformel

$$\frac{\partial \text{out}_v}{\partial \text{out}_u} = \frac{\partial \text{out}_v}{\partial \text{net}_v} \frac{\partial \text{net}_v}{\partial \text{out}_u} = \frac{\partial \text{out}_v}{\partial \text{net}_v} \sum_{p \in \text{pred}(v)} w_{vp} \frac{\partial \text{out}_p}{\partial \text{out}_u}.$$

Aber für die erste versteckte Schicht bekommen wir

$$\frac{\partial \text{net}_v}{\partial \text{out}_u} = w_{vu}, \quad \text{therefore} \quad \frac{\partial \text{out}_v}{\partial \text{out}_u} = \frac{\partial \text{out}_v}{\partial \text{net}_v} w_{vu}.$$

Diese Formel stellt den Beginn der Rekursion dar.

Sensitivitätsanalyse

Betrachte (wie üblich) den Spezialfall, bei dem

- die Ausgabefunktion die Identität ist
- und die Aktivierungsfunktion logistisch ist.

In diesem Fall lautet die Rekursionsformel

$$\frac{\partial \text{out}_v}{\partial \text{out}_u} = \text{out}_v(1 - \text{out}_v) \sum_{p \in \text{pred}(v)} w_{vp} \frac{\partial \text{out}_p}{\partial \text{out}_u}$$

und der Anker der Rekursion ist

$$\frac{\partial \text{out}_v}{\partial \text{out}_u} = \text{out}_v(1 - \text{out}_v)w_{vu}.$$