



Deep Learning

Einführung - Thema 2

Silas Hoffmann

29. März 2020

Fachhochschule Wedel

Deep Learning

2020-03-29



Deep Learning

Einführung - Thema 2

Silas Hoffmann
29. März 2020
Fachhochschule Wedel

Geschichtliche Entwicklung

McCulloch-Pitts-Neuron

Perceptron

Adeline

convolutionalNN

Aktuelle Entwicklung

Backpropagation

Multilayer Perceptron

Recurrent Neural Network

Inhalt
Geschichtliche Entwicklung
McCulloch-Pitts-Neuron
Perceptron
Adeline
convolutionalNN
Aktuelle Entwicklung
Backpropagation
Multilayer Perceptron
Recurrent Neural Network

2020-03-29

Deep Learning

└─ Geschichtliche Entwicklung

└─ McCulloch-Pitts-Neuron

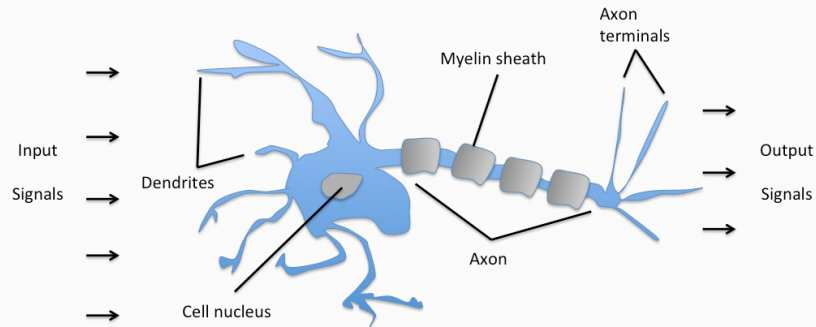
Geschichtliche Entwicklung

McCulloch-Pitts-Neuron

Geschichtliche Entwicklung

McCulloch-Pitts-Neuron

Zusammenhang - Biologisches Neuron



Schematic of a biological neuron.

2020-03-29

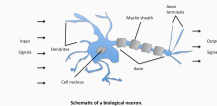
Deep Learning

└─ Geschichtliche Entwicklung

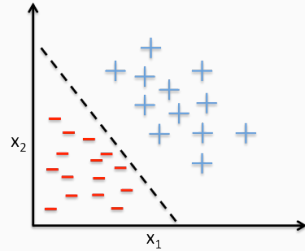
└─ McCulloch-Pitts-Neuron

└─ Zusammenhang - Biologisches Neuron

Zusammenhang - Biologisches Neuron



- Modell soll Funktionalität des biologischen Neurons imitieren
- Klassifizierungsproblem als grundlegende Problemstellung
- Lineare Entscheidungsfunktion zur binären Klassifizierung verwendet



Example of a linear decision boundary for binary classification.

2020-03-29

Deep Learning

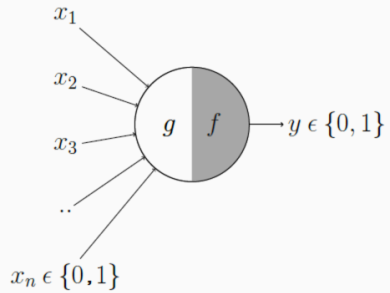
└─ Geschichtliche Entwicklung

└─ McCulloch-Pitts-Neuron

└─ MP-Neuron

- Modell soll Funktionalität des biologischen Neurons imitieren
- Klassifizierungsproblem als grundlegende Problemstellung
- Lineare Entscheidungsfunktion zur binären Klassifizierung verwendet





$$g(x_1, x_2, \dots, x_n) = g(x) = \sum_{i=1}^n x_i$$
$$f(g(x)) = \begin{cases} 1 & \text{if } g(x) \geq \theta \\ 0 & \text{if } g(x) < \theta \end{cases}$$

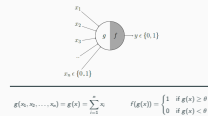
2020-03-29

Deep Learning

└─ Geschichtliche Entwicklung

└─ McCulloch-Pitts-Neuron

└─ Aufbau und Funktionsweise

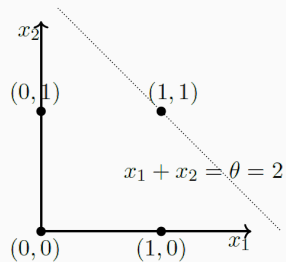


Notation AND-Gatter



AND function

$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 2$$



2020-03-29

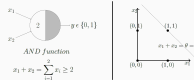
Deep Learning

└ Geschichtliche Entwicklung

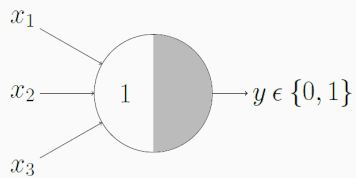
└ McCulloch-Pitts-Neuron

└ Notation AND-Gatter

Notation AND-Gatter

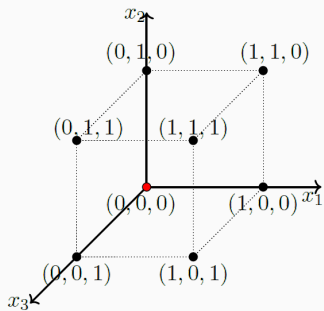


Notation OR-Gatter



OR function

$$x_1 + x_2 + x_3 = \sum_{i=1}^3 x_i \geq 1$$



Deep Learning

└ Geschichtliche Entwicklung

└ McCulloch-Pitts-Neuron

└ Notation OR-Gatter

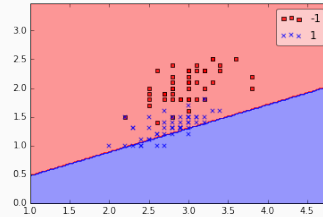
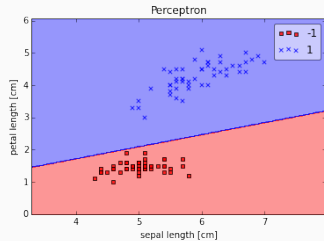
2020-03-29

Notation OR-Gatter



Nachteile

- Keine kontinuierlichen Eingabewerte (nur boolesche Werte)
- Schwelle muss manuell gesetzt werden, keine automatische Aktualisierung vorgesehen
- Keine Priorisierungsmöglichkeit der Eingabewerte möglich
- Funktionen müssen durch lineare Entscheidungsfunktion getrennt werden können



2020-03-29

Deep Learning

- └ Geschichte Entwicklung
 - └ McCulloch-Pitts-Neuron
 - └ Nachteile

Nachteile

- Keine kontinuierlichen Eingabewerte (nur boolesche Werte)
- Schwelle muss manuell gesetzt werden, keine automatische Aktualisierung vorgesehen
- Keine Priorisierungsmöglichkeit der Eingabewerte möglich
- Funktionen müssen durch lineare Entscheidungsfunktion getrennt werden können



Geschichtliche Entwicklung

Perceptron

2020-03-29

Deep Learning

└─ Geschichtliche Entwicklung

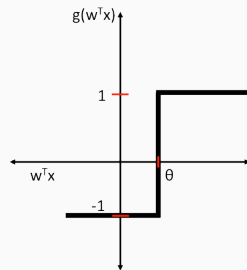
└─ Perceptron

Geschichtliche Entwicklung

Perceptron

Perceptron

- Ähnliche Aktivierungsfunktion wie beim MP-Neuron
- Jedoch gewichtete kontinuierliche Eingabewerte



Unit step function.

$$\mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

$$\begin{aligned} z &= w_1 x_1 + \dots + w_m x_m \\ &= \sum_{j=1}^m x_j w_j \\ &= \mathbf{w}^T \mathbf{x} \end{aligned}$$

2020-03-29

Deep Learning

└ Geschichte der Entwicklung

└└ Perceptron

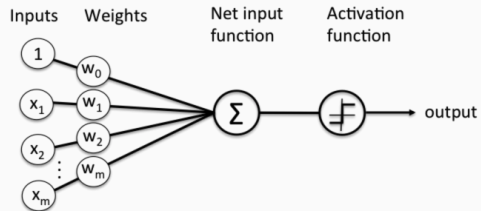
└└└ Perceptron

Perceptron

- Ähnliche Aktivierungsfunktion wie beim MP-Neuron
- Jedoch gewichtete kontinuierliche Eingabewerte



$$\begin{aligned} \mathbf{w} &= \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \\ z &= w_1 x_1 + \dots + w_m x_m \\ &= \sum_{j=1}^m x_j w_j \\ &= \mathbf{w}^T \mathbf{x} \end{aligned}$$



Schematic of Rosenblatt's perceptron.

$$g(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ 1 & \text{if } z > 0 \end{cases}$$

$$\begin{aligned} z &= \mathbf{w_0x_0} + w_1x_1 + \dots + w_mx_m \\ &= \sum_{j=0}^m x_j w_j \\ &= \mathbf{w^T x} \end{aligned}$$

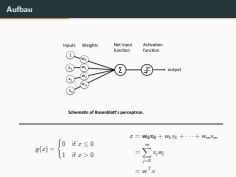
2020-03-29

Deep Learning

└ Geschichtliche Entwicklung

└ Perceptron

└ Aufbau



Lernregel - Ablauf

- Modell übernimmt selbst die Anpassung der Gewichte
- Test mittels einer Menge von gelabelten Trainingsdatensätzen

Grober Ablauf

- Initialisiere die Gewichte mit einem sehr kleinen Wert oder 0.
- Für jeden Datensatz der Menge von Trainingsdatensätzen:
 - Berechne den Ausgabewert des Systems
 - Gleiche die Gewichte an

2020-03-29

Deep Learning

└─ Geschichtliche Entwicklung

└─ Perceptron

└─ Lernregel - Ablauf

Lernregel - Ablauf

- Modell übernimmt selbst die Anpassung der Gewichte
- Test mittels einer Menge von gelabelten Trainingsdatensätzen

Grober Ablauf

- Initialisiere die Gewichte mit einem sehr kleinen Wert oder 0.
- Für jeden Datensatz der Menge von Trainingsdatensätzen:
 - Berechne den Ausgabewert des Systems
 - Gleiche die Gewichte an

Angleichung der Gewichte

- Gewichte komponentenweise angleichen: $w_j := w_j + \Delta w_j$
- Gewichtsänderung: $\Delta w_j = \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_j^{(i)}$
- Beispiel - Iteration mit zweidimensionalem Trainingsvektor:

$$\Delta w_0 = \eta (\text{target}^{(i)} - \text{output}^{(i)})$$

$$\Delta w_1 = \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_1^{(i)}$$

$$\Delta w_2 = \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_2^{(i)}$$

2020-03-29

Deep Learning

└─ Geschichtliche Entwicklung

└─ Perceptron

└─ Lernregel - Formel

Angleichung der Gewichte

- Gewichte komponentenweise angleichen: $w_j := w_j + \Delta w_j$
- Gewichtsänderung: $\Delta w_j = \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_j^{(i)}$

- Beispiel - Iteration mit zweidimensionalem Trainingsvektor:

$$\Delta w_0 = \eta (\text{target}^{(i)} - \text{output}^{(i)})$$

$$\Delta w_1 = \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_1^{(i)}$$

$$\Delta w_2 = \eta (\text{target}^{(i)} - \text{output}^{(i)}) x_2^{(i)}$$

- Trainingsdatensatz richtig erkannt:

$$\Delta w_j = \eta(1^{(i)} - 1^{(i)}) x_j^{(i)} = 0$$

$$\Delta w_j = \eta(1^{(i)} - 1^{(i)}) x_j^{(i)} = 0$$

- Trainingsdatensatz falsch erkannt:

$$\Delta w_j = \eta(1^{(i)} - -1^{(i)}) x_j^{(i)} = \eta(2) x_j^{(i)}$$

$$\Delta w_j = \eta(-1^{(i)} - 1^{(i)}) x_j^{(i)} = \eta(-2) x_j^{(i)}$$

2020-03-29

Deep Learning

└─ Geschichtliche Entwicklung

└─ Perceptron

└─ Lernregel - Trainingsbeispiele

- Trainingsdatensatz richtig erkannt:

$$\Delta w_j = \eta(1^{(i)} - 1^{(i)}) x_j^{(i)} = 0$$

$$\Delta w_j = \eta(1^{(i)} - 1^{(i)}) x_j^{(i)} = 0$$

- Trainingsdatensatz falsch erkannt:

$$\Delta w_j = \eta(1^{(i)} - -1^{(i)}) x_j^{(i)} = \eta(2) x_j^{(i)}$$

$$\Delta w_j = \eta(-1^{(i)} - 1^{(i)}) x_j^{(i)} = \eta(-2) x_j^{(i)}$$

Geschichtliche Entwicklung

Adeline

2020-03-29

Deep Learning

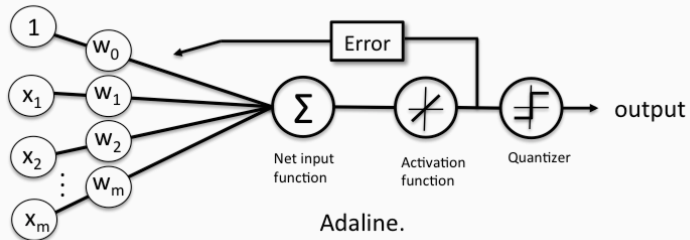
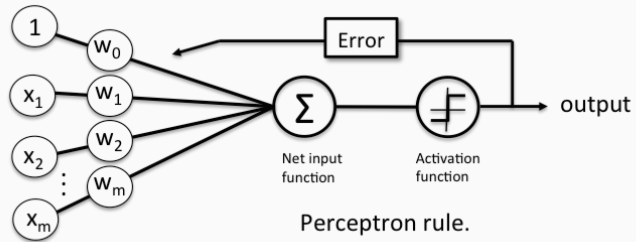
└─ Geschichtliche Entwicklung

└─ Adeline

Geschichtliche Entwicklung

Adeline

ADaptive LINear Element



2020-03-29

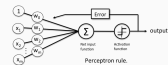
Deep Learning

└─ Geschichtliche Entwicklung

└─ Adeline

└─ ADaptive LINear Element

ADaptive LINear Element



- Lernalgorithmus durch Erfinder geprägt
- auch unter *Least-Mean-Square-Algorithmus* bekannt
- Wesentlicher Vorteil: Ableitbare Kostenfunktion

Notation

$$J(w) = \frac{1}{2} \sum_i (\text{target}^{(i)} - \text{output}^{(i)})^2 \quad \text{output}^{(i)} \in \mathbb{R}$$

2020-03-29

Deep Learning

└─ Geschichtliche Entwicklung

└─ Adeline

└─ Delta-Regel

Delta-Regel

- Lernalgorithmus durch Erfinder geprägt
- auch unter *Least-Mean-Square-Algorithmus* bekannt
- Wesentlicher Vorteil: Ableitbare Kostenfunktion

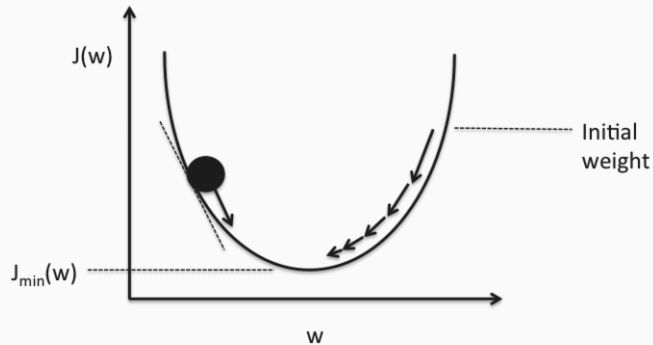
Notation

$$J(w) = \frac{1}{2} \sum_i (\text{target}^{(i)} - \text{output}^{(i)})^2 \quad \text{output}^{(i)} \in \mathbb{R}$$

Gradientenverfahren

- Ziel: Gradientenvektor für bestimmten Input bestimmen:

$$\nabla J \equiv \left(\frac{\partial J}{\partial w_1}, \dots, \frac{\partial J}{\partial w_m} \right)^T.$$



Schematic of gradient descent.

2020-03-29

Deep Learning

└ Geschichtliche Entwicklung

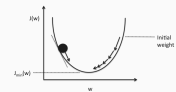
└└ Adeline

└└└ Gradientenverfahren

Gradientenverfahren

- Ziel: Gradientenvektor für bestimmten Input bestimmen:

$$\nabla J \equiv \left(\frac{\partial J}{\partial w_1}, \dots, \frac{\partial J}{\partial w_m} \right)^T.$$



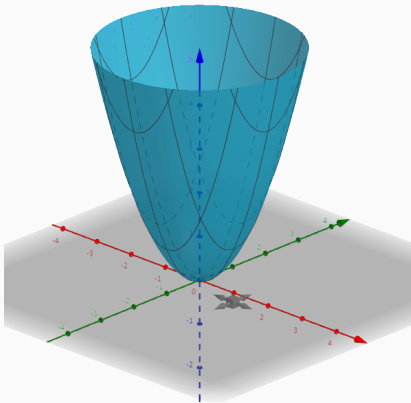
Schematic of gradient descent.

Partielle Ableitungen

- Differenzieren von Funktionen mit mehreren Eingabewerten
- Beispiel: $z = f(x) = x^2 + y^2$

Partielle Ableitung - Notation

$$\frac{\partial \text{AbzuleitendeFkt.}}{\partial \text{BetrachteteKomponente}}$$



2020-03-29

Deep Learning

└ Geschichtliche Entwicklung

└ Adeline

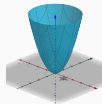
└ Partielle Ableitungen

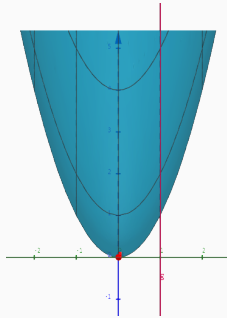
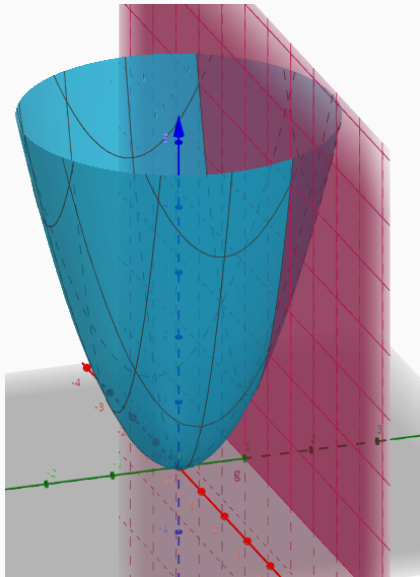
Partielle Ableitungen

- Differenzieren von Funktionen mit mehreren Eingabewerten
- Beispiel: $z = f(x) = x^2 + y^2$

Partielle Ableitung - Notation

$$\frac{\partial \text{AbzuleitendeFkt.}}{\partial \text{BetrachteteKomponente}}$$





Ableitung - Beispiel

$$z = f(x, y) = x^2 + y^2$$

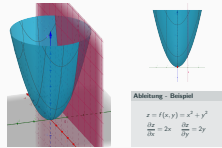
$$\frac{\partial z}{\partial x} = 2x \quad \frac{\partial z}{\partial y} = 2y$$

2020-03-29

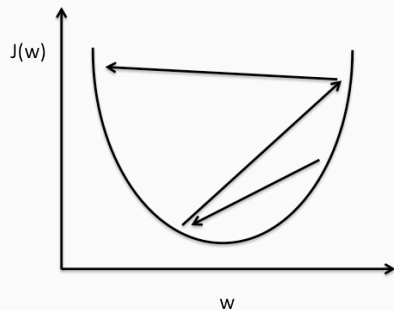
Deep Learning

└ Geschichtliche Entwicklung

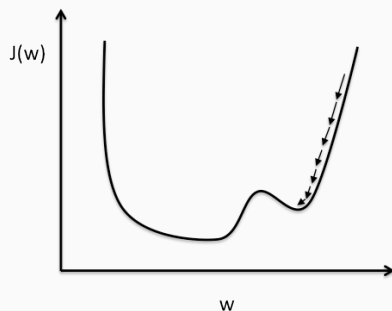
└ Adeline



Gradientenverfahren



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.

2020-03-29

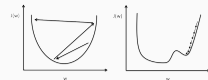
Deep Learning

└ Geschichtliche Entwicklung

└└ Adeline

└└└ Gradientenverfahren

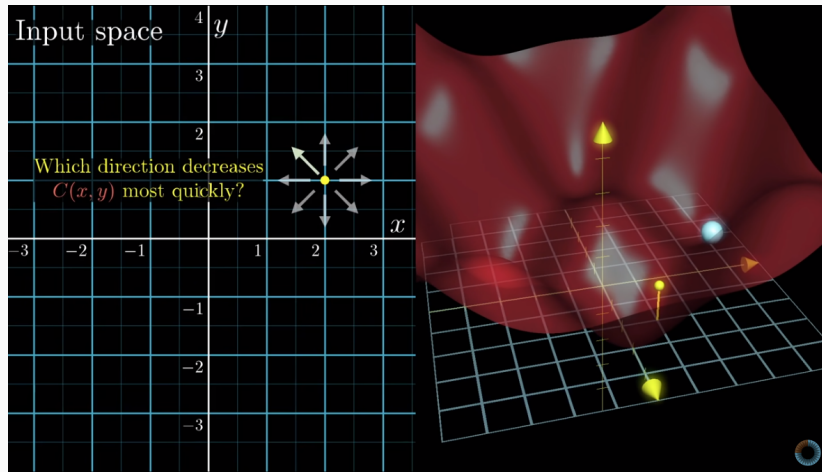
Gradientenverfahren



Large learning rate: Overshooting.

Small learning rate: Many iterations until convergence and trapping in local minima.

Gradientenverfahren



2020-03-29

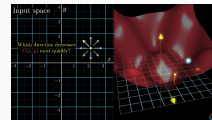
Deep Learning

└─ Geschichtliche Entwicklung

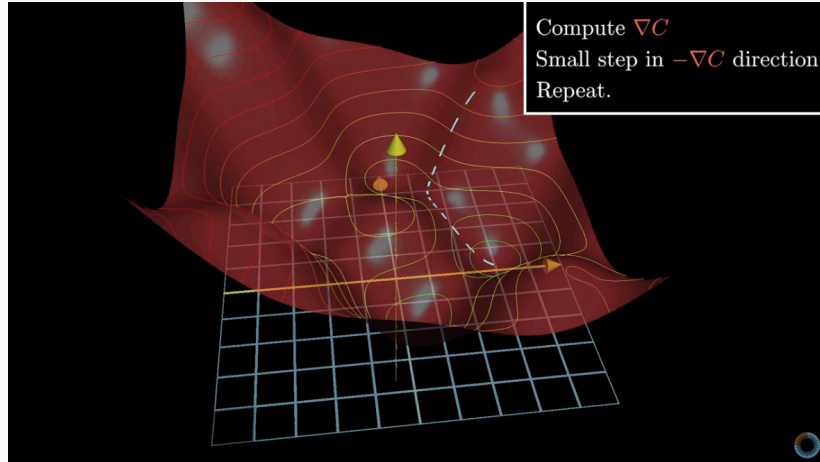
└─ Adeline

└─ Gradientenverfahren

Gradientenverfahren



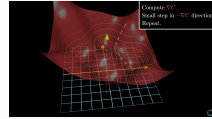
Gradientenverfahren



2020-03-29

Deep Learning
└─ Geschichtliche Entwicklung
 └─ Adeline
 └─ Gradientenverfahren

Gradientenverfahren



Gradientenverfahren - Anwendung

- Gradientenvektor

$$\nabla J \equiv \left(\frac{\partial J}{\partial w_1}, \dots, \frac{\partial J}{\partial w_m} \right)^T.$$

- Allgemein: Vektorielle Darstellung

$$\Delta w = -\eta \nabla J(w)$$

- Für die jeweiligen Gewichte: Komponentenweise Darstellung

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j}$$

- Angleichung der Gewichte $w = w + \Delta w$

2020-03-29

Deep Learning

└─ Geschichtliche Entwicklung

└─ Adeline

└─ Gradientenverfahren

Gradientenverfahren - Anwendung

- Gradientenvektor

$$\nabla J \equiv \left(\frac{\partial J}{\partial w_1}, \dots, \frac{\partial J}{\partial w_m} \right)^T.$$

- Allgemein: Vektorielle Darstellung

$$\Delta w = -\eta \nabla J(w)$$

- Für die jeweiligen Gewichte: Komponentenweise Darstellung

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j}$$

- Angleichung der Gewichte $w = w + \Delta w$

Kostenfunktion ableiten

$$\begin{aligned}\frac{\partial J}{\partial w_j} &= \frac{\partial}{\partial w_j} \frac{1}{2} \sum_i (t^{(i)} - o^{(i)})^2 \\ &= \frac{1}{2} \sum_i \frac{\partial}{\partial w_j} (t^{(i)} - o^{(i)})^2 \\ &= \frac{1}{2} \sum_i 2(t^{(i)} - o^{(i)}) \frac{\partial}{\partial w_j} (t^{(i)} - o^{(i)}) \\ &= \sum_i (t^{(i)} - o^{(i)}) \frac{\partial}{\partial w_j} \left(t^{(i)} - \sum_j w_j x_j^{(i)} \right) \\ &= \sum_i (t^{(i)} - o^{(i)}) (-x_j^{(i)})\end{aligned}$$

2020-03-29

Deep Learning

└─ Geschichtliche Entwicklung

└─ Adeline

└─ Kostenfunktion ableiten

Kostenfunktion ableiten

$$\begin{aligned}\frac{\partial J}{\partial w_j} &= \frac{\partial}{\partial w_j} \frac{1}{2} \sum_i (t^{(i)} - o^{(i)})^2 \\ &= \frac{1}{2} \sum_i \frac{\partial}{\partial w_j} (t^{(i)} - o^{(i)})^2 \\ &= \frac{1}{2} \sum_i 2(t^{(i)} - o^{(i)}) \frac{\partial}{\partial w_j} (t^{(i)} - o^{(i)}) \\ &= \sum_i (t^{(i)} - o^{(i)}) \frac{\partial}{\partial w_j} \left(t^{(i)} - \sum_j w_j x_j^{(i)} \right) \\ &= \sum_i (t^{(i)} - o^{(i)}) (-x_j^{(i)})\end{aligned}$$

Geschichtliche Entwicklung

convolutionalNN

2020-03-29

Deep Learning

└─ Geschichtliche Entwicklung

└─ convolutionalNN

Geschichtliche Entwicklung
convolutionalNN

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

2020-03-29

Deep Learning

└─ Geschichtliche Entwicklung

└─ convolutionalNN

└─ convolutionalNN

convolutionalNN

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

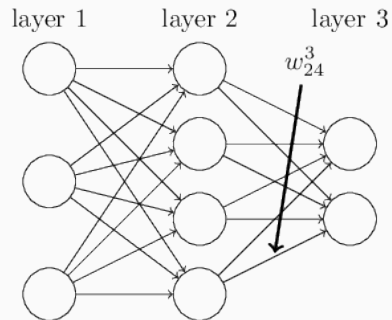
Aktuelle Entwicklung

Backpropagation

2020-03-29

Deep Learning
└ Aktuelle Entwicklung
└ Backpropagation

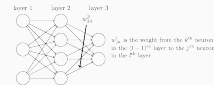
Aktuelle Entwicklung
Backpropagation



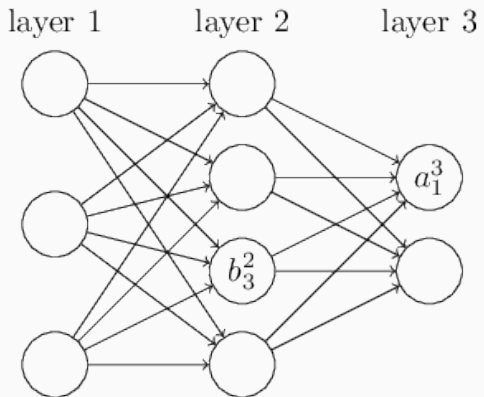
w_{jk}^l is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer

2020-03-29

Deep Learning
└ Aktuelle Entwicklung
└ Backpropagation
└ Notation



1. l: Exponent, steht für die Schicht
2. l - 1, weil man stets von hinten nach vorne schaut
3. Eingabe wird auch als eigene Schicht verstanden
4. j: Index Zielneuron
5. k: Index Startneuron



$$a_j^l = \sigma \left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l \right) \Rightarrow \begin{aligned} a^l &= \sigma(z^l) \\ z^l &= w^l a^{l-1} + b^l \end{aligned}$$

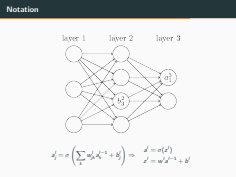
2020-03-29

Deep Learning

└ Aktuelle Entwicklung

└ Backpropagation

└ Notation



1. Ähnlich zu Gewichtsnotation
2. l bezieht sich hierbei jedoch auf aktuelle Schicht
3. j wie gehabt Index in Schicht
4. Notation gilt auch für Aktivierung a
5. Wichtig: σ bezieht sich auf Vektor \Rightarrow Vektorielle Funktion
6. Jede Komponente einzeln mit σ verarbeitet
7. Abstraktion vom Ausgabewert vor der Aktivierungsfkt. hilft später beim Ableiten

- Kostenfunktion soll minimiert werden
- Ziel: Optimale Gewichte und Schwellwerte finden
- Grobe Vorgehensweise: Iterativer Prozess
 - Fehlervektor der letzten Schicht berechnen
 - Fehler schichtweise zum Eingabelayer zurückführen
 - Parameter schichtweise nach Gradienten angleichen

2020-03-29

Deep Learning
└─ Aktuelle Entwicklung
 └─ Backpropagation
 └─ Backpropagation

Backpropagation

- Kostenfunktion soll minimiert werden
- Ziel: Optimale Gewichte und Schwellwerte finden
- Grobe Vorgehensweise: Iterativer Prozess
 - Fehlervektor der letzten Schicht berechnen
 - Fehler schichtweise zum Eingabelayer zurückführen
 - Parameter schichtweise nach Gradienten angleichen

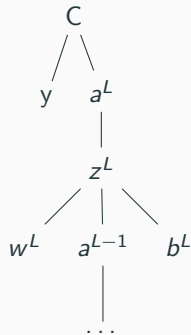
1. Kostenfunktion wie bei Gradientenabstieg / Adeline
2. Unterschied: Hier mehrschichtiges Netz
3. 1970er entwickelt, 1986 von Rumelhart, Hinton und Williams in Paper bekannt gemacht
4. Gradientenabstieg grob erläutert, ausgeblieben - Anwendung im mehrschichtigen Netz und mehrdimensionale Kostenfunktion
5. Fehlervektor der letzten Schicht berechnen
6. Fehler schichtweise zum Eingabelayer zurückführen
7. Parameter schichtweise nach Gradienten angleichen

Fehler - Ausgabeschicht

$$\begin{aligned}\delta_j^L &= \frac{\partial C}{\partial z_j^L} \\ &= \sum_k \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L} \\ &= \frac{\partial C}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} \\ &= \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)\end{aligned}$$

Anmerkung: Kettenregel

$$\frac{d}{dx} [f(u)] = \frac{d}{du} [f(u)] \frac{du}{dx}$$



- **C**: Kostenfunktion
- **y**: Erwartete Ausgabe

2020-03-29

Deep Learning

- └ Aktuelle Entwicklung
 - └ Backpropagation
 - └ Fehler - Ausgabeschicht

Fehler - Ausgabeschicht

$$\begin{aligned}\delta_j^L &= \frac{\partial C}{\partial z_j^L} \\ &= \sum_k \frac{\partial C}{\partial a_k^L} \frac{\partial a_k^L}{\partial z_j^L} \\ &= \frac{\partial C}{\partial a_j^L} \frac{\partial a_j^L}{\partial z_j^L} \\ &= \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)\end{aligned}$$

Anmerkung: Kettenregel

$$\frac{d}{dx} [f(u)] = \frac{d}{du} [f(u)] \frac{du}{dx}$$



- **C**: Kostenfunktion
- **y**: Erwartete Ausgabe

1. Baum nur für Netz mit einer einzigen Aktivierung
2. Zusammenhang mit Kettenregel erläutern
3. Großes L immer für Ausgabeschicht

Fehler - Ausgabeschicht

2020-03-29

Deep Learning

└ Aktuelle Entwicklung

└└ Backpropagation

└└└ Fehler - Ausgabeschicht

Zusammenfassung

- Um den Fehlervektor der letzten Schicht zu bestimmen:

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

- Äquivalent zu:

$$\delta^L = (a^L - y) \odot \sigma'(z^L)$$

- Um die Fehler komponentenweise zu bestimmen:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)$$

- $\nabla_a C$ entspricht dabei Vektor aller $\frac{\partial C}{\partial a_j^L}$ einer Schicht
- \odot : Komponentenweise Multiplikation zweier Vektoren

Zusammenfassung

- Um den Fehlervektor der letzten Schicht zu bestimmen:

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

- Äquivalent zu:

$$\delta^L = (a^L - y) \odot \sigma'(z^L)$$

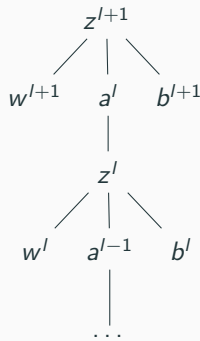
- Um die Fehler komponentenweise zu bestimmen:

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)$$

Fehler - Zwischenschicht

- Zusammenhang zwischen Fehler zweier Schichten herleiten
- Es gilt: $\delta_j^l = \partial C / \partial z_j^l$ sowie $\delta_k^{l+1} = \partial C / \partial z_k^{l+1}$

$$\begin{aligned}\delta_j^l &= \frac{\partial C}{\partial z_j^l} \\ &= \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} \\ &= \sum_k \frac{\partial z_k^{l+1}}{\partial z_j^l} \delta_k^{l+1}\end{aligned}$$



2020-03-29

Deep Learning

└ Aktuelle Entwicklung

└ Backpropagation

└ Fehler - Zwischenschicht

• Zusammenhang zwischen Fehler zweier Schichten herleiten

• Es gilt: $\delta_j^l = \partial C / \partial z_j^l$ sowie $\delta_k^{l+1} = \partial C / \partial z_k^{l+1}$



$$z_k^{l+1} = \sum_j w_{kj}^{l+1} a_j^l + b_k^{l+1} = \sum_j w_{kj}^{l+1} \sigma(z_j^l) + b_k^{l+1}$$

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} \sigma'(z_j^l)$$

Zusammenfassung

- Komponentenweise Darstellung: $\delta_j^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1} \sigma'(z_j^l)$
- Vektorielle Darstellung: $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$

2020-03-29

Deep Learning

└ Aktuelle Entwicklung

└ Backpropagation

└ Fehler - Zwischenschicht

Fehler - Zwischenschicht

$$z_k^{l+1} = \sum_j w_{kj}^{l+1} a_j^l + b_k^{l+1} = \sum_j w_{kj}^{l+1} \sigma(z_j^l) + b_k^{l+1}$$
$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} \sigma'(z_j^l)$$

Zusammenfassung

- Komponentenweise Darstellung: $\delta_j^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1} \sigma'(z_j^l)$
- Vektorielle Darstellung: $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$

2020-03-29

Deep Learning
└ Aktuelle Entwicklung
└ Multilayer Perceptron

Aktuelle Entwicklung
Multilayer Perceptron

Aktuelle Entwicklung

Multilayer Perceptron

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

2020-03-29

Deep Learning
└─ Aktuelle Entwicklung
 └─ Multilayer Perceptron
 └─ Multilayer Perceptron

Multilayer Perceptron

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

2020-03-29

Deep Learning

└ Aktuelle Entwicklung

└ Recurrent Neural Network

Aktuelle Entwicklung

Recurrent Neural Network

Aktuelle Entwicklung

Recurrent Neural Network

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Deep Learning

└ Aktuelle Entwicklung

└ Recurrent Neural Network

└ Recurrent Neural Network

2020-03-29

Recurrent Neural Network

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.



Deep Learning

Einführung - Thema 2

Silas Hoffmann

29. März 2020

Fachhochschule Wedel

2020-03-29

Deep Learning

└ Aktuelle Entwicklung

└ Recurrent Neural Network



Deep Learning

Einführung - Thema 2

Silas Hoffmann
29. März 2020
Fachhochschule Wedel

Fragen?

2020-03-29

Deep Learning

└ Aktuelle Entwicklung

└ Recurrent Neural Network

Fragen?

Sometimes, it is useful to add slides at the end of your presentation to refer to during audience questions.

The best way to do this is to include the `appendixnumberbeamer` package in your preamble and call `\appendix` before your backup slides.

metropolis will automatically turn off slide numbering and progress bars for slides in the appendix.

Backup slides

Sometimes, it is useful to add slides at the end of your presentation to refer to during audience questions.

The best way to do this is to include the `appendixnumberbeamer` package in your preamble and call `\appendix` before your backup slides.

metropolis will automatically turn off slide numbering and progress bars for slides in the appendix.



3Blue1Brown - Videokurs zur Einführung in die Neuralen Netze.

https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi.

Aufgerufen am: 16-03-2020.



Übersicht - verschiedene Architekturen.

<https://www.asimovinstitute.org/neural-network-zoo/>.

Aufgerufen am: 22-03-2020.



Definition Klassifizierungsproblem.

http://ekpwww.physik.uni-karlsruhe.de/~tkuhr/HauptseminarWS1112/Keck_handout.pdf.

Aufgerufen am: 15-03-2020.

References



Einführung Convolutional neural network.

<https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/>.

Aufgerufen am: 18-03-2020.



Öffentliche Datensätze - Übersicht.

<https://github.com/awesomedata/awesome-public-datasets>.

Aufgerufen am: 18-03-2020.



Funktionsweise - CNN.

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1890437/>.

Aufgerufen am: 18-03-2020.



Funktionsweise - CNN.

<https://bit.ly/2QGK0Ej>.

Aufgerufen am: 18-03-2020.

References



Geschichte der Convolutional neuronalen Netze.

<https://glassboxmedicine.com/2019/04/13/a-short-history-of-convolutional-neural-networks/>.
Aufgerufen am: 18-03-2020.



Khan Academy - Partielle Ableitungen (Funktion mit zwei Eingabewerten).

<https://www.youtube.com/watch?v=1CMDS4-PKKQ&t=542s>.
Aufgerufen am: 16-03-2020.



Künstliche Neuronale Netzwerke und Deep Learning - Stefan Stelle.

https://www.htwsaar.de/wiwi/fakultaet/personen/profile/selle-stefan/Selle2018e_Kuenstliche_Neuronale_Netzwerke.pdf/at_download/file.
Aufgerufen am: 24-03-2020.



McCulloch-Pitts Neuron.

<https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1>.

Aufgerufen am: 14-03-2020.



Perceptron - Python Implementierung.

<https://github.com/rasbt/mlxtend/blob/master/mlxtend/classifier/perceptron.py>.

Aufgerufen am: 16-03-2020.



Single-Layer Neural Networks and Gradient Descent.

https://sebastianraschka.com/Articles/2015_singlelayer_neurons.html.



Aufgerufen am: 14-03-2020.



M. Nielsen.

Neural Networks and Deep Learning.

Determination Press, 2015.

-  McCulloch-Pitts Neuron.
<https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1>.
Aufgerufen am: 14-03-2020.
-  Perceptron - Python Implementierung.
<https://github.com/rasbt/mlxtend/blob/master/mlxtend/classifier/perceptron.py>.
Aufgerufen am: 16-03-2020.
-  Single-Layer Neural Networks and Gradient Descent.
https://sebastianraschka.com/Articles/2015_singlelayer_neurons.html.
Aufgerufen am: 14-03-2020.
-  M. Nielsen.
Neural Networks and Deep Learning.
Determination Press, 2015.