

Algorithmik zur Optimierung in neuronalen Netzwerken

Gradient Descent und Backpropagation

Tim Hilt

Date: tbd

Hochschule Esslingen — University of Applied Sciences

Training

- Loss-Funktion

- Gradient Descent

- Backpropagation

Training

Loss-Funktion/ Cost-Funktion

- Dient zur Berechnung des Fehlers während dem Training
- Trainingsfehler soll minimiert werden
- \Rightarrow wir suchen den Punkt, an dem die Ableitung der Loss-Funktion 0 wird, der Fehler also nicht mehr abnimmt
- Es gibt eine Vielzahl an Loss- oder Cost-Funktionen, wir betrachten hier die „Quadratic Cost/ Mean Squared Error (MSE)“:

$$C(w, b) = \frac{1}{2n} \sum_x ||y(x) - a||^2$$

Loss-Funktion/ Cost-Funktion

- Dient zur Berechnung des Fehlers während dem Training
- Trainingsfehler soll minimiert werden
- \Rightarrow wir suchen den Punkt, an dem die Ableitung der Loss-Funktion 0 wird, der Fehler also nicht mehr abnimmt
- Es gibt eine Vielzahl an Loss- oder Cost-Funktionen, wir betrachten hier die „Quadratic Cost/ Mean Squared Error (MSE)“:

$$C(w, b) = \frac{1}{2n} \sum_x ||y(x) - a||^2$$

$C(w, b)$	Cost in Abhängigkeit von w und b
n	Anzahl der Trainingsinstanzen
$y(x)$	Label wenn x Input ist
a	Output des Netzwerkes, bei w und b

Backpropagation

Es werden vier Gleichungen benötigt:

Error im Output-Layer:

Error einzelner Neuronen:

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

Vektorisiert:

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

Aufgelöst, wenn MSE benutzt:

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

Error im Layer l hinsichtlich Error im nächsten Layer δ^{l+1}

$$\delta^l = \left((w^{l+1})^T \delta^{l+1} \right) \odot \sigma' (z^l)$$

- Rekursive Definition durch Verwendung von δ^l in Abhängigkeit von δ^{l+1}
- Wenn anfangs δ^L in die Gleichung gegeben wird kann der Error rekursiv für jeden vorhergehenden Layer berechnet werden

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

$$\delta^l = \left((w^{l+1})^T \delta^{l+1} \right) \odot \sigma' (z^l)$$





$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$$

Pass

Fragen?

Literaturverzeichnis

-  GÉRON, Aurélien. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.
-  BURKOV, Andriy. *The hundred-page machine learning book*. Andriy Burkov Quebec City, Can., 2019.
-  LECUN, Yann; BOTTOU, Léon; BENGIO, Yoshua; HAFFNER, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 1998, Jg. 86, Nr. 11, S. 2278–2324.
-  XIAO, Han; RASUL, Kashif; VOLLGRAF, Roland. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*. 2017.