Data Mining und Maschinelles Lernen

Prof. Kristian Kersting
ZhongjieYu
Johannes Czech

TECHNISCHE UNIVERSITÄT DARMSTADT

Sommersemester 2021
8. Juli 2022
Übungsblatt 9

Diese Übung wird am **01.07.2020** um **13:30 Uhr** besprochen und **nicht** bewertet.

9.1 Backpropagation und Stochastic Gradient Descent

Gehen Sie von folgendem Netzwerk aus, wie in Abbildung 9.1 zu sehen. Verwenden Sie dabei die Identitätsfunktion $f(x) = x$ als Aktivierungsfunktion. Zur Vereinfachung verwendet das Netzwerk keine Bias-Parameter. Nehmen Sie für den aktuellen Zielausgabewert $y^* = 0.5$ und eine Definition der Fehlerfunktion von $L(W) = \frac{1}{2} \|g - y^*\|^2$ mit einer Lernrate von $\alpha = 0.5$ an.

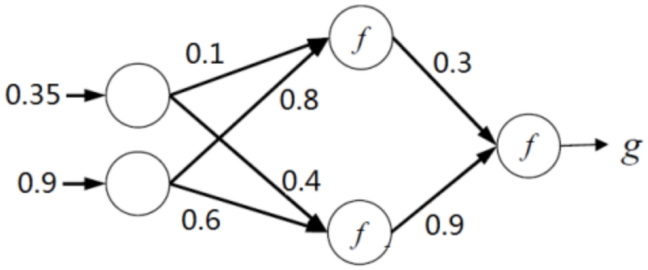

Abbildung 1: Netzsstruktur eines „Multi-Layer Perceptron“

Die Eingabe und Gewichte werden wie folgt beschrieben:
$x_1 = 0.35$, $x_2 = 0.9$, $w_1 = 0.1$, $w_2 = 0.8$, $w_3 = 0.4$, $w_4 = 0.6$, $w_5 = 0.3$, $w_6 = 0.9$.

a) **Backpropagation**
Wenden Sie Backpropagation an, um die Gewichte w_1 bis w_6 zu aktualisieren.

1. Forward Pass

$$h_1 = x_1 \cdot w_1 + x_2 \cdot w_2 = 0.755$$
$$h_2 = x_1 \cdot w_3 + x_2 \cdot w_4 = 0.68$$
$$o_1 = f(h_1) = 0.755$$
$$o_2 = f(h_2) = 0.68$$
$$h_3 = o_1 \cdot w_5 + o_2 \cdot w_6 = 0.8385$$
$$g = f(h_3) = 0.8385$$

2. Fehler berechnen

$$L(W) = \frac{1}{2} \|g - y^*\|^2 = \frac{1}{2} \|0.8385 - 0.5\|^2$$

3. Backprobagation

Gradient Descent für Gewichte anwenden

$$w_i^{new} = w_i - \alpha \cdot \frac{\partial L}{\partial w_i}$$

$$w_5^{new} = w_5 - \alpha \cdot \frac{\partial L}{\partial g} \cdot \frac{\partial g}{\partial w_5} = w_5 - \alpha \cdot (g - y^*) \cdot h_1$$

Ähnlich verhält es sich mit der Aktualisierung von w_6:

$$w_6^{new} = w_6 - \alpha \cdot \frac{\partial L}{\partial g} \cdot \frac{\partial g}{\partial w_6} = w_6 - \alpha \cdot (g - y^*) \cdot h_2$$

anschließend,

$$w_1^{new} = w_1 - \alpha \cdot \frac{\partial L}{\partial g} \cdot \frac{\partial g}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1} = w_1 - \alpha \cdot (g - y^*) \cdot w_5 \cdot x_1$$
$$w_2^{new} = w_2 - \alpha \cdot (g - y^*) \cdot w_5 \cdot x_2$$
$$w_3^{new} = w_3 - \alpha \cdot (g - y^*) \cdot w_6 \cdot x_1$$
$$w_4^{new} = w_4 - \alpha \cdot (g - y^*) \cdot w_6 \cdot x_2$$

4. Backward Pass

$$w_5^{new} = w_5 - \alpha \cdot (g - y^*) \cdot h_1 = 0.3 - 0.5 \cdot (0.8385 - 0.5) \cdot 0.755 = 0.17221625$$
$$w_6^{new} = w_6 - \alpha \cdot (g - y^*) \cdot h_2 = 0.9 - 0.5 \cdot (0.8385 - 0.5) \cdot 0.68 = 0.78491$$
$$w_1^{new} = w_1 - \alpha \cdot (g - y^*) \cdot w_5 \cdot x_1 = 0.1 - 0.5 \cdot (0.8385 - 0.5) \cdot 0.3 \cdot 0.35 = 0.08222875$$
$$w_2^{new} = w_2 - \alpha \cdot (g - y^*) \cdot w_5 \cdot x_2 = 0.8 - 0.5 \cdot (0.8385 - 0.5) \cdot 0.3 \cdot 0.9 = 0.7543025$$
$$w_3^{new} = w_3 - \alpha \cdot (g - y^*) \cdot w_6 \cdot x_1 = 0.4 - 0.5 \cdot (0.8385 - 0.5) \cdot 0.9 \cdot 0.35 = 0.34668625$$
$$w_4^{new} = w_4 - \alpha \cdot (g - y^*) \cdot w_6 \cdot x_2 = 0.6 - 0.5 \cdot (0.8385 - 0.5) \cdot 0.9 \cdot 0.9 = 0.4629075$$

5. Forward Pass

$$h_1 = x_1 \cdot w_1^{new} + x_2 \cdot w_2^{new} = 0.70765$$
$$h_2 = x_1 \cdot w_3^{new} + x_2 \cdot w_4^{new} = 0.53796$$
$$o_1 = f(h_1) = 0.70765$$
$$o_2 = f(h_2) = 0.53796$$
$$h_3 = o_1 \cdot w_5^{new} + o_2 \cdot w_6^{new} = 0.54412$$
$$g = f(h_3) = 0.54412$$

Der aktualisierte Output liegt näher an dem Zielwert.

b) **Stochastic Gradient Descent (SGD)**

Beschreiben Sie SGD und seine Beziehung zu Backpropagation.

1. Parameterinitialisierung
Zufällige Initialisierung der Parameter z. B. anhand von **Xavier Intialisierung** [1].

Alle Parameter werden zufällig anhand der Gleichverteilung $\mathcal{U}(-a, a)$ gesetzt, wobei a:

$$a = \text{gain} \times \sqrt{\frac{6}{\text{fan\_in} + \text{fan\_out}}}$$

gain ein Skalierungsfaktor, der von der Aktivierungsfunktion abhängt, und fan_in die Anzahl eingehender und fan_out die Anzahl ausgehender Verbindungen.

Alternativ kann z. B. **Kaiming Initialisierung** [2] verwendet werden. Hier werden alle Parameter zufällig nach der Gleichverteilung $\mathcal{U}(-a, a)$ bestimmt, wobei

$$a = \text{gain} \times \sqrt{\frac{3}{\text{fan\_mode}}}$$

und fan_mode entweder fan_in oder fan_out.

2. Iteration über Datensatz
*Iteriere eine Anzahl von **Epochen**, über den gesamten Trainingsdatensatz. Wenn alle Datenpunkte des Datensatzes abgearbeitet wurden spricht man von einer Epoche.*

*- Stochastic Gradient Descent: Ziehe **zufällig** einen Datenpunkt.*

- Batch Gradient Descent: Verwende den gesamten Datensatz als Batch.

*- Mini-batch Stochastic Gradient Descent: Ziehe **zufällige** Teilmenge des gesamten Datensatzes.*

3. Forward Pass
Berechnung der Aktivierungen von Eingangschicht bis zur Ausgangschicht.

4. Berechnung des Fehlers
Bestimmung des Fehlers anhand der verwendeten Fehlerfunktion, z.B. MSE:

$$L(W) = \frac{1}{2}(y\_predicted - y\_true)^2$$

5. Backpropagation des Fehlers
Melde den Fehler zurück, d.h., berechne den Gradienten für jede Schicht von Ausgangschicht zur Eingangschicht.

6. Anpassung der Gewichte/Parameter mittels Gradient Descent
Update die Parameter entgegen ihres Gradienten:

$$w_i^{new} = w_i - \alpha \cdot \frac{\partial L}{\partial w_i}.$$

c) **Backpropagation mit Batches**

Wie verhält sich der Fehler, wenn man die Backpropagation statt für einen einzelnen Datenpunkt auf einer Menge von Daten, einem Batch, gleichzeitig durchführt.

Wenn es sich bei der Eingabe um mehrere Datenpunkte handelt (ein Batch) handelt, dann ist die Fehlerfunktion:

$$L(W) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left\|g_i - y_i^*\right\|^2 = \frac{1}{m}\sum_{i=1}^{m} L_i$$

$$w_z^{new} = w_z - \alpha \cdot \frac{\partial L}{\partial w_z}$$

$$\frac{\partial L}{\partial w_z} = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial L_i}{\partial w_z}$$

Die Kettenregel kann erneut angewendet werden.

9.2 2D Convolution

In dieser Übung führen wir eine 2D-Faltung auf dem Bild mit dem gegebenen Filter durch.

Tabelle 1: Matrixdarstellung eines Bildes

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Tabelle 2: Kernel-Filter

1	0	1
0	1	0
1	0	1

a) **Resultierende Größe**

Gehen Sie von quadratischer Eingabegröße und quadratischer Kernelgröße aus. Gegeben ist die Eingabegröße i, Kernelgröße k, Stride s und Zero-Padding p. Was ist die resultierende Größe der Ausgabe o nach Anwendung der Faltungsoperation?

$$o = \left\lfloor \frac{i + 2\,p - k}{s} \right\rfloor + 1 \tag{2}$$

b) **Anwendung I**

Zeigen Sie das Ergebnis der 2D-Faltung, bei $stride = 1$ entlang beider Achsen und ohne $zero - padding$.

4	3	4
2	4	3
2	3	4

Tabelle 3: Resultierende Matrix nach Anwendung der Faltung I

c) **Anwendung II**

Zeigen Sie das Ergebnis der 2D-Faltung, bei $stride = 2$ entlang beider Achsen und $zero - padding = 1 \times 1$.

2	3	1
1	4	3
0	2	1

Tabelle 4: Resultierende Matrix nach Anwendung der Faltung II

9.3 Neuronale Netze

In dieser Übung erforschen wir neuronale Netzwerke mit dem **tensorflow playground**:
https://playground.tensorflow.org

Wir wählen den Spiral-Datensatz für den Problemtyp-Klassifizierung, stellen das Trainings-/Testverhältnis 50 % und wählenden folgende Parametereintellungen: Rauschen 20, Batch Size 10, Lernrate 0.03, Aktivierungsfunktion ReLU, Regularisierung L2, Regularisierungsrate 0.001.

a) **Konfiguration I**

Wählen Sie ein zweischichtiges Modell(4, 2) und verwenden Sie nur die Merkmale X_1 und X_2, was ist der Trainings-/Testverlust nach 200 Epochen?

ca. bei $0.45/0.48$

b) **Konfiguration II**

Wählen Sie ein vierschichtiges Modell (6, 4, 2, 2) und verwenden Sie nur die Merkmale X_1 und X_2, wie hoch ist der Trainings-/Testfehler nach 200/1600 Epochen?

ca. bei $0.5/0.3$

c) **Konfiguration III**

Wählen Sie ein dreischichtiges Modell (4, 4, 2) und nutzen Sie alle Features. Wie hoch ist der Trainings-/Testfehler nach 200 Epochen? Wie ist er bei 500 Epochen?

ca. bei $0.09/0.10$ *und ca. bei* $0.06/0.08$

Literatur

[1] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.