h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbmn
FACHBEREICH MATHEMATIK
UND NATURWISSENSCHAFTEN

Prof. Dr. H. Zisgen
Markus Schepers
Fachbereich Mathematik und Naturwissenschaften
Sommersemester 2020

Data Mining 1

Pflichtaufgaben – Teil II
- Künstliche Neuronale Netze -

Aufgabe 1 (Neuronale Netze mit Numpy)
Im Moodlekurs im Ordner "Python-Beispiel: Neuronale Netze" können Sie das Jupyter-Notebook `example_neuralnet.ipynb` (auch als HTML-Export) finden.

a) Das Modell aus dem Beispiel soll um ein Hidden-Layer(Layer1) mit m Neuronen erweitert werden. Leiten Sie die Backward Propagation (also die mathematische Beschreibung) auf der Basis des unten aufgeführten Modells (mathematische Beschreibung der Erweiterung um einen zweiten Layer inkl. der zugehörigen Forward-Propagation) her. Implementieren Sie diese Backward Propagation im Beispiel. Diese sollte für eine effiziente Berechnung mit Vektor-Matrix-Multiplikation durchgeführt werden. Es folgt das neue (erweiterte) Modell ohne Backward-Propagation:

Forward-Propagation:

Layer 1 (Hidden-Layer):
$n \in \mathbb{N}$ sei die Anzahl an Beobachtungen in der Stichprobe und $m \in \mathbb{N}$ sei die Anzahl an Neuronen im ersten Layer.

$$\begin{aligned} z_1 &= X \cdot W_1 + b_1 \\ &= \begin{pmatrix} x_{11} & x_{12} \\ \vdots & \vdots \\ x_{n1} & x_{n2} \end{pmatrix} \cdot \begin{pmatrix} w_{1;11} & \dots & w_{1;1m} \\ w_{1;21} & \dots & w_{1;2m} \end{pmatrix} + \begin{pmatrix} b_{1;1} & \dots & b_{1;m} \\ \vdots & \ddots & \vdots \\ b_{1;1} & \dots & b_{1;m} \end{pmatrix} \\ &= \begin{pmatrix} z_{1;11} & \dots & z_{1;1m} \\ \vdots & \ddots & \vdots \\ z_{1;n1} & \dots & z_{1;nm} \end{pmatrix} \end{aligned}$$

Sigmoid-Funktion: $\sigma \colon \mathbb{R} \ni x \mapsto \dfrac{1}{1+\exp(-x)} \in (0;\ 1) \subset \mathbb{R}$

$$
\begin{aligned}
a_1 &= \sigma\left(z_1\right) \\
&= \begin{pmatrix} \sigma\left(z_{1;11}\right) & \dots & \sigma\left(z_{1;1m}\right) \\ \vdots & \ddots & \vdots \\ \sigma\left(z_{1;n1}\right) & \dots & \sigma\left(z_{1;nm}\right) \end{pmatrix} \\
&= \begin{pmatrix} a_{1;11} & \dots & a_{1;1m} \\ \vdots & \ddots & \vdots \\ a_{1;n1} & \dots & a_{1;nm} \end{pmatrix}
\end{aligned}
$$

mit $X \in \mathbb{R}^{n\times 2}$, $W_1 \in \mathbb{R}^{2\times m}$, $b_1 \in \mathbb{R}^{n\times m}$ und $z_1, a_1 \in \mathbb{R}^{n\times m}$.

Layer 2 (Output-Layer):

$$
\begin{aligned}
z_2 &= a_1 \cdot W_2 + b_2 \\
&= \begin{pmatrix} a_{1;11} & \dots & a_{1;1m} \\ \vdots & \ddots & \vdots \\ a_{1;n1} & \dots & a_{1;nm} \end{pmatrix} \cdot \begin{pmatrix} w_{2;11} \\ \vdots \\ w_{2;m1} \end{pmatrix} + \begin{pmatrix} b_{2;1} \\ \vdots \\ b_{2;1} \end{pmatrix} \\
&= \begin{pmatrix} z_{2;11} \\ \vdots \\ z_{2;n1} \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
\widehat{\pi} &= a_2 \\
&= \sigma(z_2) \\
&= \begin{pmatrix} \sigma(z_{2;11}) \\ \vdots \\ \sigma(z_{2;n1}) \end{pmatrix} \\
&= \begin{pmatrix} a_{2;11} \\ \vdots \\ a_{2;n1} \end{pmatrix}
\end{aligned}
$$

mit $W_2 \in \mathbb{R}^m$, $b_2 \in \mathbb{R}^n$ und $z_2, a_2 \in \mathbb{R}^n$.

Backward-Propagation:

...

Kostenfunktion:

$$E\left(y, \widehat{\pi}\right) = \frac{1}{2} \sum_{k=1}^{n} \left(y_k - \widehat{\pi}_k\right)^2$$

Gradient descent parameter update:

$$\begin{aligned} W_1^{\text{neu}} &= W_1^{\text{alt}} - \eta \cdot \mathrm{d}W_1 \\ b_1^{\text{neu}} &= b_1^{\text{alt}} - \eta \cdot \mathrm{d}b_1 \\ W_2^{\text{neu}} &= W_2^{\text{alt}} - \eta \cdot \mathrm{d}W_2 \\ b_2^{\text{neu}} &= b_2^{\text{alt}} - \eta \cdot \mathrm{d}b_2 \end{aligned}$$

mit Lernrate η, $\mathrm{d}W_1 := \frac{\partial E(y,\widehat{\pi})}{\partial W_1}$, $\mathrm{d}b_1 := \frac{\partial E(y,\widehat{\pi})}{\partial b_1}$, $\mathrm{d}W_2 := \frac{\partial E(y,\widehat{\pi})}{\partial W_2}$ und $\mathrm{d}b_2 := \frac{\partial E(y,\widehat{\pi})}{\partial b_2}$.

Hinweis: In Python können b_1 und b_2 durch $b_1 = (b_{1;1}, \ldots, b_{1;m})^T \in \mathbb{R}^m$ und $b_2 = b_{2;1} \in \mathbb{R}$ vereinfacht dargestellt werden. Hierbei muss auf die Dimensionen aufgepasst werden, siehe Broadcasting (Hyperlink).

Bearbeitungshinweis: Erläutern Sie Ihr vorgehen im Booklet (mathematische Vorgehensweise!) und fügen Sie die Implementierung in den Anhang.

b) Variieren Sie folgende Einstellungen (einzeln) und diskutieren Sie die Ergebnisse:

i) Anzahl der Neuronen im Hidden Layer

ii) Verringerung und Vergrößerung der Iterationen des Gradientenverfahrens (`iterations`), ausgehend von 1000 Iterationen.

iii) Lernrate (`learning-rate`)

iv) Initialisierung der Gewichte

- Initialisieren Sie einmal alle Gewichte mit null und probieren andere Initialisierungen mit Werten ungleich null
- Interpretieren Sie die Bedeutung des Seed-Wertes bei der Gewichtswahl

Aufgabe 2 (Neuronale Netze mit Keras)
In dieser Aufgabe soll der australische Wetterdatensatz verwendet werden (Aufteilung wie in Booklet-Aufgabenteil I). Nutzen Sie die deep learning API `Keras`, welche in der Python-Bibliothek `TensorFlow` integriert ist. Ziel soll es sein ein möglichst gutes Modell aufzustellen.

Erläutern Sie welche Mittel und Methoden Sie verwendet haben, um Ihr "Sieger-Modell" anzupassen und anhand welcher Kriterien Sie sich für dieses Modell entschieden haben. Vergleichen Sie Ihre Ergebnisse mit denen von Aufgabenblatt 1.

Hinweise: Im Jupyter-Notebook von Aufgabe 1 können Sie im Abschnitt 4 weitere Informationen zum Umgang mit Keras und weiterführende Links finden.