1163150 - Übung 2

Name, Vorname: _____

Matrikelnummer: _____

- Abgabetermin der Übung ist der 27. Mai 2018
- Elektronische Abgaben erfolgen grundsätzlich über die Moodle-Plattform
- Handgeschriebene Lösungsaufgaben können ins Fach eingeworfen werden (WH C Etage 2)
- Für jeden Tag nach Abgabefrist werden 5 Punkte der Maximalpunktzahl abgezogen
- Über alle Übungen hinweg besitzten Sie 3 Bonustage, die Sie für eine verspätete Abgabe ohne Punktabzug verwenden können

Aufgabe:	1	2	3	Summe:
Punkte:	15	10	10	35
Ergebnis:				

1. Linear Classifier

   (a) (15 Punkte) Ihre Aufgabe ist es, einen linearen Klassifikator für den MNIST-Datensatz mit folgender Kostenfunktion basierend auf der Softmax-Funktion und L2-Regularisierung umzusetzen:

$$L = \frac{1}{M}\sum_{i=1}^{M} -log\left(\frac{e^{h(x_j,\Theta)}}{\sum_{k=1}^{K} e^{h(x_k,\Theta)}}\right)_i + \frac{\lambda}{2}\sum \Theta^2, \; wobei \;\; h(X,\Theta) = X * \Theta \tag{1}$$

   Optimieren Sie die Kosten des Klassifikators mit dem Gradientenabstiegsverfahren. Das beigelegte Jupyter-Notebook *softmax_classifier.ipynb* können Sie als Basis verwenden, müssen Sie jedoch nicht. Sie sind jedoch auf die Python importe beschränkte, die auch in diesem Notebook genutzt werden.

   (b) (5 Bonuspunkte) Machen Sie etwas Extra. Sie können bis zu 5 zusätzlichen Punkten erhalten, wenn Sie das Jupyter-Notebook um nicht geforderte Aspekte ergänzen, beispielsweise:

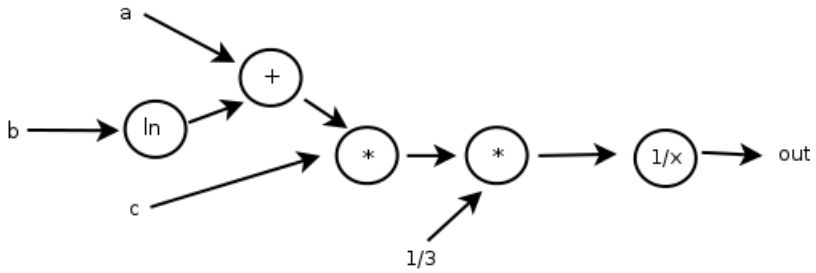
   - Durchführung einer Kreuzvalidierung für Hyperparameter
   - Erweitern der Datenaufteilung in Trainings-, Validierungs- und Testdaten
   - Testen unterschiedlicher Lernraten und Ausgeben dieser in einer Grafik
   - Implementieren des stochastischen Gradientenabstiegsverfahrens (Mini-Batch-Gradient-Descent)
   - Verbessern Sie das Notebook, beispielsweise durch Code-Optimierung, sinnvolle Kommentare, Erweiterung der textuellen Beschreibung(!) oder hilfreiche Ausgaben (print())
   - Werden Sie kreativ und denken Sie sich eine Erweiterung aus

   Die Anzahl der Bonuspunkte wird basierend auf Anspruch and Quantiät der Erweiterungen vom Dozenten vergeben.

2. Backpropagation

   (a) (5 Punkte) Berechnen Sie die partiellen Ableitungen $\frac{\partial out}{\partial a}$, $\frac{\partial out}{\partial b}$ und $\frac{\partial out}{\partial c}$ zu gegebenem 'Computational Graph' mittels Backpropagation. Die Variablen haben dabei folgende Werte:

   - $a = 2$
   - $b = e$ (eulersche Zahl, d.h. $b \approx 2.7183$)
   - $c = 3$



   Hinweise: $\frac{\partial ln(z)}{\partial z} = \frac{1}{z}$, nutzen Sie Brüche.

   (b) (5 Punkte) Entwerfen Sie eine eigene Backpropagation-Übung. Erstellen Sie einen 'Computational Graph' zu dem partielle Ableitungen mittels Backpropagation-Algorithmus berechnet werden sollen. Erstellen Sie eine Musterlösung, die Sie ihren Kommilitonen_innen zur Verfügung stellen.

   (c) (5 Bonuspunkte) Wenn Sie Ihre eigne Backpropagation-Übung als Jupyter-Notebook umsetzten erhalten Sie 5 Bonuspunkte. Nutzen Sie zum Erstellen des Graphen das Werkzeug Graphviz (https://graphviz.org/) mit einem entsprechenden Python-Binding, beispielsweise https://github.com/xflr6/graphviz. Eine Einführung zur Verwendung von Graphviz unter Python finden Sie in der Dokumentation
   (https://graphviz.readthedocs.io/en/stable/examples.html).

3. Neural Network

   Ihnen sind folgende Informationen zu einem neuronalen Netzwerk gegeben.

   Gewichtsmatrizen der Schichten, wobei das erste Element jeder Zeile einem Bias entspricht:

$$W_{HIDDEN} = \begin{pmatrix} 10 & -20 & 20 & -40 \\ 20 & -40 & 0 & 0 \end{pmatrix}$$

$$W_{OUTPUT} = \begin{pmatrix} 20 & 40 & -40 \end{pmatrix}$$

   Aktivierungsfunktion g(z), die bei allen Neuronen gleich ist:

$$g(z) = \begin{cases} 0 & z \leq -10 \\ 1 & z \geq 10 \\ 0.5 & \textit{sonst} \end{cases}$$

   (a) (5 Punkte) Zeichnen Sie das beschriebene neuronale Netzwerk mit allen Neuronen und deren Verbindungen auf. Notieren Sie Gewichte und Bias an die jeweiligen Verbindungen. Zeichnen Sie sauber, nicht leserliche Skizzen werden nicht gewertet.

   (b) (5 Punkte) Erstellen Sie aus den gegebenen Vektoren x_1, x_2, x_3 eine Mini-Batchmatrix. Berechnen Sie basierend auf diesem Input den Ouput des Netzwerks. Notieren Sie auf dem Weg ebenfalls die Präaktivitäten der verdeckten Neuronen, deren Aktivierungen und die Präaktivitäten des Ausgangs. Um die vollständige Punktzahl zu erhalten, müssen alle Schritte mittels Matrix-Operationen berechnet werden.

$$\vec{x}_{(1)} = [0,1,1],\ \vec{x}_{(2)} = [1,1,0],\ \vec{x}_{(3)} = [1,0,1]$$