Programmieraufgabe 2

In dieser Programmieraufgabe werden wir ein erstes Neuronales Netz mittels eines Abstiegsverfahrens lernen um handgeschriebene Ziffern maschinell zu klassifizieren. Als Grundlage verwenden wir einen Ausschnitt des MNIST-Datensatzes, welcher aus Trainings- und Testdaten sowie der Zuordnung zu den richtigen Ziffern besteht.

Wir werden das `Python` package `sklearn` benutzen. Die Bilder der Trainingsdaten können aus `sklearn.datasets` mit den Funktionen

- `load_digits().data, load_digits().target`
- `mask = np.isin(y, [1,5,7]), label_map = {1:0,5:1,7:2}`

geladen und nach den Klassen $\{1, 5, 7\}$ gefiltert werden. Das Trainigsset wird anschließend gesplittet mittels `sklearn.model_selection`

- `train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)`
- `scaler = StandardScaler(), scaler.fit_transform`

und passend skaliert aus `sklearn.preprocessing`.
Das Ziel der Aufgabe ist, ein Neuronales Netz mit $D = n_0 = 28^2$ input Dimensionen, zwei hidden layern der Größe $n_1 = 64$ und $n_2 = 32$ und $K = n_3 = 3$ output Dimensionen (Anzahl der möglichen Klassen) zu konstruieren und zu lernen.

1. Schreiben Sie alle nötigen Hilfsfunktionen, die in Ihrem Program auftreten, d.h. `one_hot` encoding, mit Ausgabe eines Binären 3-dimensionalen Arrays, der softplus-function und die sigmoid-function, und den zugehörigen Ableitung.
2. Zum Auswerten des Neuronalen Netzes schreiben Sie ferner eine Funktion `forward`, die Ihnen das Neuronale Netz vorwärts auswertet.
3. Ferner benötigt das Gradient-Abstiegsverfahren eine Funktion, die den Loss und die Gradienten mittels der Backpropagation ermittelt. Nutzen Sie ferner einmal die quadrierte Euklidische Norm als Loss und einmal die empirische log-Entropie Funktion.
4. Zum Durchführen des Gradient-Abstiegsverfahrens schreiben Sie eine gesonderte Funktion, die die gelernten Parameter updated.
5. Schreiben Sie eine Funktion `train_full_bath` die das Gradienten-Abstiegsverfahren ausführt. Innerhalb berechnen Sie sowohl den Loss auf den Trainings- als auch auf den Testdaten mittels der Function `accuracy_score` aus `sklearn.metrics`.

6. Speichern Sie den Verlauf über alle Epochen und plotten Sie diesen. Ferner visualisieren Sie das Klassifizierungsergebnis durch eine Confusionmatrix.

Verwenden Sie außer `skimage`, `sklearn`, `numpy` und `matplotlib` keine weiteren Pakete. Kommentiere Sie den Quellcode! Fügen Sie der Abgabe ein Hauptprogramm (Python-Skript) bei, welches eigenständig alle Ausgaben erzeugt. Beschriften Sie die Ausgaben direkt oder erläutern Sie in einer Readme-Datei in welcher Reihenfolge die Ausgaben erfolgen.