Künstliche Intelligenz WS 19/20

ÜBUNGSBLATT 1

Abgabe bis: 21. November 2019

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Vorlesung: Prof. Dr. Paul Lukowicz

Übungen: M.Sc Peter Hevesi, Kunal Oberoi, M.Sc Vitor Fortes, B.Sc Matthias Tschöpe

Hinweis: Alle Programmieraufgaben sind in Python abzugeben.

Aufgabe 1.1 (FizzBuzz)

Implementieren Sie eine Funktion `fizz_buzz(n)`, die für eine Zahl $n \in \mathbb{Z}$ folgendes zurück gibt:

$$\texttt{fizz\_buzz}(n) := \begin{cases} \text{“}Fizz\text{”}, & \text{wenn } n \bmod 3 \equiv 0 \\ \text{“}Buzz\text{”}, & \text{wenn } n \bmod 5 \equiv 0 \\ \text{“}FizzBuzz\text{”}, & \text{wenn } (n \bmod 3 \equiv 0) \wedge (n \bmod 5 \equiv 0) \\ n, & \text{sonst} \end{cases}$$

Aufgabe 1.2 (running_mean)

Es sei $l \in \mathbb{R}^n$. Zur Erinnerung, das *Arithmetische Mittel* $mean(l)$ ist wie folgt definiert:

$$mean(l) := \frac{1}{n} \sum_{i=1}^{n} l_i$$

dabei steht l_i für das i—te Element in l. Für $k \in \mathbb{N}$ mit $1 \leq k \leq n$ definieren wir eine Funktion:

$$mean\_k(l,k) := mean([l_1, l_2, \ldots, l_k]).$$

Schreiben Sie eine Funktion `running_mean(l)`, die eine Liste l als Eingabe nimmt und eine Liste:

$$l' := [mean\_k(l,1), mean\_k(l,2), \ldots, mean\_k(l,n)]$$

zurück gibt.

Für eine Eingabelist `l = [6, -5, -1, 3, 11, 1, -8, 7, 4, 9]` sollte die Ausgabe:

```
running_mean(l) = [6.0, 0.5, 0.0, 0.75, 2.8, 2.5, 1.0, 1.75, 2.0, 2.7]
```

Aufgabe 1.3

In dieser Aufgabe geht darum eine CSV-Datei einzulesen, die Daten richtig zu interpretieren und weiterzuverarbeiten. Im OLAT-Kurs finden Sie eine csv-Datei mit dem Namen `Data_Task_3.csv`. Diese enthält zwei Matrizen A und B hintereinander geschrieben und durch ein Semikolon getrennt. Die Kommata trennen die einzelnen Elemente einer Matrix.
Lösen Sie folgende Aufgaben:

(a) Lesen Sie die, in der csv-Datei, gespeicherten Matrizen A und B als numpy-Array ein.

(b) Berechnen Sie $C = A \cdot B$ und plotten Sie mindestens die ersten vier Zeilen. Was fällt Ihnen auf? Geben Sie bei Ihrer Abgabe den Plot auch als png-Datei mit an.

Ihre Ergebnisse sollten in etwas so aussehen wie in Abbildung 1.1

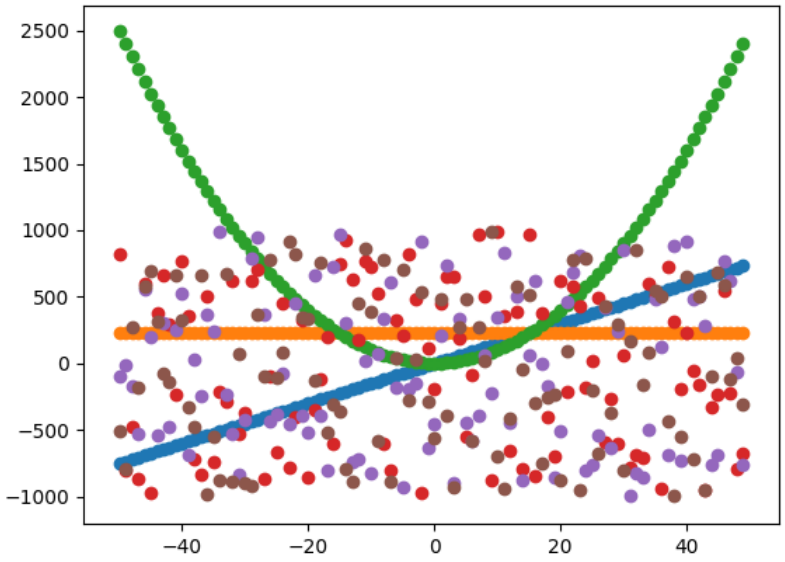

Abbildung 1.1

Aufgabe 1.4

Schreiben Sie eine Klasse `Knoten` mit einem geeigneten Konstruktor und einer Methode `add_kinder(kids)`. Der Konstruktor soll einem Knoten-Objekt einen eindeutigen Wert `key` und ggf. Nachfolgerknoten `kids` zuweisen. Mit der Methode `add_kinder(kids)` soll man nachträglich noch weitere Kinder hinzufügen können. Erstellen Sie anschließend mit Ihrer Klasse `Knoten`, einen Graphen mit mindestens fünf Knoten.

So sollte man z.B. einen kleinen Graphen erzeugen können:

```
k3 = Knoten(3, [])
k2 = Knoten(2, [])                    k3.add_Kinder([k1, k5])
k1 = Knoten(1, [k2, k3])              k2.add_Kinder(k1)
```