6. PNK-Übungsblatt

Aufgabe 18 Die Klasse der Brüche

Verwenden Sie die Klasse der Brüche aus der Übung.

a) Erweitern Sie die Klasse so, dass bei negativen Zahlen immer das Minus im Zähler steht.

b) Erweitern Sie die Klasse so, dass die Null immer ohne Vorzeichen und ohne Nenner ausgegeben wird.

c) Testen und erklären Sie was passiert, wenn Sie „verbotene Dinge“ verlangen, wie z.B. die Division durch Null.

Aufgabe 19 Die Klasse der Neuronalen Netze

Für die Implementierung eines Prototyps eines Neuronalen Netzes haben wir uns auf folgende vereinfachende Randbedingungen geeinigt:

- Der Typ des neuronalen Netzes ist Multilayer Perceptron (mlp).
- Vorerst reicht ein Netz ohne verstecke Schicht und mit jeweils zwei Ein- und Ausgabeneuronen.
- Es existieren Kanten zwischen allen Knoten benachbarter Schichten.
- Als Aktivierungsfunktion genügt die Stufenfunktion. Als Ausgabefunktion wird die Identität verwendet. Es sollen zunächst nur binäre und noch keine reellwertigen Probleme gelöst werden.
- Die Initialisierung der Kantengewichte erfolgt zufällig.
- Die Lernrate ist $\eta = 0.2$
- Als Lernverfahren soll Backpropagation mit Gradientenabstieg und Online-Training verwendet werden.

a) Implementieren Sie die Funktion zum Erstellen eines neuronalen Netzes (___init___).

b) Implementieren Sie die Funktion zum Ausgeben der Parameter und der Gewichtsmatrizen eines neuronalen Netzes (___str___).

c) Implementieren Sie die Funktion zum Bestimmen der Ausgabe für eine feste Eingabe (predict).

d) Testen Sie Ihre Implementierung durch beispielhaftes Ausführen und manuelles Prüfen der Ergebnisse.

e) Planen Sie die Implementierung der Backpropagation des Fehlers der Ausgabeschicht.

- Welche Schwierigkeiten erwarten Sie?
- Sollten einige Randbedingungen nochmals geändert werden, um die Implementierung zu erleichtern?
- Welche für das Lernen wichtige Randbedingungen sind noch offen?

Hinweise:
Wenn Sie Spyder im „scientific“-Modus verwenden, ist dort standardmäßig der Import von mehreren Modulen und der darin enthaltenen Funktionen voreingestellt. Für die aktuelle Aufgabe sollte es reichen Zufallszahlen gleichverteilt im Intervall [0;1) durch die folgenden Kommandos zu erzeugen:

```
import numpy as np #einmalig zu Beginn der Programms
np.random.rand() #jedesmal, wenn Sie eine Zufallszahl zwischen 0 und 1 brauchen
```

Wenn Sie Zufallszahlen aus einem anderen Bereich zwischen a und b benötigen, so können Sie zum Beispiel die folgende Funktion verwenden:

```
def r(a,b):
    return a+np.random.rand()*(b-a)
```

Durch übergeben weiterer Parameter an die Funktion rand(), können Sie auch gleich mehrere Zufallszahlen erhalten. Wie das genau geht, steht in der Dokumentation beschrieben, die Sie zum Beispiel unter

http://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand.html

einsehen können.
Dort können Sie auch nach weiteren hilfreichen Funktionen stöbern.