JUST DO IT
HÖHERE ABTEILUNG FÜR
INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Laborprotokoll

GK10.1

Systemtechnik Labor
5BHIT 2017/18

Martin Wölfer

Version 0.1
Begonnen am 5. April 2018
Beendet am 5. April 2018

Note:
Betreuer: BRAH

Inhaltsverzeichnis

1 Aufgabenstellung

Sie erhalten ein lauffähiges Neuronales Netzwerk (NN) inklusive Sourcecode mit parametrisierbarer Topologie und wählbarer Learnrate.

Adaptieren Sie dieses in Bezug auf Topologie, Learnrate, Trainingsset, Trainingsablauf, Trainingsdauer, sodass es möglich ist, damit das untenstehende Problem zu lösen. Versuchen Sie noch ein zweites NN zu finden, welches sehr verschieden zumindest in Bezug auf Topologie, jedoch auch in der Lage ist, die Aufgabe zu erfüllen. Trainieren Sie die gefundenen NN's entsprechend und zeigen Sie die erfolgreichen Anwendungen und dokumentieren Sie diese im Abgabeprotokoll.

Zu lösen sei die Wahrheitstabelle

0 0 0 -> 0 0 0

0 0 1 -> 0 1 1

0 1 0 -> 1 1 0

0 1 1 -> 1 0 1

1 0 0 -> 1 0 1

1 0 1 -> 1 1 0

1 1 0 -> 0 1 1

1 1 1 -> 0 0 0

2 Ergebnisse

2.1 C++ Kompilieren und Ausführen

Der erste Schritt war es C++ kompilieren zu können. Da ich auf Linux arbeite, kann ein `.cpp` File folgendermaßen kompiliert und ausgeführt werden:

```
g++ ${file} -std=c++1z -o ${fileBasenameNoExtension} && ./${fileBasenameNoExtension}
```

2.2 Topologie und Parameter anpassen

Da die Aufgabe **3** Inputs und **3** Outputs erwartet, muss die Topologie angepasst werden. Es wurde sich für einen Hidden Layer mit **5** Nodes entschieden, dadurch ergibt sich folgende Liste an Integer als Parameter:

```
    topologie{ 3,5,3 }
```

Weiters wurde statt der `ReLU` Funktion eine Sigmoid-Funktion (`eins_durch_ehoch`) verwendet kombiniert mit einer `learnRate` von **0.9** statt **0.09**

2.3 NN trainieren

Um das Netz zu trainieren wurden das Trainingset angepasst:

```
    // Beispiel fuer 001 -> 011
    p->input[0] = 0.0;
    p->input[1] = 0.0;
    p->input[2] = 1.0;
    p->trueVal[0] = 0.0;
    p->trueVal[1] = 1.0;
    p->trueVal[2] = 1.0;
    p->calc(learn{ true });
```

2.4 Ausgabe

Natürlich musste auch die Angabe angepasst werden, da nun statt **2** Inputs und **1** Output jeweils **3** Inputs und **3** Outputs vorhanden sind. Beim Ausführen ergibt sich folgende Ausgabe:

```
topologie 3 5 3
Nlayer = 3
Nnod = 11
Nwij = 38
Neural Network is up and ready
Ergebnis:
000 -> 0.000629122 6.85436e-05 0.00161121
001 -> 0.00223526 0.998225 0.997495
010 -> 0.999252 0.998233 2.96132e-06
011 -> 0.997832 0.00201509 0.999948
100 -> 0.997397 0.00209864 1
101 -> 0.997617 1 0.00285021
110 -> 0.00390483 0.996907 0.996344
111 -> 1.4412e-10 0.00284993 0.00310903
Elapsed time in seconds = 14.4039
```