Blatt 01 - Einführung

Abgabe: Bis Mittwoch, 29.Oktober 2014, 10.00Uhr
An: Tobias Rothenberger, <rothenb@informatik.uni-frankfurt.de>

Aufgaben:	Aufgabe 1		Aufgabe 2			Aufgabe 3	
	1a	1b	2a	2b	2c	3a	3b

Aufgabe 1.1

Wie wir in der Einleitung gesehen haben, ist die Aktivierung eines formalen Neurons gegeben durch $z = \sum_{i=0}^{n} x_i w_i = \underline{w}^T \underline{x}$. In dieser Aufgabe werden wir uns kurz ansehen, wie wir diese Größe anschaulich interpretieren können.

a. Schreiben Sie ein Programm, welches ein einzelnes formales Neuron mit zwei Eingaben und der linearen Ausgabefunktion $S(z) = z$ implementiert. Die Gewichte des Neurons können Sie dabei beliebig wählen; sinnvollerweise sollte aber mindestens eines der zu den Eingaben gehörenden Gewichte ungleich Null sein. Stellen Sie anschließend die durch den Gewichtsvektor definierte Hyperebene mit Hilfe des Plotters (s. Einführung ins Praktikum) graphisch dar.

nach oben

b. Wählen Sie nun zufällig einige Punkte (maximal 10) aus einem Bereich des Eingaberaums, durch den Ihre Hyperebene geht, tragen Sie diese in ihren bestehenden

Plot ein und berechnen Sie jeweils die Ausgabe des Neurons. Was fällt Ihnen auf? Wie deuten Sie die Aktivierung *z* und welche Aufgabe übernimmt das von Ihnen implementierte Neuron, wenn Sie die

Ausgabefunktion $S(z) = \begin{cases} 0, & z \leq 0 \\ 1, & z > 0 \end{cases}$ verwenden?

nach oben

Aufgabe 1.2

Betrachten wir das Neuronenmodell aus Abbildung 1.1. Beim Entwurf dieses Modells wurde versucht, die wesentlichen informationsverarbeitenden Eigenschaften einer biologischen Nervenzelle in einem möglichst einfachen Modell einzufangen und wir werden uns im Verlauf des Praktikums hauptsächlich mit künstlichen Neuronen und Netzen auf Grundlage dieses Modells beschäftigen.

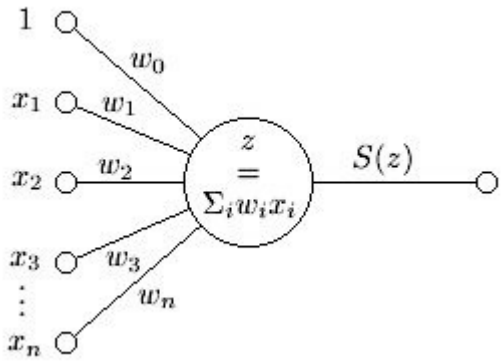

Abbildung 1.1 Neuronenmodell

Eine interessante Frage lautet nun, was dieses Modell überhaupt

kann, wo seine Grenzen und wo seine Stärken liegen. Ein erster Einblick soll in dieser Aufgabe gegeben werden.

a. Schreiben Sie ein kurzes Programm, welches mit formalen Neuronen boole'sche Gatter implementiert, aus denen sich jede beliebige boole'sche Funktion $f : \{0,1\}^n \longrightarrow \{0,1\}, n \in \mathbb{N}$ zusammensetzen läßt. Dazu ist es hinreichend, daß ihre Neuronen boole'sche Gatter berechnen, die universell für die boole'sche Algebra sind. Verwenden Sie dabei die Heaviside-Funktion (benannt nach dem britischen Physiker und Mathematiker Oliver Heaviside (1850-1925)) $S(z) = \begin{cases} 0, & z \leq 0 \\ 1, & z > 0 \end{cases}$ als Ausgabefunktion ihrer Neuronen.

   Boole'sche Gatter, die durch Neuronen berechnet werden, bezeichnet man in diesem Zusammenhang auch als Threshold-Gatter und Schaltkreise aus diesen Gattern als Threshold-Schaltkreise. Ihr Programm sollte dabei für jedes Threshold-Gatter bei Eingabe eines $\underline{x} \in \{0,1\}^n$ für beliebiges n die Ausgabe $y \in \{0,1\}$ dieses Gatters berechnen und ausgeben. Die Anzahl und Gewichte aller Neuronen können Sie nach ihrem Ermessen wählen.

   Sie haben damit gezeigt, daß Netze aus Neuronen, wie sie in Abbildung 1.1 dargestellt sind, *mindestens* über die Fähigkeit verfügen, alle erdenklichen boole'schen Funktionen $f : \{0,1\}^n \longrightarrow \{0,1\}^m, n, m \in \mathbb{N}$ repräsentieren zu können!

nach oben

b. Implementieren Sie ein einfaches kleines Neuronales Netz, welches die XOR-Funktion zweier 1-Bit Binärzahlen berechnet. Verwenden Sie als Ausgabefunktion ihrer Neuronen die Heaviside-Funktion.

   Zur Erinnerung: $XOR(a, b) = (a \vee b) \wedge \neg(a \wedge b)$

   Haben Sie eine Erklärung dafür, warum Sie zur Berechnung dieser Funktion kein einschichtiges Netz verwenden können (ein Netz ohne *hidden layer*)?

nach oben

c. Welches Problem tritt auf, wenn Sie eine Funktion $g : \{0,1\}^* \longrightarrow \{0,1\}$ berechnen möchten?

nach oben

Aufgabe 1.3

a. Implementieren Sie ein Perzeptron und trainieren Sie es mit dem Perzeptron-Lernalgorithmus und den Daten aus der Iris-Tabelle darauf, die beiden Iris-Typen korrekt zu klassifizieren. Stellen Sie dabei die beiden Datensätze und die trennende Hyperebene graphisch dar.

nach oben

b. Verwenden Sie nun das trainierte Perzeptron, um zu den letzten vier Tabelleneinträgen den Iris-Typ zu bestimmen. Überprüfen Sie das Ergebnis, indem Sie die Messwerte der vier Pflanzen in ihre Graphik aus Teil a) eintragen.

nach oben