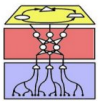

Universität Hamburg
Department Informatik, ABe NATS, WTM & TAMS
Wolfgang Menzel, Stefan Wermter, Jianwei Zhang

Kontakt: Cornelius Weber
Raum F-233
Tel 42883-2537

(für MLP Gruppen)

Übungen zum Modul: Algorithmisches Lernen

SS 2012 Blatt 7 (für MLP Gruppen)

Bearbeitung bis: 23.05.2012

Aufgabe 7.1 MLP mit Programmierfehlern

Gegeben seien Datenpaare $\{\vec{x}^\alpha, y^\alpha\}$, $\alpha = 1, \ldots, 10000$. Die Komponenten der Eingaben $\vec{x}^\alpha \in \mathbb{R}^5$ seien jeweils aus dem Intervall $[0, 100]$, Ausgaben $y^\alpha \in \mathbb{R}$ seien im Intervall $[0, 10]$ aus einem Gesetz mit Rauschen entstanden. Aufgabe sei es, ein MLP zu trainieren, das das Gesetz möglichst gut approximiert.

Korrigieren Sie die Fehler im folgenden Lösungsvorschlag und schreiben Sie notwendige Ergänzungen hinzu!

Lösungsvorschlag:

Festlegung der Netzarchitektur: 5 Eingaben (Schicht 0), 3 versteckte Neurone (Schicht 1), 1 Ausgabeneuron (Schicht 2). Ein "Schwellwertneuron" mit Aktivierung "-1", zu dem alle Neurone verbunden sind. Initialisierung aller Gewichte $w_{ij}^{\lambda\,\lambda-1}$ mit Zufallswerten im Intervall [0,1]

BEGIN {iteriere}

Alle temporären Gewichte : $\Delta w_{ij}^{\lambda\,\lambda-1} = 0$

Trainingsfehler: $E^{train} = 0$

BEGIN {für jedes Datum}

Aktivitäten der Eingabeneurone i: $S_i^0 = x_i^\alpha$

Für Schichten $\lambda{=}1$ und dann $\lambda{=}2$: $S_i^\lambda = 1/(1 + \exp(-\sum_j w_{ij}^{\lambda\,\lambda-1} S_j^{\lambda-1}))$

In Schicht $\lambda{=}2$ für das Neuron i: $\delta_i^\lambda = S_i^\lambda \cdot (1 - S_i^\lambda) \cdot (y^\alpha - S_i^\lambda)^2$

Für Schicht $\lambda{=}1$: $\delta_j^{\lambda-1} = S_j^{\lambda-1} \cdot (1 - S_j^{\lambda-1}) \cdot \sum_i \delta_i^\lambda w_{ij}^{\lambda\,\lambda-1}$

Für Schichten $\lambda{=}1$ und $\lambda{=}2$: $\Delta w_{ij}^{\lambda\,\lambda-1} = \Delta w_{ij}^{\lambda\,\lambda-1} + \delta_i^\lambda S_j^{\lambda-1}$

Mit $i{=}1$, $\lambda{=}2$: $E^{train} = E^{train} + (y^\alpha - S_i^\lambda)$

END {für jedes Datum}

Für Schichten $\lambda{=}1$ und $\lambda{=}2$: $w_{ij}^{\lambda\,\lambda-1} = w_{ij}^{\lambda\,\lambda-1} + \Delta w_{ij}^{\lambda\,\lambda-1}$

END {iteriere} STOP FALLS $\{E^{train} = 0\}$

— bitte wenden —

Wolfgang Menzel
(menzel@informatik...)
Stefan Wermter
(wermter@informatik...)
Jianwei Zhang
(zhang@informatik...)

Vorlesung:
Mi 8:15-11:45, F-009
Übungen:
Mi 12:15-13:45, F-009

Dienstgebäude
Universität Hamburg
Vogt-Kölln-Str. 30
22527 Hamburg

Öffentliche Verkehrsmittel
Buslinien 181, 281
Haltestelle: Informatikum

Aufgabe 7.2 Ableitung der Transferfunktion

Die logistische Transferfunktion ist:

$$f(x) \;=\; \frac{1}{1+e^{-\beta x}}$$

Zeigen Sie, daß $\frac{\partial f(x)}{\partial x} = \beta f(x) \cdot (1 - f(x))$. Was bringt diese Differentialgleichung für einen Vorteil für den Backpropagation Algorithmus?

Betrachten Sie nun folgende Transferfunktion:

$$g(x) \;=\; \frac{1}{2}(\tanh(\frac{\beta}{2}x) + 1)$$

Plotten Sie beide Funktionen, $f(x)$ und $g(x)$. Was sind Unterschiede? Gilt dieselbe Differentialgleichung wie für $f(x)$?

Aufgabe 7.3 MLP und XOR-Klassifierung

Gegeben ein MLP mit Eingabe-, versteckter und Ausgabeschicht. Wieviele Neurone in der versteckten Schicht braucht es, um den XOR-Datensatz, und wieviele, um den AND-Datensatz korrekt zu klassifizieren?

Falls Sie anstatt theoretischer Überlegungen das Experiment bevorzugen, befindet sich Code des Multi-Layer Perceptron (`mlp.py`) auf der Webseite des Marsland Buchs unter "Chapter 3":
`http://seat.massey.ac.nz/personal/s.r.marsland/MLBook.html`
Benutzen Sie die Demonstration des MLP an logischen Funktionen (`logic.py`).