Übungen zur Vorlesung
Grundlagen der Künstlichen Intelligenz
Sommersemester 2016

Institut für Informatik 4
Priv.-Doz. Dr. V. Steinhage
Friedrich-Ebert-Allee 144
53113 Bonn
Email: steinhage@cs.uni-bonn.de
WWW: http://net.cs.uni-bonn.de/ivs/

Blatt 9 (10 Punkte)

Abgabe durch Hochladen auf der eCampus-Seite bis Sonntag, 19.06.2016, 23:59 Uhr, in Gruppen von 2-3 Personen.

Aufgabe 9.1: AdaBoost $(\mathbf{3+1=4})$

Anhand der zwei Attribute A_1:*"linkes Knie juckt"* und A_2:*"Vögel fliegen tief"* soll entschieden werden, ob es regnet oder nicht. Dazu soll hier mittels des Algorithmus `AdaBoost` (Vorlesung 14) ein Ensemble von nur zwei gewichteten Entscheidungsstümpfen über genau diesen beiden Attributen gelernt werden. Dabei besteht das Hypothesen-Lernverfahren L(examples,w) darin, für jedes der beiden Attribute $A_m, m \in \{1,2\}$, die Hypothese (Vorfaktor) $h_m \in \{1,-1\}$ zu lernen, die den gewichteten Gesamtfehler $e_m = 0.5 \cdot \sum_{j=1}^{8} |w[j] \cdot h_m \cdot I_j(A_m) - I_j(R)|$ über der Trainingsmenge minimiert, d.h. ob es eher regnet, wenn A_m zutrifft oder wenn A_m nicht zutrifft. Gegeben sind folg. Trainingsbeispiele:

Instanz	A_1	A_2	R egen
I_1	1	1	1
I_2	-1	1	1
I_3	1	-1	1
I_4	-1	1	1
I_5	1	-1	-1
I_6	-1	1	-1
I_7	1	1	-1
I_8	1	-1	-1

a) 1) Berechnen Sie für jede der zwei Iterationen die Hypothese h_m, den gewichteten Fehler e_m und das Hypothesengewicht z_m (mit ln für log im Algm.) zum Attribut A_m:

Iteration m	h_m	e_m	z_m
#1			
#2			

2) Notieren Sie nach jeder Iteration m die neue Gewichtsverteilung über den Trainingsinstanzen w_m und den absoluten Fehler E_m des Ensembles auf der Trainingsmenge.

$$E_m = 0.5 \cdot \sum_{j=1}^{8} \left| \overbrace{\operatorname{sgn}\left[\sum_{i=1}^{m} z_i \cdot h_i \cdot I_j(A_i)\right]}^{\text{Klassifikator Ergebnis}} - \overbrace{I_j(R)}^{\text{ground truth}} \right|$$

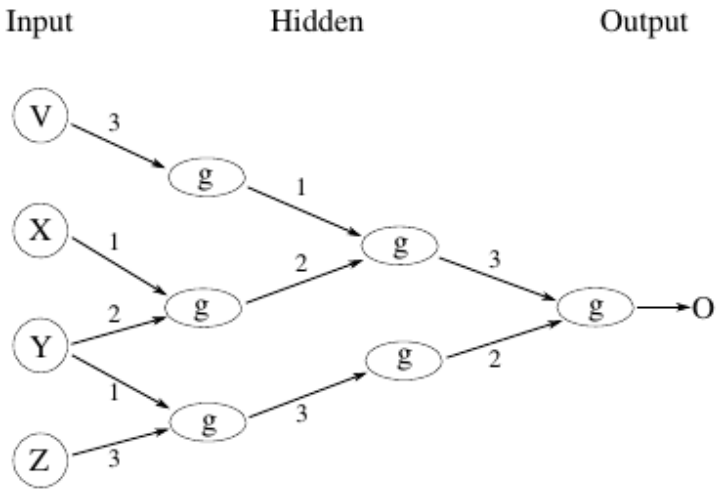
	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	E_m
$w_0[I_j]$	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	-
$w_1[I_j]$ vor Normalisierung									
$w_1[I_j]$ nach Normalisierung									
$w_2[I_j]$ vor Normalisierung									
$w_2[I_j]$ nach Normalisierung									

b) Würde es laut dem gelernten Ensemble-Klassifizierer bei den Testinstanzen $(-1, -1)$ und $(1, -1)$ regnen oder trocken bleiben?

Aufgabe 9.2: Neuronale Netze: die XOR-Funktion **(1)**

Konstruieren Sie das kleinste neuronale Netz (gemessen an der Zahl der Knoten), das die XOR Funktion von zwei (binären) Eingaben berechnet. Verwenden Sie dazu nur $step_t$-Funktionen mit beliebigem t als Aktivierungsfunktion. Jeder zusätzliche Knoten führt zum Abzug eines halben Punktes.

Aufgabe 9.3: Neuronale Netze mit $g(x) = (c \cdot x)$ **$(\mathbf{1 + 0.5 + 0.5 = 2})$**



Betrachten Sie das oben gezeigte neuronale Netz. Die Aktivierungsfunktion g sei $g(x) = 2 \cdot x$.

a) Welche Funktion wird von dem Netz berechnet?

b) Geben Sie ein Perzeptron an, das diese Funktion ebenso berechnet.

c) Welche Eigenschaft lässt sich über Feed-Forward-Netze mit linearen Aktivierungsfunktionen $(g(x) = (c \cdot x))$ schliessen? Begründen Sie Ihre Antwort!

Beachten Sie, dass von Aufgaben 9.4 und 9.5 nur eine auszusuchen ist, die abgegeben werden kann/muss. Es wird dementsprechend nur eine Aufgabe korrigiert und bepunktet. Bei Abgabe beider Aufgaben bitte kenntlich machen, welche bewertet werden soll.

Aufgabe 9.4: Programmieraufgabe: Backpropagation **(entweder hier 3)**

Vorbereitung: Laden Sie bitte das ZIP-Archiv feedforward.zip herunter von unserer eCampus-Seite unter Kursunterlagen >> Python und AIMA Python >> AIMA-Py Neural Networks. Das ZIP-Archiv enthält den Ordner *feedforward* mit dem unvollständigen Skript *backpropagationS.py* sowie dem vorgegebenen Restaurant-Trainingsdatensatz *restaurant.feat* und der vorgegebenen Netzwerkstruktur *restaurant.ffn* dafür. Fügen Sie den Ordner *feedforward* auf der gleichen Ebene wie den Ordner *aima* ein.

Damit das Hinton-Diagramm nicht in der Konsole, sondern in einem eigenem Fenster gezeigt und aktualisert wird, nehmen Sie bitte ggf. folgende Einstellung in *Spyder* vor: (1) Gehen Sie auf *Tools* → *Preferences*. (2) In dem neuen Fenster von *Preferences* links auf *IPython console*. (3) Im rechten Teil auf *Graphics*. (4) Ändern Sie *Backend* von *Inline* auf *Tkinter*.
Aufgabe:

a) Erstellen Sie auf der Basis des gegebenen unvollständigen Skripts *backpropagationS.py* ein neues Skript *backpropagation.py*, welches den Algorithmus *Backpropagation* auf dem vorgegebenen Restaurant-FF-Netz gemäß der Vorlesung umsetzt. Zu ergänzen sind die Fehlerberechnung und die Fehlerpropagation in der Funktion *main* sowie die Berechnung der Potientiale der versteckten Einheiten sowie der Ausgabe in der Funktion *run_network*.

b) Trainieren Sie mit Hilfe Ihres neu erstellten Skripts *backpropagation.py* die Gewichte des vorgegebenen Restaurant-FF-Netzes (Datei *restaurant.ffn*) anhand des ebenfalls vorgebenen Restaurant-Trainingsdatensatzs *restaurant.feat*. Dazu ist einfach die Option *Backpropagtion* in der GUI zu wählen (nach Laden Ihres entspr. Skriptes natürlich). Dabei lassen Sie bitte die Parameter auf den vorgebenen Default-Werten: *Threshold_to_stop* = 2.5, *Iterations_for_diagram_update* = 20, *Learning_rate* = 0.2 (s. *Properties* in der GUI).

c) Bitte geben Sie einen Screenshot des Hinton-Diagramms des trainierten Netzes ab.

d) Wenden Sie Ihr so trainiertes Netz auf ein neues Fallbeispiel über die Option *Insert own Example!* an. Wählen Sie dazu folgendes Beispiel für die Abgabe: Patrons = some, Wait_Estimate = 10-30, Alternate = No, Hungry = No, Reservation = No, Bar = Yes, Fri/Sat = Yes, Raining = Yes, Price = Cheap, Type = French.

e) Welches Ergebnis bzgl. des Zielprädikates *Will Wait* liefert Ihr trainiertes Netz für das vorgegebene Fallbeispiel?

Aufgabe 9.5: Backpropagation **(oder hier $1+1+1=3$)**

Gegeben sei folgendes Zwei-Schichten-FF-Netzwerk (vgl. Vorlesung 15):

- Zwei Eingabeknoten $I_k, k \in [1,2]$
- Drei innere Knoten $a_j, j \in [1,3]$
- Ein Ausgabeknoten O
- Lineare Aktivierungsfunktion $g(x) = x$ für alle Knoten
- Gewichtungen

$$W_{Ia} = \begin{bmatrix} -0.2 & 0.2 & -0.2 \\ 1.0 & -0.6 & 0.4 \end{bmatrix} \qquad W_{aO} = \begin{bmatrix} 0.5 \\ -0.9 \\ -0.5 \end{bmatrix}$$

Nutzen Sie die quadratische Fehlerfunktion $E(\mathbf{W}) = \frac{1}{2}\sum_i (T_i - O_i)^2$, eine Lernrate von $\alpha = 1.0$ und das Beispiel $e \in E$ mit $I^e = \begin{bmatrix} -1.0 & 0.5 \end{bmatrix}$ und $T^e = [-0.4]$.

Wenden Sie wie folgt eine Iteration des BACKPROPAGATION-Algorithmus (Vorlesung 15) an:

a) Bestimmen Sie zunächst die Ausgabe O^e des Netzwerkes sowie den Fehler $E(\mathbf{W})$ für Beispiel e .

b) Berechnen Sie nun für jede Schicht die Δ-Werte, also Δ_O sowie die Δ_{a_j}.

c) Bestimmen Sie die neuen Gewichtsmatrizen W'_{Ia} und W'_{aO}.