Übungen zur Vorlesung
Grundlagen der Künstlichen Intelligenz
Sommersemester 2024

Institut für Informatik 4
Priv.-Doz. Dr. V. Steinhage
Friedrich-Hirzebruch-Allee 8
53115 Bonn
Email: steinhage@cs.uni-bonn.de
WWW: http://net.cs.uni-bonn.de/ivs/

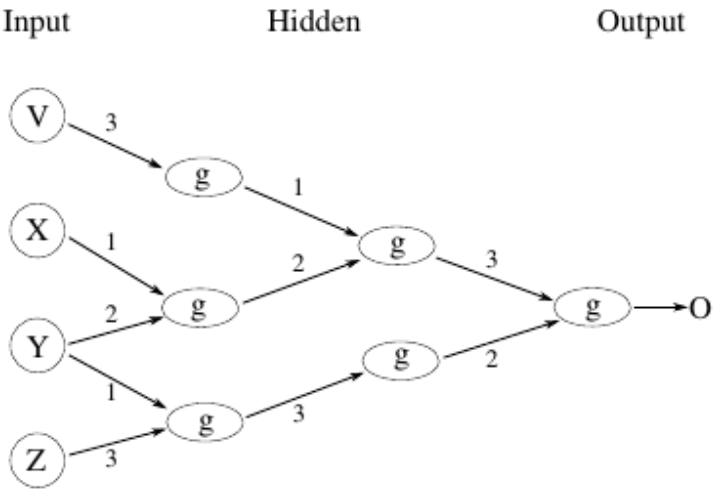
Blatt 9 (10 Punkte)

Abgabe durch Hochladen (nur PDF-Format bzw. Python-Code) auf der eCampus-Seite bis **Sonntag, 16.06.2024, 12:00 Uhr**, in Gruppen von 3 Personen.

Aufgabe 9.1: Neuronale Netze: die XOR-Funktion (1)

Konstruieren Sie das kleinste *Geschichtete Feed-forward-Netz* (GFF) 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. Es werden nur GFFs als Lösungen akzeptiert.

Aufgabe 9.2: Neuronale Netze mit $g(x) = (c \cdot x)$ $(0.5 + 0.5 + 0 = 1)$



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) Diskutieren Sie **in der der Übungsgruppe**: Welche Eigenschaft lässt sich über Feed-Forward-Netze mit linearen Aktivierungsfunktionen $(g(x) = (c \cdot x))$ schliessen?

Beachten Sie, dass von Aufgaben 9.3 und 9.4 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 - ansonsten wird Aufgabe 9.4 bewertet.

Aufgabe 9.3: 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* $\rightarrow$ *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 9.4: 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}.

Aufgabe 9.5: Lineare SVM: Trennebene **($0.5 + 0.5 + 1.5 + 1.5 = 4$)**

Gegeben seien folgende Punkte: $\mathbf{x}_1 = (2, 1)$, $\mathbf{x}_2 = (3, 2)$, $\mathbf{x}_3 = (1, 3)$, und $\mathbf{x}_4 = (1, 4)$, sowie deren Klassifikationsvektor $\mathbf{y} = (1, 1, -1, -1)$. Es soll mit Hilfe des SVM-Algorithmus (Vorlesung 16) die Maximum-Margin-Trennlinie $\langle w, x \rangle - b = 0$ bestimmt werden.

a) Berechnen und tragen Sie als Vorbereitung zunächst die Produkte $\mathrm{p}(i, j) = y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle$ für $i, j \in \{1, 2, 3, 4\}$ in die folg. Tabelle ein. *Tipp: Aus Symmetriegründen sind es nur zehn verschiedene Werte.*

p(i,j)	1	2	3	4
1				
2				
3				
4				

b) Bestimmen Sie nun unter Nutzung der in Teil a) ermittelten Produkte die vier Lagrange-Multiplikatoren α_i, indem Sie die Zielfunktion (s. Folie 14, Vorlesung 16)

$$W(\boldsymbol{\alpha}) = \sum_i \alpha_i - 0.5 \sum_{ij} [\alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle] = \sum_i \alpha_i - 0.5 \sum_{ij} [\alpha_i \alpha_j \, \mathrm{p}(i, j)]$$

bezüglich der α_i unter den Nebenbedingungen $\alpha_i \geq 0$ und $\sum_i \alpha_i y_i = 0$ maximieren. Schreiben Sie die bis auf die α_i, α_j vollständig instanziierte Zielfunktion $W(\boldsymbol{\alpha})$ und die ebenso instanziierten Nebenbedingungen auf! Die in den Aufgabenteilen c) und d) zu verwendende Lösung der Zielfunktion ist: $\alpha_1 = \frac{2}{9}, \alpha_2 = \frac{2}{9}, \alpha_3 = \frac{4}{9}, \alpha_4 = 0$.

c) Berechnen Sie nun den Normalenvektor der Trennlinie

$$\mathbf{w} = \sum_i (\alpha_i y_i \mathbf{x_i})$$

und bestimmen Sie die Indexmenge der Stützvektoren $N = \{i | \mathbf{x}_i \text{ ist Stützvektor}\}$. Hiermit errechnen Sie den Abstand b der Trennlinie zum Ursprung gemäß

$$b = \frac{1}{|N|} \sum_{i \in N} [\langle \mathbf{w}, \mathbf{x}_i \rangle - y_i]$$

d) Klassifizieren Sie $\mathbf{x}_{\text{neu}} = (4, 4)$ mittels $y_{\text{neu}} = \text{sign}(\langle \mathbf{w}, \mathbf{x}_{\text{neu}} \rangle - b)$ und zeichnen Sie die Trainingsmenge, sowie die Trennebene und den Punkt $\mathbf{x}_{\text{neu}}$.

Aufgabe 9.6: SVM: Kernel Trick **(1)**

In dieser Aufgabe soll eine Support-Vektormaschine konstruiert werden, die aus zwei Eingaben x_1 und x_2 die XOR-Funktion berechnet. Sie wissen, dass XOR nicht linear separierbar ist. Also brauchen wir den *Kernel Trick*. Es ist praktisch, als Eingaben und als Ausgaben Werte von 1 und -1 statt 1 und 0 zu verwenden:

x_1	x_2	XOR
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1

Es ist typisch, eine Eingabe (einen Datenvektor) x in einen Merkmalsraum aus fünf Dimensionen abzubilden, der aus den beiden ursprünglichen Dimensionen x_1 und x_2 und den drei Kombinationen x_1^2, x_2^2 und $x_1 \cdot x_2$ aufgespannt wird. Für diese Aufgabe betrachten wir jedoch nur die beiden Dimensionen $f_1 = x_1$ und $f_2 = x_1 \cdot x_2$. Zeichnen Sie die 4 möglichen Eingabebeispiele in diesen Merkmalsraum. Zeichnen Sie außerdem die Trennlinie ein, die den Abstand zu den Trainingsbeispielen maximiert, und den Rand (bzw. die *Margin*).