Informatik IV - Tutorium XII & XIII (SR -120)

Tut Nr. 12 – Wiederholung

David Münch

Universität Karlsruhe (TH)
Fakultät für Informatik
IBDS Prautzsch

17. Juli 2008

Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Organisatorisches

Email: muenchdavid@gmail.com

https://www.stud.uni-karlsruhe.de/~uhbro/

Tutorium 12: Donnerstags 8:00 Uhr - Raum -120
Tutorium 13: Donnerstags 9:45 Uhr - Raum -120

Übungsblattabgabe Donnerstag.

Literatur

Boehm, Prautzsch: Numerical Methods. AK Peters 1993. ISBN 3-528-06350-5
http://www.ubka.uni-karlsruhe.de/hylib-bin/suche.cgi?opacdb=UBKA_OPAC&fbt=7319953&nd=3204657

Ash: Information Theory. Dover 1990. ISBN 0-486-66521-6
http://www.ubka.uni-karlsruhe.de/hylib-bin/suche.cgi?opacdb=UBKA_OPAC&nd=9866904

Goos: Vorlesungen über Informatik. Bd. 4, Springer 1998. ISBN 3-540-60650-5
http://www.ubka.uni-karlsruhe.de/hylib-bin/suche.cgi?opacdb=UBKA_OPAC&fbt=9316367&nd=6568301

Was wollen wir heute erreichen?

Was wollen wir heute erreichen?

- Übungsblatt 12 besprechen

Was wollen wir heute erreichen?

- Übungsblatt 12 besprechen
- Simplexalgorithmus wiederholen

Was wollen wir heute erreichen?

- Übungsblatt 12 besprechen
- Simplexalgorithmus wiederholen
- Backpropagation wiederholen

Was wollen wir heute erreichen?

- Übungsblatt 12 besprechen
- Simplexalgorithmus wiederholen
- Backpropagation wiederholen
- Aufgaben von Blatt 13 rechnen

Aufgabe 69

Betrachten Sie den Hamming-Kode, der die Gleichung $A \cdot \mathbf{x} = \mathbf{0}$ löst, wenn gilt:

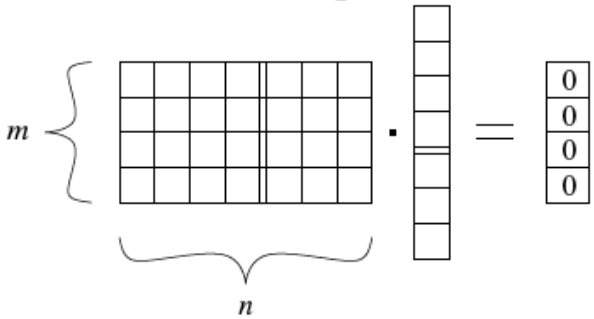
$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & a_1 \\ 1 & 1 & 0 & 0 & a_2 \\ 1 & 0 & 1 & 0 & a_3 \\ 0 & 1 & 1 & 1 & a_4 \end{bmatrix}$$

1. Geben Sie alle Kodewörter an für den Fall $a_1 = a_2 = a_3 = a_4 = 1$.
2. Geben Sie alle $\mathbf{a} = [\ a_1 \ ... \ a_4\ ]^T$ an, sodaß alle 1-Fehler korrigiert und alle 2-Fehler entdeckt werden können.
3. Für welches $\mathbf{a} = [\ a_1 \ ... \ a_4\ ]^T$ sind auch alle 2-Fehler korrigierbar?

Aufgabe 70

Studieren Sie die Eigenschaften von Hamming-Kodes.

Wir betrachten hier den Fall von $n = 7$-stelligen Kodes, bestehend aus $m = 4$ Testbits und 3 informationstragenden Bits:



Für Kodewörter $\mathbf{x}$ muß gelten: $A \cdot \mathbf{x} = \mathbf{0}$

1. Zeigen Sie, daß die vier Bitvektoren

   0010110 0110011 1101100 1001001

   linear abhängig sind.

   Welches allgemeine Kriterium können Sie dafür angeben, daß k Bitvektoren linear abhängig sind?

2. Zeigen Sie, daß die Zeilen der Matrix

$$A = \left[\begin{array}{cccc|ccc} 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{array}\right]$$

linear unabhängig sind.

Konstruieren Sie 2^{n-m} Kodewörter $\mathbf{x}_1, ..., \mathbf{x}_8$ zur Matrix A.

3. Wie groß ist der Hamming-Abstand zwischen zweien Ihrer Kodewörter mindestens? Was folgt daraus für die Fehlerkorrektur?

4. Wie groß ist die Rate r, wenn Ihr Kode zur Informationsübermittlung benutzt wird? Wieviel absolute und wieviel relative Redundanz hat Ihr Kode?

5. Ließe sich (für die gegebenen Werte n, m) ein Kode finden, der günstigere Eigenschaften bezüglich einer Fehlerdetektion und -korrektur hätte?

Vergleich des algebraischen und geometrischen Simplexverfahrens

Das Simplex-Verfahren ist ein Algorithmus zur Lösung linearer Programme.

Das Simplex-Verfahren ist ein Algorithmus zur Lösung linearer Programme.

Lineares Programm

Ein *Lineares Programm* besteht aus einer linearen Zielfunktion

$$c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$$

die maximiert/minimiert werden soll, sowie m Ungleichungen der Form

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & \begin{matrix}\leq\\=\\\geq\end{matrix} & b_1 \\ \vdots & & & & \ddots & + & & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & \begin{matrix}\leq\\=\\\geq\end{matrix} & b_m \end{array}$$

Das Simplex-Verfahren ist ein Algorithmus zur Lösung linearer Programme.

Lineares Programm

Ein *Lineares Programm* besteht aus einer linearen Zielfunktion

$$c^\top x$$

die maximiert/minimiert werden soll, sowie m Ungleichungen der Form

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & \begin{matrix}\leq\\=\\\geq\end{matrix} & b_1 \\ \vdots & & & & \ddots & + & & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & \begin{matrix}\leq\\=\\\geq\end{matrix} & b_m \end{array}$$

Das Simplex-Verfahren ist ein Algorithmus zur Lösung linearer Programme.

Lineares Programm

Ein *Lineares Programm* besteht aus einer linearen Zielfunktion

$$c^\top x$$

die maximiert/minimiert werden soll, sowie m Ungleichungen der Form

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & \leq & b_1 \\ \vdots & & & & \ddots & + & & & \vdots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & \leq & b_m \end{array}$$

Das Simplex-Verfahren ist ein Algorithmus zur Lösung linearer Programme.

Lineares Programm

Ein *Lineares Programm* besteht aus einer linearen Zielfunktion

$$c^\top x$$

die maximiert/minimiert werden soll, sowie m Ungleichungen der Form

$$Ax \leq b$$

Das Simplex-Verfahren ist ein Algorithmus zur Lösung linearer Programme.

Lineares Programm

Ein *Lineares Programm* besteht aus einer linearen Zielfunktion

$$c^\top x$$

die maximiert/minimiert werden soll, sowie m Ungleichungen der Form

$$Ax \leq b$$

Dabei sind $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ und $b \in \mathbb{R}^m$.

Je nach Verfahren (geometrisch oder algebraisch) ist eine andere *Ausgangsform* des LP notwendig.

Je nach Verfahren (geometrisch oder algebraisch) ist eine andere *Ausgangsform* des LP notwendig.

Geometrisch

Standardform

Gegeben: $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ und $c \in \mathbb{R}^n$.

Gesucht: Lösung $x \in \mathbb{R}^n$, so dass

$$c^\top x$$

maximiert unter

$$Ax \leq b \quad \text{und} \quad x \geq 0$$

Algebraisch

Normalform

Gegeben: $v \in \mathbb{R}^n$, $c \in \mathbb{R}^m$, $B \in \mathbb{R}^{m \times n}$.

Gesucht: Lösung $x \in \mathbb{R}^n$, so dass

$$z = c^\top x + v$$

maximiert unter

$$y = Bx + b \quad \geq 0$$

Simplex-Algorithmus (Geometrisch)

Das Vorgehen anschaulich:

Simplex-Algorithmus (Geometrisch)

Das Vorgehen anschaulich:

- Jede Nebenbedingung der Form

$$a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i$$

beschreibt einen *Halbraum* im $\mathbb{R}^n$.

Simplex-Algorithmus (Geometrisch)

Das Vorgehen anschaulich:

- Jede Nebenbedingung der Form

$$a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i$$

  beschreibt einen *Halbraum* im $\mathbb{R}^n$.
- Da *alle* Nebenbedingung erfüllt sein müssen definiert der *Schnitt* der Nebenbedingungen die Menge aller Lösungen – ein *konvexer Polyeder*.

Simplex-Algorithmus (Geometrisch)

Das Vorgehen anschaulich:

- Jede Nebenbedingung der Form

$$a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i$$

  beschreibt einen *Halbraum* im $\mathbb{R}^n$.
- Da *alle* Nebenbedingung erfüllt sein müssen definiert der *Schnitt* der Nebenbedingungen die Menge aller Lösungen – ein *konvexer Polyeder*.
- Der Vektor c zeigt in die zu optimierende Richtung.

Simplex-Algorithmus (Geometrisch)

Das Vorgehen anschaulich:

- Jede Nebenbedingung der Form

$$a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i$$

  beschreibt einen *Halbraum* im $\mathbb{R}^n$.
- Da *alle* Nebenbedingung erfüllt sein müssen definiert der *Schnitt* der Nebenbedingungen die Menge aller Lösungen – ein *konvexer Polyeder*.
- Der Vektor c zeigt in die zu optimierende Richtung.

Da der Lösungspolyeder konvex ist, ist die optimale Lösung stets eine *Ecke* des Polyeders.

Simplex-Algorithmus (Geometrisch)

Das Vorgehen anschaulich:

- Jede Nebenbedingung der Form

$$a_{i1}x_1 + \ldots + a_{in}x_n \leq b_i$$

  beschreibt einen *Halbraum* im $\mathbb{R}^n$.
- Da *alle* Nebenbedingung erfüllt sein müssen definiert der *Schnitt* der Nebenbedingungen die Menge aller Lösungen – ein *konvexer Polyeder*.
- Der Vektor c zeigt in die zu optimierende Richtung.

Da der Lösungspolyeder konvex ist, ist die optimale Lösung stets eine *Ecke* des Polyeders.

- Wir bewegen uns von Ecke zu Ecke entlang „verbessernder“ Kanten.

Simplex-Algorithmus (Geometrisch)

Algorithmus 1: SimplexGeom

input : Lineares Programm (A, c, b) in Standardform
output: Optimale Lösung x des LPs

$P \leftarrow$ konvexes Lösungspolyeder zu (A, c, b)

$x \leftarrow$ beliebige Ecke in P

while *es gibt verbessernde Kante* $(x, y) \in P$ **do**
    $x \leftarrow y$
end

return x

$\rightsquigarrow$ Wir bewegen uns über die Kanten des Polyeders von Ecke zu Ecke, bis wir eine optimale Ecke x gefunden haben.

Simplex-Algorithmus (Algebraisch)

Der „Ecken-Tausch“ lässt sich auch algebraisch interpretieren.

Simplex-Algorithmus (Algebraisch)

Der „Ecken-Tausch“ lässt sich auch algebraisch interpretieren.

- Statt sich von Ecke zu Ecke zu bewegen transformieren wir das Koordinatensystem:

Simplex-Algorithmus (Algebraisch)

Der „Ecken-Tausch“ lässt sich auch algebraisch interpretieren.

- Statt sich von Ecke zu Ecke zu bewegen transformieren wir das Koordinatensystem:
  - Der Nullpunkt ist stets die aktuell zu betrachtende Ecke.

Simplex-Algorithmus (Algebraisch)

Der „Ecken-Tausch“ lässt sich auch algebraisch interpretieren.

- Statt sich von Ecke zu Ecke zu bewegen transformieren wir das Koordinatensystem:
  - Der Nullpunkt ist stets die aktuell zu betrachtende Ecke.
  - Die Koordinatenachsen laufen entlang der zur Ecke inzidenten Kanten.

Simplex-Algorithmus (Algebraisch)

Der „Ecken-Tausch“ lässt sich auch algebraisch interpretieren.

- Statt sich von Ecke zu Ecke zu bewegen transformieren wir das Koordinatensystem:
  - Der Nullpunkt ist stets die aktuell zu betrachtende Ecke.
  - Die Koordinatenachsen laufen entlang der zur Ecke inzidenten Kanten.
- Zentrale Operation: Eckentausch $\leftrightarrow$ Koordinatentransformation.

Simplex-Algorithmus (Algebraisch)

Der „Ecken-Tausch“ lässt sich auch algebraisch interpretieren.

- Statt sich von Ecke zu Ecke zu bewegen transformieren wir das Koordinatensystem:
  - Der Nullpunkt ist stets die aktuell zu betrachtende Ecke.
  - Die Koordinatenachsen laufen entlang der zur Ecke inzidenten Kanten.
- Zentrale Operation: Eckentausch $\leftrightarrow$ Koordinatentransformation.
- Der Nullpunkt muss zu Anfang bereits eine Ecke des Polyeders sein!

Simplex (Algebraisch)

Algorithmus 2: SimplexAlg

input : Lineares Programm (B, b, c, v) in Normalform
output: Optimale Lösung des LPs

$y \leftarrow$ Basislösung.

if $\exists$ *Variable* x_j *in ZF mit positivem Koeffizienten* $c_j > 0$ **then**
Erhöhe x_j maximal. Finde also Ungleichung i, die das Erhöhen von x_j am stärksten einschränkt.

Tausche x_j mit y_i. Forme dazu Gleichung i nach x_j um, und setze diese für alle Vorkommen von x_j ein (auch in ZF).

Weiter in Zeile 0
end
else
$(x_1, \ldots, x_m) = 0$ ist beste Lösung.

Aufgabe

Simplex-Aufgabe

Gegeben sei folgendes lineares Programm.

$$\textbf{maximiere } 4x_1 + x_2$$

unter

$$\begin{array}{rcrcl} x_1 & + & x_2 & \leq & 16 \\ & & x_2 & \leq & 12 \\ 3x_1 & + & x_2 & \leq & 36 \end{array}$$

(a) Lösen Sie das LP geometrisch.

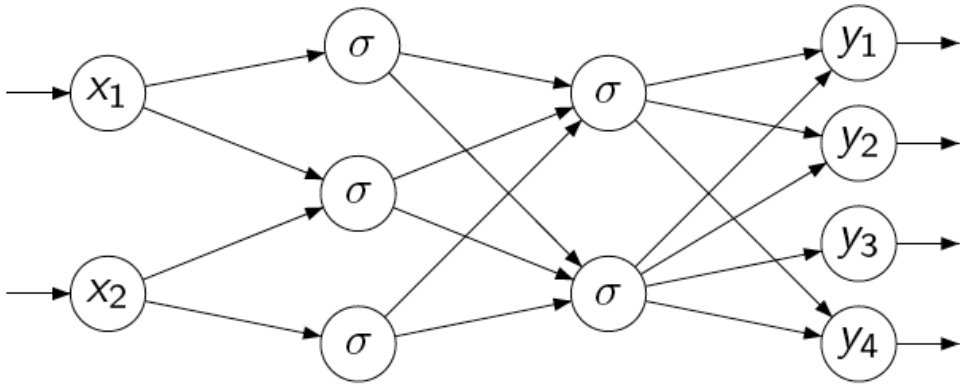
(b) Lösen Sie das LP algebraisch.

Backpropagation für mehrschichtige Neuronale Netze

Wiederholung: Perzeptron

Perzeptron

Das *Perzeptron* ist ein vorwärtsgerichtetes Neuronales Netz, bei dem die Neuronen schichtweise angeordnet sind. Verbindungen existieren nur zwischen benachbarten Schichten.
Das *einfache Perzeptron* besitzt nur eine Eingabe- sowie eine Ausgabeschicht.



Wiederholung: Neuronale Netze

Arbeitsweise der Neuronen

Die Neuronen arbeiten alle *synchron* in diskreten Zeitabständen. Jedes Neuron N_j hat einen *Zustand* (bzw. eine Ausgabe) $q_j \in Q$, der an den Ausgabekanten anliegt.

Die *Erregungsfunktion* p_j des Neurons N_j berechnet sich aus den Werten der Eingangskanten sowie dem Schwellwert von N_j:

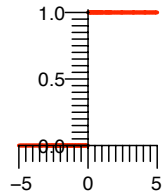
$$p_j := \sum_i w_{ij} q_i - \sigma_j$$

Die *Aktivierungsfunktion* $h_j(p_j)$ berechnet die *Ausgabe* (Zustand) des Neurons, und ist i.A. eine *sigmoide Funktion*. Es ist $q_j := h_j(p_j)$

Wiederholung: Neuronale Netze; Aktivierungsfunktionen

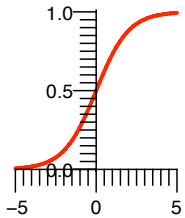
Zustandsmenge $Q = \{0, 1\}$ und

$$h(p) := \begin{cases} 1 & \text{falls} \quad p \geq 0 \\ 0 & \text{sonst} \end{cases}$$



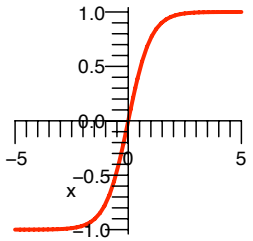
Zustandsmenge $Q = [0, 1]$ und

$$h(p) := \frac{1}{1 + e^{-\beta p}}$$



Zustandsmenge $Q = [-1, 1]$ und

$$h(p) := \tanh(\beta p)$$



Backpropagation-Algorithmus

Das Problem

Gegeben: Eine Menge von Trainingsdaten und zugehöriger Klassifikation (Ausgabe).

Gesucht: Gewichte und Schwellwerte, so dass das Netz die Trainingsdaten gut klassifiziert.

Backpropagation-Algorithmus

Das Problem

Gegeben: Eine Menge von Trainingsdaten und zugehöriger Klassifikation (Ausgabe).

Gesucht: Gewichte und Schwellwerte, so dass das Netz die Trainingsdaten gut klassifiziert.

$\rightsquigarrow$ Die gewünschte Ausgabe der versteckten Neuronen ist unbekannt!

Backpropagation-Algorithmus

Das Problem

Gegeben: Eine Menge von Trainingsdaten und zugehöriger Klassifikation (Ausgabe).

Gesucht: Gewichte und Schwellwerte, so dass das Netz die Trainingsdaten gut klassifiziert.

$\rightsquigarrow$ Die gewünschte Ausgabe der versteckten Neuronen ist unbekannt!

Idee des Algorithmus

Definiere Fehlerfunktion $E(w)$ die von den Gewichten abhängt, und minimiere diese mittels *Gradientenabstieg*.

Backpropagation-Algorithmus

Das Vorgehen anschaulich:

1. Forward-Pass (Schichtweises Durchreichen der Eingabe)
2. Bestimmung des Fehlers
3. Backpropagation (Rückrechnen des Fehlers)
4. Modifizieren der Gewichte
5. Goto (1)

Backpropagation-Algorithmus

Das Vorgehen anschaulich:

1. Forward-Pass (Schichtweises Durchreichen der Eingabe)
2. Bestimmung des Fehlers
3. Backpropagation (Rückrechnen des Fehlers)
4. Modifizieren der Gewichte
5. Goto (1)

Zwei Methoden:

- Musterlernen: Fehler wird für *ein* Trainingsdatum errechnet
- Epochelernen: Fehler wird für *alle* Trainingsdaten errechnet

Backpropagation-Algorithmus

Das Vorgehen anschaulich:

1. Forward-Pass (Schichtweises Durchreichen der Eingabe)
2. Bestimmung des Fehlers
3. Backpropagation (Rückrechnen des Fehlers)
4. Modifizieren der Gewichte
5. Goto (1)

Zwei Methoden:

- Musterlernen: Fehler wird für *ein* Trainingsdatum errechnet
- Epochelernen: Fehler wird für *alle* Trainingsdaten errechnet

Backpropagation-Algorithmus

Sei $x_1, \ldots, x_M$ eine Menge von Trainingseingaben mit den korrekten Ausgaben $a_1, \ldots, a_M$. Vermögen weiterhin W die Gewichte aller Synapsen.

Backpropagation-Algorithmus

Sei $x_1, \ldots, x_M$ eine Menge von Trainingseingaben mit den korrekten Ausgaben $a_1, \ldots, a_M$. Vermögen weiterhin W die Gewichte aller Synapsen.

- Die zu minimierende Fehlerfunktion ist

$$E(W) = \frac{1}{2}\|a - q\|^2 = \frac{1}{2}\sum_j (a_j - q_j)^2$$

Backpropagation-Algorithmus

Sei $x_1, \ldots, x_M$ eine Menge von Trainingseingaben mit den korrekten Ausgaben $a_1, \ldots, a_M$. Vermögen weiterhin W die Gewichte aller Synapsen.

- Die zu minimierende Fehlerfunktion ist

$$E(W) = \frac{1}{2}\|a - q\|^2 = \frac{1}{2}\sum_j (a_j - q_j)^2$$

- Benutze Gradientenabstieg. Also Rechenvorschrift:

$$W^+ = W - \eta \cdot \nabla E(W)$$

Bzw. aufgeschlüsselt nach Gewichten:

$$w_{ij}^+ = w_{ij} \underbrace{-\eta \cdot \frac{\partial E}{\partial w_{ij}}}_{=:\Delta w_{ij}}$$

Backpropagation-Algorithmus

Mit recht viel Rechnerei (Mehrfache Anwendung der Kettenregel) folgt

$$\Delta w_{ij} = \eta \cdot \delta_j \cdot q_i$$

Backpropagation-Algorithmus

Mit recht viel Rechnerei (Mehrfache Anwendung der Kettenregel) folgt

$$\Delta w_{ij} = \eta \cdot \delta_j \cdot q_i$$

Dabei sind

- η – Schrittweite beim Gradientenabstieg (frei wählbar)
- q_i – Zustand des betrachteten Neurons in Schicht i
- δ_j – „Fehlerkoeffizient“ aus Schicht $j = i + 1$

Backpropagation-Algorithmus

Mit recht viel Rechnerei (Mehrfache Anwendung der Kettenregel) folgt

$$\Delta w_{ij} = \eta \cdot \delta_j \cdot q_i$$

Dabei sind

- η – Schrittweite beim Gradientenabstieg (frei wählbar)
- q_i – Zustand des betrachteten Neurons in Schicht i
- δ_j – „Fehlerkoeffizient“ aus Schicht $j = i + 1$

Berechnung von δ_j nach folgender Regel:

$$\delta_j := \begin{cases} h'(p_j)(a_j - q_j) & \text{falls } N_j \text{ Ausgabe-Neuron} \\ h'(p_j) \sum_k \delta_k w_{jk} & \text{falls } N_j \text{ Hidden-Neuron} \end{cases}$$

Backpropagation-Algorithmus

Hinweise zum Backpropagation Algorithmus.

- Die Werte p_j und $q_j = h(p_j)$ sollten im Vorwärtsschritt zwischengespeichert werden.

Backpropagation-Algorithmus

Hinweise zum Backpropagation Algorithmus.

- Die Werte p_j und $q_j = h(p_j)$ sollten im Vorwärtsschritt zwischengespeichert werden.
- Gewichte nicht mit 0 initialisieren. $\rightsquigarrow$ Zufall.

Backpropagation-Algorithmus

Hinweise zum Backpropagation Algorithmus.

- Die Werte p_j und $q_j = h(p_j)$ sollten im Vorwärtsschritt zwischengespeichert werden.
- Gewichte nicht mit 0 initialisieren. $\rightsquigarrow$ Zufall.
- Alle Probleme des Gradientenabstiegs treffen hier auch zu.

Backpropagation-Algorithmus

Hinweise zum Backpropagation Algorithmus.

- Die Werte p_j und $q_j = h(p_j)$ sollten im Vorwärtsschritt zwischengespeichert werden.
- Gewichte nicht mit 0 initialisieren. $\rightsquigarrow$ Zufall.
- Alle Probleme des Gradientenabstiegs treffen hier auch zu.
- Netztopologie muss „manuell“ im Voraus festgelegt werden. $\rightsquigarrow$ Problem der Generalisierungsfähigkeit.

siehe Aufgabe 40.

Das Ende Ist Nahe. . .

Ein paar Wahr/Falsch Fragen.

Wahr/Falsch-Fragen

- Bei evolutionären Algorithmen können die Eltern im Gegensatz zu den genetischen Algorithmen immer überleben.

Wahr/Falsch-Fragen

- Bei evolutionären Algorithmen können die Eltern im Gegensatz zu den genetischen Algorithmen immer überleben. X

Wahr/Falsch-Fragen

- Bei evolutionären Algorithmen können die Eltern im Gegensatz zu den genetischen Algorithmen immer überleben. X
- Es gibt keinen Code C so dass $L(C) < L(S)$ wobei S der Shannoncode ist.

Wahr/Falsch-Fragen

- Bei evolutionären Algorithmen können die Eltern im Gegensatz zu den genetischen Algorithmen immer überleben. X
- Es gibt keinen Code C so dass $L(C) < L(S)$ wobei S der Shannoncode ist. X

Wahr/Falsch-Fragen

- Bei evolutionären Algorithmen können die Eltern im Gegensatz zu den genetischen Algorithmen immer überleben. X
- Es gibt keinen Code C so dass $L(C) < L(S)$ wobei S der Shannoncode ist. X
- In einem mehrschichtigen Perzeptron reicht eine verdeckte Schicht nicht aus, um jede berechenbare stetige Funktion beliebig gut anzunähern.

Wahr/Falsch-Fragen

- Bei evolutionären Algorithmen können die Eltern im Gegensatz zu den genetischen Algorithmen immer überleben. X
- Es gibt keinen Code C so dass $L(C) < L(S)$ wobei S der Shannoncode ist. X
- In einem mehrschichtigen Perzeptron reicht eine verdeckte Schicht nicht aus, um jede berechenbare stetige Funktion beliebig gut anzunähern. X

Wahr/Falsch-Fragen

- Bei evolutionären Algorithmen können die Eltern im Gegensatz zu den genetischen Algorithmen immer überleben. X
- Es gibt keinen Code C so dass $L(C) < L(S)$ wobei S der Shannoncode ist. X
- In einem mehrschichtigen Perzeptron reicht eine verdeckte Schicht nicht aus, um jede berechenbare stetige Funktion beliebig gut anzunähern. X
- Ein async. Hopfield-Netz erreicht immer nach endlich vielen Schritten einen stabilen Zustand.

Wahr/Falsch-Fragen

- Bei evolutionären Algorithmen können die Eltern im Gegensatz zu den genetischen Algorithmen immer überleben. X
- Es gibt keinen Code C so dass $L(C) < L(S)$ wobei S der Shannoncode ist. X
- In einem mehrschichtigen Perzeptron reicht eine verdeckte Schicht nicht aus, um jede berechenbare stetige Funktion beliebig gut anzunähern. X
- Ein async. Hopfield-Netz erreicht immer nach endlich vielen Schritten einen stabilen Zustand. ✓

Wahr/Falsch-Fragen

- Ein Kanal ist deterministisch falls $H(Y|X) = 0$.

Wahr/Falsch-Fragen

- Ein Kanal ist deterministisch falls $H(Y|X) = 0$. ✓

Wahr/Falsch-Fragen

- Ein Kanal ist deterministisch falls $H(Y|X) = 0$. ✓
- Der Kullback-Leibler-Abstand $D(p\|q)$ ist eine Metrik.

Wahr/Falsch-Fragen

- Ein Kanal ist deterministisch falls $H(Y|X) = 0$. ✓
- Der Kullback-Leibler-Abstand $D(p\|q)$ ist eine Metrik. X

Wahr/Falsch-Fragen

- Ein Kanal ist deterministisch falls $H(Y|X) = 0$. ✓
- Der Kullback-Leibler-Abstand $D(p\|q)$ ist eine Metrik. X
- In einem rekurrenten Neuronalen Netz kann es Rückwärtskanten geben.

Wahr/Falsch-Fragen

- Ein Kanal ist deterministisch falls $H(Y|X) = 0$. ✓
- Der Kullback-Leibler-Abstand $D(p\|q)$ ist eine Metrik. X
- In einem rekurrenten Neuronalen Netz kann es Rückwärtskanten geben. ✓

Wahr/Falsch-Fragen

- Ein Kanal ist deterministisch falls $H(Y|X) = 0$. ✓
- Der Kullback-Leibler-Abstand $D(p\|q)$ ist eine Metrik. X
- In einem rekurrenten Neuronalen Netz kann es Rückwärtskanten geben. ✓
- Der Lernalgorithmus für das einfache Perzeptron terminiert immer.

Wahr/Falsch-Fragen

- Ein Kanal ist deterministisch falls $H(Y|X) = 0$. ✓
- Der Kullback-Leibler-Abstand $D(p\|q)$ ist eine Metrik. X
- In einem rekurrenten Neuronalen Netz kann es Rückwärtskanten geben. ✓
- Der Lernalgorithmus für das einfache Perzeptron terminiert immer. X

Wahr/Falsch-Fragen

- Ein Kanal ist deterministisch falls $H(Y|X) = 0$. ✓
- Der Kullback-Leibler-Abstand $D(p\|q)$ ist eine Metrik. X
- In einem rekurrenten Neuronalen Netz kann es Rückwärtskanten geben. ✓
- Der Lernalgorithmus für das einfache Perzeptron terminiert immer. X
- Haben die Codeworte x_i mindestens den Abstand

$$d(x_i, x_j) \geq 2e + 1 \qquad \text{für alle } i \neq j$$

können f falsche Bits korrigiert werden, wenn $f \leq e$.

Wahr/Falsch-Fragen

- Ein Kanal ist deterministisch falls $H(Y|X) = 0$. ✓
- Der Kullback-Leibler-Abstand $D(p\|q)$ ist eine Metrik. X
- In einem rekurrenten Neuronalen Netz kann es Rückwärtskanten geben. ✓
- Der Lernalgorithmus für das einfache Perzeptron terminiert immer. X
- Haben die Codeworte x_i mindestens den Abstand

$$d(x_i, x_j) \geq 2e + 1 \qquad \text{für alle } i \neq j$$

können f falsche Bits korrigiert werden, wenn $f \leq e$. ✓

Wahr/Falsch-Fragen

- Reguläre Kodes sind stets dekodierbar.

Wahr/Falsch-Fragen

- Reguläre Kodes sind stets dekodierbar. X

Wahr/Falsch-Fragen

- Reguläre Kodes sind stets dekodierbar. X
- Bei der Übertragung von LZW kodierten Daten braucht das Wörterbuch oft den meisten Platz.

Wahr/Falsch-Fragen

- Reguläre Kodes sind stets dekodierbar. X
- Bei der Übertragung von LZW kodierten Daten braucht das Wörterbuch oft den meisten Platz. X

Wahr/Falsch-Fragen

- Reguläre Kodes sind stets dekodierbar. X
- Bei der Übertragung von LZW kodierten Daten braucht das Wörterbuch oft den meisten Platz. X
- Bei der Wavelettransformation findet kein Informationsverlust statt.

Wahr/Falsch-Fragen

- Reguläre Kodes sind stets dekodierbar. X
- Bei der Übertragung von LZW kodierten Daten braucht das Wörterbuch oft den meisten Platz. X
- Bei der Wavelettransformation findet kein Informationsverlust statt. ✓

Wahr/Falsch-Fragen

- Reguläre Kodes sind stets dekodierbar. X
- Bei der Übertragung von LZW kodierten Daten braucht das Wörterbuch oft den meisten Platz. X
- Bei der Wavelettransformation findet kein Informationsverlust statt. ✓
- Nach Dueck, Scheuer und Wallmeister ist das Simulierte Tempern dem Sintflut- und Schwellwert-Verfahren überlegen.

Wahr/Falsch-Fragen

- Reguläre Kodes sind stets dekodierbar. X
- Bei der Übertragung von LZW kodierten Daten braucht das Wörterbuch oft den meisten Platz. X
- Bei der Wavelettransformation findet kein Informationsverlust statt. ✓
- Nach Dueck, Scheuer und Wallmeister ist das Simulierte Tempern dem Sintflut- und Schwellwert-Verfahren überlegen. X

Aufgabe

Die Zeichenkette `BLABLABLABLA` soll binär kodiert werden.

1. Wie viele Bits enthält eine (zeichenweise) Huffman-Kodierung der Zeichenkette `BLABLABLABLA`?

2. Kodieren Sie die Zeichenkette `BLABLABLABLA` nach dem Lempel-Ziv-Welch-Verfahren (LZW). Die Kodierung des Eingabe-Alphabets sei durch

Eingabe	A	B	L
Ausgabe	0	1	2

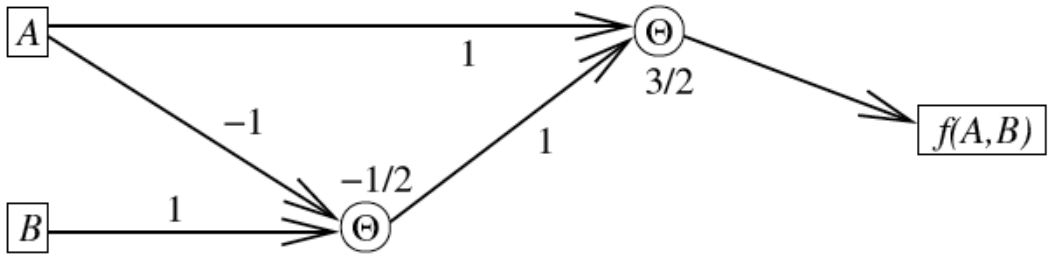
gegeben. Lassen Sie also weitere Kodierungen im Wörterbuch bei Kode-Nr. 3 anfangen:

Wie viele Bits benötigen Sie bei dieser LZW-Kodierung pro Zeichen des Ausgabealphabets, wenn die Ausgabe – wie üblich – im Blockkode (d. h. mit konstanter Kodelänge) kodiert wird?

Wie lang ist also die Kodierung der gesamten Zeichenkette?

Aufgabe

Welche boolesche Funktion f wird durch das folgende neuronale Netz modelliert?



Läßt sich f auch mit nur einem Neuron modellieren? (Wenn ja: Skizze! Wenn nein: Begründung!)
Sowohl für die Eingaben als auch für die Ausgabe des Netzes gelte:

$$\texttt{TRUE} \equiv 1 \qquad \text{und} \qquad \texttt{FALSE} \equiv 0$$

(Ein Neuron sei folgendermaßen beschaffen: ist die gewichtete Summe der Eingaben größer als sein Schwellwert Θ, so sei die Ausgabe des Neurons Eins; ist die gewichtete Summe $< \Theta$, so sei die Ausgabe Null; andernfalls ist die Ausgabe als undefiniert anzusehen!)

Quellen

Pajor, Vogel - Informatik 4 Tutorium SS2007

Prautzsch - Skript Informatik 4 SS2008

Wavelets for computer graphics: A primer
http://www.cis.udel.edu/~amer/CISC651/wavelets_for_computer_graphics_Stollnitz.pdf

Wikipedia

Noch Fragen?

Vorschau

Vorschau

- Wie wäre es, wenn einige von euch ein Informatik 3 Tutorium im WS2008/2009 halten würden?

Vielen Dank für eure zahlreiche Anwesenheit!