Softcomputing – Biologische Prinzipien in der Informatik

Neuronale Netze

Dipl. Math. Maria Oelinger
Dipl. Inform. Gabriele Vierhuff
IF TIF 08 – 2003

Überblick

- Motivation
- Biologische Grundlagen und ihre Umsetzung
- Aufbau Neuronaler Netze
- Bemerkungen zur Funktionsweise
- Fähigkeiten abstrakter Neuronen
- Lernarten
- Verschiedene Netztypen
- Zusammenfassung
- Referenzen

Motivation – 1

- digital $\leftrightarrow$ analog
- zentralisiert $\leftrightarrow$ verteilt
- sequentiell $\leftrightarrow$ parallel
- leicht interpretierbar $\leftrightarrow$ schwer interpretierbar

Motivation – 2

- Informationsverarbeitung von Lebewesen
  - adaptiv
  - massiv parallel
  - robust
  - nicht sehr präzise

- Netzparameter vom Netz selbst gefunden und eingestellt

- nicht alle Probleme lassen sich algorithmisch lösen

Biologische Grundlagen

- Dendriten
- Zellkörper
- Axon

Umsetzung biologischer Grundlagen

• Dendriten	→	• Eingabe
• Zellkörper	→	• Berechnung
• Axon	→	• Ausgabe

Aufbau Neuronaler Netze

- ein Neuron umfasst:
  - Eingabe
  - Gewichtung
  - Summation
  - Aktivierungsschwelle/-funktion
  - Ausgabe
- ein Neuronales Netz besteht aus
  - Schichten
  - gewichteten Verknüpfungen

Bemerkungen zur Funktionsweise

- jedes Neuron berechnet nur einen kleinen Teil
- Komposition primitiver Funktionen
- Verknüpfung $\rightarrow$ nacheinander mehrere Schritte berechnen
- Informationen im gesamten Netz speichern
  Aktivitätsmuster
  Kurzzeit: Neuronenaktivität
  Langzeit: Verknüpfungen, Gewichte

Bemerkungen zur Funktionsweise – 2

- Berechenbarkeitsmodell:
  primitive Funktionen
  Kompositionsregeln

- z.B. NOT, AND, OR

- Bei Neuronalen Netzen:
  primitiven Funktionen: Knoten
  (Aktivierungsfunktionen)
  Regeln: Vernetzung

Fähigkeiten abstrakter Neuronen

- synaptische Summation

  linearer Filter

  $x \rightarrow h = x * w \rightarrow y = f(h)$

- Klassifikation

  Raum durch Ebene geteilt

  linear separabel

Umsetzung Neuronaler Netze

- Mit Software auf von-Neumann simulieren
- Spezielle Hardware
- Vorteil Software:
  keine neue Hardware nötig
- Vorteil Hardware:
  höhere Verarbeitungsgeschwindigkeit
  auch für grosse Netze geeignet

Lernarten

- überwachtes Lernen
  Nachahmen
  Trainingsbeispiele, Lehrer

- unüberwachtes Lernen
  Exploration
  keine Trainingsbeispiele, kein Lehrer

- Reinforcement Lernen: halbüberwacht

Theorie des Lernens

- On-Line Lernen

- Batch Lernen

- Lernen und Generalisieren:

  E_D

  $E_\infty(w_{emp})$

  $E_\infty(w_{opt})$

Güte von Lernenmethoden

- Konvergenz: Fehler nicht wiederholen
- Konvergenzgeschwindigkeit
- Generalisierung
- Netzwerkkomplexität:
  Grosse Netze konvergieren schnell, generalisieren schlecht
  Kleine Netze konvergieren langsam, generalisieren gut

Verschiedene Netztypen

- überwacht
  - Perzeptron
  - Multi-Layer-Perzeptron MLP

- unüberwacht
  - Hebb
  - Self-Organizing Maps SOM

Perzeptron

- Funktionsweise
  - Merkmale als Eingabe
  - mehrere parallele Neuronen
  - am stärksten feuerndes Neurons
- Ziel: Bestimme die Gewichte so, dass die Klassen
  - $y(x) = \sum(w * x)$
  - eine Folge von Mustern x korrekt klassifizieren
- Konvergenz
- Grenzen: nicht linear separable Probleme

Multi-Layer-Perzeptron

- MLP = Mehrschicht-Perzeptron = Backpropagation-Netz
- erweitert/verzerrt den Merkmalsraum
- Berechnung
  - Vorwärtsphase
  - Rückwärtsphase
  - Gradientenschritt

Überblick

1. Vorwärtsphase:
   Berechne die Aktivität eines Neurons
   aus der Aktivität seiner Vorgängerneuronen

2. Rückwärtsphase:
   Berechne den Fehler eines Neurons
   aus den Fehlern, der nachfolgenden Neuronen

3. Gradientenschritt:
   Ändere die Gewichte entsprechend der Fehler
   der Neuronen

Fehlermaß beim MLP

$$E(w) = \frac{1}{2} \sum_{\alpha} (y^{\alpha} - y(x^{\alpha}, w))^2$$

- w, x und y sind Vektoren mit je α Dimensionen
- y^{α} ist die gewünschte Ausgabe
- $y(x^{\alpha}, w)$ ist die Ausgabe des Netzes bei Eingabe x^{α} und den Gewichten w

Aktivierung

Aktivierung s des $i$ten Neurons in Schicht l:
$s_i^l = \sigma(\sum_j w_{i,j}^{l,l-1} s_j^{l-1})$

$s_i^0 = x_i$
Erste Schicht: Nimm Eingabevektors x
als Aktivierung der Vorgängerneuronen

$s_i^k = y_i$
Aktivierungen der letzten Schicht ergibt Ausgabevektor

Hilfsgrößen

Berechne: $E_{i,j}^{l,l-1} = - \frac{\partial E(w)}{\partial w_{i,j}^{l,l-1}}$

Fehler zwischen Neuron i aus Schicht l
und Neuron j aus Schicht $l-1$,
Rückwärts durch das Netz reichen

Hilfsgrößen: $E_i^l = - \frac{\partial E(w)}{\partial s_i^l}$

$h_i^l = \sum_{j=1}^{n_{l-1}} w_{i,j}^{l,l-1} s_j^{l-1}$ ein
„Eingabe“ h_i^l für Neuron i in Schicht l

Rückwärtsphase

nach der Kettenregel gilt:

$$E_j^{l-1} = - \frac{\partial E}{\partial s_j^{l-1}}$$

$$= - \sum_k \frac{\partial E}{\partial s_k^l} \frac{\partial s_k^l}{\partial s_j^{l-1}}$$

$$= \sum_k E_k^l \sigma'(h_k^l) w_{k,j}^{l,l-1}$$

damit kann E_j^{l-1} aus E_k^l berechnet werden

Änderung der Gewichte

$$E_{i,j}^{l,l-1} = - \frac{\partial E}{\partial w_{i,j}^{l,l-1}}$$

$$= - \sum_k \frac{\partial E}{\partial s_k^l} \frac{\partial s_k^l}{\partial w_{i,j}^{l,l-1}}$$

$$= E_i^l \sigma'(h_i^l) s_j^{l-1}$$

Unüberwachtes Lernen: Hebb

- Vereinfachung der komplexen Umwelt
- Hauptkomponenten (PCA), Karhunen-Loewe-Entwicklung
- $\Delta w = \eta(x - w)y$

Self-Organizing Maps

- best match in Richtung des Stimulus verschieben
- Nachbarschft lernt mit
- Nachbarschaftsfunktion
- $\delta w_r = \eta h(||r - s(x)||)(x - w_r)$

Ausblick: rekursive Netze

- an alte Werte erinnern, zB Addition mit Übertrag
- Resultat wieder in das Netz eingespeist
- Signale für eine bestimmte Zeit festhalten,
  um sie wiederverwenden zu können

Neuronale Netze und Zelluläre Automaten

- Ähnliche Architekturen:
- Beide sind massiv parallel
- Lokalität von Daten
- SOMs: Nachbarschaft
- NN als Verallgemeinerung Zellulärer Automaten

Neuronale Netze und Evolutionäre Algorithmen

- Lernen in NN entspricht Optimierung:
  Fehlerfunktion des Netzes wird minimiert
- NN kann man durch EvA optimieren:
  normalerweise:
  Gleitkommawerte für die Gewichte und Schwellenwerte
  Codierung:
  zB jeden Netzparameter in 20 Bits codieren
  alle Parameter in eine Gesamtkette

Neuronale Netze und Evolutionäre Algorithmen – 2

- Ergebnisse werden besser, wenn die Trennstelle einzelne Netzparameter nicht zerschneidet

- Werte an einem Knoten gemeinsam behandeln

- Mutation der Netzparameter:
  Nicht bitweise
  Sondern durch Addition von Zufallszahlen

Zusammenfassung

- Warum Neuronale Netze?
  - adaptiv
  - massiv parallel
  - robust
  - nicht alle Probleme lassen sich algorithmisch lösen
- Funktionsweise
  - Eingabe
  - Berechung
  - Ausgabe
  - jedes Neuron berechnet nur einen kleinen Teil

Referenzen

Rojas, R.:
Theorie der neuronalen Netze – Eine systematische Einführung.
Springer. 1996