Der Backpropagation Algorithmus

Es sollen die Gewichte für ein beliebiges gerichtetes azyklisches Netz gelernt werden. Als Aktivierung wird die Sigmoid-Funktion verwendet.

Mittels Gradientenabstieg wird der quadratische Fehler minimiert. Da das Netz mehrere Ausgänge haben kann, werden die Teilfehler addiert:

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in outputs} (t_{kd} - o_{kd})^2$$

Die notwendige Anpassung des Gewichts ergibt sich wiederum aus:

$$\triangle w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} \qquad \text{mit} \qquad \frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} x_{ji}$$

$\delta_k = -\frac{\partial E_d}{\partial net_k}$ bezeichnet den Fehler in dem Knoten k.

Die Trainingsbeispiele liegen jeweils als Paare von Eingabewerten $\vec{x}$ und erwarteten Ergebniswerten $\vec{t}$ vor. $outputs$ ist die Menge der Ausgabeknoten. $downstream(k)$ ist die Menge aller Knoten zu denen eine gerichtete Verbindung von k aus besteht.

Der Backpropagation Algorithmus

Die Gewichte im Netz werden mit kleinen zufälligen Werten initialisiert.

Die folgenden Schritte werden wiederholt bis ein Zielkriterium erfüllt ist:

- Für **jedes** Trainingsbeispiel die Eingabe vorwärts durch das Netz propagieren
- den Fehler rückwärts durch das Netz propagieren
  - Für jeden Ausgabeknoten k

    $$\delta_k = (t_k - o_k) o_k (1 - o_k)$$

  - Für jeden Knoten h in einer versteckten Schicht

    $$\delta_h = o_h (1 - o_h) \sum_{k \in downstream(h)} \delta_k w_{kh}$$

- Alle Gewichte anpassen: $w_{ji} = w_{ji} + \triangle w_{ji}$ mit $\triangle w_{ji} = \eta \delta_j x_{ji}$

Modifikationen des Lernverfahrens

Ziel: bessere Konvergenz und vermeiden lokaler Minima

- Stochastischer Gradientenabstieg: Anpassung der Gewichte nach jedem Trainingsbeispiel
- Momentum ergänzen: $\triangle w_{ij}(n) = \eta \delta_i x_{ij} + \alpha \triangle w_{ij}(n-1)$
- Lernrate im Laufe des Trainings verringern
- Unterschiedliche Netze mit gleichen Trainingsdaten trainieren

Zielkriterium für das Lernenverfahren

Das Unterschreiten eines zuvor festgelegten Fehlerwertes auf den Trainingsdaten ist als Zielkriterium nicht geeignet. Während der Fehler auf den Trainingsbeispielen weiter fällt, kann er auf anderen Beispieldaten bereits wieder steigen – das Netz ist übertrainiert. Im Allgemeinen wird zum Testen eine separate Menge von Validierungbeispielen verwendet.

Das Verfahren darf aber auch nicht zu früh abgebrochen werden. Ein steigender Fehler auf den Validierungsdaten kann auch nur vorübergehend sein.

Lösungsidee: bestes bisher gelerntes Netz speichern; steigt der Validierungsfehler signifikant an – Abbruch; fällt der Validierungsfehler wieder – gespeichertes Netz ersetzen

Ausdrucksstärke vorwärtsverketteter Netze

- boolsche Funktionen mit 2-Schichten-Netz aus Perceptronen
- stetige Funktionen mit 2-Schichtennetz mit Sigmoid-Einheiten und linearer Einheit in den Ausgabeknoten
- beliebige Funktionen mit 3-Schichtennetz mit Sigmoid-Einheiten und linearer Einheit in den Ausgabeknoten

Charakterisierung des Lernenverfahren

- Hypothese: jede möglich Zuweisung von Gewichtswerten
- Hypothesenraum: $\mathbb{R}^n$, wobei n der Anzahl der Gewichte entspricht
- Suchverfahren: Gradientenabstieg
- Inductive Bias: stetige Interpolation von Zwischenwerten
- Leistungsfähigkeit: Approximation beliebiger Funktionen

Literatur

[1] T. M. Mitchell: *Machine Learning.* WCB/McGraw-Hill, 1997.

[2] G. Görz, C.-R. Rollinger, J. Schneeberger (Hrsg.): *Handbuch der Künstlichen Intelligenz.* 3., vollständig überarbeitete Auflage, Oldenbourg, 2000.

[3] P. Russel, S. Norvig: *Artificial Intelligence - A Modern Approach.* 2. Ed., Prentice Hall, 2003.

[4] G. F. Luger: *Künstliche Intelligenz - Strategien zur Lösung komplexer Probleme.* 4th Ed., Pearson, 2002.