Home | Startseite | Impressum | Kontakt | Gästebuch

**Aufgabe:** Entwickeln Sie einen hypothetischen Schachcomputer auf Basis eines neuronalen Netzwerks. Erläutern Sie, warum ein neuronales Netzwerk trotz sämtlicher Unwägbarkeiten immer noch ein deterministisches System ist.

**Lösung:** Komplexe Programmierung erfordert eine Vielzahl von Entscheidungen. In einem neuronalen Netzwerk wird das von den Sensoren bereitgestellte Eingangssignal **A** den Eingangsneuronen $A_i$ zugeführt, während das Ausgangssignal **B** anhand von Informationen der Ausgangsneuronen $B_j$ synthetisiert werden kann. Dabei sind

$$\mathbf{A} = \left(A_1, A_2, \ldots, A_m\right) \quad \text{und} \quad \mathbf{B} = \left(B_1, B_2, \ldots, B_n\right)$$

$m$- bzw. $n$-elementige Neuronenvektoren bestehend aus Nullen oder Einsen. Für $m = n$ ist ein solches Perzeptron in Abb. 1 dargestellt.

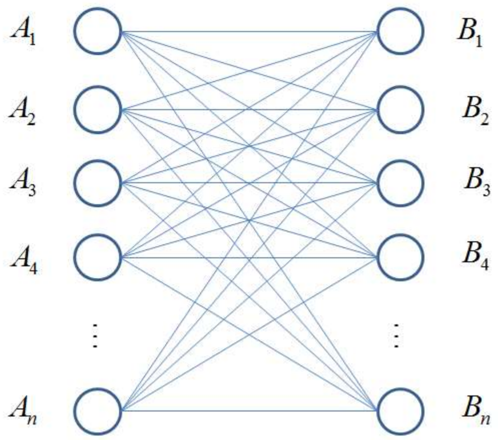


**Abbildung 1. Neuronales Netzwerk mit einer Ein- und Ausgabeschicht ohne verdeckte Schichten**

An der Schwelle zu **A** und **B** steht lediglich ein vorgeschaltetes Aktivierungspotential, das überschritten werden muß, damit das Neuron, wie man sagt, feuert und die Aktion auslöst. Die Grenze zwischen Feuern und Nichtfeuern, wahr oder falsch, entscheidet darüber, ob die Signalgröße bereits als wahr oder noch als falsch empfunden wird. Wenn eine Bedingung für ein Eingangsneuron nicht erfüllt ist, feuert dieses Neuron einfach nicht und braucht in der Betrachtung nicht weiter berücksichtigt werden. Gegebenenfalls ist dafür ein anderes Neuron aktiv, wobei die beiden sich gegenseitig ausschließen können.

Die außergewöhnliche Mächtigkeit eines neuronalen Netzwerks besteht aber darin, daß $2^n - 1$ unterschiedliche Neuronenmuster

$$\mathbf{A}_p = \left(A_{p1}, A_{p2}, \ldots, A_{pm}\right) \quad \text{und} \quad \mathbf{B}_q = \left(B_{q1}, B_{q2}, \ldots, B_{qn}\right)$$

von ein und demselben Netzwerk abgearbeitet werden können. Das entspricht einem klassischen Programmieraufwand von

 

$$\begin{aligned} \mathbf{A}_1 &\Rightarrow \mathbf{B}_1, \\ \mathbf{A}_2 &\Rightarrow \mathbf{B}_2, \\ &\vdots \\ \mathbf{A}_{2^n-1} &\Rightarrow \mathbf{B}_{2^n-1} \end{aligned}$$

Wenn-Dann-Verknüpfungen. Wie diese zustande kommen, läßt sich mit Hilfe der Binomialkoeffizientenentwicklung gemäß Tabelle 1 erklären.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | 1 |
| 2 | 1 | | | | | | | | 3 |
| 3 | 3 | 1 | | | | | | | 7 |
| 4 | 6 | 4 | 1 | | | | | | 15 |
| 5 | 10 | 10 | 5 | 1 | | | | | 31 |
| 6 | 15 | 20 | 15 | 6 | 1 | | | | 63 |
| 7 | 21 | 35 | 35 | 21 | 7 | 1 | | | 127 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | | $\vdots$ |
| $\binom{n}{1}$ | $\binom{n}{2}$ | $\binom{n}{3}$ | $\binom{n}{4}$ | $\binom{n}{5}$ | $\binom{n}{6}$ | $\binom{n}{7}$ | | $\binom{n}{n}$ | $2^n-1$ |

**Tabelle 1. Zahl der möglichen Zustände der Ausgangsneuronen in Abhängigkeit von der Zahl der Eingangsneuronen**

Dabei haben wir von der Definition

$$\binom{n}{k} = \begin{cases} \dfrac{n!}{k!(n-k)!} & \text{für} \quad 0 \le k \le n \\ 0 & \text{für} \quad 0 \le n < k \end{cases}$$

Gebrauch gemacht. Die zwei einfachsten Spezialfälle sind

$$\binom{n}{1} = n \quad \text{und} \quad \binom{n}{n} = 1.$$

Aus dem binomischen Satz

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k$$

folgt für $a = b = 1$:

$$2^n = \sum_{k=0}^{n} \binom{n}{k} \quad \text{bzw.} \quad 2^n - 1 = \sum_{k=1}^{n} \binom{n}{k} = \binom{n}{1} + \binom{n}{2} + \ldots + \binom{n}{n}$$

Mit $n$ Neuronen sind also durch parallele Ablaufsteuerung in einem einzigen Zeittakt maximal $2^n - 1$ unterschiedliche Reaktionen erzielbar, die aus allen möglichen Kombinationen der $n$

 

Einzelneuronen bestehen. Ein neuronales Netzwerk bietet demnach eine erhebliche Zeitersparnis hinsichtlich des Programmieraufwands, den eine klassische Vorgehensweise erfordern würde. Wenn man bedenkt, daß man nur jeweils 30 Neuronen verknüpfen muß, um damit 1.073.741.824 Zuweisungen vorzunehmen, dann erkennt man, welch unschätzbaren Vorteil diese Art von Programmierung beinhaltet. Ein sequentieller Programmierer, der diese Fallunterscheidungen manuell treffen müßte, würde länger als 34 Jahre ununterbrochen programmieren müssen, wenn ihn jeder einzelne Implikationsschritt eine Minute Zeit kosten würde, geschweige denn, daß ein Software-Tester diese Datenmenge in seinem Berufsleben jemals validieren könnte. Die Zeit, um dem System alle möglichen Fälle anzutrainieren, kann ein Designer neuronaler Netzwerke ebenfalls nicht erübrigen. Er wird sich daher auf eine erheblich geringere Auswahl an Trainingsmaterial beschränken müssen, in der Hoffnung, daß das Netzwerk die nicht trainierten Fälle selbständig herausfindet. Daß dies annähernd möglich ist, ist eine der verblüffendsten Eigenschaften neuronaler Netzwerke.[1] Diese sehen großzügig über Informationslücken hinweg und liefern in der Regel immer noch ein brauchbares, wenn auch nicht immer vorhersagbares Ergebnis.

Eine grundlegende Forderung an ein neuronales Netzwerk nach Zuverlässigkeit besteht also in der Eindeutigkeit der Ergebnisse, d.h. jede denkbare Kombination aus Eingangsneuronen $A_i$ darf ausgangsseitig auch nur einem Muster an Ausgangsneuronen $B_j$ entsprechen. Würde nämlich ein Eingangsmuster abwechselnd zwei verschiedene Ausgangsmuster erzeugen, so könnte man in der Tat von einem nichtdeterministischen System reden.[2] Wir sehen schon, daß unser neuronales Netzwerk ein endliches System ist und definitiv eine Lösung finden wird, was vielleicht auf Anhieb nicht sofort einsichtig ist.[3]

Schreiten wir nun zur Auslegung des Schachspiels als eines neuronalen Netzwerks: Ein Schachbrett hat 64 Felder, auf denen zu Beginn der Partie zweimal 16 Figuren aufgestellt sind, mit soundso vielen verschiedenen Möglichkeiten, die Figuren entsprechend vorgegebenen Regeln zu ziehen.

Beispiel: Wenn etwa die Dame bedroht wird, muß sie entweder abgezogen oder es muß eine andere Figur zu ihrer Deckung in Stellung gebracht werden. Zuzüglich darf die Dame nicht auf eine Stelle gesetzt werden, wo sie geschlagen oder eingekesselt werden kann. Das sind bereits vier verschiedene Forderungen, die zu erfüllen sind. Mit allen anderen Figuren verhält es sich ähnlich, wobei immer auf deren Wertigkeit zu achten ist. Es gibt allerdings Schachpartien,[4] die eine Reihe zwingend notwendiger Züge auslösen, wo unter gewaltigen eigenen Verlusten am Ende doch noch der Sieg herausgespielt wird.[5]

---

[1] Die Natur brauchte für das Ausreifen intelligenter Arten, zu denen nicht nur der Mensch zählt, gut 3 Milliarden Jahre.

[2] In der Praxis kann das bei weniger gut trainierten Systemen passieren, vor allem bei der sogenannten Backpropagation.

[3] Gegebenenfalls muß man eine oder mehrere verdeckte Schichten einbauen. Solche Probleme können auftreten, wenn das lineare Gleichungssystem nicht auf Anhieb eine Lösung besitzt.

[4] z.B. die italienische Variante

[5] Solche Zugabfolgen sind natürlich jedem Schachcomputer antrainiert worden.

 

Zu Beginn des Spiels gibt es nicht viele Möglichkeiten für den Anziehenden. Die weißen Bauern haben insgesamt 16 Möglichkeiten zu ziehen, die zwei Pferde vier. Das sind alles in allem 20 Möglichkeiten, das Spiel zu eröffnen. Wenn nur noch wenige Figuren am Platz sind, reduziert sich die Zahl der möglichen Züge ebenfalls. In der mittleren Phase des Spiels eröffnen sich die meisten Möglichkeiten, Figuren zu bewegen, von denen einige erkennbar ungünstig sind, weil sie zu Verlusten führen. Werden einem neuronalen Netzwerk nur die nachweislich erfolgversprechenden Züge antrainiert, wird es in dieser oder vergleichbaren Situationen auch nur die entsprechenden[6] Züge vorschlagen. Füttert man nun als Trainingsmaterial sämtliche[7] Meisterpartien[8] ein, die für die betreffende Farbe siegreich waren, erhält man ein Sammelsurium an erfolgversprechenden Möglichkeiten, an die sich das neuronale Netzwerk „erinnert", wobei es je nach Anlernzeit eine Spielstärke entwickelt, die bald von den besten Spielern der Welt nicht mehr übertroffen werden kann. Trotz der „unüberschaubaren" Zahl von Möglichkeiten, die das Schachspiel bietet, kann ein neuronales Netzwerk nur mit einer viel kleineren Zahl erfolgreicher Züge „abgerichtet" werden. Es müssen aber in jedem Fall so viele Neuronen vorhanden sein, daß die endlich vielen Entscheidungsmöglichkeiten des Spiels alle abgebildet werden können.

Auch wenn dem Netzwerk nun bestimmte Kombinationen nicht beigebracht wurden, so gibt es dennoch immer eine Lösung, d.h. im Ablauf der Zugfolge treten keine Elemente der leeren Menge auf. Zusätzlich sind alle Fälle reproduzierbar.[9] Manchmal allerdings kann sich das System nicht so recht für einen endgültigen Wert entscheiden, so daß Oszillationen zwischen zwei Werten vorkommen können. Auch wenn die Ergebnisse nicht in jedem Fall vorhersagbar sind, so ist ein neuronales Netzwerk dennoch völlig deterministisch. In der Praxis scheitert es meistens an der praktischen Vorausberechenbarkeit.[10]

Ein neuronales Netzwerk, auch wenn es unbestimmt ist, hat absolut keine Entscheidungsfreiheit.[11] Die Entscheidungen, die getroffen werden, hängen ausschließlich von den jeweils feuernden Eingangsneuronen ab sowie vom durchgeführten Training, das sich in den abgelegten Gewichten niederschlägt.[12] Mit oder ohne ausreichendes Training hat die aufgespannte Hyperfläche aber ein Minimum,[13] das sich beispielsweise durch die sogenannte Backpropagation approximieren läßt.

---

[6] d.h. die aus evolutionärer Sicht optimalen

[7] In der Regel wird man sich auf Meisterpartien beschränken, die besonders viele siegreiche Wendungen beinhalten.

[8] Unter den Schachspielern sind die Großmeister am fittesten.

[9] Bis auf Rundungsfehler, die allerdings immer möglich sind

[10] Das liegt in der Natur der Lösbarkeit linearer Gleichungssysteme, die nicht immer eindeutige Lösungen haben. Erst durch Einführung einer oder mehrerer verdeckter Schichten werden auch diese Systeme lösbar.

[11] Es gibt demnach auch keine autonomen Entscheidungen, alle Entscheidungen sind extern induziert und damit deterministisch. Der Begriff Autonomie bezieht sich lediglich darauf, wo sich die Instanz befindet, welche die Entscheidungen trifft. Das kann ein menschliches Gehirn oder ein mit künstlicher Intelligenz ausgestattetes Elektronengehirn sein.

[12] Die menschliche Entscheidungsfreiheit besteht lediglich darin, daß wir uns überlegen können, ob wir nach der nächsten folgenschweren Entscheidung noch leben werden. Oftmals geht diese Rechnung nicht auf, weil uns dafür die nötige Erfahrung fehlt.

[13] Dieses Minimum kann unscharf sein.

 

Ein Schachbrett mit 64 Feldern $P_1, \ldots, P_{64}$ erfordert für die eindeutige Zuordnung eines Feldes 6 Neuronen. Jeder Schachzug erfordert demnach zwei Positionsangaben, eine für die Anfangsposition der zu ziehenden Figur, eine zweite für die Endposition. Das sind zusammen 12 Neuronen zur Beschreibung eines einzelnen Schachzugs von $P_i$ nach $P_j$. Gezogen werden können nur die eigenen Figuren, und davon gibt es insgesamt 6 Klassen (Bauer, Turm, Läufer, Springer, Dame, König). Damit ist die Ausgangsseite vollständig beschrieben.

Eingangsseitig haben wir maximal 32 Figuren, unter denen 12 verschiedene Klassen vorkommen, die wir mit 4 Neuronen in jedem Fall unterscheiden können. Das entspricht etwas weniger als

$$2^4 \cdot 2^6 = 2^{10} = 1024$$

verschiedenen Stellungen, die sich mit 10 Neuronen eindeutig abbilden lassen.

Zur Generierung der neuronalen Eingangssignale können Fotos der jeweiligen Stellung von einer über dem Schachbrett aufgehängten Kamera dienen. Die Bilderkennung muß in der Lage sein, die entsprechende Figur auf dem jeweiligen Feld zu klassifizieren und aus dieser Information ein neuronales Muster erzeugen.

Die Trainingssignale können entweder durch ein Nachstellen der jeweiligen Konfiguration auf dem Brett gemäß dem oben beschriebenen Verfahren generiert werden oder durch synthetisch erzeugte Eingaben. Die Verknüpfung mit den Ausgangsneuronen geschieht über einen Satz gesammelter und sich über einen langen Zeitraum erstreckenden Meisterpartien, wo man genau den Zug auswählt, der auch den Meister zum Erfolg geführt hat.[14] Diese Fußarbeit, die Suche nach der optimalen Lösung, kann dem Architekten eines neuronalen Netzwerks nicht abgenommen werden, und dementsprechend lange kann sie dauern. In der Regel muß diese Arbeit daher auf viele Zuarbeiter verteilt werden, womit die Herstellung künstlicher Intelligenz selbstverständlich auch ihren Preis hat. Die Gewichte des neuronalen Netzwerks können nur mittels gleichzeitiger Verarbeitung sämtlicher Trainingsmuster ermittelt werden, die alle im Speicher des Rechners Platz haben müssen. Man stelle sich dazu ein lineares Gleichungssystem von $m$ Gleichungen mit $n$ Unbekannten vor, dessen $m \times n$ Koeffizienten aus allen $2^n - 1$ möglichen Neuronen-Kombinationen bestimmt werden müssen. Könnte man dies wirklich leisten, hätte man ein vollständig beschriebenes System vor sich, in dem jede mögliche Eingangskombination genau einer möglichen Ausgangskombination als Lösung entspricht. Ein derart an Erfahrungen reiches System wird man aber niemals haben, weil der Aufwand das Menschenmögliche übersteigt. Der Nutzen neuronaler Netzwerke erschließt sich demgemäß daraus, wieviel kumulatives Vorwissen aufgrund gemachter positiver Erfahrungen bereits vorhanden ist, das zur Assoziation bei anstehenden Entscheidungen herangezogen werden kann.

[14] Hiermit ist dem Evolutionsprinzip in einem neuronalen Netzwerk Genüge getan.

 