Handbuch: 7.2.3. Rekurrente Netze

Die grundlegende Idee eines rekurrenten neuronalen Netzes ist es, die Zukunft abhängig zu machen von der Vergangenheit, und zwar durch eine Form der Funktion, die dem Perceptron-Modell ähnelt. Ein sehr einfaches rekurrentes Modell könnte beispielsweise sein: $x(t+1) = a(W \cdot x(t) + b)$. Beachten Sie bitte sorgsam drei wichtige Eigenschaften dieser Gleichung, die sie von dem zuvor besprochenen Perceptron unterscheiden: (1) Sowohl Eingaben als auch Ausgaben sind jetzt keine Wert-Vektoren mehr, sondern zeitabhängige Funktions-Vektoren; (2) es gibt keine getrennten Eingaben und Ausgaben; vielmehr sind Eingaben und Ausgaben identisch, aber zu unterschiedlichen Zeiten; und (3) weil es sich um zeitabhängige rekurrente (d.h. wiederholbare) Beziehungen handelt, benötigen wir zur Beurteilung gewisse Anfangsbedingungen, den Zustand $x(0)$.

Das Netz, dass wir einsetzen wollen, verwendet das Konzept eines Reservoirs, was im Wesentlichen nur aus einem Satz Knotenpunkte besteht, die irgendwie miteinander verbunden sind. Die Verbindung zwischen diesen Knotenpunkten wird durch eine Gewichts-Matrix W zum Ausdruck gebracht, die auf bestimmte Weise initialisiert wird. Im Allgemeinen wird sie völlig zufallsbedingt initialisiert, aber von großem Vorteil ist es, wenn sie mit Hilfe einer gewissen Struktur initialisiert wird. Gegenwärtig bedarf es noch einer gewissen Zauberei, die richtige Struktur dafür zu bestimmen, in jedem Fall hängt sie von dem zu lösenden Problem ab. Der augenblickliche Zustand jedes Knotenpunktes im Reservoir wird in einem Vektor $x(t)$ abgespeichert, der ebenfalls zeitabhängig ist. Es ist die Zeitreihe, die wir tatsächliche messen können und die wir voraussagen wollen. Der Eingabeprozess wird gesteuert durch eine Eingabe-Gewichts-Matrix W^{in}, welche die Eingabe-Vektor-Werte jedem gewünschten Neuron im Reservoir zur Verfügung stellt. Die Ausgabe des Reservoirs wird gegeben als Vektor $y(t)$. Der Systemzustand entwickelt sich im Laufe der Zeit gemäß $x(t+1) = f(W \cdot x(t) + W^{in} \cdot u(t+1) + W^{fb} \cdot y(t))$, wobei W^{fb} eine Feedback-Matrix ist, die optional ist für solche Fälle, bei denen wir das Ausgabe-Feedback wieder zurück ins System einspeisen wollen, und wobei $f(...)$ eine Sigmoidfunktion ist, gewöhnlich $tanh(...)$.

Die Ausgabe $y(t)$ wird vom erweiterten Systemzustand $z(t) = [x(t); u(t)]$ berechnet, unter Verwendung von $y(t) = g(W^{out} \cdot z(t))$ wobei W^{out} eine Ausgabe-Gewichts-Matrix darstellt und $g(...)$ eine sigmoidale Aktivierungsfunktion ist, z.B. $tanh(...)$.

Die Eingabe und Feedback-Matrizen sind Teil der Problemspezifikation und müssen deshalb dem Nutzer zur Verfügung gestellt werden. Die interne Gewichtsmatrix wird irgendwie initialisiert und bleibt danach unberührt. Erfüllt die Matrix W bestimmte komplizierte Bedingungen, so hat das Netz sogenannte Echo-State-Eigenschaft. Das heißt, dass die Vorhersagen irgendwann unabhängig von den Anfangsbedingungen sind. Das ist von entscheidender Bedeutung, weil es dann keine Rolle mehr spielt, zu welcher Zeit die Modellierung beginnt. Solche Netze sind theoretisch in der Lage, nahezu jede Funktion (mit bestimmten technischen Voraussetzungen) darzustellen, wenn sie korrekt angeordnet sind.