Spikende Neuronale Netze - Theorie und Simulation

Rene Mayrhofer

Februar 2001

Inhaltsverzeichnis

1 Einleitung

Neuere biologische Untersuchungen zur Funktionsweise natürlicher neuronaler Netze zeigen nun mit immer höherer Wahrscheinlichkeit, dass bei der Übertragung von Information zwischen Neuronen nicht nur die Rate der abgefeuerten Impulse (*Post Synaptic Potential, PSP, Spike*), sondern noch andere Faktoren eine Rolle spielen. Dadurch ist es möglich, mit Hilfe eines einzelnen Spikes Information in der Größenordnung mehrerer Bits zu übertragen (vgl. [1], Seite 17). Die dazu verwendeten Kodierungsarten sind ([2]):

- Feuerrate: Die eigentliche Information ist in der Feuerrate der Neuronen enthalten, also im Durchschnitt der pro Zeiteinheit abgefeuerten Spikes.
- temporale Kodierung: Die exakten Feuerzeitpunkte der Spikes tragen die Information.
- Populationskodierung: Der Code wird durch die Aktivität verschiedener Populationen von Neuronen gebildet.

Ansätze bisheriger künstlicher neuronaler Netze (*Artificial Neural Networks, ANNs*) gingen nur auf die erste Art der Kodierung ein, wobei die Feuerrate üblicherweise durch einfache reelle Zahlen modelliert wird. Allerdings zeigte sich in jüngsten Experimenten, dass diese Art der Kodierung nicht ausreichend ist, um genügend Information für eine visuelle Mustererkennung innerhalb von 150 ms zu übertragen. Da der Mensch aber durchaus dazu fähig zu sein scheint, liegt der Schluss nahe, dass für schnelle Informationsverarbeitung zeitliche Kodierung entscheidend ist. Aus diesem Grund wurden Spikende Neuronale Netze (*Spiking Neural Networks, SNNs*) entwickelt, die Gegenstand dieser Arbeit sind.

Bei SNNs steht die temporale Kodierung im Vordergrund, es werden explizit die einzelnen von Neuron zu Neuron übertragenen Spikes modelliert. Bis jetzt ist nur wenig über die sich dadurch bietenden Möglichkeiten bei der Simulation und möglichen Implementierung von SNNs in Hardware bekannt. Folgende wichtige Fragestellungen müssen geklärt werden, um SNNs sinnvoll in der Praxis einsetzen zu können (vgl. [2]):

- Welchen Einfluss haben bestimmte Eigenschaften biologischer neuronaler Netze auf die Berechnungspotenz von SNNs ?
- Wie kann Lernen mit SNNs realisiert werden ?
- Wie kann das Timing einzelner Spikes dazu benutzt werden, Muster abzuspeichern und zu analysieren, oder um selbst-organisierendes Verhalten zu erreichen ?

Diese Fragen werden zum Teil in [2] beantwortet, weshalb sich die ersten Kapitel dieser Arbeit zu einem großen Teil darauf stützen. Einige Kapitel wurden direkt als Zusammenfassung der entsprechenden Kapitel aus [2] übernommen, allerdings mit leichten Anpassungen an das hier verwendete mathematische Modell und die Konventionen für Variablenbezeichnungen.

2 Mathematisches Modell

Auf eine Beschreibung der biologischen Modelle natürlicher Neuronen wird in dieser Arbeit verzichtet, da dies bereits mehrmals sehr ausführlich dokumentiert wurde, so z.B. auch in [2].

Im folgenden Dokument wird diese Definition eines SNN verwendet, welche in [3] eingeführt wurde. Diese Definition ist allgemein gehalten, da die Schwellwertfunktion der Neuronen sowie die Antwortfunktion der Synapsen offen gelassen wurden.

Ein SNN besteht aus:

- *einem endlichen gerichteten Graphen (V, E) (die Elemente von V werden als "Neuronen", die Elemente von E als "Synapsen" bezeichnet)*

- *einer Teilmenge $V_{in} \subseteq V$ von Eingangsneuronen*
- einer Teilmenge $V_{out} \subseteq V$ von Ausgangsneuronen
- für jedes Neuron $v \in V - V_{in}$ eine Schwellwertfunktion $\Theta_v : R^+ \to R \cup \{\infty\}$
- für jede Synapse $\langle u, v \rangle \in E$ eine Antwortfunktion $\varepsilon_{u,v} : R^+ \to R$ sowie ein Gewicht $w_{u,v} \in R^+$

Wir nehmen an, dass das Feuern der Eingangsneuronen $v \in V_{in}$ von außerhalb bestimmt wird, also die Mengen $F_v \subseteq R^+$ von Feuerzeitpunkten (spike trains) für die Neuronen $v \in V_{in}$ als die Eingabe (Input) des SNN gegeben werden.

Für ein Neuron $v \in V - V_{in}$ wird die Menge F_v von Feuerzeitpunkten rekursiv definiert. Das erste Element von F_v ist $inf\{t \in R^+ : P_v(t) \geq \Theta_v(0)\}$, und für irgend ein $s \in F_v$ ist das nächst größere Element von F_v gleich $inf\{t \in R^+ : t > s \, and \, P_v(t) \geq \Theta_v(t-s)\}$, wobei die Potentialfunktion $P_v : R^+ \to R$ definiert ist als

$$P_v(t) := \sum_{\substack{u \in V \\ \langle u,v \rangle \in E}} \sum_{\substack{s \in F_u \\ s < t}} w_{u,v} \cdot \varepsilon_{u,v}(t-s)$$

Die sich daraus ergebenden Feuerzeitpunkte (spike trains) F_v der Ausgabeneuronen $v \in V_{out}$ werden als die Ausgabe (Output) des SNN interpretiert.

Die Komplexität einer Berechnung in einem SNN wird durch Zählen jedes einzelnen Spikes als Berechnungschritt ausgewertet.

Ist die Antwortfunktion $\varepsilon_{u,v}$ positiv, so spricht man von einem EPSP (*Excitatory PostSynaptic Potential*), ist sie negativ, von einem IPSP (*Inhibitory PostSynaptic Potential*). Weiters nimmt man an, dass $\varepsilon_{u,v}(t) = 0$ für $t \in [0, d_{u,v}]$, wobei $d_{u,v}$ die Übertragungsverzögerung zwischen der Generierung des Aktionspotentials (der Überschreitung der Schwelle Θ_v) und der anfänglichen Auswirkung des PSP (*PostSynaptic Potential*) auf das nächste Neuron v darstellt. $P_v(t)$ gibt die Auswirkungen aller ankommenden EPSPs und IPSPs an und wird bei Abwesenheit jeglicher Eingabe mit $P_v = 0$ als Ruhewert angenommen. Die Schwellwertfunktion Θ_v schließlich kann dazu verwendet werden, um die bei biologischen Netzwerken vorhandenen Abhängigkeit des Schwellwertes vom Feuern des Neurons abzubilden. Üblicherweise werden dazu zwei Konstante $\tau_{v,ref}$ und $\tau_{v,end}$ mit $0 < \tau_{v,ref} < \tau_{v,end}$ eingeführt, wobei der Schwellwert $\Theta_v(t)$ unmittelbar nach dem Feuern, also für $t < \tau_{v,ref}$ (*absolute refractory period*), mit ∞ angenommen wird, um ein erneutes Feuern innerhalb dieser Zeitspanne zu unterbinden (nach dem Vorbild der biologischen Neuronen). Im Intervall $\tau_{v,ref} < t < \tau_{v,end}$ (*relative refractory period*) nähert sich die Schwellwertfunktion rapide dem konstanten Wert $\Theta_v(0)$ an, sodass $\Theta_v(t) = \Theta_v(0)$ für alle $t > \tau_{v,end}$.

Mathematisch interpretiert berechnet das Netzwerk eine Funktion, welche einen Vektor von Feuerzeitpunkten $\langle F_v \rangle_{v \in V_{in}}$ auf einen anderen Vektor von Feuerzeitpunkten $\langle F_v \rangle_{v \in V_{out}}$ abbildet. Allerdings sind die Netzein- und -ausgaben Vektoren von Zeitserien anstelle von Vektoren von Zahlenwerten wie in herkömmlichen Neuronalen Netzen. Dies könnte wichtig für das Verständnis der Berechnung in biologischen Neuronalen Netzen sein, welche schnellstmöglich auf sich ständig ändernde Umgebungen reagieren müssen. Da die künstliche Variante dieser Netzwerke, die SNNs, ebenfalls gut für diese Art der zeitabhängigen Signalverarbeitung geeignet sind, ist dieses Anwendungsgebiet von besonderem Interesse für das Design von SNNs (aus [7]).

3 Berechnungspotenz

In [2] werden folgende Kernaussagen bezüglich der Berechnungspotenz von SNNs verschiedener Typen aufgestellt und bewiesen. Auf die eigentlichen Beweise der Aussagen wird hier verzichtet, da sie für das Verständnis der später folgenden Kapitel nicht entscheidend sind.

Der folgende verwendete Begriff der Simulation in Echtzeit bedeutet, dass eine Maschine M durch eine Maschine M' derart simuliert wird, dass jeder Rechenschritt von M nur eine konstante Anzahl von Rechenschritten von M' benötigt.

- Es gibt Funktionen, welche von einem einzigen spikenden Neuron berechnet werden können, bei der Verwendung von ANNs mit Schwellwert- bzw. Sigmoidfunktionen jedoch eine große Anzahl von versteckten Neuronen benötigen würden.

- Es ist möglich, ein SNN endlicher Größe mit entsprechenden rationalen Gewichtswerten aus [0,1] zu konstruieren, das eine beliebige Turing-Maschine in Echtzeit simuliert, sofern die Antwort- und Schwellwertfunktionen des SNNs bestimmte Eigenschaften erfüllen (hauptsächlich die Eigenschaft, dass die von den Neuronen ausgesandten Impulse, also die PSPs, kleine Segmente mit monoton steigender Charakteristik enthalten müssen).

- Ein SNN mit stückweise konstanten Antwort- und Schwellwertfunktionen ist für Berechnungen mit beschränkten, reell-wertigen Ein- und Ausgaben echtzeit-äquivalent zu N–RAMs.
  Bemerkung: N–RAMs sind Random Access Machines (RAMs) mit einer konstanten Anzahl von Eingabe-, Ausgabe- und Arbeits-Registern und einem ausgezeichneten Register als Akkumulator. Sie können beliebige endliche Programme zusammengesetzt aus den Befehlen ADD(b), SUBTRACT(b), IF COMPARE(R) THEN GOTO, GOTO, LOAD(R), STORE(R) und HALT ausführen, wobei LOAD und STORE auf dem Akkumulator arbeiten und ADD sowie SUBTRACT nur konstante Werte zum Akkumulator addieren bzw. vom Akkumulator subtrahieren können. Für eine genaue Definition siehe [2].

- SNNs mit stückweise konstanten Antwortfunktionen und stückweise monotonen und kontinuierlichen Schwellwertfunktionen sind äquivalent zu SNNs mit stückweise konstanten Antwort- und Schwellwertfunktionen.
  Die letzten beiden Aussagen zeigen sehr deutlich, dass die Einschränkung der Antwort- und Schwellwertfunktionen auf stückweise konstante Funktionen zu einer sehr deutlichen Reduktion der Berechnungspotenz von SNNs führt. Daher werden eher stückweise lineare Funktionen verwendet, durch welche diese Reduktion nicht auftritt.

- SNNs mit stückweise konstanten Antwortfunktionen und stückweise linearen Schwellwertfunktionen mit rationalen Parametern sind für boolsche Eingabewerte echtzeit-äquivalent zu endlichen Automaten (Deterministic Finite Auomatas, DFAs).

- Kein SNN mit stückweise konstanten Antwortfunktionen und stückweise monotonen und kontinuierlichen Schwellwertfunktionen kann eine Turing-Maschine in polynomialer Zeit simulieren. Dies bedeutet, dass die Berechnungspotenz für boolsche Eingabewerte (Bit Strings) und der bereits genannten Einschränkung der Antwortfunktion auf stückweise konstante Funktionen noch weiter sinkt.

Aufgrund dieser Schlüsse werden im weiteren Verlauf dieser Arbeit Netzwerktypen verwendet, welche die oben aufgeführten Einschränkungen nicht aufweisen und somit echtzeit-äquivalent zu Turing-Maschinen sind. Die einfachsten für die Antwort- und Schwellwertfunktionen möglichen Funktionen, um die Echtzeit-Turing-Äquivalenz aufrecht zu erhalten, sind stückweise linear.

4 Elementare Berechnungen

Bis jetzt ist wenig über die Berechnungsmöglichkeiten von SNNs unter Rückgriff auf das Timing einzelner Spikes bekannt. Allerdings ist für ANNs das sogenannte "Sigmoidal Gate", ein Neuron mit der Sigmoidfunktion als Aktivierungsfunktion, der meist verwendete Baustein für Berechnungen. Ein "Sigmoidal Gate" verwendet die Gewichtswerte w_i, $1 \leq i \leq m-1$ und einen "Bias" w_m, bekommt die Werte $s_i, ..., s_{m-1} \in [0, \gamma]$ als Eingabe und berechnet die Funktion $\sigma\left(\sum_{i=1}^{m-1} w_i s_i + w_m\right)$. Die Aktivierungsfunktion σ wird Sigmoidfunktion genannt und ist monoton, auf $[0, \gamma]$ beschränkt und nicht konstant. Oft wird die Sigmoidfunktion stückweise linear definiert:

$$\pi_\gamma(y) = \begin{cases} 0, & wenn \quad y < 0 \\ y, & wenn \quad 0 \leq y \leq \gamma \\ \gamma, & wenn \quad y > \gamma \end{cases}$$

In den folgenden Unterkapiteln wird auf die Berechnung einfacher Funktionen hingeführt, wobei der Berechnung einer gewichteten Summe aufgrund der Anwendung in "Sigmoidal Gates" spezielle Bedeutung zukommt. Wenn ein "Sigmoidal Gate" mittels spikender Neuronen simuliert werden kann, können die bekannten Netzwerktypen dadurch automatisch auf SNNs angewandt werden, auch wenn die Lernregeln nicht direkt übertragbar sind. Einige Unterkapitel wurden dabei in Anlehnung an die entsprechenden Kapitel aus [2] und [7] dargestellt, wobei hier in einigen Bereichen nur Zusammenfassungen präsentiert werden und auf die Details verzichtet wird.

4.1 Simulation von McCulloch-Pitts Neuronen

Ein McCulloch-Pitts Neuron ist der einfachste, sinnvoll anwendbare Typ von Neuronen. Es hat reelle Gewichte $\alpha_{u,v}$, einen Schwellwert Θ und empfängt n binäre oder reell-wertige Eingangswerte $x_1, ..., x_n$. Sein Ausgang ist definiert als

$$\begin{cases} 1, & wenn \quad \sum_{i=1}^{n} \alpha_{u,v} \cdot x_u \geq \Theta \\ 0, & sonst \end{cases}$$

Die Verwendung von Sigmoidfunktionen zum Erzeugen eines reell-wertigen Ausganges anstelle der Fallunterscheidung erlaubt es gegenüber den einfacheren McCulloch-Pitts Neuronen nicht nur, reell-wertige Funktionen zu berechnen, sondern erhöht auch die Berechnungspotenz für binäre Funktionen.

Man sieht sofort aus der Definition eines SNN, dass ein spikendes Neuron ein McCulloch-Pitts Neuron mit binären Eingängen simulieren kann (folgende Konstruktion aus [7], jedoch erweitert für das in dieser Arbeit verwendete, etwas komplexere, mathematische Modell). Dazu nehmen wir an, dass alle Antwortfunktionen $\varepsilon_{u,v}$ der Synapsen identisch mit Ausnahme ihres Vorzeichens (welches wir negativ für $\alpha_{u,v} < 0$ und positiv für $\alpha_{u,v} > 0$ wählen) sind und alle presynaptischen Neuronen u, die überhaupt feuern, zum selben Zeitpunkt $t_u = T_{input}$ feuern. In diesem Fall kann man aus der Definition des ersten Element von F_v ableiten, dass das spikende Neuron v genau dann und nur dann feuert, wenn

$$\sum_{u\, feuert\, zur\, Zeit\, T_{input}} w_{u,v} \cdot \varepsilon_{u,v} \geq \Theta_v(0)$$

wobei $\varepsilon_{u,v}$ der Extremwert von $\varepsilon_{u,v}(s)$ (also $\varepsilon_{u,v} = \max_s(\varepsilon_{u,v}(s))$ wenn $\varepsilon_{u,v}$ einen EPSP darstellt bzw. $\varepsilon_{u,v} = \min_s(\varepsilon_{u,v}(s))$ wenn $\varepsilon_{u,v}$ einen IPSP darstellt), $w_{u,v} > 0$ ein Synapsengewicht und $\Theta_v(0)$ der konstante Schwellwert des Neurons v bei Abwesenheit von unmittelbar zuvor gefeuerten Impulsen sind. Dann kann das spikende Neuron v mit $w_{u,v} := \alpha_{u,v}/\varepsilon_{u,v}$ jedes beliebige McCulloch-Pitts Neuron nach obiger Definition simulieren, wenn die binären Eingabewerte $x_1, ..., x_n$ durch das Feuern bzw. Nicht-Feuern der presynaptischen Neuronen $u = 1, ...n$ zum gemeinsamen Zeitpunkt T_{input} kodiert werden und der binäre Ausgang durch das Feuern bzw. Nicht-Feuern des Neurons v im relevanten Zeitfenster kodiert wird.

Allerdings zeigt sich bei näherer Betrachtung, dass es bedeutend schwieriger ist, diese Simulation von McCulloch-Pitts Neuronen durch spikende Neuronen für Netzwerke mit mehreren Schichten durchzuführen. Dies rührt daher, dass der exakte Feuerzeitpunkt des Neurons v von seinem konkreten Eingabevektor $x_1, ..., x_n$ abhängt. Ist die Summe $\sum_{u\, feuert\, zur\, Zeit\, T_{input}} w_{u,v} \cdot \varepsilon_{u,v} - \Theta_v(0)$ deutlich über 0, so wird der Schwellwert $\Theta_v(0)$ früher überschritten als bei einem Eingabevektor, dessen Summe $\sum_{u\, feuert\, zur\, Zeit\, T_{input}} w_{u,v} \cdot \varepsilon_{u,v} - \Theta_v(0)$ zwar positiv, aber nahe bei 0 liegt. Daher würden die spikenden Neuronen der ersten Schicht in einer Simulation nach oben beschriebener Konstruktion im allgemeinen Fall zu leicht unterschiedlichen Zeitpunkten feuern,

auch wenn alle Eingabeneuronen zum selben Zeitpunkt T_{input} feuern würden. Durch den Verlust der Synchronität nach bereits einer Schicht ist die Konstruktion von mehrschichtigen Netzwerken nach diesem Modell also unmöglich, da die Eingabewerte für die zweite Schicht die erforderliche Eigenschaft der Synchronität bereits nicht mehr aufweisen.

Um dies auszugleichen, werden separate Synchronisationsmechanismen benötigt, wie z.B. zusätzliche spikende Neuronen. Anstatt ein Neuron v wie oben beschrieben direkt zum Feuern zu bringen, wird dann sichergestellt, dass es bei $\sum_{u\, feuert\, zur\, Zeit\, T_{input}} w_{u,v} \cdot \varepsilon_{u,v} \geq \Theta_v(0)$ nicht durch die zusätzlichen spikenden Neuronen am Feuern gehindert wird. Diese zusätzlichen hindernden Neuronen werden direkt mit den Eingabeneuronen verbunden, die den Eingabevektor $x_1, ..., x_n$ durch ihr Feuern bzw. Nicht-Feuern kodieren, wobei die anregende Kraft für das Feuern des Neurons v nun von Neuronen ausgeht, die unabhängig von der Eingabe sind. Mit dieser in [3] genauer beschriebenen Konstruktion ist es prinzipiell möglich, jedes beliebige boolsche Schaltnetz und in der Abwesenheit von Rauschen sogar jede beliebige Turing-Maschine zu simulieren.

Allerdings ist es auch möglich, eine Beinahe-Synchronisation der Feuerzeitpunkte mit Hilfe einer gemeinsamen Hintergrunderregung, z.B. einer anregenden Hintergrundoszillation $\sin(\omega t)$, welche zu den Potentialen aller spikenden Neuronen addiert wird, zu erreichen. Durch geeignete Einstellung aller Parameter kann so erreicht werden, dass die Schwellwerte aller Neuronen nur dann überschritten werden können, wenn die Hintergrundoszillation nahe ihres Maximums ist (für eine genaue Beschreibung vgl. [8]). Es zeigt sich auch, dass Berechnungen mit binären Kodierungen höchstwahrscheinlich in solchen biologischen neuronalen Systemen vorkommen, die eine stark oszillierende Komponente aufweisen.

Allerdings muss die Frage gestellt werden, ob eine erzwungene Synchronisation in allen Fällen die bessere Wahl darstellt, da in den kleinen Unterschieden zwischen den Feuerzeitpunkten wertvolle Information kodiert ist, welche durch die Synchronisation zerstört wird. Es könnte sinnvoll sein, mit asynchronen oder lose synchronisierten Systemen zu arbeiten.

4.2 Berechnungen mit temporaler Kodierung

Bei dieser Art der Kodierung werden reelle Zahlen in den exakten Feuer- bzw Ankommenszeitpunkten einzelner Spikes kodiert, was besonders für schnelle Informationsverarbeitung große Vorteile bietet. Ein einzelner Spike trägt bereits die entscheidende Information, es ist nicht nötig, mehrere Spikes zu empfangen, bevor die Berechnung in einem Neuron durchgeführt werden kann (so wie dies z.B. bei Verwendung von Feuerraten, also der Anzahl von empfangenen Spikes in einem bestimmten Zeitfenster, nötig wäre).

4.2.1 Erkennung von Gleichheit

Ein spikendes Neuron kann zur Erkennung von Gleichzeitigkeit von ankommenden Spikes dienen, also bei Kodierung von reellen Zahlen in den Ankommenszeitpunkten der einzelnen Spikes erkennen, ob diese Zahlen den gleichen bzw. beinahe gleichen Wert haben. Für diese Operation findet sich keine Äquivalenz bei herkömmlichen neuronalen Netzen, bei welchen dies nur mit signifikantem Aufwand und unter Einsatz mehrere Neuronen bewerkstelligt werden kann.

Zur Konstruktion nehmen wir an, dass alle Neuronen $u = 1, ..., n$ direkte Vorgänger des Neurons v sind, alle die selbe Übertragungsverzögerung $d_{u,v}$ aufweisen und alle Synapsengewichte $w_{u,v} = 1$ für $u = 1, ..., n$ sind. Weiters nehmen wir an, dass die Antwortfunktionen $\varepsilon_{u,v}$ definiert sind als

$$\varepsilon_{u,v} := \frac{1}{1-(\tau_s/\tau_m)} \cdot \left[exp\left(-\frac{s-d_{u,v}}{\tau_m}\right) - exp\left(-\frac{s-d_{u,v}}{\tau_s}\right)\right] \cdot H(s-d_{u,v})$$

mit Zeitkonstanten $0 < \tau_s < \tau_m$. Ein EPSP von derartiger Form besteht aus einer annähern linear ansteigenden Phase für kleine s, exponentiellem Abfall für große s und einem glatten Übergang zwischen beiden Phasen, wenn er den maximalen Wert erreicht. Für alle beliebigen Werte von τ_s und τ_m mit $\tau_s < \tau_m$ lassen sich Konstante $0 < c_1 < c_2$ und $\Theta_v(0)$ finden, sodass $P_v(t) < \Theta_v(0)$ für beliebige Eingaben bestehend aus einer beliebigen Anzahl von EPSPs, deren zeitliche Distanz $\geq c_2$ und $P_v(t) \geq \Theta_v(0)$ für zwei EPSPs mit zeitlicher Distanz $\leq c_1$.

Wenn nun z.B. n reelle Zahlen $x_1, ..., x_n$ durch die Feuerzeitpunkte der n direkten Vorgängerneuronen von v kodiert werden, und das Feuern von v als "1" bzw. das Nicht-Feuern von v als "0" interpretiert wird, dann berechnet das Neuron v die partielle Funktion $ED_n : R^n \to \{0, 1\}$:

$$ED_n(x_1, ..., x_n) = \begin{cases} 1, & wenn\ \exists u \neq u' : |x_u - x_{u'}| \leq c_1 \\ 0, & wenn\ \forall u \neq u' : |x_u - x_{u'}| \geq c_2 \\ undefiniert, & sonst \end{cases}$$

Andererseits benötigt ein beliebiges vorwärtsgekoppeltes, in Schichten strukturiertes neuronales Netzwerk, bestehend aus McCulloch-Pitts Neuronen oder sigmoidalen Gattern mit beliebigen Gewichts- und Schwellwerten mindestens $\log(n!) \geq \frac{n}{2} \cdot \log(n)$ Gatter in der ersten Schicht, um ED_n berechnen zu können. Ein Beweis dieser Behauptung kann in [11] nachgelesen werden.

4.2.2 Radiale Basisfunktionen (RBF)

Wenn statt gleicher Übertragungsverzögerungen $d_{u,v}$, wie im vorherigen Kapitel verwendet, für jedes presynaptischen Neuron u verschiedene Verzögerungen $d_{u,v}$ zum Neuron v verwendet werden, kann die Konstruktion zur Erkennung von Gleichheit erweitert werden, um komplexere temporale Muster zu erkennen. Diese Verzögerungen $d_{u,v}$ repräsentieren eine neue Art von Parametern, welche ebenfalls keine Entsprechung bei herkömmlichen neuronalen Netzen haben. In verschiedenen biologischen neuronalen Netzen scheint es, dass diese Verzögerungen durch Lernen angepasst werden. Zusätzlich dazu können auch die Gewichte $w_{u,v}$ und der Schwellwert $\Theta_v(0)$ angepasst werden, um spezifische temporale Muster besser erkennen zu können. Im Extremfall kann $\Theta_v(0)$ so weit angehoben werden, dass alle Spikes der presynaptischen Neuronen u beinahe simultan ankommen müssen, um das Neuron v zum Feuern zu bringen. In diesem Fall verhält sich das Neuron v wie eine RBF-Einheit herkömmlicher neuronaler Netzwerkmodelle. Es wird nur dann feuern, wenn alle presynaptischen Neuronen u zu Zeitpunkten t_u feuern, sodass für eine Konstante T_{input} für alle u gilt, dass $t_u \approx T_{input} - d_{u,v}$, wobei der Vektor $\langle d_{u,v} \rangle$ die Rolle des Zentrums der RBF-Einheit einnimmt (vgl. [8]).

In [8] wird auch aufgezeigt, dass es vorteilhaft sein kann, die externen sensorischen Inputs y_u logarithmisch zu kodieren als $x_u = \log(y_u)$ für die Feuerzeitpunkte $t_u = T_{input} - x_u$. Weil spikende Neuronen temporale Muster unabhängig von additativen Konstanten in deren Ankommenszeitpunkten erkennen können, können sie mit Hilfe logarithmischer Kodierung auch konstante Faktoren λ in den Sensor-Inputs $\langle \lambda \cdot y_u \rangle$ ignorieren. Dies könnte mit der Fähigkeit biologischer neuronaler Netze zusammenhängen, Muster über ein breites Spektrum von Intensitäten hinweg klassifizieren können, wie z.B. visuelle Muster bei stark unterschiedlichen Beleuchtungsbedingungen.

4.2.3 Gewichtete Summe

Bei den bisher gezeigten Berechnungsarten in temporaler Kodierung besteht das Problem, dass Input und Output der Neuronen nicht das selbe Kodierungsschema verwenden. So z.B. bei der Erkennung von Gleichheit, wo die Input-Werte als Feuerzeitpunkte, der Output-Wert jedoch binär, also durch Feuern bzw. Nicht-Feuern, kodiert werden. Für Netzwerke mit mehreren Layern ist dies offensichtlich nicht wünschenswert, da eine direkte Kopplung eines Neurons mit einem anderen Neuron mit gleichem Aufbau unmöglich wird. Statt dessen sollten die Neuronen das selbe Kodierungsschema für Inputs sowie Output verwenden, wobei der Feuerzeitpunkt des Neurons v von den den Feuerzeitpunkten der presynaptischen Neuronen abhängen und durch die internen Parameter $w_{u,v}$ und $d_{u,v}$ kontrolliert werden soll. Dazu nehmen wir an, dass die Antwortfunktionen $\varepsilon_{u,v}$ ein wie früher angegebenes Verhalten aufweisen, also $\varepsilon_{u,v}(s) = 0$ für $s \leq d_{u,v}$ und anschließend annähernd linear mit Faktor $\lambda_{u,v}$ für ein Intervall der Länge mindestens $R > 0$ ansteigen bzw. abfallen. Weiters sollen sie presynaptischen Neuronen $u = 1, ..., n$ zu den Zeiten $t_u = T_{input} - x_u$ feuern, wobei x_u der jeweils zu kodierende Input-Wert ist. Wenn der Schwellwert $\Theta_v(0)$ des Neurons v von dessen Zustandsvariable $\sum_{u \in V, \langle u,v \rangle \in E} w_{u,v} \cdot \varepsilon_{u,v}(t - t_u)$ zum Zeitpunkt t_v überschritten

wird, solange alle Antwortfunktionen $\varepsilon_{u,v}(t - t_u)$ in deren linearer Phase der Länge $\geq R$ sind, so ist der Feuerzeitpunkt t_v des Neurons v bestimmt durch die Gleichung

$$\sum_{\substack{u \in V \\ \langle u,v \rangle \in E}} w_{u,v} \cdot \varepsilon_{u,v}(t_v - t_u) = \sum_{\substack{u \in V \\ \langle u,v \rangle \in E}} w_{u,v} \cdot \lambda_{u,v} \cdot (t_v - t_u - d_{u,v}) = \Theta_v(0)$$

Durch Umformen dieser Gleichung unter Ersetzung von $\sum_{u \in V, \langle u,v \rangle \in E} w_{u,v} \cdot \lambda_{u,v} = \lambda$ und $t_u = T_{input} - x_u$ erhält man

$$t_v = \frac{\Theta_v(0)}{\lambda} + \sum_{\substack{u \in V \\ \langle u,v \rangle \in E}} \frac{w_{u,v} \cdot \lambda_{u,v}}{\lambda} \cdot (T_{input} - x_u + d_{u,v})$$

oder

$$t_v = T_{output} - \sum_{\substack{u \in V \\ \langle u,v \rangle \in E}} \alpha_{u,v} \cdot x_u$$

für die von den Input-Werten abhängige Variable $T_{output} := \frac{\Theta_v(0)}{\lambda} + \sum_{u \in V, \langle u,v \rangle \in E} \frac{w_{u,v} \cdot \lambda_{u,v}}{\lambda} \cdot (T_{input} + d_{u,v})$ und formale Gewichte $\alpha_{u,v} := \frac{w_{u,v} \cdot \lambda_{u,v}}{\lambda}$, welche automatisch normalisiert sind (laut Definition von λ ist die Summe aller $\alpha_{u,v}$ gleich 1). Dabei ist der Ausgabewert $\sum_{u \in V, \langle u,v \rangle \in E} \alpha_{u,v} \cdot x_u$ durch das selbe Kodierungsschema ausgedrückt wie die analogen Eingabewerte x_u, nämlich durch den Feuerzeitpunkt t_v.

4.2.4 Simulation von sigmoidalen Gattern

Zur Simulation eines sigmoidalen Gatters ist es nötig, dass das Neuron v zum Zeitpunkt T_{output} feuert, wenn $\sum_{u \in V, \langle u,v \rangle \in E} \alpha_{u,v} \cdot x_u < 0$ und zum Zeitpunkt $T_{output} - \gamma$ wenn $\sum_{u \in V, \langle u,v \rangle \in E} \alpha_{u,v} \cdot x_u > \gamma$ ist (für eine Wahl der Konstante γ siehe [2]). Dies kann durch zusätzliche anregende und hemmende presynaptische Neuronen erreicht werden, wenn auch nur bis zu einer gegebenen Genauigkeit ε mit $\gamma > \varepsilon > 0$. Dabei hindern die hemmenden Neuronen das Neuron v vom Feuern vor dem Zeitpunkt $T_{output} - \gamma - \varepsilon$ und die anregenden Neuronen sorgen dafür, dass v nicht später als zum Zeitpunkt T_{output} feuert. Für Details dazu siehe [2].

4.2.5 Universelle Approximation kontinuierlicher Funktionen

Durch die in den vorherigen Kapiteln gezeigten Berechnungsverfahren können SNNs konstruiert werden, welche beliebige beschränkte, kontinuierliche Funktionen mit temporaler Kodierung berechnen. Da bei der Simulation von sigmoidalen Gattern die Inputs und der Ouput das selbe Kodierungsschema verwenden, können die Outputs einer Schicht von Neuronen als Inputs einer anderen Schicht von Neuronen verwendet werden. Unter der Annahme, dass die initialen Segmente der Antwortfunktionen $\varepsilon_{u,v}(s)$ linear sind, kann folgendes Theorem bewiesen werden (Beweis siehe [10]): *Jedes beliebige vorwärtsgekoppelte oder periodische analoge neuronale Netzwerk (zum Beispiel jedes beliebige mehrschichtige Perzeptron), bestehend aus s sigmoidalen Gattern mit einer bestimmten Sättigungsfunktion, kann beliebig genau durch ein Netzwerk von s + c spikenden Neuronen mit analogen, temporal kodierten Ein- und Ausgängen simuliert werden, wobei c eine eine kleine Konstante ist. Dies kann auch durchgeführt werden, wenn die spikenden Neuronen Rauschen ausgesetzt sind.*

Dies zeigt wiederum, dass die Berechnungspotenz spikender neuronaler Netze größer ist als die herkömmlicher Netzwerktypen, da jedes beliebige sigmoidale neuronale Netz zwar von einem nur unwesentlich größeren (die Größe wird als die Anzahl der verwendeten Neuronen definiert) spikenden neuronalen Netzwerk mit temporaler Kodierung simuliert werden kann, bestimmte Typen

von spikenden Neuronalen Netzwerken jedoch nur durch erheblich größere sigmoidale neuronale Netzwerke simuliert werden können (wie z.B. die Erkennung von Gleichheit).

Es ist bekannt, dass vorwärtsgekoppelte sigmoidale neuronale Netzwerke jede beliebige beschränkte, kontinuierliche Funktion $F : [0,1]^n \to [0,1]^m$ mit beliebiger Genauigkeit approximieren können. Aus dem oben genannten Theorem ergibt sich damit automatisch, dass beschränkte, kontinuierliche Funktionen auch beliebig genau durch SNNs mit analogen, temporal kodierten Ein- und Ausgängen approximiert werden können.

4.3 Berechnungen mit Populationskodierung

In biologischen neuronalen Netzen zeigen sich auch Verhaltensstrukturen, die die Verwendung von Populationskodierung (*population code, space rate code*) nahe legen. Dabei werden analoge Werte $x \in [0,1]$ durch den Prozentsatz von Neuronen einer bestimmten Gruppe / Population, welche innerhalb eines kleinen Zeitfensters feuern, kodiert. Obwohl empierische Ergebnisse zeigen, dass diese Art der Kodierung in vielen kortikalen Systemen verwendet wird, ist bisher unklar, wie Berechnungen mit Hilfe eines solchen Codes durchgeführt werden können. Eine Methode zur Approximation beliebiger kontinuierlicher Funktionen $[0,1]^n \to [0,1]^m$ mit populations-kodierten Ein- und Ausgaben durch SNNs mit 3 Schichten kann in [7] nachgelesen werden, wegen der derzeit fehlenden praktischen Relevanz wird jedoch in dieser Arbeit nicht näher darauf eingegangen.

4.4 Berechnungen mit Feuerraten

In bisherigen künstlichen neuronalen Netzen (ANNs) wurde die Verbindung zu den biologischen Vorbildern hergestellt, indem die Feuerrate der zwischen den natürlichen Neuronen übertragenen Spikes als reelle Zahl zwischen 0 und 1 modelliert wurde. Daraus lässt sich ableiten, dass bei künstlichen Neuronen mit sigmoidalen Aktivierungsfunktionen der Ausgangswert einen Zusammenhang mit der Summe der Eingangswerte aufweist. Auch bei natürlichen neuronalen Netzwerken besteht ein Zusammenhang zwischen der Feuerrate eines Neurons und den Feuerraten dessen presynaptischer Neuronen.

Es existieren hinreichende empierische Beweise, dass Information über äußere Reize in biologischen neuronalen Netzen in Feuerraten der Sensoren kodiert ist. So wird z.B. eine kontinuierliche Belastung eines Dehnungs-Rezeptors direkt auf die Feuerrate bzw. Frequenz der von ihm ausgesendeten Spikes abgebildet (vgl. dazu [1], Seite 6f). Allerdings gibt es Zweifel darüber, ob die Feuerrate eines natürlichen Neurons v wirklich von den Feuerraten x_u der presynaptischen Neuronen u in einer Form abhängt, die sich durch einen Term $g(\sum_{u \in V, \langle u,v \rangle \in E} w_{u,v} \cdot x_u)$ ausdrücken lässt.

Aktuelle biologische Untersuchungen zeigen, dass sich bei biologischen Synapsen oberhalb einer "Grenzfrequenz" von ungefähr 10 Hz die Amplituden der PSPs (Spikes) umgekehrt proportional zur Feuerrate x_u des jeweiligen presynaptischen Neurons verhalten. Um diese Erkenntnisse im mathematischen Modell für natürlichen Neuronen abbilden zu können, müssten die bisher konstant angenommenen Gewichtsfaktoren $w_{u,v}$, welche die biologischen Synapsenstärken repräsentieren, durch Funktionen $w_{u,v}(x_u)$, welche von den Feuerraten x_u der presynaptischen Neuronen u abhängen, ersetzt werden. Diese Funktionen $w_{u,v}(x_u)$ müssten proportional zu $\frac{1}{x_u}$ sein. Dann allerdings würde die gewichtete Summe $\sum_{u \in V, \langle u,v \rangle \in E} w_{u,v}(x_u) \cdot x_u$, welche das Potential der biologischen Neuronen modelliert, eines Neurons v nicht mehr von den Feuerraten x_u derer presynaptischen Neuronen u abhängen, welche oberhalb der Grenzfrequenz feuern.

Eine weitere Besonderheit in Zusammenhang mit Kodierung durch Feuerraten wurde in [9] präsentiert. Es wird gezeigt, dass bei einem zur Erkennung von Gleichzeitigkeit arbeitenden spikenden Neuron, welches als Eingabe Spikefolgen aus stochastisch unabhängigen Poisson-Prozessen erhält, die Feuerrate seines Ausganges in bestimmten Parameterbereichen proportional zum Produkt der Feuerraten seiner Eingabe-Spikefolgen ist. Dies folgt daher, dass bei einer Menge von unabhängigen Poisson-Prozessen die Wahrscheinlichkeit des Auftretens eines Ereignisses innerhalb des selben, kleinen Zeitfensters in allen Prozessen proportional zum Produkt der Auftrittswahrscheinlichkeiten der einzelnen Poisson-Prozesse ist. Wenn man also annimmt, dass ein Neuron v genau

dann feuert, wenn alle seine presynaptischen Neuronen innerhalb eines kleinen Zeitfensters feuern, dann ist seine Feuerrate ungefähr proportional zum Produkt der Feuerraten der presynaptischen Neuronen u.

Diese Ergebnisse zeigen bereits, dass die Kodierung mit Feuerraten in SNNs einige Probleme bereitet und vermutlich deshalb noch wenige Arbeiten zu diesem Thema existieren.

4.5 Berechnungen mit temporalen Korrelationen

In [7] wird auf die Möglichkeit hingewiesen, die Korrelationen bzw. Gleichzeitigkeit von Feuerzeitpunkten verschiedener Neuronen in Berechnungen einfließen zu lassen. Dieses Konzept hat kein Äquivalent bei herkömmlichen neuronalen Netzen und kann auf mikroskopischer Ebene - bei den Zeitpunkten einzelner Spikes - sowie auf makroskopischer Ebene - bei den statistischen Korrelationen von Spikefolgen - angewandt werden. Es wird auch ein mathematisches Modell präsentiert, welche diese Kodierung verwendet und eine quantitativ größere Berechnungspotenz als bisher verwendete Kodierungsverfahren aufzuweisen scheint.

5 Emulation von Hopfield-Netzwerken

Dieses Kapitel stellt eine Zusammenfassung von [5] mit Anpassung an die in dieser Arbeit verwendeten Konventionen dar.

Bei der Emulation von Hopfield-Netzwerken mit SNNs wird von der in vorherigen Kapiteln beschriebenen temporalen Kodierung von Werten $x \in [-1,1]^n$ Gebrauch gemacht, bei welcher die Feuerzeitpunkte der Neuronen u durch die reellen Zahlen $x_u \in [-1,1]$ als $T - c \cdot x_u$ (für sinnvoll gewählte Konstante T und c) gegeben werden. Zur Konstruktion von SNNs zur Approximation beliebiger Hopfield-Netzwerke wird für die dazu verwendeten Neuronen das übliche Modell *leaky integrate-and-fire neuron with noise* oder das etwas allgemeinere *spike response model* verwendet. Die einzige konkrete, für die Konstruktion nötige Annahme ist, dass die initialen Segmente der Antwortfunktionen linear ansteigend (bei EPSPs) bzw. linear abfallend (bei IPSPs) sind.

Gegeben sei ein beliebiges Hopfield-Netzwerk H mit stufiger Antwort (im Gegensatz zu binärer Antwort) und synchronem Update. Es wird angenommen, dass H aus n sigmoidalen Gattern $u = 1, ..., n$ mit beliebigen Gewichten $w_{u,v} \in R$ und einer stückweise linearen Aktivierungsfunktion σ mit $\sigma(x) = -1$ für $x < -1$, $\sigma(x) = x$ für $-1 \leq x \leq 1$ und $\sigma(x) = 1$ für $x > 1$.

Dann kann jede Berechnung von H durch ein periodisches SNN S_H aus n spikenden Neuronen (mit $O(1)$ zusätzlichen Hilfsneuronen) mit temporaler Kodierung approximiert werden. Eine Eingabe, Ausgabe oder ein interner Zustand von H der Form $\langle x_1, ..., x_n \rangle \in [-1,1]^n$ wird in S_H repräsentiert durch temporale Kodierung, also einem Muster von Spikes in S_H, bei welchem ein Neuron u zum Zeitpunkt $k \cdot T - c \cdot \widetilde{x}_u$ feuert, wobei $|x_u - \widetilde{x}_u|$ beliebig klein werden kann. Die Referenzzeitpunkte $k \cdot T$ für $k = 0, 1, ...$ werden definiert von $O(1)$ periodisch feuernden Hilfsneuronen in S_H. Jeder Fixpunkt von H entspricht einem stabilen periodischen Muster von Spikes in S_H. Die Gewichte $w_{u,v}$ in S_H sind gleich (entsprechend skaliert) denen der korrespondierenden Kantengewichte in H.

Der Beweis dieser Konstruktion (siehe [6]) gründet darauf, dass bei bestimmter Wahl der Parameter ein spikendes Neuron die Berechnung einer gewichteten Summe durchführen kann, wie in einem vorherigen Kapitel beschrieben.

In [5] werden auch die Ergebnisse von Simulationen dieser Konstruktion unter Zuhilfenahme realistischerer Modelle in GENESIS präsentiert. Dabei stellt sich heraus, dass die Berechnung einer gewichteten Summe durch spikende Neuronen, also die Grundberechnungsart der obigen Konstruktion, durch die schnell ansteigenden EPSPs theoretisch ein sehr kurzes Zeitfenster (im Bereich von 1 ms) erfordern würde, damit alle EPSPs während der Summenbildung in ihrer annähernd linearen Phase sind. Diese kurze Zeitspanne wäre allerdings schon temporalen Ungenauigkeiten und Rauschen unterworfen. Die Simulationen in realistischeren Modellen zeigen aber, dass im Gegensatz zur Theorie die Berechnung noch besser funktioniert und ein zusätzlicher Oszillator zur Erzeugung der Referenzzeitpunkte $k \cdot T$ für $k = 0, 1, ...$ nicht nötig ist.

6 Überwachtes Lernen

In den bisherigen Kapiteln wurde auf die Emulation bekannter Netzwerktypen mit SNNs eingegangen, ohne jedoch den Lernvorgang zur Bestimmung der Synapsengewichte miteinzubeziehen. In den folgenden Unterkapiteln, welche aus [2] zusammengefasst wurden, werden Möglichkeiten zum überwachten Training von Synapsengewichten in SNNs präsentiert.

Üblicherweise verwendet man bei Neuronalen Netzen die Hebb'sche Regel als Grundlage der Lernverfahren, also eine Synapse wird verstärkt, wenn sowohl das presynaptische also auch das postsynaptische Neuron zur selben Zeit eine hohe Feuerrate aufweisen. Da hier das Lernen mit temporaler Kodierung (die Werte werden in den Feuerzeitpunkten einzelner Spikes kodiert) betrachtet wird, wird auch eine abgewandelte Hebb'sche Lernregel verwendet. Dabei ändert sich das Synapsengewicht abhängig von der Zeitdifferenz der einzelnen presynaptischen und postsynaptischen Feuerzeitpunkte. In den folgenden Unterkapiteln werden Lernverfahren präsentiert, bei denen ein bestimmtes Fehlersignal verfügbar ist. Das Ziel des Lernvorganges ist, dass das Synapsengewicht derart verändert wird, dass der presynaptische Spike das postsynaptische Neuron nach einer vorbestimmten Zeit zum Feuern bringt. Die Überwachung des Lernvorganges erfolgt durch ein zusätzliches presynaptisches Neuron, welches anzeigt, wann das postsynaptische Neuron durch den ersten presynaptischen Spike zum Feuern angeregt hätte werden sollen.

6.1 Einzelne Synapsen

In diesem Kapitel wird eine mögliche Lernregel angegeben, die das Erlernen vorgegebener Gewichte für einzelne Synapsen erlaubt, wobei die Änderungen des Synapsengewichtes nur durch einzelne pre- und postsynaptische Spikes bestimmt wird. Die Synapse erhält zwei Spikes und das Ziel ist, dass das Synapsengewicht auf einen Wert abhängig von der zeitlichen Differenz dieser beiden Spikes eingestellt wird.

Da eine Synapse nicht das Feuern der Spikes am Neuron, sondern nur die Ankunft des Spikes wahrnehmen kann, wird in diesem Kapitel der Einfachheit halber mit Ankunftszeiten der presynaptischen Spikes anstatt deren Feuerzeitpunkten gearbeitet. Wenn also von Feuerzeitpunkten gesprochen wird, sind eigentlich die jeweiligen Ankunftszeitpunkte an den entsprechenden Synapsen gemeint, welche sich durch $d_{u,v}^{pre}$ von den Feuerzeitpunkten unterscheiden (Verzögerung bis zur Synapse). Weiters wird wiederum angenommen, dass die initialen Segmente der EPSPs bzw. IPSPs linear ansteigen bzw. abfallen.

Wie nehmen an, dass Neuron u eine anregende Verbindung zu Neuron v hat und das Gewicht der Synapse $\langle u, v\rangle$ erlernt werden soll. Neuron u feuert zu einer bestimmten Zeit t_u und bringt dadurch Neuron v zum Feuern zur Zeit t_v. Neuron u feuert ein zweites Mal zur Zeit $t_0 > t_u$, wodurch sich der Eingangswert von v an $\langle u, v\rangle$ als $t_0 - t_u$ ergibt und die Lernregel zur Änderung des Gewichtswertes wie folgt definiert werden kann:

$$\Delta w_{u,v} = \eta \cdot (t_v - t_0)$$

Dabei gibt $\eta > 0$ die Lernrate an, welche vor Beginn des Lernens gewählt und während der gesamten Lernphase konstant gehalten werden muss. Die Zeitdifferenz $t_v - t_0$ zwischen dem Ankunftszeitpunkt t_0 des presynaptischen Spikes an u und dem Feuerzeitpunkt t_v des postsynaptischen Spikes an v kann als Fehler interpretiert werden, welcher durch das Lernen verringert wird und gegen 0 konvergieren soll.

Folgend soll ein Beweis des folgenden Theorems angedeutet werden: *Gegeben sind zwei Neuronen u und v mit einer anregenden Synapse $\langle u, v\rangle$, welche eine Antwortfunktion $\varepsilon_{u,v}$ mit initialen linearen Segmenten aufweisen. Das Intervall $[w_{min}, w_{max}] \subseteq R^+$ sei der mögliche Wertebereich für das Synapsengewicht $w_{u,v}$ und die Lernrate η sei eine beliebige reelle Zahl im Bereich $0 < \eta \leq w_{min}^2/(\Theta_v - P_v^{rest})$, wobei P_v^{rest} das Ruhepotential des Neurons v angibt, wenn dieses keinen Input erhält. Wenn Neuron u zweimal innerhalb jedes Lernzyklusses mit konstanter Zeitdifferenz $t_0 - t_u$ feuert und $w_{u,v}$ nach der Regel $\Delta w_{u,v} = \eta \cdot (t_v - t_0)$ verändert wird, dann wird die Folge der Gewichtswerte für beliebige Anfangswerte aus $[w_{min}, w_{max}]$ gegen $\widetilde{w} = (\Theta_v - P_v^{rest})/(t_0 - t_u)$ konvergieren.*

Zum Beweis dieses Theorems muss zuerst gezeigt werden, dass der einzige Fixpunkt der Lernregel

$$\widetilde{w} = \frac{\Theta_v - P_v^{rest}}{t_0 - t_u}$$

ist. Die Anwendung der oben angeführten Lernregel zur Änderung der Gewichtswerte ändert $w_{u,v}$ genau dann nicht, wenn der zweite Spike von u zum Zeitpunkt t_0 und der Spike von v zum Zeitpunkt t_v exakt zum selben Zeitpunkt auftreten, also $t_v = t_0$. Durch die Annahme der Linearität der initialen Segmente der EPSPs erhält man

$$P_v^{rest} + \widetilde{w} \cdot (t_v - t_u) = \Theta_v$$

Durch Ersetzen von $t_v = t_0$ erhält man

$$t_0 - t_u = \frac{\Theta_v - P_v^{rest}}{\widetilde{w}}$$

was äquivalent zu oben angegebenem Fixpunkt ist. Weiters muss zum vollständigen Beweis des Theorems auch gezeigt werden, dass jedes beliebige $w \in [w_{min}, w_{max}]$ durch wiederholte Anwendung der Lernregel gegen $\widetilde{w}$ konvergiert. Dabei ist eine Fallunterscheidung für die beiden Fälle $w_{u,v} > \widetilde{w}$ und $w_{u,v} < \widetilde{w}$ nötig. Die Details dieses Beweises werden hier nicht näher ausgeführt und können in [2] nachgelesen werden.

Der Ankunftszeitpunkt t_0 des zweiten von u gesendeten Spikes bestimmt daher, wann das Neuron v feuern soll, wodurch eine gezielte Einstellung des Synapsengewichtes abhängig vom gewünschten Feuerzeitpunkt von v bei gegebenem Ankunftszeitpunkt t_u des presynaptischen Spikes möglich wird. Der Wert des Synapsengewichtes $w_{u,v}$ selbst muss zur Erreichung dieses Ziels nicht bekannt sein.

6.2 Mehrere Synapsen

Mit der im vorherigen Kapitel beschriebenen Methode können die Gewichte einzelner Synapsen gelernt werden. Da jedoch ein Neuron v meist Inputs von mehreren Neuronen $u = 1, ..., n$ mit entsprechenden Gewichtswerten $w_{u,v}$ erhält, müssen alle diese Gewichtswerte gelernt werden, um das Neuron v steuern zu können. Diese Gewichte können auch durch die bereits beschriebene Methode sequentiell gelernt werden, indem das Neuron v in jedem Lernzyklus nur Inputs von jeweils einer einzelnen Synapse empfängt. Der Lernvorgang wäre wesentlich effizienter, wenn diese Gewichte parallel gelernt werden könnten, warauf nun folgend eingegangen wird. Bei der zuvor beschriebenen Methode würde sich beim parallelen Training mehrerer Synapsen das Problem ergeben, wie der durch $t_v - t_0$ definierte Fehler auf die verschiedenen Synapsen aufgeteilt werden soll, um die jeweiligen $\Delta w_{u,v}$ zu berechnen.

Eine mögliche Lösung wird in [2] präsentiert und hier zusammengefasst. Sie stützt sich auf eine Normalisierung aller Gewichtswerte nach jedem Lernzyklus, wodurch ein unendliches Anwachsen von Gewichtswerten verhindert wird. Wenn der Gewichtsvektor $\underline{w} = \langle w_{u,v} \rangle$ für $u = 1, ..., n$ so normalisiert wird, dass $\|\underline{w}\| = 1$ ist, kann eine Hebb'sche Lernregel wie folgt definiert werden: Das Ziel des Lernens ist, dass $\underline{w} = \underline{\widetilde{w}}$ für einen gewünschten Gewichtsvektor $\underline{\widetilde{w}}$ mit $\|\underline{\widetilde{w}}\| = 1$. Das Neuron v empfängt nun von einer zusätzlichen Synapse mit ausreichend starkem Gewichtswert einen Input-Spike, sodass v genau zum Zeitpunkt t_0 feuert. Die Synapse $\langle u, v \rangle$ erhält einen Input-Spike vom Neuron u zum Zeitpunkt t_u, sodass $t_0 - t_u = \widetilde{w}_{u,v}$. Weiters wird durch ausreichend viele IPSPs von zusätzlichen hemmenden Synapsen sichergestellt, dass v nicht vor t_0 feuert. Die Lernregel wird zwei-schrittig definiert:

(1) $\Delta w_{u,v} = \eta \cdot (t_0 - t_u),\ 1 \leq u \leq n$
(2) *normalisiere den entstandenen Gewichtsvektor* $\underline{w}$

Wenn $\eta > 0$ ist, gilt folgendes Theorem über die Konvergenz des Lernverfahrens: *Gegeben sei ein Neuron v mit anregenden Synapsen $\langle u, v \rangle$ für $u = 1, ..., n$ und sei $\underline{\widetilde{w}}$ ein Gewichtsvektor mit*

$\|\underline{\widetilde{w}}\| = 1$ *und* $\eta > 0$ *die Lernrate. Dann konvergiert die durch obige Lernregel erzeugte Folge von Gewichtsvektoren für jeden beliebigen Startvektor gegen* $\underline{\widetilde{w}}$.

Zum Beweis dieses Theorems ist es wiederum nötig zu zeigen, dass $\underline{\widetilde{w}}$ ein Fixpunkt der Lernregel ist und dass die Folge der $w_{u,v}$ konvergiert. Letzteres wird in [2] gezeigt und wird deshalb hier nicht weiter angeführt. Zum Beweis des ersten Kriteriums (in [2] nicht explizit aufgeführt) nehmen wir an, dass

$$\underline{w}^m = \underline{\widetilde{w}}$$

also dass der Gewichtsvektor nach dem m-ten Berechnungsschritt gleich $\underline{\widetilde{w}}$, dem zu erlernenden Gewichtsvektor, ist.

Durch Einsetzen der Definition von t_u in (1) der Lernregel ergibt sich

$$\Delta w_{u,v} = \eta \cdot \widetilde{w}_{u,v}$$

und somit für den im ersten Schritt neu berechneten neuen Gewichtsvektor

$$\underline{w}^{m+1'} = \underline{w}^m + \Delta\underline{w} = \underline{w}^m + \eta \cdot \underline{\widetilde{w}} = (1+\eta) \cdot \underline{w}^m.$$

Durch das anschließende Normalisieren von $\underline{w}^{m+1}$ ergibt sich also (mit $\|\underline{w}^m\| = 1$)

$$\underline{w}^{m+1} = \frac{\underline{w}^{m+1'}}{\|\underline{w}^{m+1'}\|} = \frac{(1+\eta) \cdot \underline{w}^m}{\|(1+\eta) \cdot \underline{w}^m\|} = \frac{(1+\eta) \cdot \underline{w}^m}{(1+\eta) \cdot \|\underline{w}^m\|} = \frac{(1+\eta) \cdot \underline{w}^m}{(1+\eta)} = \underline{w}^m.$$

Somit ist $\underline{\widetilde{w}}$ ein Fixpunkt der Lernregel.

Mit entsprechenden Transformationen kann das beschriebene Verfahren auch zum Lernen von Gewichtswerten außerhalb des Intervalls $[0,1]$ in einem allgemeineren Intervall $[w_{min}, w_{max}] \subseteq R^+$ angewandt werden. Im Gegensatz zum Lernen mit einzelnen Synapsen wird hier nicht verlangt, dass die EPSPs ein initiales lineares Segment aufweisen.

Allerdings besitzt diese parallele Lernregeln einen gravierenden Nachteil zum Lernen einzelner Synapsen: Der Zusammenhang zwischen dem Feuerzeitpunkt von v und dem eingestellten Gewichtsvektor $\underline{\widetilde{w}}$ geht vollständig verloren. Es ist nicht mehr möglich, einen gewünschten Feuerzeitpunkt t_v des Neurons v bei gegebenem Eingangsmuster ohne Kenntnis des dazu notwendigen Gewichtsvektors zu erlernen. Wenn die Lernphase beendet ist, also die zusätzlichen EPSPs bzw. IPSPs nicht mehr an das Neuron v angelegt werden, so wird dieses nicht mehr zum davor fixierten Zeitpunkt t_0 feuern, auch wenn das selbe, während der Lernphase verwendete Eingangsmuster weiterhin angelegt wird. Das hier beschriebene Lernverfahren ermöglicht lediglich das Lernen eines vorgegebenen Gewichtsvektors $\underline{\widetilde{w}}$, nicht jedoch eine Interpretation der Gewichtswerte im Sinne der temporalen Kodierung. Allerdings ermöglicht es das Lernen dieser Gewichte für die jeweiligen Synapsen mit nur lokal an den Synapsen verfügbarer Information, eine globale Information über die Feuerzeitpunkte aller beteiligten Synapsen ist nicht nötig.

7 Unüberwachtes Lernen

Dieses Kapitel widmet sich dem unüberwachten Lernen in SNNs mit temporaler Kodierung. In früheren Varianten unüberwacht lernender SNNs, wie z.B. ersten Implementierungen von SOMs wurden die Werte wiederum als Feuerraten kodiert. Da allerdings die Erkennung des Gewinnerneurons meist mit periodischen Netzwerken implementiert wurde, konnte das Gewinnerneuron stets erst nach Erreichen eines Gleichgewichtszustandes bestimmt werden. Durch diese Erfordernis kann der Vorteil temporaler Kodierung zur schnellen Informationsverarbeitung nicht ausgenutzt werden.

In den folgenden Unterkapiteln wird eine Möglichkeit zum Berechnen und Lernen in SNNs mit temporaler Kodierung präsentiert. Im Gegensatz zu herkömmlichen Implementierungen von SOMs kann hierbei das Gewinnerneuron einer Gruppe konkurrierenden Neuronen schnell und nur durch lokal verfügbare Information bestimmt werden, was durch anregende bzw. hemmende

Querverbindungen erreicht wird. Diese Querverbindungen bestimmen auch die Nachbarschaftsbeziehungen zwischen den Neuronen. Es wird angenommen, dass am Anfang topologisch nahe beieinander liegende Neuronen starke anregende Querverbindungen und entfernte Neuronen starke hemmende Querverbindungen aufweisen. Während der Lernphase werden die Gewichte dieser Querverbindungen verringert.

Durch Computersimulationen wurde in [2] nachgewiesen, dass die hier beschriebenen Modelle die selben Charakteristika aufweisen wie SOMs und dass die Ergebnisse nach einer bestimmen Anzahl von Lernzyklen unabhängig davon waren, ob die Gewichtswerte nach jedem Lernzyklus normalisiert wurden oder nicht. Dies ist im Hinblick auf Realitätnähe wichtig, da die Normalisierung der Gewichte bei Betrachtung ihrer biologischen Relevanz umstritten ist.

7.1 Wettbewerbslernen

Auf Basis der früher beschriebenen Konstruktion sigmoidaler Gatter mit Hilfe spikender Neuronen kann Wettbewerbslernen in SNNs wie folgt realisiert werden: Gegeben sei eine Menge S m-dimensionaler Input-Vektoren $\underline{s}^l = (s_1^l, ..., s_m^l) \in [0, \gamma]^m$ und ein SNN mit $m+1$ Input-Neuronen und n konkurrierenden Neuronen, wobei jedes konkurrierende Neuron v_j synaptischen Input "vorwärtsgekoppelt" von jedem Input-Neuron u_i mit Gewicht $w_{i,j}$ und synaptischen Input "lateral" von jedem konkurrierenden Neuron v_k, $k \neq j$ mit Gewicht $\widetilde{w}_{k,j}$ (Querverbindungen) erhält. In jedem Lernzyklus wird ein $\underline{s}^l \in S$ zufällig ausgewählt und die Input-Neuronen werden innerhalb des Kodierungsintervalls so zum Feuern gebracht, dass sie temporär $\underline{s}^l$ kodieren. Wie früher beschrieben wird ein zusätzliches Neuron u_0, welches zur Zeit T_{input} feuert, verwendet, um den Wert 0 zu kodieren. Dessen Gewichte $w_{0,j}$ werden so gewählt, dass $\sum_{i=0}^m w_{ij} = \lambda$ für eine Konstante $\lambda > 0$, die so gewählt werden soll, dass die Neuronen v_j tatsächlich feuern.

Jedes konkurrierende Neuron startet dann mit der Berechnung von $\sum_{i=0}^m w_{i,j} \cdot s_i^l$, wobei davon ausgegangen wird, dass sich alle EPSPs und IPSPs in ihrer initialen, linearen Phase befinden. Allerdings beschreibt hier nur derjenige Feuerzeitpunkt des konkurrierenden Neurons, welches zuerst feuert, die zugehörige gewichtete Summe, die Feuerzeitpunkte aller anderen konkurrierenden Neuronen werden um den Feuerzeitpunkt dieses einen Neurons verschoben. Wenn nun angenommen wird, dass sowohl der Input-Vektor als auch der Gewichtsvektor für jedes Neuron normalisiert sind, dann repräsentiert diese gewichtete Summe die Ähnlichkeit der beiden Vektoren in Bezug auf den euklidischen Abstand. Daher gilt, je eher ein Neuron v_j feuert, desto ähnlicher ist sein Gewichtsvektor dem Input-Vektor, d.h. der Gewinner innerhalb der Schicht der konkurrierenden Neuronen feuert zuerst.

Wenn die Querverbindungen stark hemmend sind, sodass das Feuern des Gewinnerneurons v_k alle anderen Neuronen der konkurrierenden Schicht vom Feuern abhält, so kann Wettbewerbslernen sehr einfach implementiert werden: Die Standardregel für Wettbewerbslernen wird einfach auf das Gewinnerneuron v_k angewandt. Diese Regel wird definiert als

$$\Delta w_{i,k} = \eta \cdot (s_i^l - w_{i,k}),\ i = 1, ..., m$$

wobei $\underline{s}^l$ das aktuelle Input-Muster und $\eta > 0$ die Lernrate ist. Diese Regel schließt nicht explizit den Fall aus, dass Gewichte negativ werden können. Falls ein Neuron allerdings auf eine Menge von Input-Vektoren spezialisiert wurde, so wird sein Gewichtsvektor nur positive Werte enthalten. Eine Vorzeichenänderung von Gewichtswerten kann auch durch Einführen einer Sättigungsfunktion, welche die möglichen Werte auf ein bestimmtes Intervall einschränkt, formal verhindert werden.

7.2 Selbstorganisierende Karten (SOMs)

Um diese Realisierung von Wettbewerbslernen zu einer Realisierung von Selbstorganisation zu erweitern, muss anhand lokal verfügbarer Information eine Nachbarschaftsmatrix $(m_{k,j})_{1 \leq k,j \leq n}$ implementiert werden, wobei $m_{k,j}$ den Abstand zwischen dem k-ten und dem j-ten konkurrierenden Neuron beschreibt. Dazu wird eine monoton wachsende Funktion $m_{k,j} \mapsto \widetilde{w}_{kj}$ verwendet, sodass

die Gewichte der Querverbindungen $\widetilde{w}_{k,j}$ zwischen den konkurrierenden Neuronen die Struktur der Nachbarschaft widerspiegeln: am Anfang haben topologisch nahe beieinander liegenden Neuronen starke anregende Querverbindungen, entfernte Neuronen haben starke hemmende Querverbindungen. Dies bedeutet, dass das Feuern des Gewinnerneurons v_k zum Zeitpunkt t_k das Feuern von Neuronen in der Nachbarschaft von v_k in Richtung t_k treibt, also die Neuronen zum Feuern zu einem früheren Zeitpunkt veranlasst und somit die durch diese Neuronen kodierten Werte erhöht. Das Feuern entfernter Neuronen hingegen wird durch die hemmenden Querverbindungen hinausgeschoben, wodurch die kodierten Werte verringert werden. Es wird dazu die folgende Lernregel vorgeschlagen:

$$\Delta w_{i,j} = \eta \cdot \frac{T_{output} - t_j}{T_{output}} \cdot (s_i^l - w_{i,j})$$

wobei t_j den Feuerzeitpunkt des j-ten konkurrierenden Neurons darstellt. Die Lernregel wird nur auf Neuronen angewandt, welche vor einer bestimmten Zeit T_{ouput} gefeuert haben. Der Faktor $(T_{output} - t_j)/T_{output}$ realisiert die Nachbarschaftsfunktion, welche am größten für das Gewinnerneuron ist und für Neuronen, welche zu späteren Zeitpunkten feuern, abnimmt.

Während des Lenzprozesses werden die Gewichte der Querverbindungen $\widetilde{w}_{k,j}$ verringert, wodurch die Größe der Nachbarschaft reduziert wird. Diese stetige Verringerung der Gewichte verlangt allerdings einen Vorzeichenwechsel während des Lernprozesses, was biologisch nicht plausibel erscheint. Es ist jedoch möglich, jeweils zwei Querverbindungen zwischen den konkurrierenden Neuronen zu betrachten, eine anregenden und eine hemmende. Während des Lernprozesses werden die anregenden Verbindungen dann verringert und die hemmenden verstärkt.

Wie in Standard-SOMs wird η während des Lernens langsam verringert.

7.3 Lernen mit Radialen Basisfunktionen

RBF-Netze sind bei ANNs bereits sehr erfolgreich in den Bereichen der Musterklassifizierung, Approximierung von Funktion und Clustering. In einem früheren Kapitel wurde bereits kurz auf die einfache Simulation einzelner RBF-Neuronen eingegangen, in diesem Kapitel wird ein Lernalgorithmus zur Realisierung von RBFs mit spikenden Neuronen eingeführt, wobei als Kodierungsart wiederum die temporale Kodierung verwendet wird. Der Lernalgorithmus ermöglicht das unüberwachte Finden von Cluster-Zentren und ist in [2] genau beschrieben. Hier wird nur eine Zusammenfassung dieser Arbeit präsentiert, in deren Zuge auch Computersimulationen mit interessanten Ergebnissen durchgeführt wurden. Die RBF Neuronen konvergierten einerseits auch in Gegenwart von Rauschen zuverlässig zu den Clusterzentren, waren andererseits aber auch zur dynamischen Rekonfiguration in der Lage, wenn während des Lernprozesses Cluster hinzugefügt oder entfernt wurden. Weiters zeigte sich, dass RBF-Neuronen unter Zuhilfenahme von PSPs verschiedener Breite in der Lage waren, temporale Sequenzen von Spikes zu erkennen, auch wenn diese auf verschiedene Arten gestört wurden.

7.3.1 Modell

Das Modell zur Simulation von RBF-Neuronen wird hier insofern erweitert, als dass nicht nur das Feuern bzw. Nicht-Feuern der Neuronen betrachtet wird, sondern auch die Feuerzeitpunkte mit berücksichtigt werden. Es bestehen große Ähnlichkeiten zu dem zuvor vorgestellte Modell zur Simulation von Wettbewerbslernen, das SNN besteht also wieder aus m Input-Neuronen $u_1, ..., u_m$ und n Output-Neuronen $v_1, ..., v_n$, welche als RBF-Neuronen bezeichnen werden. Im einfachsten Fall besteht von jedem Input-Neuron u_i zu jedem RBF-Neuron v_j jeweils eine synaptische Verbindung mit Gewicht $w_{i,j}$ und Verzögerung $d_{i,j}$.

Das Kodierungsschema wird ähnlich den früheren Konstruktionen so festgelegt, dass jedes Input-Neuron u_i genau einmal zum Zeitpunkt t_i innerhalb das Kodierungsintervalls $[T_{input} - \widetilde{\gamma}, T_{input}]$ feuert, wobei $\widetilde{\gamma}$ eine später noch festzulegende Konstante ist. Daher wird die Eingabe relativ zum ersten Spike innerhalb des Kodierungsintervalls kodiert, d.h. kein expliziter Referenzspike wird benötigt. Input-Vektoren dieser Kodierungsart werden mit $\underline{x}$ bezeichnet und sind definiert als $\underline{x} = \langle x_1, ..., x_m \rangle$ mit $x_i = \max\{t_i \mid 1 \leq i \leq m\} - t_i$.

Eine Eingabe $\underline{x}$ ist nahe dem Zentrum $\underline{c}_j$ eines RBF-Neurons v_j, wenn die Spikes der Input-Neuronen v_j durch die Verzögerungen $d_{i,j}$ zu ähnlichen Zeiten erreichen, also wenn $\|\underline{x} - \underline{c}_j\|$ klein ist. Der Feuerzeitpunkt von v_j beschreibt, wie nahe der Input-Vektor zum Zentrum liegt. Wenn der Abstand zu groß ist, wird v_j nicht feuern. Wenn aus einer Menge von RBF-Neuronen $\{v_1, ..., v_n\}$ für verschiedene j die Distanz $\|\underline{x} - \underline{c}_j\|$ klein genug ist, um die Neuronen v_j zum Feuern zu bringen, dann wird dasjenige Neuron zuerst feuern, dessen Zentrum der Eingabe $\underline{x}$ am nächsten liegt. Daher kann eine solche Menge von RBF-Neuronen zum Trennen von Eingaben in verschiedene Cluster verwendet werden.

7.3.2 Lernen von Clustern

Um Cluster-Zentren unüberwacht durch RBF-Neuronen finden zu lassen, wird die obige Konstruktion erweitert. Anstatt jeweils einer einzelnen Verbindung zwischen jedem Input-Neuron u_i und jedem RBF-Neuron v_j wird nun eine Menge von Synapsen mit verschiedenen presynaptischen Verzögerungen angenommen. Die Menge $D = \{d^{(1)}, ..., d^{(l)}\}$ von Verzögerungen ist gleich für alle Inputs mit $d^{(k)} - d^{(k-1)} > 0$ für $2 \leq k \leq l$. Die totale Verzögerung $d_{i,j}^{(l)}$ zwischen dem presynaptischen Feuern und der Auswirkung auf das postsynaptische Neuron entsteht daher durch die Zeit $d^{(l)}$, welche der von u_i emittierte Spike benötigt, um die Synapse $\langle u_i, v_j \rangle$ zu aktivieren und der als konstant angenommenen Zeit d^{post}, welche das Potential von v_j benötigt, um durch den resultierenden PSP eine Auswirkung zu zeigen. Die Zeitspanne $\tilde{\gamma}$ des Kodierungsintervalls, innerhalb welcher die Input-Neuronen feuern dürfen, wird als kleiner als $d^{(l)} - d^{(1)}$ angenommen. Die Verzögerung der Länge $d^{(k)}$ für die Verbindung von u_i zu v_j wird als aktiv bezeichnet, wenn das zugehörige Gewicht $w_{i,j}^{(k)}$ signifikant größer als 0 ist. Weiters wird jedes RBF-Neuron durch eine langsame hemmende Synapse mit sich selbst und durch schnelle hemmende Synapsen mit jedem anderen RBF-Neuron verbunden.

Das Ziel des Lernalgorithmus kann wie folgt definiert werden: Gegeben sei ein Cluster C von Input-Vektoren $\underline{x}$ im Eingaberaum, ein RBF-Neuron v_j, welches für jede Input-Koordinate eine solche Verzögerung aktivieren soll, dass für das daraus resultierende Zentrum $\underline{c}_j$ der Ausdruck $\sum_{x \in C} \|\underline{x} - \underline{c}_j\|$ minimiert wird. Zusätzlich sollen einige Verzögerungen in Querverbindungen aktiviert werden können, da das Zentrum des Clusters möglicherweise durch die endliche Anzahl von verfügbaren Verzögerungen nicht repräsentierbar ist. Eine größere Anzahl von aktivierten Verzögerungen in Querverbindungen kann auch auf eine höhere Varianz der Input-Cluster hindeuten.

Für eine solche Konstellation von mehreren aktiven Verzögerungen in Querverbindungen wird das Zentrum $\underline{c}_j$ eine RBF-Neurons v_j definiert als $\underline{c}_j = \langle c_{j,1}, ..., c_{j,m} \rangle$ mit $c_{j,i} = d_{i,j} - \min\{d_{i,i} \mid 1 \leq i \leq m\}$, wobei $d_{i,j}$ die mittlere Verzögerung für jede Input-Koordinate ist und definiert wird als

$$d_{i,j} = \sum_{k=1}^{l} \frac{w_{i,j}^{(k)} \cdot d^{(k)}}{\sum_{k=1}^{l} w_{i,j}^{(k)}}$$

Die Konvergenz eines RBF-Neurons zum Zentrum eines Clusters kann erreicht werden, indem am Anfang alle Gewichte mit zufälligen Werten so gewählt werden, dass kein RBF-Neuron feuern kann, bevor es mindestens einen Spike von jedem Input-Neuron u_i, $1 \leq i \leq n$ innerhalb des Kodierungsintervalls erhalten hat. Sobald ein RBF-Neuron feuert, wird der Spike rückwärts zu dessen Synapsen propagiert, wo deren Gewicht anhand einer Lernfunktion $L(\Delta t)$ geändert wird, wobei $\Delta t = t_{pre} - t_{post}$ die Differenz zwischen den Ankunftszeitpunkten t_{pre} des presynaptischen und t_{post} das postsynaptischen Spikes angibt. $L(\Delta t)$ wird so gewählt, dass die Gewichte jener Synapsen, welche kurz vor dem postsynaptischen Feuern einen presynaptischen Spike erhielten, maximal verstärkt werden und Synapsen, welche einen presynaptischen Spike lange davor oder danach erhielten, verringert werden. D.h. das Maximum von $L(\Delta t)$ liegt bei einem kleinen negativen Wert.

Das Gewicht $w_{i,j}^{(k)}$ der Synapse zwischen dem Input-Neuron u_i und dem RBF-Neuron v_j mit

Verzögerung $d^{(k)}$ wird während des Lernprozesses nach folgender Regel geändert:

$$\Delta w_{i,j}^{(k)} = \eta \cdot L(t_{pre} - t_{post}) = \eta \cdot L\left((t_i + d^{(k)}) - (t_j + d^{back})\right)$$

für jedes Paar aus Input-Feuerzeitpunkt t_i und Feuerzeitpunkt t_j des RBF-Neurons. $\eta > 0$ bezeichnet die Lernrate, die Verzögerung d^{back} gibt die Zeit an, die ein postsynaptischer Spike benötigt, um zurück zu den Synapsen propagiert zu werden. Es wird angenommen, dass d^{back} konstant für all Synapsen, also unabhängig von deren Verzögerung ist. Diese Eigenschaft wird durch biologische Untersuchungen belegt und stellt daher vermutlich eine korrekte Annahme dar. Weiters wird angenommen, dass die Gewichtswerte einer Sättigungsfunktion im Bereich $[0, w_{max}]$ für ein $w_{max} > 0$ unterworfen sind.

Durch das Hinzufügen schneller hemmender Querverbindungen kann eine winner-take-all Funktion implementiert werden, sodass nur dasjenige RBF-Neuron feuert, welches dem Input-Vektor am nächsten liegt. Daher werden auch nur die Gewichtswerte des Gewinnerneurons verändert, wodurch dessen Zentrum noch weiter in Richtung des Input-Vektors verschoben wird. Die langsamen, hemmenden Verbindungen der Neuronen zu sich selbst vermindern die Wahrscheinlichkeit, dass das selbe Neuron zweimal innerhalb weniger Iterationen der Lernregel feuert. Dadurch wird das übliche Clustering-Problem gelöst, dass manche RBF-Neuronen auf eine Menge von Clustern spezialisieren während andere unbenutzt bleiben.

8 Simulation

Dieses Modell stellt einen ersten Entwurf für ein objektorientiertes Framework zur Simulation spikender neuronaler Netze dar. Die Ziele dieses Frameworks sind hohe Flexibilität und Geschwindigkeit bei der Simulation. Dabei wird allerdings das erste dieser beiden im Gegensatz zueinander stehenden Ziele mit höherer Priorität bewertet, die Geschwindigkeit während der Simulation wird also nur insofern optimiert, als die Flexibilität nicht dadurch beeinträchtigt wird. Da alle derzeit untersuchten Netzwerkmodelle für SNNs prinzipiell im Framework modellierbar sein sollen, ist ein großer Grad an Abstraktion notwendig, so dass bei den obersten Klassen in der Hierarchie nur die Basis des in dieser Arbeit verwendeten mathematischen Modells enthalten ist. Konkrete Implementierungen, wie z.B. das *leaky integrate-and-fire* Modell werden dann durch Überladen der entsprechenden abstrakten Methoden erzeugt.

Eine hohe Simulationsgeschwindigkeit soll durch Verwendung ereignisbasierter Simulation so weit wie möglich erreicht werden. Jedoch ist ereignisbasierte Simulation nur für solche Modelle einsetzbar, deren mathematische Funktionen auf explizites Berechnen der jeweiligen Zeitvariablen umformbar sind, da von dem jeweils aktuellen Simulationsstatus aus das jeweils nächste Ereignis in der Simulation und dessen exakter Zeitpunkt berechnet werden müssen. Daher kann es sein, dass bestimmte mathematische Funktionen für die Antwort- und Schwellwertfunktionen nicht abgebildet werden können. Allgemeine Simulation durch numerische Integrationsverfahren sowie Kombination allgemeiner Simulation mit ereignisbasierter Simulation soll zwar im Modell des Frameworks prinzipiell möglich sein, kann aber vermutlich in der ersten konkreten Implementierung aus Zeitgründen nicht berücksichtigt werden. In der ersten Implementierung dieses Frameworks wird nur Support für ereignisbasierte Simulation enthalten sein. Für realistische Modelle spikender neuronaler Netzwerke sind daher bereits existierende und speziell dafür ausgelegte Programmpakete wie z.B. GENESIS besser geeignet und zu bevorzugen.

8.1 Ereignisbasierte Simulation

Diskrete ereignisbasierte Simulation spikender neuronaler Netzwerke kann einen entscheidenen Geschwindigkeitsvorteil gegenüber herkömmlicher, getakteter Simulation bringen, wenn nur verhältnismäßig wenige Neuronen im gesamten Netzwerk zur selben Zeit aktiv sind. Die Ruhephasen zwischen den Aktivitäten müssen nicht explizit berechnet werden, die Simulation behandelt nur die aktiven Phasen. Durch die dadurch entstehende Geschwindigkeitssteigerung könnte eine Simulation deutlich größerer Netzwerke auch auf handelsüblichen Workstations möglich werden.

Das zu entwickelnde Simulationsframework soll daher die Möglichkeiten der diskreten ereignisbasierten Simulation (*DEVS, discrete event simulation*) so gut wie möglich ausnutzen und nur in solchen Fällen auf kontinuierliche (getaktete) Simulation ausweichen, die über ereignisbasierte Simulation nicht behandelt werden können. Allerdings legen die folgend präsentierten Konzepte in Verbindung mit den zuvor behandelten Aussagen über die Berechnungspotenz spikender Neuronaler Netzwerke den Schluss nahe, dass alle zur Erfüllung der Echtzeit-Turing-Äquivalenz nötigen Voraussetzungen durch ereignisbasierte Simulation erfüllt werden können. Dadurch sollte es somit möglich sein sollte, die qualitativen Eigenschaften der in den vorherigen Kapiteln dargelegten theoretischen Untersuchungen in der Simulation zu reproduzieren. Zur Untersuchung quantitativer Eigenschaften können derzeit nur Vermutungen angestellt werden, welche durch den Vergleich von Ergebnissen realistischer Simulationen in GENESIS und mit ereignisbasierter Simulation durchgeführten Experimenten entweder untermauert oder widerlegt werden müssen. Allerdings ist es prinzipiell möglich, die Genauigkeit in der ereignisbasierten Simulation beliebig zu erhöhen und damit einige der nötigen Vereinfachungen zu kompensieren.

Nachfolgende Kapitel wurden durch Diskussionen mit Michael Affenzeller, Herbert Prähofer und Alexander Friedl angeregt.

8.1.1 Konzepte

Eine ereignisbasierte Simulation ist zwar mit Funktionen beliebiger Art möglich, es scheint jedoch im Hinblick auf die zur Erreichung von Echtzeit-Turing-Äquivalenz nötigen Eigenschaften und dem generellen Ziel möglichst hoher Simulationsgeschwindigeit von Vorteil, die im System verwendeten Funktionstypen (für die Antwort- und Schwellwertfunktionen) auf stückweise lineare Funktionen einzuschränken. Diese können sehr einfach durch eine Liste von Segmenten, bestimmt durch Steigungswert mit zugeordnetem Zeitintervall, vollständig spezifiziert werden, ohne dabei eine Enschränkung auf bestimmte Kurvenformen zu erzeugen (siehe Abbildung 2). Durch Aufspalten der zusammengesetzten linearen Segmente auf mehrere Teile mit unterschiedlichen Steigungen können kontinuierliche, nicht-lineare Funktionen beliebig genau angenähert werden, wodurch eine Untersuchung des Einflusses der genauen Kurvenform auf die quantitativen Eigenschaften von SNNs möglich ist.

Sollte sich wider Erwarten herausstellen, dass für die Reproduktion bestimmter Ergebnisse aus Simulationen in GENESIS stetige Funktionen nötig sind, so kann ereignisbasierte Simulation auch zur Generierung dieser Funktionstypen verwendet werden. Es wäre dann allerdings nötig, die stückweise linearen Funktionen durch stückweise polynomiale Funktionen zu ersetzen, wodurch Stetigkeit der Funktionen erreicht werden kann. An der Simulation ändert sich in einem solchen Fall nur, dass die einzelnen Segmente der Funktionen statt einem durch mehrere Parameter beschrieben werden (z.B. zwei Parameter bei Verwendung von Polynomen zweiten Grades für die Beschreibung der Segmente). Weiters können die Segmente durch Verwendung von Bezier- oder Basis-Splines so zusammengesetzt werden, dass die Ableitungen der Funktion stetig sind (der maximale Grad der Ableitung hängt dabei vom Grad der Polynome ab), wodurch die zusammengesetzten Funktionen beliebig "rund" werden können.

Im weiteren Verlauf dieser Arbeit wird aber mit stückweise linearen Funktionen gearbeitet, um die Berechnungen innerhalb der ereignisbasierten Simulation möglichst einfach und performant zu halten.

Bei den hier verwendeten Funktionsdefinitionen wird von einer wie in Abbildung 1 skizzierten Struktur ausgegangen. Es gilt also, dass Neuron v von den Neuronen $u = 1, ..., n$ Input erhält, wobei die Form der presynaptischen Spikes der Neuronen u in der Simulation nicht berücksichtigt wird. Ein Feuern eines Neurons u, also die Erzeugung eines presynaptischen Spikes, wird als Ereignis (*Event*) an alle angebundenen Synapsen $\langle u, v\rangle$ weitergeleitet und von diesen nach einer Verzögerung von $d_{u,v}$ und mit einer Gewichtung durch $w_{u,v}$ in PSPs, ausgedrückt durch stückweise lineare Funktionen $\varepsilon_{u,v}(t)$, wie in Abbildung 2 umgesetzt. Neuron v addiert nun alle einkommenden PSPs und erzeugt selbst einen Spike in Form eines Events, sobald die Summe der einkommenden PSPs $P_v(t)$ zu irgend einem Zeitpunkt den Schwellwert $\Theta_v(t)$ übersteigt. Ein mehrfaches Feuern innerhalb einer kurzen Zeitspanne wird durch die zeitabhängige Modellierung

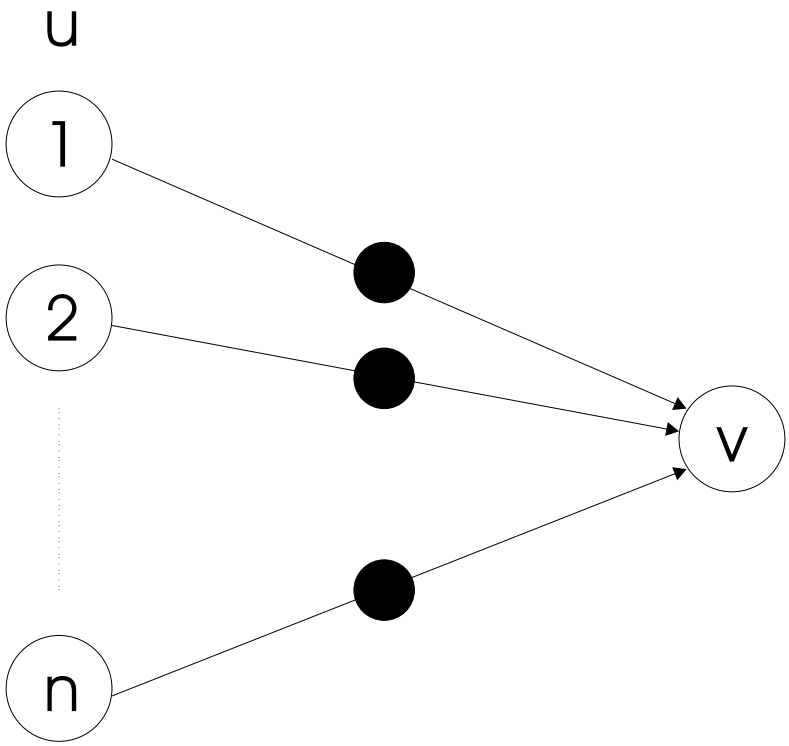

Abbildung 1: Struktur der Verbindungen zwischen Neuronen $u = 1, ..., n$ und v.

von $\Theta_v(t)$, wie in früheren Kapiteln beschrieben, verhindert.

Die hier eingeführen stückweise linearen Funktionen werden wie folgt definiert: Zwischen den Zeitpunkten s_i und s_{i+1} hat die Funktion die Steigung λ_i, wobei $s_0 := 0$, $\lambda_0 := 0$ und $\lambda_m := 0$ und m die Anzahl der Steigungsänderungen angibt, wobei laut Definition die letzte Steigungsänderung m den Wert der Steigung auf 0 setzt, wobei vorausgesetzt, wird, dass der Funktionswert zum Zeitpunkt s_m exakt 0 erreicht hat. Daher muss die letzte Steigung λ_{m-1} zwischen s_{m-1} und s_m für alle Funktionen entsprechend gewählt werden. Die Antwortfunktionen $\varepsilon_{u,v}(t)$ wird daher definiert als

$$\varepsilon_{u,v}(t) := \sum_{i=1}^{k_{u,v}-1} (s_{u,v,i+1} - s_{u,v,i}) \cdot \lambda_{u,v,i} + (t - s_{u,v,k_{u,v}}) \cdot \lambda_{u,v,k_{u,v}}$$

und $\varepsilon_{u,v}(0) := 0$, wobei $k_{u,v} := \max_i(s_{u,v,i} < t)$, also den Index der letzten Steigungsänderung vor dem aktuellen Zeitpunkt t angibt und $\lambda_{u,v,k_{u,v}}$ damit die zum Zeitpunkt t aktuelle Steigung von $\varepsilon_{u,v}(t)$ ist. Wenn nun nach jeder Steigungsänderung der Funktionswert $\varepsilon_{u,v}(s_{u,v,k_{u,v}})$ der Antwortfunktion unmittelbar vor dieser letzten Steigungsänderung berechnet wird, so ergibt sich die rekursive Form

$$\varepsilon_{u,v}(t) = \varepsilon_{u,v}(s_{u,v,k_{u,v}}) + (t - s_{u,v,k_{u,v}}) \cdot \lambda_{u,v,k_{u,v}}$$

Die Annahme, dass $\varepsilon_{u,v}(t) = 0$ für $t \in [0, d_{u,v}]$, also die initiale Verzögerung einer Synapse zwischen dem Ankommen des presynaptischen Spikes und der ersten Auswirkung auf das postsynaptische Potential kann einfach dadurch modelliert werden, dass $s_{u,v,1} = d_{u,v}$ gewählt wird (λ_0 im Intervall $[0, s_{u,v,1}]$ ist definiert als 0).

Das Potential eines Neurons v wird, in Einklang mit dem mathematischen Modell, welches für die theoretischen Überlegungen der vorherigen Kapitel verwendet wurde, definiert als

$$P_v(t) := \sum_{\substack{u \in V \\ \langle u,v \rangle \in E}} \sum_{\substack{s_u \in F_u \\ s_u < t}} w_{u,v} \cdot \varepsilon_{u,v}(t - s_u)$$

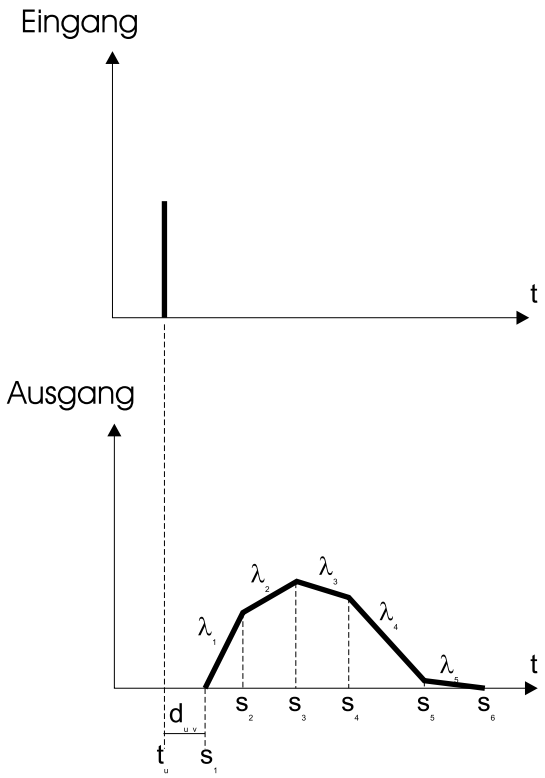

Abbildung 2: Die stückweise lineare Antwortfunktion der Synapse $< u, v >$ mit $m = 6$ (Anzahl der Steigungsänderungen).

und $P_v(0) := 0$, wobei V die Menge der Neuronen, E die Menge der Synapsen und F_u die Menge der Feuerzeitpunkte des Neurons u angeben. Das Potential $P_v(t)$ ist daher die lineare Überlagerung aller einkomenden PSPs von den presynaptischen Neuronen u, wobei mehrere PSPs einer Synapse $\langle u, v \rangle$, jeweils startend zu den Zeitpunkten $s_u \in F_u$ gleichzeitig aktiv sein können.

Die Schwellwertfunktion $\Theta_v(t)$ wird ebenfalls stückweise linear definiert als

$$\Theta_v(t) := \begin{cases} \infty & falls \quad \max(s_{v,i}) \leq t < \max(s_{v,i}) + \tau_{v,ref} \\ \Theta_{v,ref} & \\ -\sum_{i=1}^{l_v-1}(s_{\Theta_v,i+1} - s_{\Theta_v,i}) \cdot \lambda_{\Theta_v,i} & falls \quad \begin{array}{l} \max(s_{v,i}) + \tau_{v,ref} \leq t \\ < \max(s_{v,i}) + \tau_{v,ref} + \tau_{v,end} \end{array} \\ +(t - s_{\Theta_v,l}) \cdot \lambda_{\Theta_v,l} & \\ \Theta_v(0) & sonst \end{cases}$$

und $\Theta_v(0) := 0$, wobei $s_{v,i}$ die Feuerzeitpunkte des Neurons v, $\Theta_{v,ref}$ den initialen Wert von $\Theta_v(\max(s_{v,i}) + \tau_{v,ref})$ nach der Phase mit $\Theta_v(t) = \infty$, l_v die Anzahl von Steigungsänderungen und $\lambda_{\Theta_v,i}$ die Steigungen in den Intervallen $[s_{\Theta_v,i}, s_{\Theta_v,i+1}]$ angeben (siehe Abbildung 3).

Das Ziel der ereignisbasierten Simulation ist nun, ausgehend von den aktuellen Daten jeweils den nächsten Zeitpunkt s_v (in Abbildung 3 aus Gründen der Übersichtlichkeit als t_v bezeichnet) zu berechnen, zu dem das Neuron v feuern wird. Dazu sind mehrere Varianten denkbar, von denen auf zwei näher eingegangen wird. Beide Varianten haben den Vorteil, dass zur Simulation keine dynamischen Speicherstrukturen benötigt werden, sondern zu Beginn der Simulation die Anzahl benötigter Variablen berechnet werden kann und sich während der Simulation nicht mehr ändert.

8.1.2 Simulationsvariante 1

In dieser Variante senden die Synapsen pro auftretender Steigungsänderung der Antwortfunktionen einen Event an die Neuronen, wobei sich überlagernde PSPs der selben Synapse in der Folge

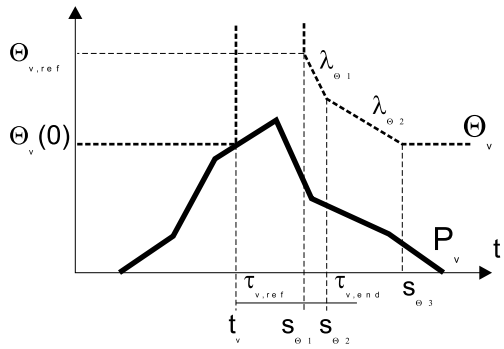

Abbildung 3: Feuerzeitpunkt eines Neurons v, wenn die Potentialfunktion P_v die Schwellwertfunktion Θ_v schneidet.

der übertragenen Steigungsänderungen gesammelt enthalten sind. Daher muss mehr Intelligenz in die Synapsen verlagert werden, um die Überlagerung der gleichzeitig aktiven PSPs, also der zeitlich verschobenen Antwortfunktionen, zu berechnen und in Form einer simplen Folge von Steigungsänderungen an das postsynaptische Neuron zu übertragen. Die Berechnung des Potentials eines Neurons v vereinfacht sich dadurch zu

$$P_v(t) = \sum_{\substack{u \in V \\ \langle u,v \rangle \in E}} w_{u,v} \cdot \varepsilon_{u,v}(t)$$

und $P_V(0) := 0$. Durch Einsetzen der Definition von $\varepsilon_{u,v}(t)$ ergibt sich dann

$$\begin{aligned}
P_v(t) &= \sum_{\substack{u \in V \\ \langle u,v \rangle \in E}} w_{u,v} \cdot \left(\varepsilon_{u,v}(s_{u,v,k_{u,v}}) + (t - s_{u,v,k_{u,v}}) \cdot \lambda_{u,v,k_{u,v}}\right) = \\
&= \sum_{\substack{u \in V \\ \langle u,v \rangle \in E}} w_{u,v} \cdot \varepsilon_{u,v}(s_{u,v,k_{u,v}}) + w_{u,v} \cdot (t - s_{u,v,k_{u,v}}) \cdot \lambda_{u,v,k_{u,v}} = \\
&= \sum_{\substack{u \in V \\ \langle u,v \rangle \in E}} w_{u,v} \cdot \varepsilon_{u,v}(s_{u,v,k_{u,v}}) + \sum_{\substack{u \in V \\ \langle u,v \rangle \in E}} w_{u,v} \cdot (t - s_{u,v,k_{u,v}}) \cdot \lambda_{u,v,k_{u,v}} = \\
&= P_v\left(\max(s_{u,v,k_{u,v}})\right) + t \cdot \sum_{\substack{u \in V \\ \langle u,v \rangle \in E}} w_{u,v} \cdot \lambda_{u,v,k_{u,v}} - \sum_{\substack{u \in V \\ \langle u,v \rangle \in E}} w_{u,v} \cdot s_{u,v,k_{u,v}} \cdot \lambda_{u,v,k_{u,v}}
\end{aligned}$$

wobei $P_v\left(\max(s_{u,v,k_{u,v}})\right)$ dem Potential des Neuron zum Zeitpunkt der letzten empfangenen Steigungsänderung von einer der Synapsen entspricht. Um den nächsten Feuerzeitpunkt s_v des Neurons v zu berechnen, muss nun $P_v(t)$ mit $\Theta_v(t)$ gleichgesetzt und explizit auf t umgeformt werden. Dabei wird davon ausgegangen, dass das Neuron v sich selbst die Steigungsänderungen der Schwellwertfunktion $\Theta_v(t)$ als Events schickt, äquivalent zu den von den Synapsen an das Neuron v versendeten Events. Aufgrund der in der Definition von $\Theta_v(t)$ enthaltenen Fallunterscheidung ergeben sich folgende drei Fälle:

1. $\max(s_{v,i}) \leq t < \max(s_{v,i}) + \tau_{v,ref}$, also das Neuron befindet sich noch in der Phase der absoluten Hemmung. Ein Feuern ist nicht möglich, der nächste Feuerzeitpunkt ist somit (bis zur Änderung von $\Theta_v(t)$ auf einen Wert ungleich ∞):

$$s_v = \infty$$

2. $\max(s_{v,i}) + \tau_{v,ref} \leq t < \max(s_{v,i}) + \tau_{v,ref} + \tau_{v,end}$, also das Neuron befindet sich in der

Phase der relativen Hemmung.

$$s_v = \frac{\Theta_{v,ref} - \sum_{i=1}^{l_v-1}(s_{\Theta_v,i+1} - s_{\Theta_v,i}) \cdot \lambda_{\Theta_v,i} - s_{\Theta_v,l} \cdot \lambda_{\Theta_v,l}}{\sum\limits_{\substack{u \in V \\ \langle u,v \rangle \in E}} w_{u,v} \cdot \lambda_{u,v,k_{u,v}} - \lambda_{\Theta_v,l}} + \frac{-P_v\left(\max(s_{u,v,k_{u,v}})\right) + \sum\limits_{\substack{u \in V \\ \langle u,v \rangle \in E}} w_{u,v} \cdot s_{u,v,k_{u,v}} \cdot \lambda_{u,v,k_{u,v}}}{\sum\limits_{\substack{u \in V \\ \langle u,v \rangle \in E}} w_{u,v} \cdot \lambda_{u,v,k_{u,v}} - \lambda_{\Theta_v,l}}$$

3. sonst, also das Neuron befindet sich nicht mehr in einer Phase der Hemmung (der letzte Feuerzeitpunkt liegt lange genug zurück).

$$s_v = \frac{\Theta_v(0) - P_v\left(\max(s_{u,v,k_{u,v}})\right) + \sum\limits_{\substack{u \in V \\ \langle u,v \rangle \in E}} w_{u,v} \cdot s_{u,v,k_{u,v}} \cdot \lambda_{u,v,k_{u,v}}}{\sum\limits_{\substack{u \in V \\ \langle u,v \rangle \in E}} w_{u,v} \cdot \lambda_{u,v,k_{u,v}}}$$

8.1.3 Simulationsvariante 2

In dieser Simulationsvariante übertragen die Synapsen mit jeweils einem Event die volle Information über einen PSP, also alle Parameter $s_{u,v,i}$ und $\lambda_{u,v,i}$ der Antwortfunktion. Bei Empfang eines solchen Events wird die enthaltene Information dann in eine einzige Folge von Steigungsänderungen, welche innerhalb des Neurons v zwischengespeichert ist, integriert (*merge*). Diese Folge von Steigungsänderungen entspricht damit den Steigungsänderungen des gesamten Potentials $P_v(t)$ des Neurons und entsteht durch gewichtete Überlagerung der verschiedenen empfangenen Folgen von Steigungsänderungen der einkommenden PSPs. Das Potential das Neurons ergibt sich damit zu

$$P_v(t) = \sum_{i=1}^{k_v-1} \lambda_{v,i} \cdot (s_{v,i+1} - s_{v,i}) + \lambda_{v,k_v} \cdot (t - s_{v,k_v})$$

wobei $k_v := \max_i(s_{v,i} < t)$, also dem Index der letzten Steigungsänderung vor der aktuellen Zeit t entspricht.

Bei Empfang eines PSPs (also eines Events mit einer Folge von Steigungsänderungen) von einer Synapse wird dann der nächste Feuerzeitpunkt wie folgt bestimmt:

1. Berechnen von $P_v\left(s_{v,k_v}\right)$, also dem Potential des Neurons v zum Zeitpunkt der letzten Steigungsänderung vor der aktuellen Zeit t, durch gewichtetes Aufsummieren wie oben definiert. Diese letzte Steigungsänderung muss noch außerhalb der neu empfangenen Segmente enthalten sein, da die neu empfangene Folge von Steigungsänderungen nur Zeitpunkte enthalten kann, die nach der aktuellen Zeit t liegen. Dadurch ergibt sich das Potential des Neurons zu späteren Zeitpunkten als

$$P_v(t) = P_v\left(s_{v,k_v}\right) + \sum_{i=j_v}^{k_v-1} \lambda_{v,i} \cdot (s_{v,i+1} - s_{v,i}) + \lambda_{v,k_v} \cdot (t - s_{v,k_v})$$

2. Berechnen von j_v, sodass bereits in $P_v\left(s_{v,k_v}\right)$ enthaltene Segmente der stückweise linearen Funktion $P_v(t)$ nicht mehr in den Steigungsänderungen mit Indizes größer als j_v enthalten sind, d.h. nach der Berechnung von $P_v\left(s_{v,k_v}\right)$ wird j_v auf den Wert von k_v gesetzt (bevor die aktuelle Simulationszeit t fortschreitet).

3. Überlagern der empfangenen Steigungsänderungen mit den gespeicherten.

4. Berechnen des nächsten Feuerzeitpunktes s_v äquivalent zur Berechnung in Simulationsvariante 1, wobei eine Summation über alle Synapsen nicht mehr nötig ist, da die gesammelte Information komprimiert in der einzelnen Liste von Steigungsänderungen im Neuron v enthalten ist.

8.2 Klassendiagramm

Dieses Klassendiagramm (siehe Abbildung 4) stellt nur die von der Simulationsvariante unabängigen Klassen dar und ist lediglich als Grobkonzept zu sehen, welches sich während der frühen Implementierungsphasen unter Umständen noch ändern kann. Dies sind die obersten Klassen in der Hierarchie und müssen zur Simulation um spezialisierte Klassen erweitert werden. Aus Gründen der Übersichtlichkeit werden die Klassen nur konzeptuell, ohne genaue Angabe der Attribute und Funktionen, skizziert.

Diejenigen Klassen, deren Namen mit "DEVS" beginnen, stellen die für ereignisbasierte Simulation speziellen Teile dar. Dabei gibt es im System zwei Klassen von Events: die SpikeEvents, welche das Feuern eines Neurons (das Versenden eines Spikes an die folgenden Synapsen) modellieren und die ResponseEvents, welche die PSPs (die durch die Synapsen hervorgerufenen Auswirkungen auf das Potential des postsynaptischen Neurons) repräsentieren.

Weiters ist entscheidend, dass die Neuronen und Synapsen im System unabhängig von der verwendeten Kodierungsart (Feuerraten, Populationskodierung oder temporale Kodierung) implementiert werden und lediglich die Netzwerkein- und Ausgänge, also die Sensoren und Aktoren, speziell an die Kodierungsart angepasst werden. Innerhalb des Netzes können somit verschiedene Kodierungsarten gleichzeitig verwendet werden, ohne dass die Neuronen und Synapsen dazu speziell angepasst werden müssten.

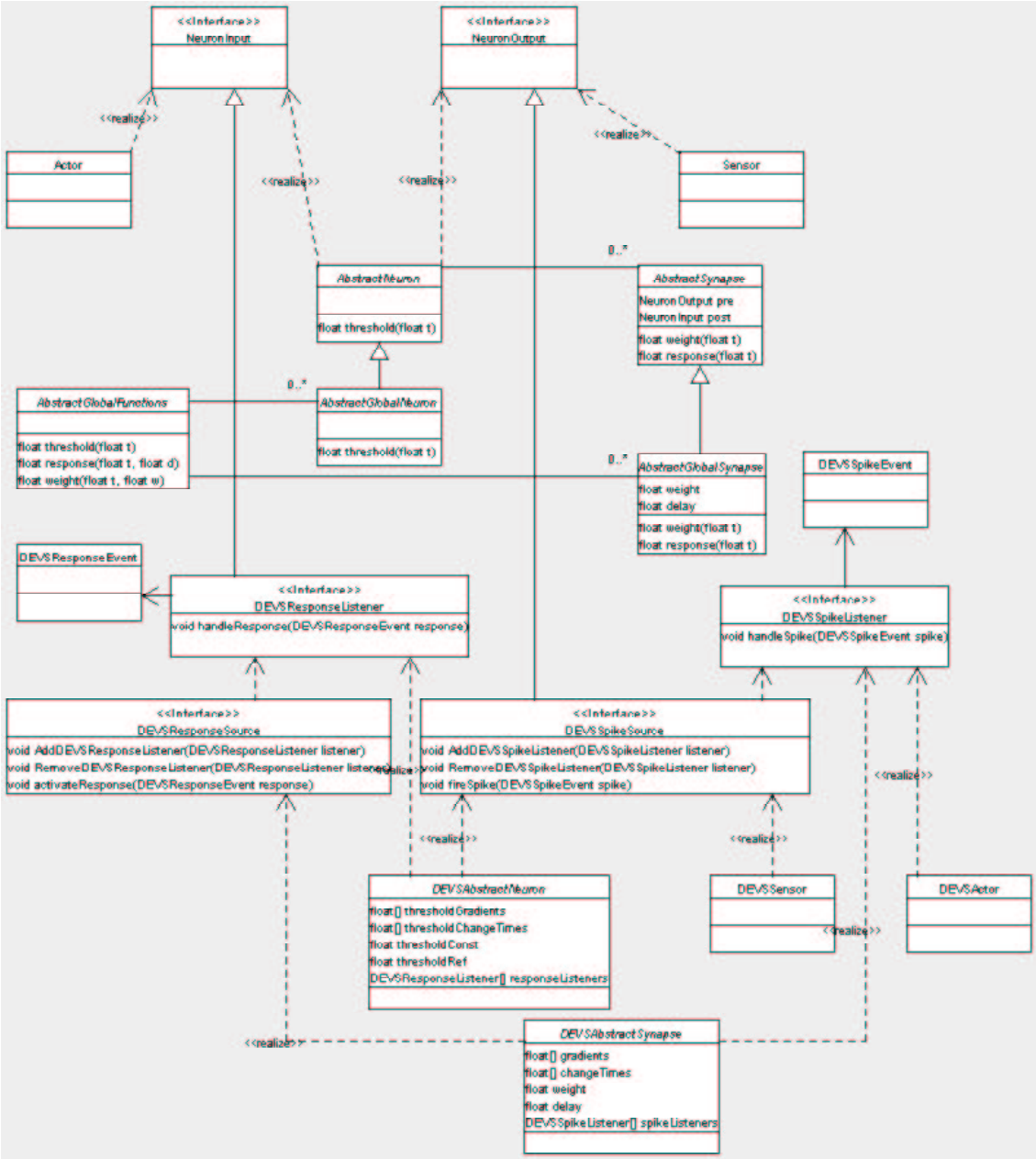

Abbildung 4: Grobkonzept der obersten Klassen im Simulationsframework

9 Ausblick

Durch ereignisbasierte Simulation Spikender Neuronaler Netze könnte ein erheblicher Geschwindigkeitsvorteil entstehen, durch welchen die performante Simulation großer Netzwerke mit mehreren hundert bis zu mehreren tausend Neuronen auf Standard-Workstations ermöglich wird. Es bleibt allerdings noch durch empierische Untersuchungen zu zeigen, dass die für die ereignisbasierte Simulation getroffenen Annahmen und Vereinfachungen keine signifikanten Auswirkungen auf das qualitative Verhalten zeigen. Durch Experimente mit direktem Vergleich der Ergebnisse aus realistischen Simulationen mit GENESIS, ereignisbasierten Simulationen und theoretischen Überlegungen können hierbei aussagekräftige Schlüsse gezogen werden.

Die hier behandelte Art der Simulation könnte weiters die Auftrennung großer Netzwerke in verschiedene Module unterschiedlichen Typs mit einfachen Verbindungen untereinander ermöglichen, wodurch eine Mischung unterschiedlicher Netzwerktypen in einem Anwendungsgebiet erleichtert wird. Diese Möglichkeiten werden allerdings in den konkreten Konzepten des Simulationsframeworks noch nicht behandelt und könnten Gegenstand weiterer Untersuchungen sein.

Literatur

[1] Fred Rieke, David Warland, Rob de Ruyter von Steveninck, William Bialek: Spikes - Exploring the Neural Code, A Bradford Book, First MIT Press paperback edition, 1999

[2] Berthold Ruf, Doctoral Thesis: Computing and Learning with Spiking Neurons - Theory and Simulations, 1998

[3] Wolfgang Maass, Lower bounds for the computational power of networks of spiking neurons, Neural Computationm 8:1-40, 1996

[4] Andreas Zell, Simulation neuronaler Netze, R. Oldenbourg Verlag München Wien, 1994

[5] Wolfgang Maass, Thomas Natschläger: Emulation of Hopfield networks with spiking neurons in temporal coding

[6] Wolfgang Maass, Thomas Natschläger: Networks of Spiking Neurons can emulate arbitrary Hopfield networks in temporal coding, Network: Computation in Neural Systems, 8(4):355-372, 1997

[7] Wolfgang Maass: Paradigms for computing with Spiking Neurons, Models of Neural Networks, volume 4, Springer, 1999

[8] J.J. Hopfield: Pattern recognition computation using action potential timing for stimulus representation, Nature, 3676:33-36, 1995

[9] M.V. Srinivasan, G.D. Bernard: A proposed mechanism for multiplication of neural signals., Biol. Cybernetics, 21:227-236, 1976

[10] Wolfgang Maass: Fast sigmoidal networks via spiking neurons, Neural Computation, 3:105-119, 1997

[11] Wolfgang Maass: Networks of spiking neurons: the third generation of neural network models. Neural Networks, 10(9):1659-1671, 1997

[12] Wolfgang Maass, Anthony M. Zador: Computing and Learning with Dynamic Synapses, Pulsed Neural Networks 321-336, MIT Press, 1998