Modellierung des Wärmeübergangs beim Blasensieden in freier Konvektion mit künstlichen neuronalen Netzen Teil I: Literaturübersicht

Gerardo Diaz, Josef Schmadl, Thomas Schutt, Mihir Sen

1. Einleitung

Ein künstliches neuronales Netz (KNN) ist dem biologischen neuronalen Netz, wie es z. B. im menschlichen Gehirn vorliegt, nachempfunden. Dieses besteht aus 10^{10} bis 10^{11} Nervenzellen (Neuronen), wobei jeweils eine mit 10^3 bis 10^4 anderen Zellen verbunden ist. Eine Zelle besteht u. a. aus dem Zellkörper mit Zellkern, Dendriten und Synapsen. Über die Dendriten erhält die Zelle elektrische Impulse, die im Zellkern aufsummiert werden. Wird ein Schwellwert (*engl.* bias) überschritten, so wird ein Signal über die Synapsen an andere Neuronen gesendet. Analog enthält ein KNN „künstliche", formal-mathematische Neuronen, die in Schichten (*engl.* layer) angeordnet und über Synapsen nach bestimmten Regeln miteinander verknüpft sind. Durch die Vielzahl von Verknüpfungsmöglichkeiten ergeben sich eine Vielzahl von KNN-Typen und -Anwendungen. In dieser Arbeit werden das Funktionsprinzip des formalen Neurons, wichtige Netztypen und KNN-Anwendungen in der Wärmeübertragung besprochen.

2. Das formale Neuron

Das formale Neuron als Grundbaustein jedes KNNs ist in Abbildung 1 schematisch dargestellt.

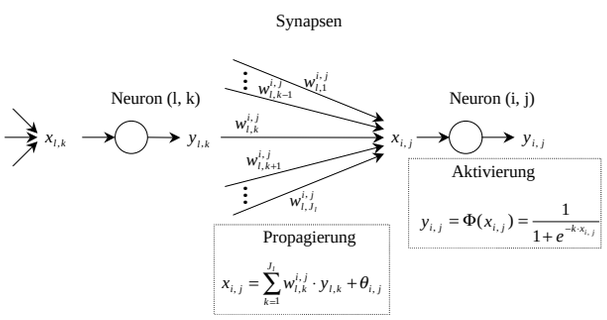

Abb. 1: Formales Neuron

Bezeichnet man mit (i, j) das Neuron j in der Schicht i, mit $x_{i,j}$ und $y_{i,j}$ Ein- bzw. Ausgangsgröße, mit $\theta_{i,j}$ Bias-Wert dieses Neurons und mit $w_{l,k}^{i,j}$ das synaptische Gewicht zweier vektoriell verknüpfter Neuronen (l, k) → (i, j), dann sind Eingangsgröße $x_{i,j}$ des Neurons (i, j) und Ausgangsgröße des Neurons (l, k) durch eine Propagierungsfunktion ξ miteinander verknüpft:

$$x_{i,j} = \xi(y_{l,k}, w_{l,k}^{i,j}, \theta_{i,j}) \tag{1}$$

Das Gewicht $w_{l,k}^{i,j}$ als Kenngröße der vektoriellen Verbindung bestimmt die relative Bedeutung des Ausgangssignals $y_{l,k}$ des Senderneurons (l, k) für das Empfängerneuron (i, j). Der Bias-Wert $\theta_{i,j}$ gehört zum Neuron selbst und stellt einen Freiheitsgrad dar, der in der Trainingsphase zusätzliche Flexibilität ermöglicht. In manchen Netztypen hat jedes Neuron seinen Bias-Wert, in anderen ist der Bias-Wert gleich für alle Neuronen einer Schicht. Mathematisch kann die Propagierungsfunktion ξ verschiedene Formen haben, meist werden die gewichteten Ausgänge aller Senderneuronen (n, k) aufsummiert, manchmal aber auch aufmultipliziert, und um den Bias-Wert korrigiert, z. B. nach der Gleichung:

$$\xi(y_{l,k}, w_{l,k}^{i,j}, \theta_{i,j}) = \theta_{i,j} + \sum_k w_{l,k}^{i,j} \cdot y_{l,k} \tag{2}$$

Die Ausgangsgröße $y_{i,j}$ des Neurons (i,j) hängt mit seiner Eingangsgröße $x_{i,j}$ über eine Aktivierungsfunktion Φ zusammen:

$$y_{i,j} = \Phi(x_{i,j}) \tag{3}$$

Die Aktivierungsfunktion trifft, analog dem natürlichen Denkvorgang, die Entscheidung, ob das Neuron sein Signal unterdrückt oder an seine Empfängerneuronen weitergibt. Als Aktivierungsfunktionen werden häufig eingesetzt:

– Sprungfunktionen von der Form

$$y_{i,j} = \Phi(x_{i,j}) = \begin{cases} 1 & x_{i,j} > 0 \\ 0 & x_{i,j} \le 0 \end{cases} \tag{4}$$

– Schwellwertfunktionen von der Form

$$y_{i,j} = \Phi(x_{i,j}) = \begin{cases} 0 & x_{i,j} < 0 \\ x_{i,j}/\theta & 0 < x_{i,j} \le \theta \\ 1 & x_{i,j} > \theta \end{cases} \tag{5}$$

– sigmoide Funktionen von der Form:

$$y_{i,j} = \Phi\left(x_{i,j}\right) = \frac{1}{1+e^{-k \cdot x_{i,j}}} \qquad \text{für i} > 1 \tag{6}$$

mit einer Konstanten k > 0.

Gl. (6) stellt eine Näherungslösung der Sprungfunktion (4) dar, hat aber zum Unterschied von ihr kontinuierliche Ableitungen und eignet sich deshalb besonders für natur-

wissenschaftlich-technische Anwendungen. Da ihr Wert sich zwischen 0 und 1 bewegt, ist es üblich, alle Eingangsdaten für die Eingangsschicht von Netzen mit sigmoiden Aktivierungsfunktionen mit Hilfe des kleinsten und größten Eingangswertes zu normieren und erst die Ausgangsdaten der letzten Schicht wieder zu entnormieren.

Die Eingangswerte eines Netzes werden so von Neuron zu Neuron über Propagierungs- und Aktivierungsschritte verändert bis zu Ausgangswerten als Rechenergebnis, das mit zuverlässigen Trainingsdaten verglichen wird. In Abhängigkeit von der Qualität dieses Rechenergebnisses werden nun Gewichte und Bias-Werte iterativ so lange verändert, bis ein optimales Ergebnis erreicht ist. Das ist die Trainings- oder Lernphase. So trainiert, ist das KNN dann in der Lage, mehr oder weniger gut zu „denken", d. h. Testdaten zu verarbeiten und zu berechnen. Das Lernverhalten kann im Allgemeinen überwachtes Lernen (*engl.* supervised learning), unüberwachtes Lernen (*engl.* unsupervised learning) oder verstärkendes Lernen (*engl.* reinforcement learning) sein. Dazu werden unterschiedliche Lernregeln gewählt, z. B. ein Gradientenabstiegsverfahren für überwachtes Lernen oder die Hebb'sche Lernregel für unüberwachtes Lernen. In allen Fällen werden Gewichte und Bias-Werte iterativ bis zum Erreichen eines Optimums angepasst. Die Qualität eines KNN hängt von der Generalisierungsleistung und dem Konvergenzverhalten ab. Die Generalisierungsleistung kenn-

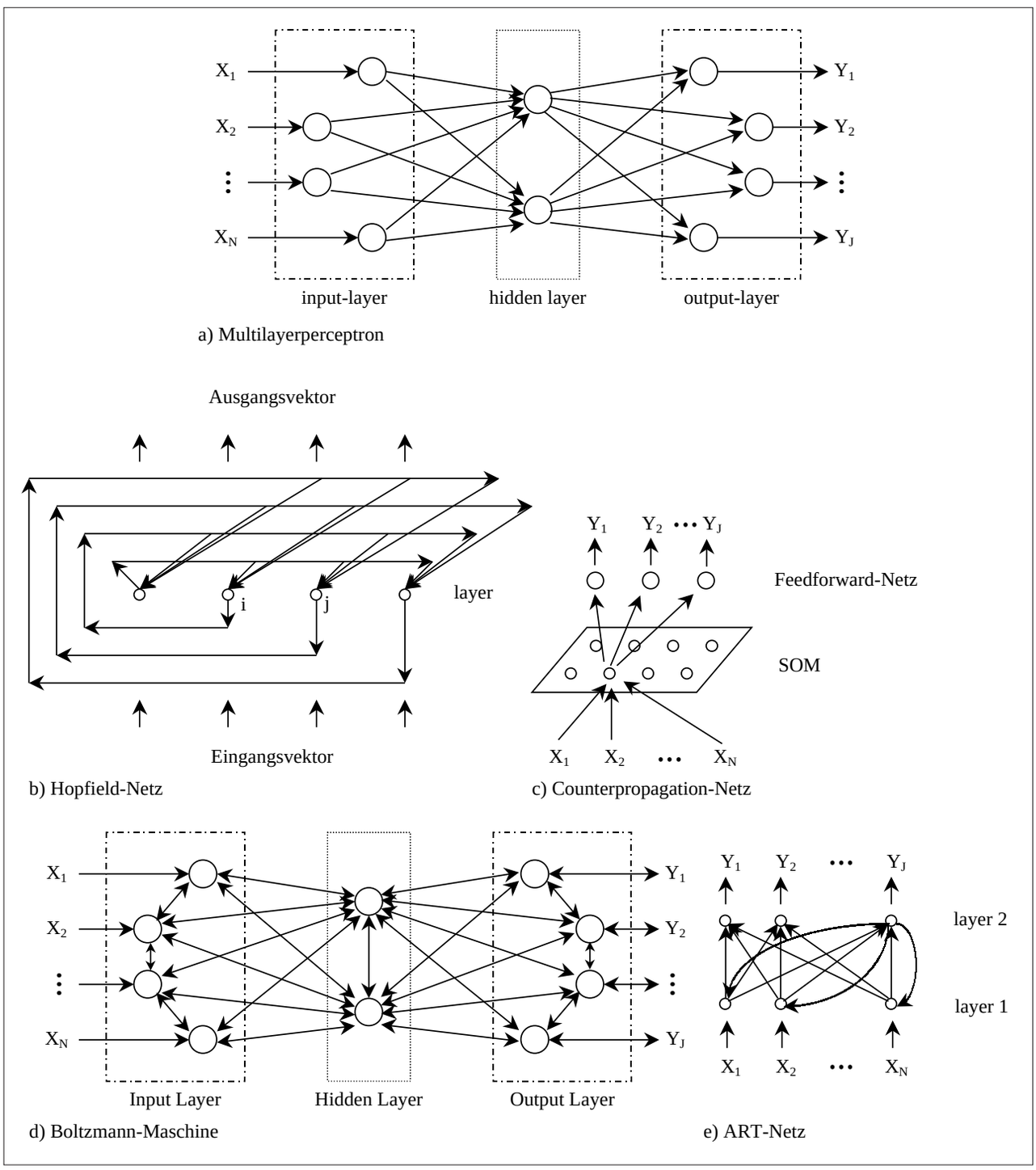

Abb. 2a-e: Neuronenanordnung und -verknüpfung in verschiedenen KNN-Modellen

zeichnet die Interpolationsfähigkeit, das Konvergenzverhalten die Fähigkeit des KNN zur Annäherung an ein stabiles Trainingsergebnis.

3. Wichtige KNN-Typen

In der Literatur sind zahlreiche KNN-Typen beschrieben, die sich in der Art der Neuronenanordnung und -verknüpfung, der angewendeten Lernregel und anderen Merkmalen unterscheiden [1-3]. In Abbildung 2a-e ist eine für technisch-naturwissenschaftliche Anwendungen in Frage kommende Auswahl schematisch dargestellt.

Abbildung 2a enthält ein Schema des besonders in Natur- und Ingenieurwissenschaften vielverwendeten Multilayerperceptrons (MLP). Dieser Typ besteht aus mehreren, aber mindestens drei Schichten, einem inputlayer, einem oder mehreren hidden layer und einem output-layer. Wenn jedes beliebige Neuron (i, j) synaptisch verbunden ist mit jedem Neuron (i-1, k) der davorliegenden Schicht, spricht man von einem voll verknoteten KNN. Die Neuronen derselben Schicht sind untereinander nicht verknüpft. Synapsen sind vektoriell immer zu Neuronen der nachfolgenden, nicht zu solchen davorliegender Schichten ausgerichtet. Die Zahl der Neuronen in der Eingangsschicht J_1 ist gleich der Zahl der Eingangsgrößen. Die Neuronenzahl der Ausgangsschicht ist gleich der Zahl der zu berechnenden Ausgangsgrößen. Die verwendete Propagierungsfunktion ist definiert durch Gl. (1) und (2) mit l = i-1:

$$x_{i,j} = \theta_{i,j} + \sum_{k=1}^{J_{i-1}} w_{i-1,k}^{i,j} \cdot y_{i-1,k} \tag{7}$$

Dies bedeutet, dass die Eingangsgröße $x_{i,j}$ eines beliebigen Neurons (i, j) sich zusammensetzt aus der Summe aller nodalen Ausgangsgrößen $y_{i-1,k}$ der vorherigen Schicht i-1, modifiziert mit den internodalen Gewichten $w_{i-1,k}^{i,j}$ zuzüglich des Bias-Wertes $\theta_{i,j}$ dieses Neurons. Die Aktivierungsfunktion Φ ist sigmoid gemäß Gl. (6) für die Neuronen aller Schichten i>1. Ausnahme davon ist die Eingangsschicht, in der das Eingangssignal nicht verändert wird, die Eingangsgröße $x_{i,j}$ aber in normierter Form eingeht:

$$y_{1,j} = \Phi_{1,j}(x_{1,j}) = x_{1,j} \qquad \text{für i = 1} \tag{8}$$

mit linear oder logarithmisch normiertem Eingangssignal:

$$0 < x_{1,j} = \frac{X_{1,j} - X_{min}}{X_{max} - X_{min}} < 1 \tag{9}$$

$$x_{1,j} = \frac{log_{10} X_{1,j} - X_{min}}{X_{max} - X_{min}} \tag{10}$$

X_{min} und X_{max} sind die beiden Extremwerte aus den verwendeten Trainingsdatensätzen. Eingangs- und Ausgangswerte der ersten Schicht, sämtliche Ausgangswerte und ebenso die Ausgangswerte der letzten Schicht variieren somit zwischen 0 und 1.

Das Netzwerk wird mit dem Errorbackpropagation-Algorithmus, einem Gradientenverfahren, trainiert. Dabei werden M Trainingszyklen durchlaufen. Ein beliebiger Trainingszyklus m generiert einen vollständigen Satz Gewichte $w_{i-1,k}^{i,j}$ und Bias-Werte $\theta_{i,j}$ sowie einen Satz Ergebnisse $y_{I,j}$. Um den Erfolg zu beurteilen, wird die hälftige Fehlerquadratsumme betrachtet:

$$E(m) = \sum_{j=1}^{J_I} \varepsilon_{I,j} = \frac{1}{2} \sum_{j=1}^{J_I} \left(t_{I,j} - y_{I,j}\right)^2$$
$$\text{mit} \quad m = 1, \ldots, M \tag{11}$$

Darin ist J_I die Zahl der Neuronen = Zahl der Ausgangsdaten der letzten Schicht I, t der gemäß Gl. (9) oder (10) normierte Vergleichswert aus dem Trainingsdatensatz für das entsprechende y. In manchen Anwendungen wird ein gemittelter Fehler wie in Gl. (11) oder ähnlich, in anderen der maximale Fehler eines Zyklus m betrachtet. Dieser Fehler wird nun in iterativen Trainingszyklen minimiert. Dabei gelten jeweils für den Endwert $y_{I,j}$ die Gl. (1), (2) und (6) mit i = I und l = I-1, so dass dieser Endwert sich als Funktion f von der Summe der gewichteten Ausgangswerte der vorletzten Schicht I-1 darstellen lässt:

$$y_{I,j} = \Phi\left[\xi\left(\theta_{I,j}, \sum_{k=1}^{J_{I-1}} w_{I-1,k}^{I,j} \cdot y_{I-1,k}\right)\right] =$$
$$f\left(\sum_{k=1}^{J_{I-1}} w_{I-1,k}^{I,j} \cdot y_{I-1,k}\right) \tag{12}$$

Wird Gl. (12) in Gl. (11) eingesetzt, so erhält man den Einzelfehler als Funktion der Gewichte:

$$\varepsilon_{I,j} = \frac{1}{2}\left[t_{I,j} - f\left(\sum_{k=1}^{J_{I-1}} w_{I-1,k}^{I,j} \cdot y_{I-1,k}\right)\right]^2 \tag{13}$$

Würde diese Gleichung grafisch dargestellt, ergäbe das eine Fehlerlandschaft mit etlichen lokalen Minima. Im ersten Trainingszyklus werden die Gewichte zufällig initialisiert und befinden sich irgendwo in der Fehlerlandschaft. In Abbildung 3 wird diese Fehlerlandschaft vereinfacht zweidimensional in der Form $E(m) = f(w)$ als Parabel betrachtet.

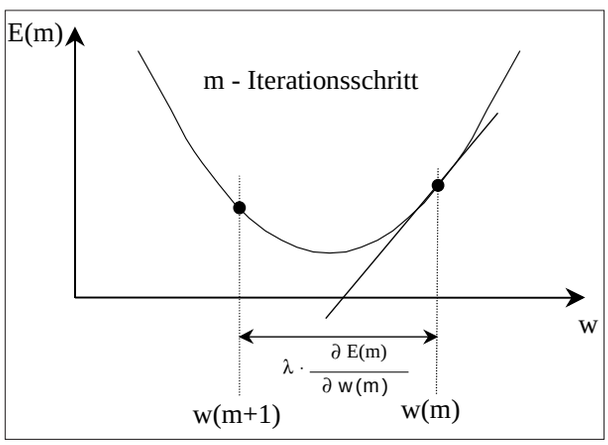

Abb. 3: Abhängigkeit des Fehlers E(m) vom Gewicht w in zwei aufeinanderfolgenden Iterationsschritten m und m+1

Der Fehler im Zyklus m befindet sich irgendwo auf dem linken oder rechten Parabelast. Wird in diesem Punkt eine Tangente angelegt, zeigt der negative Anstieg in Richtung Minimum. Um in darauffolgenden Zyklus m+1 eine Gewichtsveränderung zu erzeugen, die die Entfernung zum Fehlerminimum verringert, wird die Änderung proportional zum negativen Anstieg angenommen:

$$\Delta w = w(m+1) - w(m) \sim -\frac{\partial E(m)}{\partial w(m)} \tag{14}$$

Um daraus eine Gleichung zu bilden, wird ein Proportionalitätsfaktor (oder -funktion) eingeführt und als Lernrate λ bezeichnet:

$$\Delta w = -\lambda \cdot \frac{\partial E(m)}{\partial w(m)} \tag{15}$$

Damit ändert sich das Gewicht im nächstfolgenden Iterationsschritt m+1 um den Betrag Δw.

Gl. (15) wird zunächst auf den output layer angewendet. Um die darin enthaltene partielle Ableitung zu ermitteln, wird Gl. (11) mit Hilfe der Kettenregel partiell abgeleitet und ein δ nach Rigoll [1] eingeführt:

$$\frac{\partial E(m)}{\partial w_{I-1,k}^{I,j}} = \frac{\partial E(m)}{\partial y_{I,j}} \cdot \frac{\partial y_{I,j}}{\partial \xi_{I,j}} \cdot \frac{\partial \xi_{I,j}}{\partial w_{I-1,k}^{I,j}} = -\delta_{I,j} \cdot \frac{\partial \xi_{I,j}}{\partial w_{I-1,k}^{I,j}} \tag{16}$$

Mit ξ gemäß Gl. (1) und (2) und y gemäß Gl. (6) ergibt sich für δ in Gl. (16):

$$\delta_{I,j} = -\frac{\partial E(m)}{\partial \xi_{I,j}} = \left(t_{I,j} - y_{I,j}\right) \cdot y_{I,j} \cdot \left(1 - y_{I,j}\right) \tag{17}$$

und

$$\frac{\partial \xi_{I,j}}{\partial w_{I-1,k}^{I,j}} = y_{I-1,k} \tag{18}$$

Damit erhält man die gesuchte partielle Ableitung des Fehlers nach dem Gewicht:

$$\frac{\partial E(m)}{\partial w_{I-1,k}^{I,j}} = -\left(t_{I,j} - y_{I,j}\right) \cdot y_{I,j} \cdot \left(1 - y_{I,j}\right) \cdot y_{I-1,k} \tag{19}$$

Das in Gl. (17) beschriebene δ gilt nur für den outputlayer. Im letzten hidden layer I-1 und analog in allen vorangehenden hidden layer kann der Fehler analog zu Gl. (11) bis (13) beschrieben werden, wobei die Abhängigkeit des Fehlers E(m) vom Gewicht $w_{I-2,n}^{I-1,k}$ über $y_{I,j}$ hergestellt wird. Während aber im output layer $w_{I-1,k}^{I,j}$ jeweils nur ein $\xi_{I,j}$ beeinflusst, verändert jedes $w_{I-2,n}^{I-1,k}$ alle $\xi_{I,j}$. Deshalb gilt für den Zusammenhang zwischen E(m) und $w_{I-2,n}^{I-1,k}$:

$$E(m) = f\left(\xi_{I,1}(w_{I-2,n}^{I-1,k}), \xi_{I,2}(w_{I-2,n}^{I-1,k}), \ldots, \xi_{I,j}(w_{I-2,n}^{I-1,k})\right) \tag{20}$$

Anders als nach Gl. (16) und (19) erhält man damit für hidden layer I-1 die Ableitung:

$$\frac{\partial E(m)}{\partial w_{I-2,n}^{I-1,k}} = \frac{\partial E(m)}{\partial \xi_{I,1}} \cdot \frac{\partial \xi_{I,1}}{\partial w_{I-2,n}^{I-1,k}} + \ldots +$$

$$\frac{\partial E(m)}{\partial \xi_{I,j}} \cdot \frac{\partial \xi_{I,j}}{\partial w_{I-2,n}^{I-1,k}} + \ldots = \sum_{j=1}^{J_I} \frac{\partial E(m)}{\partial \xi_{I,j}} \cdot \frac{\partial \xi_{I,j}}{\partial w_{I-2,n}^{I-1,k}} \tag{21}$$

mit:

$$\frac{\partial \xi_{I,j}}{\partial w_{I-2,n}^{I-1,k}} = \frac{\partial \xi_{I,j}}{\partial y_{I-1,k}} \cdot \frac{\partial y_{I-1,k}}{\partial \xi_{I-1,k}} \cdot \frac{\partial \xi_{I-1,k}}{\partial w_{I-2,n}^{I-1,k}} \tag{22}$$

und analog zu Gl. (18):

$$\frac{\partial \xi_{I-1,k}}{\partial w_{I-2,n}^{I-1,k}} = y_{I-2,n} \tag{23}$$

Gl. (22) und (23) werden in Gl. (21) eingesetzt:

$$\frac{\partial E(m)}{\partial w_{I-2,n}^{I-1,k}} = \sum_{j=1}^{J_I} \frac{\partial E(m)}{\partial \xi_{I,j}} \cdot \frac{\partial \xi_{I,j}}{\partial y_{I-1,k}} \cdot \frac{\partial y_{I-1,k}}{\partial \xi_{I-1,k}} \cdot$$

$$\frac{\partial \xi_{I-1,k}}{\partial w_{I-2,n}^{I-1,k}} = -\delta_{I-1,k} \cdot y_{I-2,n} \tag{24}$$

Darin ist $\delta_{I-1,k}$ analog aber anders als in Gl. (16) definiert als:

$$\delta_{I-1,k} = -\frac{\partial E(m)}{\partial \xi_{I-1,k}} = -\frac{\partial y_{I-1,k}}{\partial \xi_{I-1,k}} \cdot \sum_{j=1}^{J_I} \frac{\partial E(m)}{\partial \xi_{I,j}} \cdot \frac{\partial \xi_{I,j}}{\partial y_{I-1,k}} \tag{25}$$

Für die darin enthaltenen Ableitungen ergeben sich mit δ aus Gl. (16), ξ aus Gl. (2) und y aus Gl. (6) und (12):

$$-\frac{\partial E(m)}{\partial \xi_{I,j}} = \delta_{I,j} \text{ und } \frac{\partial \xi_{I,j}}{\partial y_{I-1,k}} = w_{I-1,k}^{I,j} \text{ sowie}$$

$$\frac{\partial y_{I-1,k}}{\partial \xi_{I-1,k}} = \left(1 - y_{I-1,k}\right) \cdot y_{I-1,k}$$

Einsetzen in Gl. (25) führt zu folgendem δ für ein beliebiges Neuron k der Schicht I-1:

$$\delta_{I-1,k} = \left(1 - y_{I-1,k}\right) \cdot y_{I-1,k} \cdot \sum_{j=1}^{J_I} \delta_{I,j} \cdot w_{I-1,k}^{I,j} \tag{26}$$

und damit für die Gewichte im hidden layer I-1 nach Gl. (15) und (25):

$$w_{I-2,n}^{I-1,k}(m+1) = w_{I-2,n}^{I-1,k}(m) + \lambda \cdot \delta_{I-1,k} \cdot y_{I-2,n} \tag{27}$$

Analog wird für alle Zwischenschichten i je ein Satz neuer Gewichte erzeugt:

$$w_{i-1,k}^{i,j}(m+1) = w_{i-1,k}^{i,j}(m) + \lambda \cdot \delta_{i,j} \cdot y_{i-1,k} \tag{28}$$

Die Bias-Werte ändern sich in allen Schichten $1 < i \leq I$ um einen mit λ und δ proportionalen Wert, z. B.:

$$\theta_{i,j}(m+1) = \theta_{i,j}(m) + \lambda \cdot \delta_{i,j} \tag{29}$$

Die Verwendung des Produktes $\lambda \cdot \delta$ hat einen empirischen Hintergrund. Andere Zusammenhänge sind

denkbar. Mit den so in jedem Iterationszyklus angepassten Gewichten w und Bias-Werten θ wird das Netz trainiert, getestet und im Erfolgsfalle angewendet.

Wie viele hidden layer gewählt und wie viele Neuronen jeweils darin angeordnet werden, liegt im Ermessen des Anwenders. Sen und Yang [4] empfehlen insbesondere im Falle vieler Eingangsgrößen mit 2 hidden layer zu starten und diese Zahl sukzessive zu erhöhen, wobei die Zahl der Neuronen pro Schicht nicht zu hoch gesetzt werden sollte. Eine zu hohe Neuronenzahl wirkt ähnlich wie ein Polynom, das mit zunehmendem Grad an Interpolationsfähigkeit einbüßt. Andererseits wird mit steigender Neuronenzahl pro Schicht die Konvergenzkapazität des Netzes erhöht. Es muss deshalb ein Optimum der Neuronenzahl durch Versuche gefunden werden. Dieser Netzwerktyp wird z. B. zur komplexen Musterklassifikation und zur Approximation beliebiger stetiger Funktionen verwendet.

Weitere KNN-Typen sind in Abbildung 2b-e dargestellt. Abbildung 2b zeigt den prinzipiellen Aufbau des Hopfield-Netzes. Es besteht aus einem layer mit indirekten Rückkopplungen und symmetrischen Gewichten ($w_{ij}=w_{ji}$). In der Trainingsphase werden einzelne Muster gespeichert, die in der Gebrauchsphase durch verrauschte bzw. unvollständige Muster aufgerufen werden. Dieses Netz wird in der Telekommunikation, für Optimierungs- und Logistik- und andere organisatorische Aufgaben angewendet.

In Abbildung 2c ist das Counterpropagation-Netz schematisch dargestellt. Es besteht aus einer Self-Organizing Map (SOM) und einem einschichtigen Feedforward-Netz. Die SOM übernimmt in diesem Netz die Musterklassifikation und das Feedforward-Netz die Generierung eines Musterschlüssels. Die Netzwerkarchitektur wird bei der Robotermanipulation eingesetzt.

Die in Abbildung 2d dargestellte Boltzmann-Maschine ist eine eher seltene Anwendung. Die Neuronen sind in mehreren Schichten mit indirekten Rückkopplungen, darunter – analog zum MLP – mindestens ein hidden layer zwischen input- und output-layer, angeordnet. Anders als die übrigen hier dargestellten Netztypen enthält sie ein wahrscheinlichkeitstheoretisches Modell zur Berechnung der Aktivierungsfunktion. Als Lernregel wird das „simulated annealing“ eingesetzt. Verarbeitet werden binäre Werte. Sie wird zur Optimierung in kaufmännischen Bereichen eingesetzt.

Das in Abbildung 2e beschriebene ART-Netz ist zweischichtig und selbstorganisierend mit Rückkopplungen, wie beispielhaft am Neuron m der Schicht 2 dargestellt. Selbstorganisierend bedeutet, dass nicht vom Benutzer vorgegebene Klassen verwendet werden, sondern das Netz von sich aus die Klasseneinteilung vornimmt. Die Gewichte werden mit dem leader-clustering-Algorithmus ermittelt. Das Besondere an diesem Netzwerk ist, dass es auch während der Gebrauchsphase noch lernen kann. Es dient der Musterklassifikation z. B. bei der Überführung gedruckter Zeichen in Bitmuster.

4. KNN-Anwendungen in der Wärmeübertragung

Sen und Yang [4] geben eine Übersicht zur Anwendung genetischer Algorithmen einschließlich KNNe in der Wärmeübertragung. Diese Übersicht haben wir durch eine eigene Literaturrecherche ergänzt [5-16]. Wie die Literaturzitate zeigen, werden häufig die in der Kerntechnik wichtige kritische Wärmestromdichte beim Übergang vom Blasen- zum Filmsieden, in anderen Fällen verschiedene thermophysikalische Stoffdaten berechnet. Berechnungen zum Wärmeübergang bei dem technisch wichtigen Blasensieden wurde bislang in der Literatur nicht gefunden.
Sen und Yang [4] berechneten die Leistung zweier Rippenrohrwärmeüberträger für den Fall der konvektiven Wärmeübertragung ohne Änderung des Aggregatzustandes mit einem selbstentwickelten KNN. In einem Fall wurde der Einfluss der KNN-Konfiguration und der Normierungsgrenzen betrachtet und der Wärmestrom mit Hilfe des KNN berechnet. Im zweiten Fall ging es um die Berechnung der Colburn-Faktoren für die sensible und latente Wärme und den Vergleich der KNN-Leistungsfähigkeit mit den konventionellen Korrelationen vom Typ $Nu = Nu(\mathrm{Re}, \mathrm{Pr})$. Außerdem wurde ein dynamisches Modell zur Vorhersage des zeitlichen Verlaufes der Austrittstemperatur untersucht mit dem Ziel der Prozessregelung.

Mazzola [5] beschreibt die Anwendung eines KNN vom Typ MLP in Kombination mit einer herkömmlichen Gleichung zur Berechnung der kritischen Wärmestromdichte bei unterkühltem Sieden. In der Gleichung tritt ein empirischer Faktor k auf, der mit Hilfe des Netzwerkes vorausgesagt wird. Das Netz besteht aus 3 Schichten, mit 6 Eingangsgrößen und einer Ausgangsgröße. Im hidden layer sind 7 formale Neuronen angeordnet. Der Bias-Wert ist für jedes Neuron einer Schicht gleich. Als Aktivierungsfunktion wird eine Tangens-Hyperbolikus-Funktion eingesetzt. Die Werte werden zwischen –0,8 und +0,8 skaliert. Zur Ermittlung der Gewichte wird die Errorbackpropagation mit Momentumterm verwendet. Der sogenannte Lernfaktor ist – anders als bei Sen und Yang – keine Konstante, sondern eine Funktion der Iterationsstufe mit einem sogenannten Momentum als ein konstanter Faktor (0,9), dessen Wert aber auf Null fällt, sobald ein Iterationsschritt zur Verschlechterung des Gesamtfehlers führt. Zum Training wurden 1808 Datensätze verwendet, wobei diese sich in zwei Datenmengen unterteilen. Eine Untermenge wurde durch die zufällige Wahl von 904 Daten gebildet. Die eine Menge wurde zum Training verwendet, während die andere zur Validierung der Generalisierungsleistung genutzt wurde. Die Anpassung der Gewichte erfolgte nach 5000 Iterationen. Der Vergleich mit der rein empirischen Berechnung (Standardabweichung: 20,2 %) erbrachte eine Verbesserung durch die Anwendung des KNN (Standardabweichung: 12,1 %).

5. Zusammenfassung

Künstliche neuronale Netze werden in technisch-naturwissenschaftlichen Anwendungen zunehmend eingesetzt. In vielen Fällen können sie konventionelle empirische und halbempirische Korrelationen von experimentellen Daten ersetzen und liefern bei interpolativen Vorhersagen bessere Ergebnisse als diese. Zum Einsatz kommt überwiegend das Multilayerperceptron. Sen und Yang geben eine Übersicht zu KNN-Anwendungen in der Wärmeübertragung, die mit dieser Arbeit ergänzt wurde. KNN-Anwendungen zur Berechnung des Wärmeübergangs beim Behältersieden wurden bislang in der Literatur nicht gefunden.

6. Danksagung

Für die finanzielle Unterstützung dieses Projektes danken wir dem Ministerium für Wissenschaft, Forschung und Kultur des Landes Brandenburg.

7. Literatur

[1] Rigoll, G.: Neuronale Netze – Eine Einführung für Ingenieure, Informatiker und Naturwissenschaftler; expert verlag, Renningen-Malmsheim 1994.

[2] Bothe, H.-H.: Neuro-Fuzzy-Methoden – Einführung in Theorie und Anwendungen; Springer-Verlag, Berlin, Heidelberg 1998.

[3] Brause, R.: Neuronale Netze – Eine Einführung in die Neuroinformatik; 2. Aufl.; B. G. Teubner; Stuttgart 1995 [1991].

[4] Young, K.T.; Sen, M. et al: Applications of Artificial Neural Networks and Genetics Algorithms in Thermal Engineering. In: CRC Handbook of Thermal Engineering, Section 4, 620-661.

[5] Mazzola, A.: Integrating artificial neural networks and empirical correlations for the prediction of water-subcooled critical heat flux. In: Revue Generale de Thermique 36, 799-806, 1997.

[6] Goll, E. S.; Jurs, P. C.: Prediction of the normal boiling points of organic compounds from molecular structures with a computational neural network model. In: Journal of chemical information and computer sciences 39 (6), 974-983, 1999.

[7] Goll, E. S.; Jurs, P. C.: Prediction of vapor pressures of hydrocarbons and halohydrocarbons from molecular structure with a computational neural network model. In: Journal of chemical information and computer sciences 39 (6), 1081-1089, 1999.

[8] Homer, J.; Generalis, S. C.; Robson, J. H.: Artificial neural networks for the prediction of liquid viscosity, density, heat of vaporization, boiling point and Pitzer's acentric factor – Part I: Hydrocarbons. In: Physical Chemistry Chemical Physics 1 (17), 4075-4081, 1999.

[9] Bunz, A. P.; Braun, B.; Janowsky, R.: Quantitative structure-property relationships and neural networks: correlation and prediction of physical properties of pure components and mixtures from molecular structure. In: Fluid Phase Equilibria 160, 367-374, 1999.

[10] Tetteh, J.; Suzuki, T.; Metcalfe, E.; Howells, S.: Quantitative structure-property relationships for the estimation of boiling point and flash point using a radial basis function neural network. In: Journal of chemical information and computer sciences 39 (3), 491-507, 1999.

[11] Lombardi, C.; Mazzola, A.: A criterion based on independent parameters for distinguishing departure from nucleate boiling and dryout in water cooled systems. In: Revue Generale de Thermique 37, 31-38, 1998.

[12] Moon, S. K.; Baek, W. P.; Chang, S. H.: Parametric trends analysis of the critical heat flux based on artificial neural networks. In: Nuclear Engineering and Design 163 (1-2), 29-49, 1996.

[13] Moon, S. K.; Chang, S. H.: Classification and prediction of the critical heat-flux using fuzzy theory and artificial neural networks. In: Nuclear Engineering and Design 150 (1), 151-161, 1994.

[14] Na, M. G.: Application of a genetic neuro-fuzzy logic to departure from nucleate boiling protection limit estimation. In: Nuclear Technology 128 (3), 327-340, 1999.

[15] Baermann, F.; Greye, G.-R.; Lieberam, A.: Einsatz neuronaler Netze zur Prognose von thermophysikalischen Stoffkonstanten reiner Stoffe aus Strukturinformationen. In: Chemie-Ingenieur-Technik 65 (3), 310-314, 1993.

[16] Egolf, L.M.; Jurs, P.C.: Prediction of boiling points of organic heterocyclic compounds using regression and neural network techniques. In: Journal of chemical information and computer sciences 33 (4), 616-625, 1993.

Autoren

Gerardo Diaz
Prof. Dr. Mihir Sen
University of Notre Dame
Department of Aerospace and Mechanical Engineering
Notre Dame, IN 46556, USA
E-Mail: Mihir.Sen.1@nd.edu

Prof. Dr.-Ing. Josef Schmadl
Thomas Schutt
Technische Fachhochschule Wildau
Technikum für Thermische Verfahrenstechnik
E-Mail: jschmadl@igw.tfh-wildau.de