

Prof. Dr. Andreas Mielke
Andreas-Hofer-Weg 47
D-69121 Heidelberg
Germany

www.andreas-mielke.de
e-mail info@andreas-mielke.de

Neuronale Netze

Andreas Mielke

4. März 2011

Neuronale Netze können in vielen Bereichen zur Simulation eingesetzt werden, zum Beispiel in der Medizin oder in der Wirtschaft. Dabei müssen einige Punkte beachtet werden, die wesentlich für die erfolgreiche Anwendung eines neuronalen Netzes sind. In diesem Dokument werden folgende Fragen diskutiert: Was ist bei der Planung eines neuronalen Netzes zu beachten? Wie wird ein neuronales Netz entworfen? Wie lassen sich Fehler, die bei der Simulation auftreten, behandeln, und welche Informationen kann man daraus gewinnen? Wie kann man mit Hilfe des neuronalen Netzes Regeln entwickeln? Wie läßt sich das neuronale Netz optimieren, und wie lassen sich die erzielten Ergebnisse kontrollieren? Am Ende wird eine Beispielanwendung aus dem medizinischen Bereich vorgestellt.

Inhaltsverzeichnis

1	Möglichkeiten und Grenzen neuronaler Netze	4
2	Planung	6
2.1	Einführung	6
2.2	Problemstellung	7
2.3	Zielvorgaben	7
2.4	Ressourcen	7
2.5	Personal	9
2.6	Projektkosten	9
2.7	Dokumentation	9
3	Daten	10
3.1	Einführung	10
3.2	Datenkodierung	10
3.3	Qualität und Quantität	11
4	Netzwerk	11
4.1	Einführung	12
4.2	Entwurf eines Mehr-Schicht Perceptrons	12
4.3	Zwischenschichten	12
4.4	Lernalgorithmen	13
5	Fehler	14
5.1	Grundlegende Bemerkungen	14
5.2	Fehlerquellen	14
5.3	Fehler bei Klassifizierungen	15
5.4	Fehler bei kontinuierlichen Ausgabedaten	15
5.5	Abschätzung der Fehler	15
5.6	Bedeutung und Interpretation von Fehlern	16
5.7	Spezialfälle	16
6	Regeln	17
6.1	Einführung	17
6.2	Methoden zum Finden von Regeln	17
6.3	Beurteilung von Regeln	18
7	Optimierung und Kontrolle	19
7.1	Einführung	19
7.2	Optimierung der Struktur des Netzes	19
7.3	Optimierung und Kontrolle der Software	20
7.4	Optimierung der Hardware	20
8	Anwendung	21
8.1	Beschreibung des Projekts	21
8.2	Vorarbeiten	21
8.3	Implementierung eines neuronalen Netzes	21

<i>Neuronale Netze</i>	3
8.4 Ausblick	22
9 Links zu anderen WWW-Seiten über neuronale Netze	22

1 Möglichkeiten und Grenzen neuronaler Netze

Neuronale Netze, oder besser künstliche neuronale Netze sind Software-Produkte. Neuronale Netze imitieren in sehr abstrakter Weise einen Teil eines Nervensystems. Neuronale Netze wurden zuerst entwickelt, um Abläufe im Gehirn besser zu verstehen. Inzwischen nutzt man sie für kognitive Anwendungen, z.B. für Lernzwecke oder Optimierungen. Im Idealfall lernt das neuronale Netz ähnlich wie ein Gehirn an Beispielen. In anthropomorpher Sprechweise kann man die Arbeitsweise eines neuronalen Netzes etwa so beschreiben: Gibt man dem neuronalen Netz genügend viele Beispiele zum Lernen, dann kann es die gewonnene Erfahrung nutzen, um sie auf andere Fälle anzuwenden. Diesen Lernprozeß bezeichnet man als Training eines neuronalen Netzes.

Neuronale Netze können in sehr unterschiedlichen Bereichen zur Modellierung eingesetzt werden. Typische Anwendungsbereiche sind medizinische Diagnostik, Analyse von Wirtschaftsdaten, Kontrolle von Fertigungsprozessen, Vorhersage von Devisenkursen, Robotik, etc. In diesen Anwendungen besteht die Aufgabe des neuronalen Netzes darin, aus vorgegebenen Daten weitere Informationen zu gewinnen. Neuronale Netze werden vor allem in Fällen eingesetzt, wo es nicht möglich ist, diese Informationen auf einfachere Art und Weise zu erhalten.

Im Prinzip kann man jede mathematische Funktion mit Hilfe eines neuronalen Netzes lernen. Neuronale Netze lassen sich besonders gut dann verwenden, wenn man eine Funktion nicht kennt, aber viele Daten zur Verfügung hat. Speziell Klassifizierungsprobleme oder Probleme, bei denen eine komplizierte Funktion aus Daten approximativ dargestellt werden soll, lassen sich mit Hilfe neuronaler Netze sehr gut bearbeiten.

Bevor man eine Simulation mit einem neuronalen Netz startet, sollte genau überlegt werden, welche Alternativen es gibt und ob die Fragestellung mit einem neuronalen Netz sinnvoll bearbeitet werden kann. Beispielsweise würde es keinen Sinn machen, die Lottozahlen der nächsten Woche aus den Ziehungen der vergangenen fünf Wochen berechnen zu wollen. In diesem Fall steht fest, daß zwischen verschiedenen Ziehungen keine Korrelationen bestehen. Deshalb ist es sinnlos zu erwarten, eine solche Rechnung könnte erfolgreich sein. Eine Wettervorhersage aufgrund des Wetters der vergangenen fünf Tage könnte dagegen mit einem neuronalen Netz möglich sein. Allerdings gibt es hier sehr viele zusätzliche Gesetzmäßigkeiten, die in Wetterberechnungen benützt werden; ein neuronales Netz kann diese Informationen nicht nutzen und andere Methoden sind dem neuronalen Netz daher überlegen. Ein Text über allgemeine mathematische Modelle [ms.html](#) diskutiert unterschiedliche Aspekte solcher Modelle und ihre Simulation. In vielen Problemen erwartet man Korrelationen, kennt aber keine zusätzlichen Gesetzmäßigkeiten. Solche Probleme sind prädestiniert für die Anwendung eines neuronalen Netzes. Ein sehr interessantes Anwendungsbeispiel ist ein Programm, das mit Hilfe weniger Eingabeparameter eine Risikoabschätzung für Prostatakrebs <http://www.charite.de/ch/uro/schwerpunkte/probiopsie.html> liefert. Dazu werden wenige Parameter, deren genaue Korrelation nicht explizit bekannt sind, verwendet. Ein neuronales Netz wurde mit Datensätzen von vielen Prostatapatienten trainiert und liefert Aussagen zum Risiko für Prostatakrebs. Ein komplizierteres Beispiel für eine medizinische Anwendung wird im Abschnitt 8 (medizinische Anwendung) vorgestellt. Weitere Anwendungsbeispiele finden Sie unter den Links, die wir im Abschnitt 9 (Links) angeben.

In künstlichen neuronalen Netzen werden einige Strukturen eines Nervensystems in karrikativer Weise imitiert, um so ein Programm zu erhalten, mit dem Daten in einer bestimmten Weise verarbeitet werden können. Ein künstliches neuronales Netz besteht aus einer Menge von Kno-

ten und deren Verbindungen untereinander, wobei jeder Knoten eine einzelnen Nervenzelle modelliert. Jeder Knoten wird durch eine Variable beschrieben, die seinen Zustand anzeigt. Für jede Verbindung zwischen zwei Knoten wird eine weitere Variable eingeführt, die die Stärke der Verbindung zwischen den Nervenzellen modelliert. Die Abbildung zeigt eine Skizze eines neuronalen Netzes.

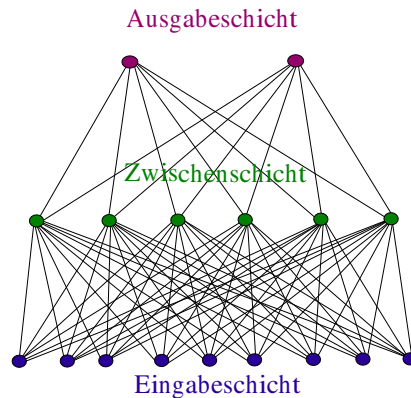


Abbildung 1: Ein neuronales Netz

Neben den genannten Größen gibt es noch weitere Parameter, die zum Verständnis weniger wichtig sind. Häufig betrachtet man neuronale Netze, die aus verschiedenen Schichten bestehen, einer Eingabeschicht, mehreren Zwischenschichten und einer Ausgabeschicht. Die Aufgabe eines solchen Netzes kann zum Beispiel in einer Klassifizierung bestehen: Die Eingabedaten werden in die Eingabeschicht eingespeist und die Ausgabeschicht sortiert diesen Eingabedatensatz in eine bestimmte Klasse. Die oben erwähnten medizinischen Beispiele fallen in diese Klasse

Neuronale Netze werden mit vollständigen Datensätzen trainiert. Ein vollständiger Datensatz besteht aus den Eingabedaten und den Ausgabedaten. Man variiert die Parameter, die das Netz beschreiben, im wesentlichen also die Kopplungen zwischen den Knoten, in geeigneter Weise, so daß das Netz bei den vollständigen Datensätzen die richtigen Ausgabedaten liefert. In einem zweiten Durchgang wird das neuronale Netz mit einem zweiten, unabhängigen Satz von vollständigen Datensätzen getestet. Beide Datensätze, der zum Trainieren und der zum Testen müssen groß genug sein und vor allem richtig verteilt sein. Was das genau heißt, werden wir im Abschnitt 3 (Daten) sehen. Für das obige medizinische Beispiel bedeutet das, daß man die Laborwerte von sehr vielen Patienten, beispielsweise aus mehreren großen Kliniken oder Studien zum Trainieren und Testen des Netzes benutzt. Sind die Tests erfolgreich, dann enthält das Netz quasi die Information aller dieser Daten in komprimierter Form und kann nun aus den Labordaten eines neuen Patienten Vorhersagen darüber machen, ob dieser Patient die Krankheit hat. Da das neuronale Netz mit den Daten von sehr vielen Patienten trainiert wurde, enthält es quasi eine Erfahrung, die, soweit es die Interpretation der Labordaten angeht, über die eines einzelnen Arztes eventuell deutlich hinausgeht. Die Ausgabe des neuronalen Netzes kann deshalb für den Arzt eine wichtige Hilfe sein. Natürlich sollte er sich nie allein darauf verlassen, sondern versuchen, mit weiteren diagnostischen Verfahren die Vermutung zu erhärten oder zu widerlegen.

Folgende Schritte sind wichtig, wenn man beginnt, mit einem neuronalen Netz zu arbeiten:

1. Man sollte überprüfen, ob es andere, effizientere Methoden gibt um das Problem zu lösen. Auch eine Machbarkeitsstudie ist sinnvoll: Kann das Problem mit einem neuronalen Netz behandelt werden.
2. Das Projekt muß sorgfältig geplant werden. Dazu gehört die Planung von Ressourcen, Personal, Kosten und Dokumentation.
3. Der nächste Schritt besteht darin, Standards für die Datenbeschaffung und -kodierung festzulegen. Qualität und Quantität der Daten müssen überprüft werden.
4. Jetzt kann man das neuronale Netz entwickeln. Meist werden mehrere Trainingszyklen benötigt um eine optimale Struktur zu erhalten.
5. Für das trainierte Netz müssen Fehlerschranken für die Ausgabedaten sorgfältig abgeschätzt werden.
6. Es ist möglich, mit Hilfe des trainierten Netzwerks Regeln zu entwickeln. Regeln sind für Optimierung und Kontrolle in der späteren Nutzung wichtig.
7. Das Netzwerk kann jetzt benutzt werden. Dabei sollten permanente Kontrollen vorgesehen werden. Es ist auch möglich, kleine Optimierungen vorzunehmen.

2 Planung

Sie finden hier einige, eigentlich selbstverständliche Bemerkungen, die bei der Planung jedes Projekts zu berücksichtigen sind, sowie Aspekte, die bei der Planung von neuronalen Netzen im Besonderen zu beachten sind.

2.1 Einführung

In sehr vielen Fällen besteht die Aufgabe eines neuronalen Netzes darin, bestimmte Ausgabedaten aus gegebenen Eingabedaten zu berechnen. Der Zusammenhang zwischen Ein- und Ausgabedaten muß eindeutig sein, d.h. einem vorgegebenen Satz von Eingabedaten wird genau ein Satz von Ausgabedaten zugeordnet. Anders ausgedrückt: Es gibt eine Funktion, die die Eingabedaten auf die Ausgabedaten abbildet. Diese Funktion soll von dem neuronalen Netz gelernt werden. Dazu wird das neuronale Netz mit einer Menge von vollständigen Datensätzen, bestehend aus Ein- und Ausgabedaten trainiert. Ferner benötigt man eine Testmenge von vollständigen Datensätzen, um zu überprüfen, wie gut das neuronale Netz die gesuchte Funktion gelernt hat. Da man in der Regel nur eine endliche Menge von Trainings- und Testdaten zur Verfügung hat, kann die Funktion nur approximativ gelernt werden. Das trainierte neuronale Netz liefert eine Funktion, die die gesuchte Funktion mehr oder weniger gut approximiert. Da die gesuchte Funktion aber ohnehin nicht eindeutig festliegt, spielt diese Einschränkung keine wesentliche Rolle. Eine wesentliche Aufgabe bei der Arbeit mit neuronalen Netzen besteht aber darin, die Fehler der Ausgabedaten, die das neuronale Netz liefert, zuverlässig abzuschätzen. Dieser Aspekt wird in Anwendungen leider häufig vernachlässigt. Wir werden auf die Fehlerabschätzung im Abschnitt 5 (Fehler) ausführlich eingehen.

Da die Funktion, die das neuronale Netz lernen soll, zumindest in praktischen Anwendungen unbekannt ist, ist eine genaue Planung des Vorgehens nötig. Einige Hinweise, was dabei zu

beachten ist, werden im folgenden gegeben. Zuerst muß die Problemstellung festgelegt werden. Folgende Fragen sind zu klären:

- Was ist die Problemstellung?
- Welche Ziele sollen erreicht werden?
- Welche Ressourcen stehen zur Verfügung?
- Wer wird an dem Projekt mitarbeiten?

2.2 Problemstellung

Bei der Problemstellung sind alle möglichen Methoden in Betracht zu ziehen, die zu einer Lösung des Problems führen können. Häufig ist die Simulation durch ein neuronales Netz nur eine Möglichkeit von vielen. In vielen Fällen sind bestimmte Gesetzmäßigkeiten bekannt, die für ein zu simulierendes System gelten. In einem neuronalen Netz lassen sich solche Gesetzmäßigkeiten in der Regel nicht implementieren. Ein neuronales Netz ist dann eventuell keine optimale Simulationsmöglichkeit und andere Verfahren liefern bessere Resultate. Andererseits kann ein neuronales Netz benutzt werden, um Regeln oder Gesetzmäßigkeiten abzuleiten. Wir kommen im Abschnitt 6 (Regeln) darauf zurück. Sind umgekehrt Gesetzmäßigkeiten bekannt, können diese benutzt werden, um die Konsistenz des neuronalen Netzes zu überprüfen.

2.3 Zielvorgaben

Gerade bei der Simulation mit neuronalen Netzen ist die genaue Definition der Ziele wichtig. Ohne zu wissen, welche Aufgaben ein neuronales Netz erfüllen soll, ist eine Programmierung eines solchen oft reine Spielerei. Ein neuronales Netz kann zudem keine Information erzeugen, die nicht schon in versteckter Form in den Eingabedaten vorhanden ist. Das gilt natürlich für jede Modellierung eines Systems. Gerade bei der Anwendung neuronaler Netze wird dieser Aspekt aber gerne übersehen.

2.4 Ressourcen

Neuronale Netze sind meist leicht zu programmieren und können auf allen heute gängigen Rechnern laufen. Die Anforderungen an die *Hardware* sind in der Regel durch die Komplexität des Problems gegeben. Einschränkungen bestehen nur dann, wenn man sich von vornherein auf spezielle *Software* festlegt, die ihrerseits spezielle Anforderungen an die Hardware stellt. Eine solche Festlegung ist aber meist nicht notwendig und sollte nur dann vorgenommen werden, wenn mit dem Produkt schon positive Erfahrungen gemacht wurden. Ein wichtiger Punkt ist aber die Stabilität der Installation: Gerade bei langfristigen Projekten ist darauf zu achten, daß Hardware und Software hinreichend stabil sind und die Software leicht auf neue Hardware portierbar ist.

Die Programmierung eines neuronalen Netzes kann entweder direkt in einer Programmiersprache geschehen, für die dann ein Compiler vorhanden sein muß, oder sie kann mit Hilfe eines Programms vorgenommen werden, das speziell für die Simulation von neuronalen Netzen

geschrieben wurde. Solche *neural network simulators* sind in großer Zahl auf dem Markt erhältlich. Beide Methoden haben unterschiedliche Vor- und Nachteile. Ein spezielles Simulationsprogramm erfordert in der Regel keine oder nur geringe Programmierkenntnisse, es ist oft mit einer intuitiv zu bedienenden, graphischen Oberfläche ausgestattet und enthält schon eine große Vielzahl von möglichen Optionen. Es erlaubt eine rasche Implementierung eines neuronalen Netzes, und die Implementierung ist in der Regel flexibel. Änderungen an dem Netz sind leicht durchzuführen. Ein Nachteil ist, daß diese Programme unter Umständen deutlich mehr Speicherplatz und Rechenzeit benötigen, als ein direkt programmiertes neuronales Netz. Auch bei der direkten Programmierung kann man häufig auf fertige Bibliotheken zurückgreifen. Ein oft nützliches Vorgehen besteht darin, das Netz zunächst mit einem Simulationsprogramm zu entwickeln und zu testen und anschließend in einer gängigen Programmiersprache ein entsprechendes Programm zu schreiben.

Arbeitet man mit einem fertigen Simulationsprogramm, so sind vor der Anschaffung folgende Punkte zu klären:

- *Welches know how wird beim Benutzer vorausgesetzt?* Viele Programme sind sehr komplex und setzen eine genaue Kenntnis von neuronalen Netzen voraus, die in der Regel nur Spezialisten haben.
- *Welche Komplexität erlaubt das Programm?* Oft sind besondere Eigenschaften für ein neuronales Netz wichtig, die in einfachen Programmen nicht zur Verfügung stehen. Andererseits ist ein komplexes Programm oft auch schwieriger zu bedienen.
- *Gibt es eine gute Dokumentation für das Programm und ist eine schnelle Einarbeitung möglich?* Dieser Punkt wird häufig unterschätzt. Ein Programm kann nur dann sinnvoll benutzt werden, wenn es ausreichend dokumentiert ist. Ist dies nicht der Fall, so wird viel Zeit mit unnötigen Arbeiten verbraucht, was zusätzliche Kosten verursacht.
- *Was kostet das Programm?* Simulationsprogramme für neuronale Netze können sehr teuer sein, es gibt aber auch sehr gute, freie Alternativen. Einen Überblick kann man sich in den Neural Network FAQs <ftp://ftp.sas.com/pub/neural/FAQ.html> (Abschnitte 5 und 6) verschaffen.
- *Ist das Programm mit vorhandener Software kompatibel?* Die Eingabedaten für das Programm werden in der Regel von vorhandener Software oder Hardware geliefert. Die Ausgabedaten werden vielfach mit vorhandener Software weiterverarbeitet. Beide Möglichkeiten müssen sichergestellt sein.
- *Ermöglicht das Programm **unsupervised learning**?* Bei Problemstellungen, die kompliziert sind und lange Lernphasen erfordern, ist es günstig, wenn das Programm in der Lage ist, während des Lernens zum Beispiel Optimierungen des Netzes selbständig vorzunehmen. Diese Möglichkeit ist wichtig, da sonst für das Lernen ein hoher Personalaufwand und damit hohe Kosten entstehen
- *Wie alt ist das Programm, wie lange wurde es entwickelt, liegen Erfahrungsberichte vor?* Sehr neue Programme sind unter Umständen noch nicht vollständig getestet und damit fehleranfällig. In zu alten Programmen sind moderne Entwicklungen eventuell nicht berücksichtigt. Besonders positive oder negative Erfahrungsberichte können wertvolle Hinweise über die Einsatzmöglichkeiten von Simulationsprogrammen in der Praxis liefern.

2.5 Personal

Ein Projekt, in dem mit einem neuronalen Netz gearbeitet werden soll, bedarf einer Reihe spezieller Qualifikationen. In der Regel kann ein solches Projekt nicht ohne einen Spezialisten für neuronale Netze durchgeführt werden.

Der Projektmanager sollte immer darauf achten, daß Zwischenziele im vorgegebenen Zeitrahmen erreicht werden und veranschlagte Kosten nicht überstiegen werden. Dazu muß er eine gewisse Erfahrung im Umgang mit neuronalen Netzen haben, um Probleme frühzeitig zu erkennen. Neuronale Netze sind extrem flexibel und daher für sehr unterschiedliche Projekte einsetzbar. Man kann insbesondere auch noch dann brauchbare Resultate erhalten, wenn andere Methoden nicht anwendbar sind. Andererseits ist die Verwendung von neuronalen Netzen in vielen Bereichen ein recht neues Verfahren. Ein Erfolg ist also nicht garantiert. Es besteht immer die Gefahr, daß die vorhandenen Daten für die Simulation nicht ausreichen und das Projekt damit kein Resultat liefert. Ein mit neuronalen Netzen erfahrener Projektmanager kann die Effektivität des Verfahrens beurteilen, Schwierigkeiten rechtzeitig erkennen und das Projekt gegebenenfalls stoppen.

2.6 Projektkosten

Die Projektkosten setzen sich wie üblich aus Personalkosten, Kosten für Hard- und Software, Kosten für die Datenerhebung und sonstigen Projektkosten zusammen. Ein wichtiger Aspekt bei neuronalen Netzen ist, das sie oft lange Rechenzeiten benötigen. Die Kosten für die Rechenzeit sind daher in der Regel recht hoch.

2.7 Dokumentation

Wichtig ist, daß jeder Schritt ausführlich dokumentiert wird. Da die Entwicklung eines neuronalen Netzes oft experimentell verläuft, spielt die Dokumentation eine wichtige Rolle. Nur in einem gut dokumentierten Projekt kann man schnell eine gute Implementierung eines neuronalen Netzes erreichen. Die Dokumentation ist vergleichbar mit dem Laborjournal in einem experimentell arbeitenden Labor.

Die Dokumentation soll zudem die Einarbeitung neuer Mitarbeiter erleichtern. Insbesondere bei lang laufenden Projekten, oder bei Projekten, an denen unterschiedliche Firmen oder Institute beteiligt sind, ist es günstig, die Dokumentation in einem nicht-proprietären Standard-Format abzulegen. Beispielsweise bietet es sich an, XML <http://www.xml.org> für die Dokumentation zu verwenden und für jeden einzelnen Schritt eine Dokumenttyp-Definition (DTD) festzulegen. Diese Vorgehensweise erlaubt es zum einen, Standards für die Dokumentation festzulegen und einzuhalten, zum anderen, die Dokumentation in unterschiedlichen Formaten zur Verfügung zu stellen.

Eine besonders wichtige Rolle spielt die Dokumentation für die spätere Kontrolle des neuronalen Netzes im laufenden Betrieb.

3 Daten

Datenerhebung, Datenkontrolle und Datenkodierung sind wichtige Aspekte bei der Simulation mit neuronalen Netzen. Insbesondere muß die Quantität und die Qualität der Daten überprüft werden.

3.1 Einführung

Die typische Aufgabe eines neuronalen Netzes besteht darin, aus einer Reihe von Eingabedaten gesuchte Ausgabedaten zu produzieren. Neuronale Netze können sehr verschiedene Formen von Daten verarbeiten, sofern diese in einer geeigneten Form vorliegen. Das bedeutet, daß die Daten in der Regel kodiert werden müssen. Die Kodierung wird hauptsächlich von der Problemstellung bestimmt. Die Problemstellung gibt vor, in welcher Form die Eingabedaten vorliegen können und welche Ausgabedaten gesucht sind. Sofern die Daten nicht schon mit der gegebenen Fragestellung vorliegen, ist es sinnvoll, Kodierungsschemata vor der Datenbeschaffung zu entwerfen. Erst das Kodierungsschema legt fest, welche Eingabedaten wirklich benötigt werden. Oft ist gerade die Datenbeschaffung kostenintensiv, so daß es sinnvoll ist, in der Kodierung nicht benötigte Daten auch nicht zu erheben. Ein typisches Beispiel für eine Kodierung von Daten ist die Reskalierung, kompliziertere Beispiele sind die Fouriertransformation einer Zeitreihe oder die Kodierung mehrerer Rohdaten in einer Variable. In dem später beschriebenen Beispiel (siehe Abschnitt 8 (Anwendung)) werden wir eine solche kompliziertere Kodierung verwenden.

Ein zweiter Aspekt betrifft *Qualität* und *Quantität* der Daten. Liegen genügend Daten vor, um das neuronale Netz zu trainieren? Liegen genügend Daten vor, um die Fragestellung mit einem neuronalen Netz beantworten zu können? Mit welchen Fehlern sind die Rohdaten behaftet? Diese Fragen müssen geklärt werden, bevor langwierige Trainingszyklen mit einem neuronalen Netz vorgenommen werden.

In diesem Abschnitt wird diskutiert, wie die Kodierung vorgenommen werden kann, welche Bedingungen dabei zu beachten sind, welche Auswirkungen das auf die Beschaffung der Daten hat und wie am Ende die Ausgabedaten rekodiert werden. Es wird ferner diskutiert, wie man Qualität und Quantität der Daten beurteilen kann.

3.2 Datenkodierung

Rohdaten können aus einer kontinuierlichen Datenmenge stammen oder aus einer diskreten Datenmenge. Sie können zudem in bestimmten Einheiten vorliegen und unterschiedlich skaliert sein. Ein erster Schritt besteht darin, für die Daten einfache statistische Größen wie den Mittelwert und die Standardabweichung zu bestimmen, bei diskreten Daten die Häufigkeitsverteilung. Oft ist es sinnvoll, die Rohdaten zu reskalieren, was mit Hilfe dieser Informationen möglich ist. Zum anderen können diese Informationen dazu dienen, statistische Ausreißer in den Rohdaten für Trainingszwecke zu eliminieren. Kommt es bei der Fragestellung dagegen gerade auf die statistischen Ausreißer an, so sind diese in geeigneter Weise (oft auch wieder durch Umskalierung) hervorzuheben. Das ist z.B. möglich, indem man nichtlineare Skalenfunktionen benutzt oder indem man unterschiedliche Bereiche in unterschiedlicher Weise skaliert. Fertige

Programmpakete zur Simulation neuronaler Netze bieten verschiedene Skalenfunktionen an, die sich in der Praxis mehr oder weniger bewährt haben.

Das neuronale Netz lernt eine Funktion, die die Eingabedaten auf die Ausgabedaten abbildet. Das Lernverhalten des Netzes hängt entscheidend von der Krümmung dieser Funktion ab: Führen kleine Änderungen in den Eingabedaten in einem bestimmten Bereich zu großen Änderungen in den Ausgabedaten, dann wird dieser Bereich eventuell schlechter gelernt und man benötigt dort eine höhere Dichte von Datensätzen. Eine geeignete nicht-lineare Reskalierung der Daten kann das Verhalten des Netzes in solchen Bereichen verbessern.

Das neuronale Netz liefert Ausgabedaten in der Form, wie sie durch die Kodierung der Trainings- und Testdaten vorgegeben wurde. Beispielsweise kann das Ausgabedatum ein relativer Zahlenwert zwischen 0 und 1 sein, der anschließend reskaliert wird. Es ist aber auch möglich, die Kodierung so vorzunehmen, daß das Ausgabedatum direkt verwendet werden kann.

3.3 Qualität und Quantität

Die Qualität der Daten hängt in entscheidender Weise davon ab, wie die Daten erhoben wurden. Zum einen spielen systematische und statistische Fehler eine Rolle, zum anderen können Daten zum Teil redundant sein. Beide Probleme können durch eine sorgfältige Planung vor der Erhebung der Daten reduziert werden. Oft lassen sich diese Schwierigkeiten aber nicht ganz vermeiden. Statistische Test können hilfreich sein, wenn die Qualität der Daten sichergestellt sein soll.

Eine andere Frage ist, wieviele Daten für eine Untersuchung erhoben werden müssen. Je komplexer ein neuronales Netz ist, desto mehr Daten werden für das Training benötigt. Oft ist es günstig, ein Netz mit einem oder wenigen Ausgabedaten zu benutzen. Es kann auch sinnvoll sein, die Anzahl der Eingabedaten zu reduzieren, um so die Komplexität des Netzes zu begrenzen. Beides kann sinnvoll sein, da die Zahl der benötigten Daten und damit der Aufwand bei der Erhebung der Daten exponentiell mit der Komplexität des Netzes wächst.

In diesem Zusammenhang ist es wichtig, sich klar zu machen, daß man keine neue Information mit Hilfe eines neuronalen Netzes erzeugen kann. Die gesuchte Information muß schon (in einer nicht erkennbaren Weise) in den Eingabedaten enthalten sein. Es ist also beispielsweise nicht möglich, aus wenigen Eingabedaten mehr unabhängige Ausgabedaten zu erzeugen. Ebenso ist es nicht möglich, mit einer geringen Anzahl von Datensätzen ein neuronales Netz zu trainieren.

4 Netzwerk

Der Entwurf eines neuronalen Netzes hat häufig experimentellen Charakter. Das Netzwerk darf weder zu komplex sein, da dann zu viele Parameter aus den Daten zu bestimmen sind und das Netz die Fähigkeit verliert zu verallgemeinern, noch zu wenig komplex, da dann die Korrelationen der Daten durch das Netzwerk nicht erfaßt werden können. In diesem Abschnitt werden einige wichtige Punkte diskutiert, die für den Entwurf eines neuronalen Netzes wichtig sind.

4.1 Einführung

Bis die optimale Struktur eines neuronalen Netzes gefunden ist, muß man in den meisten Fällen eine Reihe von Vorentwürfen ausprobieren und schrittweise verbessern. Es ist zwar häufig möglich, das Verhalten eines gängigen neuronalen Netzes wie etwa eines Mehr-Schicht Perceptron abzuschätzen, Details müssen aber in der Regel variiert werden. Trotzdem ist damit eine gewisse Planung eines neuronalen Netzes möglich.

Wir werden sehen, daß insbesondere die korrekte Komplexität eines Netzes schon bei der Planung zu berücksichtigen ist.

4.2 Entwurf eines Mehr-Schicht Perceptrons

Das Mehr-Schicht Perceptron stellt das am meisten verwendete neuronale Netz dar. Der Entwurf eines solchen Netzes soll hier exemplarisch dargestellt werden, viele Punkte sind für andere neuronale Netze in ähnlicher Weise gültig.

Ein Mehr-Schicht Perceptron besteht aus einer Eingabeschicht, einer oder mehrerer Zwischenschichten und einer Ausgabeschicht (siehe Abbildung). Es dient meist der Vorhersage von Daten: Aus den in der Eingabeschicht eingegebenen Daten werden die Daten der Ausgabeschicht vorhergesagt. Die Verarbeitung der Daten geschieht in den Zwischenschichten. Ein wesentlicher Parameter des Netzes ist die Anzahl der Neuronen. Sie bestimmt im wesentlichen die Komplexität des Netzes. Die Anzahl der Neuronen der Ein- und Ausgabeschicht ist durch die Aufgabe des Netzes und durch die Kodierung der Daten vorgegeben. Variiert werden kann die Zahl der Zwischenschichten und die Zahl der Neuronen in den Zwischenschichten.

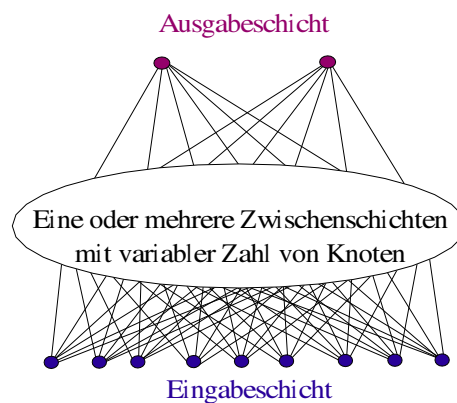


Abbildung 2: Ein neuronales Netz

4.3 Zwischenschichten

Die Zwischenschichten determinieren die Eigenschaften des Netzes, insbesondere die Fähigkeit zur Verallgemeinerung. Die Anzahl der Neuronen in den Zwischenschichten muß groß genug sein, um die gestellte Aufgabe zu erfüllen, sie muß klein genug sein, um eine sinnvolle

Verallgemeinerung durch das Netz zu ermöglichen. Außerdem bestimmt die Anzahl der Neuronen in den Zwischenschichten die Anzahl der Verknüpfungen innerhalb des Netzes und damit die Komplexität des Netzes. In einem Netz mit einer Zwischenschicht ist die Anzahl der Verknüpfungen proportional zu der Anzahl der Neuronen in der Zwischenschicht. Für jede Verknüpfung gibt es ein Gewicht. Man benötigt eine ausreichende Zahl von Daten für das Training des Netzes, um die Gewichte hinreichend genau bestimmen zu können. Diese Bedingungen wirken unterschiedlich.

In der Praxis besteht oft ein weiteres Problem: Die Eingabedaten enthalten unter Umständen redundante Information, die auch nicht durch eine geeignete Kodierung aufgehoben werden kann. In diesem Fall sollte das Netz die redundante Information herausprojizieren können, also eine Kompression der Eingabedaten leisten können. Dazu muß die Zwischenschicht weniger Neuronen haben, als die Eingabeschicht.

Es wurden vielfach Vorschläge zu Berechnung der Größe der Zwischenschichten gemacht, diese widersprechen sich aber zum Teil und sind praktisch von geringer Bedeutung. Wie schon oben erwähnt, besteht die Entwicklung eines neuronalen Netzes zum großen Teil aus experimenteller Arbeit. Häufig startet man mit einer bestimmten Anzahl von Neuronen in einer Zwischenschicht und variiert diese solange, bis das Netz die gewünschten Eigenschaften hat. Dazu erzeugt man sich geeignete Mengen von Trainings- und Testdaten, trainiert das Netz und testet die Verallgemeinerungsfähigkeiten mit den Testdaten. Je nach Ergebnis kann man dann die Zahl der Neuronen variieren. Folgende heuristischen Regeln haben sich bewährt:

1. Wenn der Fehler im Trainingszyklus klein ist, der Fehler beim Testen dagegen groß, verallgemeinert das Netz nicht. In der Regel bedeutet das, daß die Zahl der Gewichte und damit die Zahl der Neuronen in der Zwischenschicht zu groß ist.
2. Wenn der Fehler im Trainingszyklus groß ist, hat das Netz eventuell zu wenig Neuronen in der Zwischenschicht.
3. Wenn alle Gewichte von etwa der gleichen Größenordnung sind, dann hat das Netz meist zu wenig Neuronen.
4. Man kann immer Neuronen hinzufügen, die Fehler können aber auch andere Ursachen haben! Beispielsweise kann man nicht erwarten, daß ein Netz mit einem geringen Fehler vorhersagt, wenn die Rohdaten selbst mit einem höheren statistischen oder systematischen Fehler behaftet sind.

4.4 Lernalgorithmen

Es gibt unterschiedliche Lernalgorithmen, mit denen ein neuronales Netz trainiert werden kann. Für ein Mehr-Schicht-Perceptron, wie wir es hier diskutieren, stehen Standard-Algorithmen zur Verfügung. Der einfachste Lernalgorithmus ist ein Gradientenverfahren. Dabei wird versucht, den Fehler der Ausgabedaten zu minimieren, indem man untersucht, in welche Richtung sich die Kopplungen zwischen den Neuronen ändern müssen, damit der Fehler abnimmt. Das einfache Gradientenverfahren wird heute eigentlich nicht mehr benutzt. Populär und in den meisten Fällen effizient sind weiterentwickelte Verfahren, die unter dem Stichwort *error back-propagation* bekannt sind. Welcher Lernalgorithmus benutzt wird, hängt häufig von den Anforderungen ab, die man hat. Benutzt man fertige Softwarepakete, so stellen diese in der Regel eine Zahl von bewährten Lernalgorithmen zur Verfügung und geben in der Dokumentation an, welche Eigenschaften die unterschiedlichen Algorithmen haben.

5 Fehler

In diesem Abschnitt erfahren Sie, warum bei einer Simulation mit einem neuronalen Netz Fehler auftreten können und wie diese zu behandeln sind.

5.1 Grundlegende Bemerkungen

Hat man mit Hilfe eines neuronalen Netzes Ausgabedaten erhalten, stellt sich die Frage, mit welchem Fehler diese Daten behaftet sind. Es ist nie möglich, mit Hilfe eines neuronalen Netzes eine fehlerlose Vorhersage zu machen. Deshalb ist es besonders wichtig, Fehler abzuschätzen.

Mögliche Fehlerquellen stecken zum einen in der eingeschränkten Vorhersagefähigkeit des Netzes, zum anderen in dem immer vorhandenen Fehler der Eingabedaten. Das bedeutet, daß für eine Fehlerabschätzung unterschiedliche Fehlerquellen zu berücksichtigen sind.

Oft hört oder liest man, daß ein neuronales Netz im Prinzip eine beliebige Präzession erlaubt. Das ist richtig, wenn beliebig viele Trainingsdaten beliebig hoher Genauigkeit zur Verfügung stehen und wenn man beliebig komplexe Netze simulieren kann. Ersteres ist der Fall, wenn die zu lernende Funktion schon bekannt ist. Für praktische Anwendungen ist dies aber nie der Fall.

5.2 Fehlerquellen

Systematische Fehler der Eingabedaten müssen bei der Datenaufnahme soweit möglich vermieden werden. In Fällen, wo das nicht möglich ist, kann ein bekannter systematischer Fehler eventuell bei der Kodierung eliminiert werden.

Statistische Fehler der Eingabedaten lassen sich mit geeigneten statistischen Methoden meist abschätzen. In vielen Fällen spielen die statistischen Fehler der Eingabedaten nur eine untergeordnete Rolle. In Fällen, wo dies nicht der Fall ist, ist zu klären, wie sich die Fehler in dem neuronalen Netz fortpflanzen und welchen Einfluß sie auf die Ausgabedaten haben. Statistische Fehler der Eingabedaten sind aber im strengen Sinn kein Fehler des neuronalen Netzes.

Fehler des neuronalen Netzes selbst können verschiedene Ursachen haben. Zum einen ist es möglich, daß die Daten, mit denen das Netz trainiert wurde, nicht die für die Aufgabe nötige Information enthalten haben. Zum zweiten kann es sein, daß das Netz nicht das für die Aufgabe nötige Design aufweist, also beispielsweise zu einfach konstruiert ist und damit die gestellte Aufgabe nur unvollständig erfüllen kann. Zum dritten besteht die Möglichkeit, daß die Eingabedaten in einem Parameterbereich liegen, in dem das Netz nicht trainiert worden ist. In diesen drei Fällen kann man die Fehler durch eine geeignete Modifikation des Netzes oder durch eine bessere Auswahl der Trainingsdaten vermeiden. Wenn ein Netz trotz solcher Modifikationen große Fehler macht, läßt sich das gestellte Problem mit Hilfe eines neuronalen Netzes eventuell nicht lösen.

Wenn man davon ausgeht, daß das Netz richtig modelliert wurde und die Trainingsdaten die nötige Information enthalten, bleibt schließlich die Abschätzung der statistischen Fehler des Netzes.

5.3 Fehler bei Klassifizierungen

Eine typische Aufgabe eines neuronalen Netzes ist die Klassifizierung von Datensätzen. Die Ausgabedaten sind in diesem Fall diskret. Ein typisches Beispiel einer Klassifizierung hat man in der Fertigungskontrolle: Ein Produkt ist entweder in Ordnung oder mangelhaft. Die Klassifizierung der Produkte in diese zwei Klassen kann in unter Umständen mit einem neuronalen Netz automatisiert werden.

Um diskrete Ausgabedaten zu erhalten, wird in der Regel ein Schwellenwert eingeführt. In dem einfachen Beispiel der Fertigungskontrolle gibt es zwei Klassen, die mit 1 (Produkt ist in Ordnung) und 0 (Produkt ist mangelhaft) bezeichnet werden können, und in die kontinuierlich verteilte Daten eingeordnet werden. Die Schwelle wird in diesem Fall zwischen 0 und 1 liegen. Je nach der Struktur der Daten kann man den Fehler in solchen Fällen durch eine Variation der Schwelle beeinflussen. In welchem Bereich die Schwelle variiert werden kann, hängt dabei von der konkreten Problemstellung ab. Die statistischen Aussagen, die mit einem solchen Netz möglich sind, sind von dem Typ, daß ein Datensatz mit einer bestimmten Wahrscheinlichkeit zu einer bestimmten Klasse gehört.

Im Fall von Klassifizierungen geben die Ausgabedaten schon selbst einen Hinweis darauf, wie gut die Klassifizierung gelungen ist. Typischerweise wird die Klassifizierung umso ungenauer, je näher man der Schwelle kommt. Einen guten Hinweis auf Fehler bei Klassifizierungen bekommt man, wenn man um die Schwelle eine Ausschlußregion fester Breite festlegt und bei einer Variation der Schwelle und der Breite feststellt, wieviele Datensätze in diese Ausschlußregion fallen.

5.4 Fehler bei kontinuierlichen Ausgabedaten

Für kontinuierliche Ausgabedaten können absolute Fehler oder relative Fehler angegeben werden. Typischerweise wird in diesen Fällen der Fehler auch davon abhängen, in welchem Parameterbereich man sich befindet. Man kann die Abbildung der Eingabedaten auf die kontinuierlichen Ausgabedaten als eine nichtlineare Funktion auffassen, die das Netz lernen soll. Hat man homogen verteilte Trainingsdaten, so wird diese Funktion in Regionen geringer Krümmung häufig leichter gelernt als in Regionen starker Krümmung. Das führt dazu, daß der Fehler in Regionen starker Krümmung größer wird. Stark inhomogene Verteilungen der Fehler können also auf solche Regionen hindeuten, sie können ihre Ursache aber auch in der Verteilung der Trainingsdaten haben.

5.5 Abschätzung der Fehler

Die Fehlerabschätzung für ein neuronales Netz basiert in der Regel auf Testläufen. Die Menge vollständiger Datensätze wird in Trainings- und Testdaten zerlegt, das Netz wird mit den Trainingsdaten trainiert und anschließend mit den Testdaten getestet. Die dabei auftretenden Fehler sind die Grundlage für die Fehlerabschätzung. Da in der Regel nur endlich viele vollständige Datensätze vorliegen, besteht hier ein Dilemma: Auf der einen Seite benötigt man viele Trainingsdaten, um zu gewährleisten, daß das Netz hinreichend gut trainiert ist, zum anderen benötigt man hinreichend viele Testdaten, um die Fehler abschätzen zu können. Wie die Aufteilung in Test- und Trainingsdaten vorgenommen wird, hängt von den Gegebenheiten ab und variiert von Problemstellung zu Problemstellung.

Eine weitere Möglichkeit der Fehlerabschätzung, die besonders bei der Berücksichtigung von statistischen Fehlern der Eingabedaten eine Rolle spielt, ist die Fehlerfortpflanzung. Kennt man die Parameter eines (trainierten) Netzes, dann kann die Fortpflanzung des Fehlers der Eingabedaten abgeschätzt werden. Diese Abschätzungen erlauben eventuell, Regionen starker Krümmung zu identifizieren.

Eine weitere Möglichkeit, Fehler abzuschätzen, besteht darin, die Datensätze selbst mittels eines Zufallszahlengenerators mit einem bekannten Fehler zu versehen und die Reaktion des Netzes auf die so verrauschten Daten zu untersuchen.

5.6 Bedeutung und Interpretation von Fehlern

Aus den (sauber abgeschätzten) Fehlern kann man häufig etwas über das Verhalten des Netzes lernen. Beispielsweise haben wir oben schon gesehen, daß eine inhomogene Verteilung der Fehler auf starke Krümmungen der zu lernenden Funktion hinweisen kann. Unter Umständen besteht dann die Möglichkeit, dem Netz in diesen Parameterbereichen weitere Trainingsdaten zur Verfügung zu stellen. Man kann auf diese Weise auch Regionen im Parameterraum identifizieren, die von dem Netz schlecht gelernt werden, um damit dann das Design des Netzes geeignet anzupassen.

Ein grundsätzliches Problem, das bei der Fehlerabschätzung auftritt, besteht darin, daß in vielen Fällen die Verknüpfungen innerhalb des Netzes nicht eindeutig sind. Eine typische Lernstrategie, bei der der Fehler der Testdaten minimiert wird, liefert bei unterschiedlichen Trainingsläufen oft unterschiedliche Konfigurationen, die einen vergleichbar kleinen Fehler aufweisen. Dies tritt insbesondere dann auf, wenn die Eingabedaten redundante Informationen enthalten. Oft kann dieses Problem aber nicht behoben werden. Damit wird die Fehlerabschätzung zusätzlich erschwert. Eine Möglichkeit besteht dann darin, mit einem Ensemble von Netzen zu arbeiten, die mit Gewichten versehen sind, die ihrerseits vom Fehler der Testläufe abhängen. Gerade bei medizinischen Anwendungen, bei denen zum einen oft nicht sehr viele Datensätze zur Verfügung stehen, zum anderen die Daten selbst größere statistische Fehler aufweisen, hat sich dieses Verfahren bewährt. Wir kommen auf diese Variante später zurück.

5.7 Spezialfälle

Es gibt spezielle Aufgabenstellungen und damit spezielle Varianten von Mehrschicht-Perceptronen, bei denen dem Fehler eine besondere Bedeutung zukommt. Ein Beispiel ist die Vorhersage der Eingabedaten aus den Eingabedaten. Zunächst scheint diese Aufgabe sinnlos zu sein. Führt man aber eine oder mehrere Zwischenschichten mit weniger Neuronen als Ein- und Ausgabedaten ein, so wird diese Aufgabe nichttrivial. Der Sinn einer solchen Anwendung kann darin bestehen, die Datensätze zu einer Klasse zusammenzufassen und zu erkennen, ob neue Datensätze in diese Klasse passen. In der kleineren Zwischenschicht liegen die Eingabedaten in komprimierter Form vor. Die Voraussetzung, die hier gemacht wird, besteht also darin, daß alle Datensätze die gleiche Informationsdichte haben und in ähnlicher Weise komprimiert werden können. Tritt bei einem neuen Datensatz ein großer Fehler in den Ausgabedaten auf, stimmen diese also nicht mehr mit den Eingabedaten überein, so hat man ein Indiz dafür, daß dieser neue Datensatz nicht zu der Klasse gehört.

6 Regeln

Aus den Parametern des neuronalen Netzwerks läßt sich sein Verhalten erklären und Regeln für das Verhalten ableiten.

6.1 Einführung

Eine grundsätzliche Kritik gegen die Anwendung neuronaler Netze ist, daß ein neuronales Netz häufig als *black box* angesehen wird, deren Funktionsweise unverstanden bleibt. Das ist aber nicht grundsätzlich so. Es gibt unterschiedliche Methoden, das Verhalten des neuronalen Netzes zu verstehen und eventuell Regeln abzuleiten. Das Hauptproblem besteht darin, daß das neuronale Netz die Ausgabedaten als eine explizit nicht bekannte, nicht lineare Funktion der Eingabedaten liefert. Daher ist kein einfacher, vor allem aber kein linearer Zusammenhang zwischen den Eingabedaten und den Ausgabedaten ableitbar.

Eine andere Frage ist, wozu überhaupt Regeln abgeleitet werden sollen. Wenn ein neuronales Netz funktioniert, ist es dann nicht gleichgültig, warum es auf diese Weise funktioniert? Es gibt eine Reihe von Algorithmen, mit Hilfe derer aus einem neuronalen Netz Regeln extrahiert werden können. Je nachdem, welche Ansprüche man an die Güte der Regeln stellt, kann die Aufstellung dieser Regeln kostenintensiv sein. Gute Gründe, Regeln zu suchen, sind

- Besseres Verständnis des Netzes durch die Benutzer.
- Verbesserung der Verallgemeinerungsfähigkeit des Netzes durch Kenntnis von Regeln.
- Vergleich der Regeln mit bekannten Gesetzmäßigkeiten als zusätzlicher Test des Netzes.
- Weitere Verwendung der Regeln in anderen Bereichen des Projekts.

Information ist in einem neuronalen Netz in der Netzwerkarchitektur und in den Parametern enthalten, die das Netz beschreiben. Das Aufstellen von Regeln bedeutet in jedem Fall eine Interpretation der so gespeicherten Information.

6.2 Methoden zum Finden von Regeln

Es gibt unterschiedliche Methoden, die es erlauben, aus den Parametern des neuronalen Netzes Regeln zu extrahieren. Man kann sie grob in analytische Methoden und synthetische Methoden einteilen.

Analytische Methoden versuchen, das neuronale Netz durch die Analyse der Gewichte der einzelnen Kopplungen in kleinere Untereinheiten zu zerlegen. Danach wird versucht, das Verhalten in diesen Untereinheiten zu erfassen, indem man versucht, festzustellen, welche Mengen von Eingabedaten ein bestimmtes Verhalten des Netzes erzeugen. Das kann entweder durch numerische Tests oder durch die direkte Analyse der Kopplungen geschehen. Anschließend kann man zum Beispiel versuchen, die Struktur des Netzes mit Hilfe der so gewonnenen Kenntnisse zu optimieren. Aus dem optimierten Netz kann man dann wieder Regeln ableiten. Die Regeln dienen also zur Verbesserung des Netzes, und lassen sich so verfeinern.

Synthetische Methoden betrachten das Netz als eine *black box*, die die Ausgabedaten als Funktion der Eingabedaten liefert. Diese Funktion kann zum Beispiel durch ihre lokalen Ableitungen

charakterisiert werden. Die Matrix dieser Ableitungen enthält Informationen über das lokale Verhalten des Netzes. Die Ableitungen lassen sich analytisch aus den Parametern des Netzes herleiten oder, indem man numerisch untersucht, wie die Ausgabedaten auf kleine Variationen der Eingabedaten reagieren. Ableitungen lassen sich natürlich nur bei kontinuierlichen Ausgabedaten benutzen. Bei neuronalen Netzen, die eine Kategorisierung durchführen, läßt sich aber auch das Verhalten des Netzes auf kleine Änderungen der Eingabedaten untersuchen. Aus dem so untersuchten Verhalten des Netzes lassen sich eventuell auch Rückschlüsse auf mögliche Optimierungen des Netzes ziehen. Stellt sich beispielsweise heraus, daß die Ableitung der Ausgabedaten nach einem Eingabewert sehr klein sind, dann ist dieser Eingabewert in diesem Problem weniger wichtig und kann eventuell weggelassen werden. Das ist natürlich nur dann richtig, wenn die Ableitungen global klein sind.

6.3 Beurteilung von Regeln

Es gibt unterschiedliche, und sich zum Teil widersprechende Kriterien, nach denen Regeln beurteilt werden können, die aus einem neuronalen Netz gewonnen wurden. Folgende vier Kriterien sind wichtig:

Qualität.

Die abgeleiteten Regeln sollen natürlich 'stimmen'. Das nachzuprüfen, ist aber häufig nicht einfach. Die Regeln müssen widerspruchsfrei sein. Ferner ist die Genauigkeit zu überprüfen, mit der eine Regel gilt. Und schließlich ist wichtig zu wissen, welche Aussagen die Regel über Datensätze macht, die nicht in dem Bereich liegen, in dem das Netz trainiert wurde.

Komplexität.

Die Regeln müssen später angewandt werden können. Es macht keinen Sinn, ein komplexes Regelwerk etwa im Rahmen eines mengentheoretischen Kalküls zu entwerfen, dessen algorithmische Umsetzung und Anwendung mehr Rechenkapazität erfordert als das neuronale Netz selbst.

Verständlichkeit.

Eng mit der Komplexität ist die Verständlichkeit der Regeln verknüpft. Wenn die Regeln von einem späteren Nutzer des Netzes verwendet werden sollen, etwa um die Konsistenz von Ergebnissen zu überprüfen, muß er die Regeln auch verstehen können. Die Verständlichkeit der Regeln hängt davon ab, wie sie formuliert werden, und natürlich davon, wer sie verstehen soll. Wenn es darum geht, die Regeln maschinell zu verarbeiten, wird man eventuell eine formale Sprache wählen, die sie für einen Menschen weniger verständlich macht.

Anwendbarkeit.

Regeln können global oder nur auf Teile des möglichen Parameterbereichs der Eingabedaten anwendbar sein. Sie können das Verhalten für alle Ein- oder Ausgabedaten erklären oder nur für einen Teil dieser Daten.

Es ist klar, daß sich diese Kriterien widersprechen können. Eine höhere Qualität der Regeln kann eine höhere oder unerwünschte Komplexität zur Folge haben, die die Regeln dann unverständlicher macht. Daher ist es wichtig, vorher klarzustellen, welches Ziel man mit den Regeln hat, und danach, welche Kriterien die Regeln erfüllen müssen, um das Ziel zu erreichen.

7 Optimierung und Kontrolle

Neuronale Netze können (und müssen) auch im laufenden Betrieb kontrolliert und optimiert werden. Dazu gibt es unterschiedliche Möglichkeiten. Speziell die mit Hilfe des Netzes entwickelten Regeln (siehe Abschnitt 6 (Regeln)) können für Kontrollzwecke genutzt werden.

7.1 Einführung

Ein funktionierendes neuronales Netz kann unter unterschiedlichen Aspekten optimiert und kontrolliert werden:

Kosten:

Der Betrieb eines neuronalen Netzes verursacht Kosten. Dabei ist nicht nur an die Kosten für die Rechenanlage und das bedienende Personal zu denken, sondern auch an die Kosten für die Erhebung der Eingabedaten. In medizinischen Anwendungen kann es zum Beispiel sehr teuer sein, Eingabedaten zu bestimmen, da dabei Laborkosten anfallen. Wir haben gesehen, daß die Bestimmung von Ableitungen der Ausgabedaten nach einzelnen Eingabedaten benutzt werden können, um einzelne Eingabedaten zu eliminieren und so Kosten zu senken.

Qualität:

Ein neuronales Netz liefert die Ausgabedaten immer nur als Schätzwerte, die mit einem Fehler behaftet sind. Eine Optimierung des Netzes kann mit dem Ziel erfolgen, die Schätzwerte zu verbessern und die Fehler zu reduzieren. Dazu lassen sich unter Umständen die Regeln benutzen, die mit Hilfe des Netzes gewonnen wurden.

Geschwindigkeit:

Die Berechnungen mit einem neuronalen Netz benötigen zum Teil nicht unerhebliche Rechenzeit. Bei Problemstellungen, wo es auf die Rechenzeit ankommt, kann man versuchen, durch verschiedene Maßnahmen die Rechengeschwindigkeit zu erhöhen. Möglichkeiten sind zum Beispiel die Umstellung des Netzes von einem fertigen Programmpaket (*neural network simulator*) auf ein in einer höheren Programmiersprache programmiertes Netz oder die Anschaffung schnellerer Hardware.

Auch hier können sich die unterschiedlichen Aspekte widersprechen. Deshalb ist es auch hier wichtig, die Ziele, die erreicht werden sollen, genau festzulegen.

Eine wesentliche Voraussetzung für eine mögliche Kontrolle eines neuronalen Netzes ist eine ausreichende Dokumentation des Projekts (siehe Abschnitt 2.7 (Dokumentation)). Dies ist schon bei der Planung zu berücksichtigen und kann am besten durch die Einhaltung festgelegter Standards gesichert werden.

7.2 Optimierung der Struktur des Netzes

Die Optimierung der Netzstruktur, also die Optimierung der Anzahl der Knoten, ihrer Eigenschaften und ihrer Verknüpfungen, gehört meist noch in den Bereich der Entwicklung des Netzes. Für die Kontrolle spielt sie weniger eine Rolle. Doch auch für ein fertig erstelltes Netz kann

eine solche Optimierung sinnvoll sein. Im laufenden Betrieb ist beispielsweise immer zu kontrollieren, ob die Eingabedaten in dem Bereich liegen, in dem das Netz trainiert wurde. Erkennen lassen sich solche Eingabedaten dadurch, daß sie in der Statistik als Ausreißer auftreten oder etablierte Regeln verletzen. Wenn solche Eingabedaten auftreten, bedeutet dies meist, daß die Trainingsmenge nicht optimal gewählt war. Das neuronale Netz kann zwar für diese Daten unter Umständen trotzdem ein befriedigendes Ergebnis liefern, man sollte dann aber in Erwägung ziehen, das Netz neu zu trainieren und die neuen Bereiche von Eingabedaten zu berücksichtigen. Auch für die Rechengeschwindigkeit kann eine Optimierung des Netzes sinnvoll sein, wobei aber immer darauf zu achten ist, inwieweit dabei Qualitätseinbußen auftreten. Bei dieser Optimierung sind nicht nur die Kosten für den Betrieb des Netzes selbst zu berücksichtigen. Die Kosten für die Beschaffung der Eingabedaten können unter Umständen recht hoch sein. Die Optimierung der Struktur des Netzes kann also auch unter der Frage betrieben werden, ob man wirklich alle Eingabedaten benötigt oder ob es ohne wesentliche Qualitätsverluste möglich ist, auf einen Teil der Eingabedaten zu verzichten.

7.3 Optimierung und Kontrolle der Software

Neuronale Netze sind letztlich ein Software-Produkt, daß wie andere Software-Produkte auch der Kontrolle unterliegen muß. Es kann passieren, daß Fehler erst nach längerem Betrieb auftreten, zum Beispiel deshalb, weil dann zum ersten Mal Datensätze auftreten, die aus einem Bereich kommen, der nicht hinreichend trainiert wurde. Im Gegensatz zu herkömmlicher Software basiert ein neuronales Netz aber nicht auf einem eindeutigen Algorithmus. Es liefert mit Fehlern behaftete Ausgabewerte.

In vielen Fällen werden neuronale Netze mit fertigen Programmpaketen entwickelt (siehe Abschnitt 2.4 (Ressourcen)). Diese Programmpakete sind häufig relativ einfach zu bedienen und bieten dem Anwender ein Fülle von Analysemöglichkeiten, die auf graphischen Darstellungen basieren. Dadurch entsteht für die spätere Anwendung des Netzes der Nachteil, daß dieses mehr Rechenzeit und viel mehr Speicher benötigt, als bei einer direkten Programmierung des Netzes auftreten würde. Es ist daher sinnvoll, die Software, mit der das Netz betrieben wird, für den Betrieb umzustellen. In der Regel ist dies mit einmaligen Kosten verbunden, die durch den geringeren Ressourcenbedarf wieder eingespart werden können. Ist die Rechenzeit für den laufenden Betrieb eine kritische Größe, dann sollte man diese Optimierung in jedem Fall vornehmen. Zudem ist die Kontrolle eines kleineren Programmpakets in der Regel sehr viel einfacher.

7.4 Optimierung der Hardware

Neuronale Netze können auf beliebigen Rechnern programmiert werden. Bei der Optimierung der Hardware spielen zwei Aspekte eine Rolle: Rechengeschwindigkeit und Stabilität. Bei dem Betrieb eines neuronalen Netzes kann die Geschwindigkeit durch neuere Hardware oft gesteigert werden. Meist bringt eine Optimierung der Software für diesen Punkt aber mehr. Ein zweiter Aspekt, der auch für die Kontrolle eine Rolle spielt, ist die Stabilität. In vielen Fällen soll das neuronale Netz über mehrere Jahre zuverlässig arbeiten. Dann muß die Hardware so ausgelegt sein, daß sie diesen Anforderungen Stand hält. Dazu gehört natürlich auch die Auswahl des Betriebssystems und die Pflege des Systems.

8 Anwendung

Abschließend wird hier eine Beispielanwendung aus dem medizinischen Bereich vorgestellt.

8.1 Beschreibung des Projekts

Im Rahmen einer medizinischen Studie zur aktiv-spezifischen Immuntherapie (ASI) <http://people.freenet.de/~muelke/> für Krebs werden Patienten mit dieser Therapie behandelt und gleichzeitig werden von diesen Patienten eine Reihe von klinischen Daten und Immundaten bestimmt. Im Rahmen der Therapie erhält ein Patient bis zu drei Impfungen. Vor und nach jeder Impfung wird ein Immunstatus des Patienten erstellt. Es ist sehr schwer, aus diesen Daten Schlüsse zu ziehen, da die Bedeutung der einzelnen Immundaten für die Therapie nicht klar ist. Daher soll mit Hilfe eines neuronalen Netzes versucht werden, den Therapiefortschritt in Abhängigkeit von den klinischen Daten und den Immundaten zu erhalten. Langfristig besteht das Ziel, aus den Patientendaten einen Index zu berechnen, der angibt, welchen Erfolg man von einer Immuntherapie für einen einzelnen Patienten erwarten kann. Dieser Index kann aus den klinischen Daten und dem anfänglichen Immunstatus oder eventuell aus der Reaktion auf die erste Impfung ermittelt werden.

8.2 Vorarbeiten

Bevor die Entwicklung eines neuronalen Netzes möglich ist, ist es notwendig, den gesuchten Index zu definieren. Diese Definition benötigt man, da das neuronale Netz mit vollständigen Datensätzen trainiert werden muß. Die Patienten sind über einen unterschiedlich langen Zeitraum beobachtet worden. Das Hauptproblem besteht darin, daß der Fortschritt der Therapie natürlich vom Beobachtungszeitraum abhängt. Man muß deshalb für eine unterschiedliche Gewichtung der einzelnen Datensätze beim Training des Netzes und beim Test sorgen, wobei sich die Gewichtung nach dem Beobachtungszeitraum richtet. Die Berechnung des Index ist ein Beispiel für eine Kodierung der Ausgabedaten (siehe Abschnitt 3.2 (Datenkodierung)). Natürlich werden auch alle Eingabedaten kodiert, allerdings nur in trivialer Weise: Sie müssen für die Verarbeitung geeignet normiert werden.

8.3 Implementierung eines neuronalen Netzes

Zu Beginn des Projekts standen von etwa 200 Patienten Daten zur Verfügung. Da diese Daten nicht immer systematisch aufgenommen worden waren, waren nicht alle Datensätze vollständig. Ein Dilemma bestand also darin, daß die Rechnung zum Beispiel mit einem reduzierten Datensatz und mehr Patienten, oder mit einem vollständigen Datensatz und weniger Patienten durchgeführt werden konnte. Da a priori nicht klar ist, welche Daten besonders relevant sind, haben wir zunächst Rechnungen mit vollständigen Datensätzen durchgeführt. Da die Zahl der Patienten aber sehr klein war, waren die statistischen Fehler relativ groß. Es deutete sich aber an, daß von den Immundaten und den klinischen Daten nicht alle wichtig sind, so daß anschließend mit kleineren Datensätzen von mehr Patienten weitergerechnet werden konnte. Das Ergebnis dieser Rechnungen war, daß ein Mehr-Schicht Perceptron mit einer Zwischenschicht in der Lage ist, die Daten zu lernen und mit einem gewissen Fehler auch Vorhersagen

zu machen. Der Fehler lag bei diesen Rechnungen allerdings noch zu hoch. Der Grund dafür waren zu große statistische Fehler bei den Eingabedaten.

Um die Fehler bei den Eingabedaten zu minimieren, wurde die Bestimmung der Immunstatus automatisiert und standardisiert. Seit einiger Zeit läuft eine neue Studie zur ASI, an der 600 Patienten teilnehmen sollen. Die statistischen Fehler der Immunstatusdaten sind für diese Patienten geringer, die Zahl der Patienten ist zudem größer, so daß man erwarten kann, daß mit diesen neuen Daten ein neuronales Netz erfolgreich trainiert werden kann. Dabei sind die bisher gewonnenen Erfahrungen natürlich von großem Nutzen.

8.4 Ausblick

Die Dauer der Studie ist auf mehrere Jahre angelegt. Auch wird man erst nach mehreren Jahren den Erfolg der Therapie an den an der neuen Studie teilnehmenden Patienten wirklich beurteilen können. Erste Voruntersuchungen mit einem Teil der neuen Patientendaten sollten aber in naher Zukunft schon möglich sein.

9 Links zu anderen WWW-Seiten über neuronale Netze

Diese Liste ist zur Zeit noch sehr unvollständig. Arbeiten Sie in diesem Bereich? Haben Sie eine WWW-Seite, auf der Sie Informationen zu neuronalen Netzen und Anwendungen bereitstellen? Dann kontaktieren Sie uns bitte, wir nehmen Ihre Seite gerne in diese Liste auf.

- Die oben erwähnte Risikoabschätzung für Prostatakrebs <http://www.charite.de/ch/uro/html/p> ist ein praktisches Beispiel für die Anwendung eines neuronalen Netzes. Die Autoren bieten auf ihrer Seite das Programm zum Herunterladen und viele Hinweise, auch zu weiterführender Literatur.
- Die Neural Network FAQs <ftp://ftp.sas.com/pub/neural/FAQ.html> stellen einen guten Ausgangspunkt dar. Sie enthalten Links auf weiterführende Seiten, die aber in vielen Fällen nicht funktionieren.
- NeuroDimension Inc. stellt eine Seite <http://www.nd.com/appcornr/purpose.htm> mit Links zu verschiedenen Anwendungen bereit, die allerdings auch teilweise nicht mehr vorhanden sind.