Deutsche Telekom AG – Fachhochschule Leipzig

Automatische Artefaktzurückweisung in EEG-Signalen durch künstliche neuronale Netze

Diplomarbeit

Andreas Romeyke

Deutsche Telekom AG – Fachhochschule Leipzig

Diplomarbeit

Automatische Artefaktzurückweisung in EEG-Signalen durch künstliche neuronale Netze

Andreas Romeyke
Matrikel 00337

2004

Betreut durch Prof. Dr. Ing. Michael Dlabka und
Dr. phil. Thomas C. Gunter

In der vorliegenden Arbeit wird die Eignung von künstlichen neuronalen Netzen für die automatisierte Zurückweisung von Artefakten in Elektroenzephalogrammen untersucht. Dabei wird ein Multi-Layer-Perceptron mit Backpropagation als Klassifikator eingesetzt und es werden verschiedene Konfigurationen und Vorverarbeitungsschritte beleuchtet. Dabei wird auch auf Probleme und deren Lösungen eingegangen, die während der praktischen Umsetzung in C-Programme gefunden wurden.

Danksagung

Diese Arbeit wurde unterstützend begleitet von meinen Betreuern
Dr. phil. Thomas C. Gunter (Max-Planck-Institut für Kognitions- und Neurowissenschaften, Leipzig) und
Prof. Dr. Ing. Michael Dlabka (Deutsche Telekom AG – Fachhochschule Leipzig). Sie wäre nicht möglich gewesen ohne die technische Unterstützung des Administratorenteams des MPI um Dr. Hayd, Frank Burkhardt und Felix Botner. Ein Danke geht an Maren Grigutsch für ihre kritischen Hinweise und ihr stets offenes Ohr für programmiertechnische Fragestellungen, sowie Dirk Koester für die Bereitstellung und manuelle Klassifikation der EEG-Daten.

An dieser Stelle möchte ich mich vor allem noch bei Stefan Bauer für seine hilfreichen Hinweise, Anregungen, Ideen und die Schaffung einer angenehmen Arbeitsatmosphäre bedanken. Nicht unerwähnt bleiben soll Jeanette Barrios, die mir half Prioritäten zu setzen und auch sonst unterstützend auf mich einwirkte. Nichtzuletzt geht noch ein kleines Danke an meine Töchter Lea Pascale und Leni Denise, die während des Studiums zu Recht ihre Aufmerksamkeit einforderten.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einführung

> Das Gehirn ist wie ein Rechen, nur der Mist bleibt drin hängen...
>
> *(Klehn)*

1.1 Problemstellung

Das Max-Planck-Institut für Kognitions- und Neurowissenschaften in Leipzig beschäftigt sich mit Grundlagenforschung hinsichtlich kognitiver Prozesse[1], vor allem mit der Untersuchung ereignisbedingter Potentiale (event related potentials – ERPs) von Elektroenzephalogrammen (EEG). Diese ERPs sind über die Kopfhaut abgegriffene elektrische Aktivitäten (hauptsächlich während der Verarbeitung wiederholt präsentierter Stimuli[2]) und reflektieren in gewissem Maße die Bedeutungserfassung, z. B. während des Lesens von Texten.

Für die Verarbeitung der EEG-Daten werden Programme benutzt, die aus den Versuchsdaten die ERPs ermitteln. Bevor allerdings ein Durchschnitt berechnet werden kann, müssen Artefakte, wie elektrische Signale von Augenzwinkern und Muskelbewegungen, aus den Datensätzen entfernt werden. Diese Prozedur erfolgt typischerweise in zwei Schritten. Zuerst wird ein primitives automatisches Zurückweisungsprogramm auf die Datensätze angesetzt, welches vor allem die vertikalen und horizontalen Elektro-Oculogramme (EOG) durch Verwendung eines bestimmten Schwellwertkriteriums analysiert. Danach wird in einem zweiten Schritt vom Wissenschaftler eine manuelle Zurückweisung vorgenommen um die Datensätze sauberer zu bekommen.

Dieser zweite Schritt kostet eine Menge Zeit, da jede Sekunde, jedes Sample eines EEGs visuell analysiert werden muß. Wenn dann andere als Augen-Artefakte vorhanden sind, müssen diese per Mausklick in einem Programm manuell markiert werden.

Wenn dieser Vorgang durch ein Neuronales Netz (ANN – artificial neural network) automatisiert würde, wäre dies eine enorme Arbeitserleichterung.

[1] d. h. , wie der Mensch Informationen, z. B. Sprache etc. verarbeitet
[2] z. B. Wörter

Dazu muß ein geeigneter ANN-Typ gefunden werden und das Netz lernen, was ein Artefakt ist und was nicht. In der Lernphase kann dabei auf eine große Menge von Datensätzen aus Hunderten von Experimenten zurückgegriffen werden.

Das Ziel der vorliegenden Diplomarbeit ist die Eignung von ANNs für diese geforderte Automatisierung zu untersuchen.

Das ANN soll unter Linux laufen, der Code in ANSI-C programmiert und als Open Source international für potentielle Benutzer verfügbar gemacht werden.

1.2 Existierende Lösungen am Max-Planck-Institut für Kognitions- und Neurowissenschaften

Am Max-Planck-Institut werden die EEG-Daten überwiegend einer Vorverarbeitung mit einem automatischen Zurückweisungprogramm unterworfen. Dabei werden über bestimmte Schwellwertkriterien (meist Standardabweichung) Zurückweisungsdateien erzeugt. In der praktischen Arbeit werden die so behandelten EEG-Aufzeichnungen von den Wissenschaftlern im Bereich der Stimuli nochmals manuell auf korrekte Artefaktklassifizierung überprüft.

Seit einiger Zeit wird von einigen Wissenschaftlern die ICA-basierte Artefaktentfernung über die frei verfügbare *EEG Toolbox* (Delorme und Makeig, 2004) benutzt.

1.3 Lösungsansätze in der Literatur

Über Robert und Gaudy (2002), einem sehr guten Überblicksartikel zu EEG-Datenverarbeitung mit Neuronalen Netzen, findet man eine recht gute Zusammenfassung zum Thema und einen repräsentativen Querschnitt zu weiterführenden Artikeln. Trotz unterschiedlicher Ergebnisse der verschiedenen Autoren, wird der EEG-Artefaktbearbeitung mit Neuronalen Netzen eine positive Entwicklung vorausgesagt.

Den ersten Einstieg in das Thema erleichtert Schleimer (2003). Im Rahmen eines Praktikums wurden die Möglichkeiten der Klassifikation von EEG-Signalen mit verschiedenen Neuronalen Netzen untersucht. Der Autor geht nicht tief in das Thema hinein, allerdings wird die Orientierung erleichtert. So wird aufgezeigt, daß die Hauptkomponentenanalyse (PCA – principle component analysis) kein hinreichend befriedigender Ansatz ist. Support Vector Machines (SVM)[3] lieferten

[3] für Details, ebenda oder Zell (1994)

ebenfalls keine befriedigenden Ergebnisse.

Um sich mit EEGs im allgemeinen und den verschiedenen Artefakttypen vertraut zu machen, empfiehlt es sich einen Blick in Ebe und Homma (1992) zu werfen. Es handelt sich um ein einsteigerfreundliches Bildkompendium. Die dort aufgeführten Artefakttypen machen den Umfang und die Schwere der Problemstellung deutlicher und vermitteln ein Gefühl dafür, welche Anforderungen an ein ANN gestellt werden müssen und stellt damit eine Ergänzung zu Novák u. a. (2001) dar.

Cooper u. a. (1984) ist ein sehr umfangreiches Buch über Elektroenzephalographie. Neben der geschichtlichen Entwicklung, den physiologischen Vorgängen, über Elektrodenkonfiguration und Arten der Ableitung, sowie Geräteeigenschaften, bietet es im Kapitel *EEG-Signalanalyse* einen umfangreichen Einblick in die verschiedenen Standardmethoden der Signalanalyse. So werden statistische Eigenschaften und Frequenzkennwerte beschrieben, die sich als hilfreich für die Transformation der EEG-Daten in für ein ANN geeignete Werte erweisen könnten.

In Novák u. a. (2001) wird auf alle gängigen Basisalgorithmen für die EEG-Datenverarbeitung eingegangen. Im ersten Teil werden neben einer Einführung zum Aufbau des Gehirns und der (biologischen) Neuronen, die typischen EEG-Muster inklusive verschiedener Artefakttypen dargestellt. Die EEG-Signalverarbeitung wird anhand der FFT, der SFFT, der CSA, Wavelet-Tranformation, Amplituden-und Spektralmapping, sowie der ICA, etc. behandelt. In dem Report findet man weiterhin eine Einführung in ERPs, sowie die Elektrodenkonfiguration des Internationalen 10-20-Systems[4].

Einen Ansatz, mit Hilfe von Wavelet-Transformationen, Artefakte in EEG-Daten zu erkennen wird neben Durka u. a. (1996) auch von Ksiezyk u. a. (1998) verfolgt. Die dort aufgeführten Ergebnisse bei Eingabe roher EEG-Daten bzw. statistisch beschriebener EEG-Daten ins ANN decken sich interessanterweise mit den später selbstgemachten Erfahrungen des Autors.

In jüngerer Zeit wurde auch der ICA-Ansatz zur Artefaktentfernung stärker verfolgt. Dies zeigen beispielhaft die Arbeiten von Makeig u. a. (1996), Jung u. a. (1998), aber auch Delorme und Makeig (2004). Weiterhin finden sich bei Bell (2000), einer sehr detaillierten Abhandlung rund um ICA und Blind Source Separation, Beispiele zur Artefakterkennung in EEG-Daten und Verweise auf weitere Quellen.

Die Arbeit von Vuckovic u. a. (2002) zeigt, daß eine Klassifizierung von EEG-Daten mit Neuronalen Netzen bei geeigneter Transformation der Eingangsdaten, z. B. durch gleitende Ermittlung der Kreuz-Spektral-Dichte möglich ist. Die Autoren

[4] s. h. auch 2.1.2 auf Seite 6

vergleichen drei verschiedene ANN-Typen: Lineares Netzwerk mit Widrow-Hoff-Regel[5], nicht-lineares ANN mit Levenberg-Marquardt-Regel und Learning Vector Quantization Neural Network um in EEG-Daten zwischen Wach- und Schlafzuständen zu unterscheiden.

Vielversprechend scheint auch der Ansatz, der von Bogacz u. a. (1999) verfolgt wird. Die Autoren untersuchen drei verschiedene Klassifikatoren: k-neighbours, RBF Netzwerke und MLP mit Backpropagation. Dabei schneidet das MLP mit Backpropagation am besten ab. Das Paper liefert wertvolle Informationen um die Komplexität des ANNs niedrig zu halten.

Als Referenzquelle erweist sich Weber u. a. (2004) als sehr hilfreich. Die *EEG and MRI Matlab Toolbox* implementiert alle gängigen Algorithmen der EEG-Signalverarbeitung in Matlab und wird aktiv weiterentwickelt.

1.4 Verfolgte Lösungsansätze

Aus den in 1.1 auf Seite 1 aufgeführten Anforderungen kristallisieren sich nach der Analyse der Fachliteratur in 1.3 auf Seite 2 folgende Möglichkeiten heraus:

- Klassifikator über ANN mit
  - Rohdaten
  - statistisch–beschreibenden Kenngrößen
  - Kenngrößen der bisherigen EEG-Signalverarbeitung (Frequenz, Spektrale Dichte usw.)
  - Wavelet-Koeffizienten
- Prädiktor über ANN, der anhand der nicht artefaktbehafteten Daten ein EEG-Signal extrapoliert, ab einer bestimmten Abweichung wird ein Artefakt erkannt.
- ICA mit nachgeschaltetem Klassifikator über ANN

Nach eingehender Analyse und ersten Vortests (s. h. 4.4 auf Seite 24) werden die ICA und der Prädiktor verworfen. Die ICA reduziert das Problem aus Sicht der Netzwerkkomplexität nicht. Sie liefert für n-Eingangsgrößen auch n-Ausgangsgrößen, wobei die n-Ausgangsgrößen in beliebiger Reihenfolge vorkommen und von einem Klassifikator korrekt einzelnen charakteristischen Komponenten zugeordnet werden müssten. Eine *Augenbewegung*-Komponente könnte

[5] auch als Delta-Regel bekannt

also bei einer Berechnung am Ausgang n_i, bei einer anderen bei n_j (mit $i \neq j$) zu finden sein. Auch wird die Artefaktzurückweisung von vielen Wissenschaftlern am MPI der Artefaktentfernung mittels ICA vorgezogen, da bestimmte Voraussetzungen[6] der ICA in Bezug auf EEG-Daten nicht zweifelsfrei geklärt sind.

Der Prädiktor zur Extrapolation nicht artefaktbehafteter EEG-Zeitreihen wird nicht weiter verfolgt, da sich bei ersten Vortests Schwierigkeiten des verwendeten Netztyps mit den Rohdaten zeigen. Dieser Ansatz sollte im Rahmen einer anderen Arbeit nichtsdestotrotz weiter verfolgt werden.

Im Verlauf der vorliegenden Arbeit werden daher nur die folgenden Konfigurationen (siehe auch Abbildungen 5.2 bis 5.6 auf Seiten 30–35) untersucht:

ANN als Klassifikator mit

- Rohdaten
- Kenngrößen der klassischen EEG-Signalverarbeitung
- statistisch–beschreibenden Kenngrößen

[6] Quellsignale nicht Gaußverteilt, lineare und instantane Mischung der Quellen, Anzahl der Meßpunkte größer/gleich Anzahl der Quellen u. a. , für Details s. h. Hennig (1998) und Novák u. a. (2001)

2 Grundlagen

Wer sich keinen Punkt denken kann, der ist einfach zu faul dazu

(Wilhelm Busch)

2.1 Einführung EEG-Daten und Artefakte

2.1.1 Allgemeines

Mit Hilfe von verteilten Oberflächenelektroden werden elektrische Potentialunterschiede von der Kopfhaut abgegriffen. Diese Spannungen werden verstärkt und anschliessend quantisiert, digitalisiert und komprimiert abgelegt.

2.1.2 Das Internationale 10-20-System

Die Verteilung der Oberflächenelektroden erfolgt meist nach dem *Internationalen 10-20-System* beziehungsweise den erweiterten Varianten. Es basiert auf der Beziehung der Elektrodenanordnung zum darunterliegenden Gebiet des zerebralen Kortex. Jeder Seite ist für die Identifikation ein Buchstabe zugeordnet der den Gehirnlappen spezifiziert und eine Zahl oder ein zweiter Buchstabe, der die Hemisphere angibt. Die Bezeichnung *10-20* bezieht sich auf den Abstand zwischen den Elektroden. Von der Nasenwurzel[1] bis zum Inion[2] werden in 10%, 20%, 20%, 20%, 20% und 10% des Abstandes Elektroden gelegt, s. h. 2.1.2 auf der nächsten Seite. Die verwendeten Buchstaben sind:

[1] auch: nasion, bezeichnet Punkt zwischen Vorderschädel und Nase
[2] 'Sprung' zwischen Hinterkopf und Nacken

- *F* frontaler Lappen[3]
- *T* temporaler Lappen[4]
- *C* zentraler Lappen[5]
- *P* parietaler Lappen[6]
- *O* Occipital-Lappen[7]

Die ungeraden Ziffern (1,3,5,7) verweisen auf die linke Hirnhälfte, die geraden (2,4,6,8) auf die rechte. *Z* verweist auf Elektroden auf der Mittellinie. *Fp* steht für *Front polar*. Je kleiner die Zahlen sind, desto näher liegen die Elektroden an der Mittellinie (vergl. Novák u. a. (2001)).

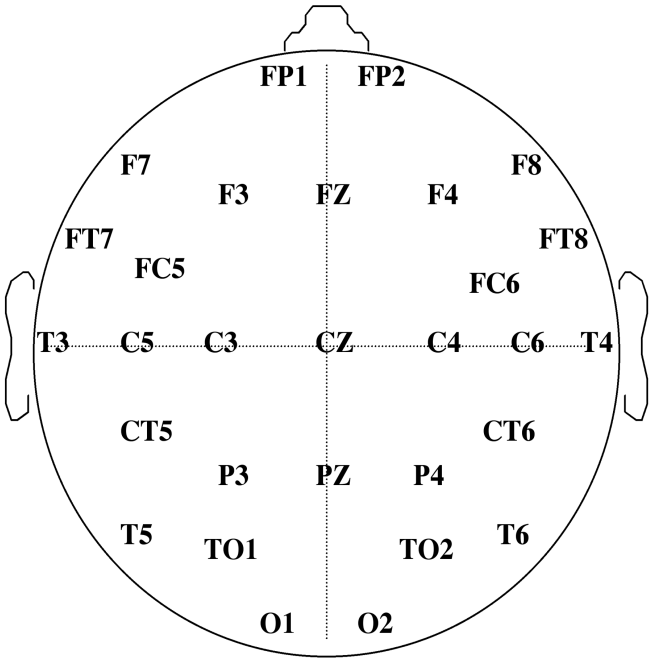


Abbildung 2.1: Elektrodenkonfiguration nach
Internationales 10-20-System, Draufsicht

Zum Internationalen 10-20-System gehören folgende Elektrodenbezeichnungen: *eogh*,*eogv*,*fp1*,*fp2*,*f7*,*f8*,*f3*,*fz*,*f4*,*t3*,*c3*,*cz*,*c4*,*t4*,*a1*,*a2*,*p3*,*pz*,*p4*,*t5*,*t6*,*o1* und *o2*.

[3]Lobus frontalis
[4]Lobus temporalis, auch Schläfenlappen
[5]Im zerebralen Kortex gibt es in dem Sinne keinen zentralen Lappen, das *C* wird hier nur für die Identifikation verwendet.
[6]Lobus parietalis, auch Scheitellappen
[7]Lobus occiptalis

2.1.3 EEG-Aufzeichnung und Parameter

Es gibt verschiedene Varianten der Potentialableitung. Üblich ist die Verwendung von *m1* oder *m2*[8] als Referenzelektrode, in manchen Konfigurationen wird auch das *nasion* als Referenz benutzt. Alternativ können die Elektroden bipolar verschaltet werden. Für das ANN spielt die Art der Potentialableitung keine Rolle, sollte dann aber dokumentiert und bei der Verwendung von trainierten Netzen beachtet werden, damit die Daten vom ANN nicht falsch interpretiert werden.

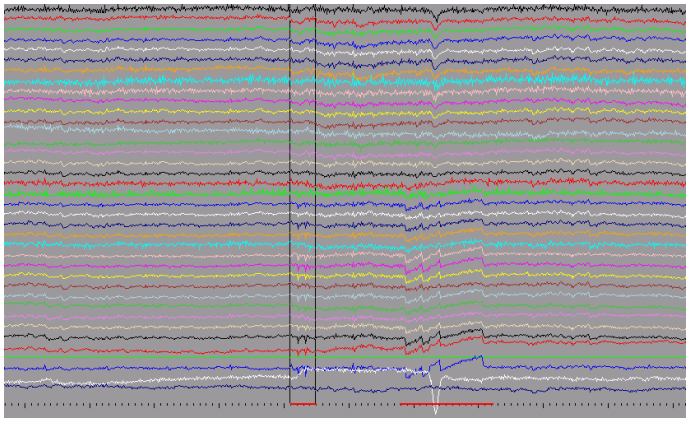


Abbildung 2.2: EEG-Beispiel

Bei der EEG-Aufzeichnung werden die abgegriffenen Spannungen in der Regel mit einer Frequenz von 250 Hz abgetastet, so daß nach Shannon-Theorem Frequenzen bis zu 125 Hz sicher gespeichert und rekonstruiert werden können. Die Spannungswerte werden üblicherweise mit 22bit linear quantisiert, wobei ein Dynamikbereich von ± 3 V überstrichen wird. Pro Samplepunkt und pro Kanal ergeben sich damit 3 Byte Rohdaten, 180 Bytes pro Sekunde. Eine übliche Aufzeichnung dauert ca. 90 min und umfaßt bis zu 63 Kanäle, dabei fallen knapp 58 MByte an Daten an. Durch spezielle Datenreduktionsverfahren läßt sich diese Menge verlustlos auf knapp 15 MByte komprimieren.

Eine besondere Rolle spielen die Augenkanäle *eogh* und *eogv*. Diese Kanäle werden am Max-Planck-Institut durch bipolare Elektrodenanordnung in vertikaler (*eogv*) und horizontaler Richtung (*eogh*) als Differenzspannung bzw. EOG gemessen.

[8] linkes bzw. rechtes Ohrläppchen

2.1.4 Artefakte

In der EEG-Aufzeichnung am Max-Planck-Institut sind folgende Artefakttypen relevant:

- Augenartefakte:
  - Augenzwinkern (eye blinks)
  - Augennachführung (eye tracking)
- Elektrodenartefakte
- Muskelartefakte
- Bewegungsartefakte
- line noise, Einstreuungen von Störspannungen auf Ableitungen
- Herzschlag

Andere Artefakttypen sind für die Bestimmung von ERPs nicht von Belang, da sie in der Regel unter einem bestimmten Schwellwert liegen und durch die Berechnung der ERPs herausgemittelt werden.

2.2 Einführung in Neuronale Netze

2.2.1 Multi Layer Perceptron

Das Multi Layer Perceptron (MLP) ist ein *feedforward*-Netzwerk basierend auf Neuronen nach dem Modell von Minsky und Papert (1988)[9]. Dabei werden die Eingangswerte i_k eines Neurons summiert. Der Ausgangswert o des Neurons bestimmt sich in der Regel durch die Anwendung einer nichtlinearen Funktion $g(x)$ auf diese Summe:

$$o = g\left(\sum_k i_k\right) \tag{2.1}$$

Die Grundstruktur eines MLP findet man in Abbildung 2.3. Dabei ist jede Schicht S_i mit der Folgeschicht S_{i+1} vollständig verbunden. Jeder Verbindung ist ein Gewicht w_{ij} zugeordnet. Für die Zahl der Verbindungen gilt demnach:

$$\dim(w_{ij}) = \dim(S_i)\dim(S_{i+1}) \tag{2.2}$$

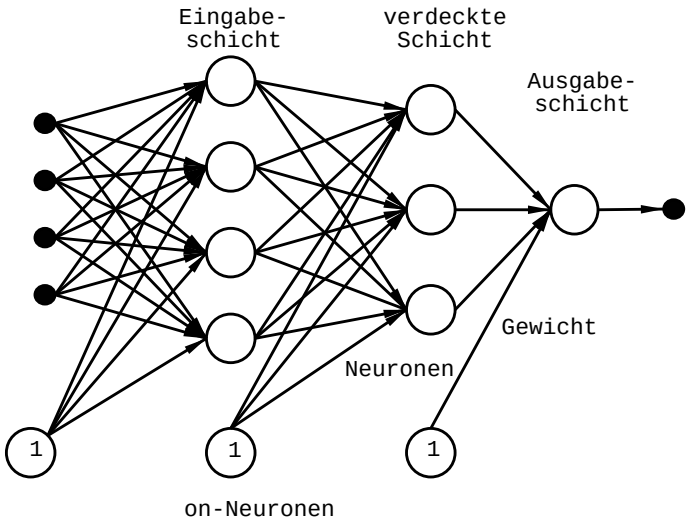


Abbildung 2.3: Grundstruktur Neuronales Netz

[9] Ursprünglich von Frank Rosenblatt, durch Minsky und Papert aber genauer definiert. Details finden sich bei Zell (1994)

2.2.2 Backpropagation

Der Backpropagation-Algorithmus ist ein Lernverfahren, mit dem ein *feedforward*-Netzwerk, z. B. ein MLP, trainiert werden kann. Er gehört zu den Gradientenabstiegsverfahren, d. h. die Änderungen der Gewichte Δw_{ij} erfolgen in Richtung des negativen Gradienten bzw. steilsten Abstiegs der Fehlerfunktion:

$$\Delta w_{ij} = -\nu \frac{\delta E}{\delta w_{ij}} \tag{2.3}$$

Der Grad der Änderung wird durch den Lernfaktor ν beeinflußt.

Der Algorithmus benötigt pro Lernschritt zwei Netzdurchläufe. Im ersten werden von der Eingangsschicht zur Ausgangsschicht aus den Eingangsdaten die Ausgaben o_j des ANN berechnet. Diese werden mit den Sollwerten t_j verglichen, der entsprechende Fehler E[10]:

$$E_p = \frac{1}{2} \sum_j \left(t_j - o_j\right)^2 \tag{2.4}$$

$$E = \sum E_p \tag{2.5}$$

wird dann in einem zweiten Durchlauf von der Ausgangsschicht zu den Eingangsneuronen zurückpropagiert und der Anteil berechnet, den jedes Gewicht der Neuronenverbindungen und damit jedes Neuron zum Fehler am Ausgang beiträgt.

Der Backpropagation-Algorithmus stellt im Prinzip nichts anderes als die Ableitung des Fehlervektors E nach den Neuronengewichten dar:

$$\frac{\delta E}{\delta w_{ij}} = \frac{\delta E}{\delta o_j} \frac{\delta o_j}{\delta \mathrm{net}_j} \frac{\delta \mathrm{net}_j}{\delta w_{ij}} \tag{2.6}$$

Dabei ist in Anlehnung an Callan (2003) und Zell (1994) net_j die Eingabe eines Neurons:

$$\mathrm{net}_j = \sum_{i=0}^{n} x_i w_{ij} \tag{2.7}$$

mit x_i dem Ausgangssignal der vorhergehenden Neuronen , w_{ij} dem Gewichtsfaktor der Verbindung für das aktuelle Neuron.

[10] Der Faktor $\frac{1}{2}$ in Formel 2.4 entsteht durch die Herleitung der Fehlerberechnung. Für Details sei auf Dlabka (2004) verwiesen. Der Laufindex der Summe in Gleichung 2.5 wird nach Lernverfahren gewählt. Für online- oder Musterlernen gilt: $E = E_p$.

Der Laufindex i beginnt mit 0, da der Bias der einzelnen Neuronen als *on*-Neuron mit dem Ausgangssignal x_0=1 definiert wird. Die Anzahl n gibt die maximale Zahl der Neuronen der vorherigen Schicht an, die mit dem aktuellen Neuron verbunden sind.

Die Ausgabe oder auch Aktivierung des aktuellen Neurons wird mit o_j bezeichnet:

$$o_j = g\left(\text{net}_j\right) \tag{2.8}$$

Siehe auch Gleichung 2.8.

Als Aktivierungsfunktion $g(x)$ wird meist der *tangens hyperbolicus*:

$$g(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.9}$$

mit dem Wertebereich $-1 \leq g(x) \leq 1$ oder die Sigmoidfunktion:

$$g(x) = s_c(x) = \frac{1}{1 + e^{-cx}} \tag{2.10}$$

mit $c = 1$ und dem Wertebereich $0 \leq g(x) \leq 1$ verwendet, da Backpropagation nach der klassischen Definition (Zell, 1994) eine nichtlineare[11], stetige und differentierbare Aktivierungsfunktion verlangt.

Backpropagation weist eine Reihe von Beschränkungen und Problemen auf:

- symmetry breaking, d. h. wenn die Initialgewichte einer Ebene gleich groß sind, so kann das Netz bei Backpropagation keine unterschiedlichen Gewichte in den vorderen Schichten ausbilden, da der Fehler zu gleichen Teilen auf jedes Gewicht zurückpropagiert wird. Um dies zu vermeiden wird das ANN mit kleinen Zufallswerten um 0 herum initialisiert.
- Lokale Minima der Fehlerfläche. Wie alle Gradientenverfahren neigt auch Backpropagation zum Hängenbleiben in lokalen Minima. Ein weiteres Problem sind Oszillationen in steilen Schluchten der Fehlerfläche, sowie Stagnation des Lernfortschritts auf flachen Plateaus. Nach Zell (1994) findet Backpropagation bei genügend kleinem Lernfaktor ν trotzdem in sehr vielen Anwendungen ein Minimum, welches sehr nahe am globalen liegt.

Durch die Einführung eines Momentumterms, der über den Parameter $0.5 < \alpha < 0.9$ eine Art Tiefpaßglättung der Gewichtsänderung bewirkt, wird zumindest der letzte Punkt entschärft:

$$\Delta w_{ij}(t) = \frac{\delta E}{\delta w_{ij}} + \alpha \Delta w_{ij}(t-1) \tag{2.11}$$

[11] Backpropagation funktioniert natürlich auch mit linearen Aktivierungsfunktionen

Gleiches gilt für die Verwendung einer sich anpassenden Lernrate (adaptives Lernen), bei der die Lernrate ν über den MQLE variiert wird.

Eine weitere Verbesserung bringt die Einführung einer nichtlinearen Fehlerfunktion. Diese sorgt dafür, daß Ausgabeneuronen, die stärker vom gewünschten Ergebnis abweichen, ein deutlich stärkeres Fehlersignal liefern.

Der Backpropagation-Algorithmus in der FANN-Bibliothek arbeitet aus Geschwindigkeitsgründen (Nissen, 2003) wahlweise mit stückweise linear angenäherter Sigmoid- oder tanh-Funktion.

Eine weitergehende Betrachtung der Backpropagation liefern Zell (1994) und Rojas (1996)[12].

[12] Eine populärwissenschaftliche, anschauliche und humourvolle Darstellung zu ANNs findet sich bei Tutrin (2004)

2.3 Einführung in grundlegende statistische Kriterien

2.3.1 Bereich

Der Bereich (range) ist die Differenz des Maximums und des Minimums einer Verteilung:

$$x_r = x_{\max} - x_{\min} \tag{2.12}$$

2.3.2 Arithmetischer Mittelwert

Der Arithmetische Mittelwert (mean, im folgenden nur Mittelwert) wird berechnet:

$$\hat{x} = \mu = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{2.13}$$

2.3.3 Erwartungswert

Der Erwartungswert $E(x)$ einer Zufallsvariable x ist derjenige Wert, von dem man *erwartet*, daß er sich bei einer genügend großen Zahl n von Experimenten als Durchschnitt (s. h. Mittelwert) ergibt. Er errechnet sich aus der Summe des Produktes der Werte x_i und dazugehöriger Einzelwahrscheinlichkeiten p_i dieser Zufallsvariablen:

$$E(x) = \sum_{i}^{n} x_i p_i \tag{2.14}$$

2.3.4 Zentrale Momente

Zentrale Momente der Ordnung k sind in der Statistik definiert als:

$$k_k(\mu) = E\left((X - \mu)^k\right) \tag{2.15}$$

mit dem Erwartungswert $E(X)$ und dem Arithmetischen Mittel μ.

2.3.5 Varianz und Standardabweichung

Die Varianz (variance) als zentrales Moment der Ordnung zwei beschreibt die Verteilung der Merkmalsausprägung einer Variablen um den Erwartungswert. Sie kann unter Verwendung des Mittelwerts $\hat{x}$ (Gleichung 2.13 auf der vorherigen Seite) berechnet werden:

$$\hat{\sigma} = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \hat{x})^2 \tag{2.16}$$

In der Literatur wird häufig die folgende Variante empfohlen:

$$\hat{\sigma} \approx \hat{x^2} - \hat{x}^2 \tag{2.17}$$

Nach Press u. a. (1993) sollte die Varianz lieber in zwei Durchläufen berechnet werden, da diese Variante schneller ist und Rundungsfehler vermeidet. Dabei wird in einem ersten Lauf der Mittelwert, und in einem zweiten die Varianz ermittelt:

$$\hat{\sigma} = \frac{1}{N-1} \left\{ \sum_{i=1}^{N} (x_i - \hat{x})^2 - \frac{1}{N} \left[\sum_{i=1}^{N} (x_i - \hat{x}) \right]^2 \right\} \tag{2.18}$$

Die Standardabweichung(standard deviation) ergibt sich unter Verwendung von Gleichung 2.16 aus:

$$\sigma = \sqrt{\hat{\sigma}} \tag{2.19}$$

2.3.6 Schiefe

Die Schiefe (skewness) als zentrales Moment der dritten Ordnung gibt den Grad der Rechts- bzw. . Linksverschiebung des Maximums einer Verteilung an. Für Gaußsche Normalverteilung ist die Schiefe $v = 0$.

Die Schiefe v für eine Anzahl von N Werten wird berechnet (Bartsch, 1999):

$$v = \frac{\sum_{i=1}^{N} (x_i - \hat{x})^3}{N\sigma^3} \tag{2.20}$$

2.3.7 Wölbung

Die Wölbung w ist ein zentrales Moment der Ordnung vier und wird für eine Anzahl von N Werten berechnet (Bartsch, 1999):

$$w = \frac{\sum_{i=1}^{N} (x_i - \hat{x})^4}{N\sigma^4} - 3 \tag{2.21}$$

Nach Wikipedia (2004) ist bei Verteilungen, deren maximale Dichte gleich der maximalen Dichte der Normalverteilung entspricht, die Wölbung (auch Kurtosis) Null. Ist sie negativ, hat die Verteilung eine geringere maximale Dichte als die Normalverteilung – sie ist flacher. Ist die Dichte positiv, ist die maximale Dichte größer als bei der Normalverteilung, die Werte sind dichter und die Verteilung stärker gewölbt.

3 Vorbetrachtung

Think early, think often

3.1 Analyse verfügbarer C-Bibliotheken

Über die FAQ[1] der Newsgroup `news:comp.ai.neural-nets` findet man im Teil 5 *Free software* (Sarle, 2004) einige Verweise auf Quellcode von Neuronalen Netzen. Eine größere Auswahl an freien Projekten findet man bei Inc. (2004).

Neben vielen Implementierungen in C++, die aufgrund der Bedingungen in 1.1 auf Seite 1 nicht verwendet werden konnten, seien kurz folgende Projekte vorgestellt:

- Stuttgarter Neuronale Netze Simulator (SNNS), ursprünglich an der Universität Stuttgart entwickelt, befindet sich jetzt unter der Obhut der Universität Tübingen (Zell, 2004). Der SNNS wird nach Zell (1994) in vielen Projekten erfolgreich eingesetzt. Die aktuelle Version des SNNS ist nur in der Programmiersprache Java verfügbar. Die älteren und ausgereiften C-Versionen liegen zwar im Quellcode vor, lassen sich aber nicht mehr mit heutiger Umgebung fehlerfrei übersetzen, da Headerfiles und Programmkonstrukte veraltet und nur wenig Dokumentation verfügbar ist. Die C-Versionen werden bedauerlicherweise nicht weiter gepflegt.

  Die aktuelle Java-Implementierung ist dagegen sehr brauchbar, konnte aber aufgrund der Restriktionen in 1.1 auf Seite 1 nicht für diese Arbeit verwendet werden.

- Logically Advanced Neural Engine (NeuroQuest AI, 2004), unterstützt ff. Netzwerke:
  - Adaline Netzwerk
  - Backpropagation Netzwerk
  - Bi-Directional Associative Memory Network
  - Bi-Directional Associative Memory System Network

[1] FAQ – frequently asked questions, Antworten auf häufig vorkommende Fragen

  - Backpropagation mit Epochenlernen
  - Self Organizing Maps

  Negativ ist die mangelnde Dokumentation und die geringe Aktivität in der Entwicklung.

- Light Weight Neural Network (van Rossum, 2004) wurde im Hinblick auf Leichtgewichtigkeit und leichter Verwendbarkeit entworfen. Sie unterstützt nur MLP-Netzwerke. Als Aktivierungsfunktion steht nur die Sigmoid-Funktion zur Verfügung. Als Lernverfahren wird Backpropagation mit Momentumterm verwendet. Die Bibliothek ist recht schnell, die Aktivierungsfunktion wurde über Lookup-Tables implementiert.
- Fast Artificial Neural Network Library (Nissen, 2004) wurde von Steffen Nissen im Rahmen seiner Diplomarbeit (Nissen, 2003) als schnelle ANN-Bibliothek entworfen. Sie erhebt den Anspruch leicht verwendbar, vielseitig anwendbar, gut dokumentiert und portabel zu sein. Die Bibliothek verwendet MLP mit Backpropagation als Lernverfahren und die Sigmoid-Funktion, die *tanh*-Funktion und Sprungfunktion als Aktivierungsfunktionen.

In die nähere Auswahl kommen auf Grund der guten Dokumentation, der Schnelligkeit und aktiven Weiterentwicklung die Bibliotheken *Light Weight Neural Network* (LWN) und *Fast Artificial Neural Network Library* (FANN).

Die API[2] der LWN ist flexibel, bedingt aber erhöhten Programmieraufwand. Die API der FANN dagegen ist sowohl flexibel, als auch einfach und unkompliziert zu verwenden. Nachteilig bei der FANN ist die Beschränkung[3] auf reines Backpropagation.

Durch

- das Studium der Arbeit von Nissen (2003), die einen objektiven Vergleich zwischen FANN und LWN enthält
- dem Entwurf von Prototypen, die beide Bibliotheken verwendeten

wird sich im folgenden auf die Verwendung der FANN Bibliothek beschränkt. Während des Diplomarbeitszeitraumes wurde die FANN stark weiterentwickelt und verbessert und hat einen stabilen Status erreicht. Auch der Support durch die Entwickler rund um Steffen Nissen erweist sich als excellent. Die Bibliothek verhält sich äußerst schnell (Neuronenupdate auf aktuellen PCs unter 150 ns), ist sauber entworfen und eignet sich damit auch für größere ANN-Projekte.

[2] API – Application Programming Interface

[3] Die FANN Bibliothek wird in der nächsten Version um adaptives Lernen, Epochen- bzw. Batchlernen, wie auch um Backpropagation mit Momentumterm ergänzt. Zukünftige Versionen unterstützen Cascaded Correlation Networks und andere ANN-Typen.

3.2 EEG-Daten und Artefaktbeschreibung

Im Rahmen dieser Arbeit wird ein einfaches, binäres Datenformat für die EEG-Daten verwendet. Die *eeg*-Datei besteht aus dem String `#EEGbu`, gefolgt von einem Header nach Listing 3.1, der die Zahl der Samples und Datenkanäle, sowie die erfolgte Normierung beschreibt. Anschliessend folgt ein Datenstrom mit den Werten der Samplepunkte als einfache Fließkommazahlen. Dabei werden die Werte pro Samplepunkt Kanalweise hintereinander abgelegt.

```
/** this stores necessary values in decompressed EEG-data
   */
struct s_cntconvhead {
    /** count of channels */
    unsigned int channelcount;
    /** count of samples per channel (all channels with
       same count) */
    unsigned int samplecount;
    /** which normalization is used */
    enum e_norm norm;
};
```

Listing 3.1: Header Definition für *eeg*-Format

Detailliertere Informationen hierzu finden sich in Romeyke (2004).

Die Artefakte dagegen werden in einer separaten Text-Datei beschrieben. *rej*-Dateien enthalten pro Zeile einen artefaktbehafteten Zeitabschnitt. Die Angabe erfolgt in Sekunden mit Angabe auf 3 Nachkommastellen. Einen Auszug aus einer *rej*-Datei findet man in Listing 3.2.

```
0.076-0.476
4.620-5.020
7.768-8.168
10.128-10.444
12.188-12.504
15.536-15.940
16.640-17.044
21.144-21.508
```

Listing 3.2: Auszug *rej*-Datei

Alle Artefakttypen einer EEG-Aufzeichnung werden am Max-Planck-Institut in einer *rej*-Datei zusammen erfasst.

4 Vorbereitende Aufgaben

Invention is 1% of inspiration
and 99% of transpiration.

(Thomas Alva Edison)

4.1 Dekompression der Daten

Die EEG-Daten am Max-Planck-Institut liegen in der Regel komprimiert im *cnt*-Format (Nowagk, 2004) der Firma Advanced Neural Technology (Technology, 2004)[1] vor. Für die Dekompression wird die ursprünglich am Max-Planck-Institut für Kognitions- und Neurowissenschaften entwickelte *opencnt*-Bibliothek (Nowagk, 2004) benötigt. Durch den proprietären Charakter dieser Bibliothek ist die Dekompression[2] in ein externes Programm `cntconv.c` ausgegliedert.

Diese Vorgehensweise hat primär lizenzrechtlich-politische Gründe, ermöglicht es zukünftig aber auch andere EEG-Datenformate leichter zu integrieren.

Nach der Dekompression wird ein Datenfile in einem internen *eeg*-Format erzeugt, für Details sei auf Romeyke (2004) verwiesen. Die entstehenden Daten haben ca. einen vierfachen Umfang der komprimierten *cnt*-Dateien.

[1] Nicht zu verwechseln mit dem *cnt*-Format(Bourke, 2004) der Firma Neuroscan (Ltd., 2004)

[2] wie auch die Normalisierung, bzw. Präparation der Daten und das Mapping auf das Internationale 10-20-System

4.2 Normalisierung der EEG-Daten

Die Normalisierung der EEG-Daten ist notwendig, da die Ableitungswerte[3] zwischen verschiedenen Versuchen und Versuchspersonen schwanken. Üblicherweise können die Spannungswerte in einem Bereich zwischen ± 3 V auftreten. Die Spannungswerte werden mit 22 Bits digitalisiert, so daß die Auflösung effektiv bei $\pm 10^{-6}$ V/bit liegt.

Die Normierung ist auch noch aus einem anderen Grund wichtig. Die verwendete FANN-Bibliothek unterstützt die C-Datentypen *float*,*double* und Festkommaarithmetik. Der Wertebereich dieser Datentypen ist eingeschränkt, so daß bei Verwendung von Werten $x > \pm 20$ der *tanh* immer zu ± 1 gesättigt wird. Der Tangens hyperbolicus oder die Sigmoid-Funktion wird in der FANN-Bibliothek als Aktivierungsfunktion der Neuronen verwendet. Wenn die Werte nicht in einen geeigneten Bereich normiert werden, lernt das ANN schlecht oder gar nicht, da bei der Backpropagation die aus der Sättigung resultierenden Werte nicht zum Eingang hin für die Gewichtsbestimmung zurückberechnet werden können.

Die Normalisierung im vorliegenden Programm ist recht einfach gehalten, die skalierten[4] Werte aus den *cnt*-Files werden einfach durch 400 dividiert. Überläufe zu ± 1 werden zu ± 1 satuiert. Der Wert von 400 ist empirisch ermittelt.

$$p = \frac{x_i}{400} \tag{4.1}$$

$$x_{i_{\text{norm}}} = \begin{cases} -1 & \text{für } p < -1 \\ 1 & \text{für } p > 1 \\ p & \text{sonst} \end{cases} \tag{4.2}$$

Eine andere mögliche Normierung wäre folgende:

$$x_{i_{\text{norm}}} = \frac{2x_i - (x_{\max} + x_{\min})}{x_{\max} - x_{\min}} \tag{4.3}$$

Diese Normierung ist empfindlich gegenüber Ausreißern zwischen verschiedenen

[3] Übergangswiderstand Haut–Elektrode, Verstärkereinstellungen

[4] im *cnt*-Format werden die Samplewerte und deren Skalierungsfaktoren getrennt gespeichert

Versuchen. Besser eignet sich nach Dlabka (2004):

$$\hat{x} = \frac{1}{N} \sum_{k=1}^{N} x_k \tag{4.4}$$

$$\sigma = \sqrt{\frac{1}{N-1} \sum_{k=1}^{N} (x_k - \hat{x})^2} = \sqrt{\hat{\sigma}} \tag{4.5}$$

$$x_{i_{\text{norm}}} = \frac{x_i - \hat{x}}{\sigma} \tag{4.6}$$

Diese Normalisierung, auch Standardisierte Zufallsvariable (Bartsch, 1999) ist wegen ihrer Mittelwertfreiheit $\hat{x} = 0$ und der Varianz $\hat{\sigma} = 1$ besonders für ANNs geeignet.

Da sich im folgenden die Normierung nach Gleichung 4.2 auf der vorherigen Seite als völlig ausreichend erweist und der Berechnungsaufwand um Größenordnungen niedriger liegt als bei den anderen Varianten, wird auf die Implementierung von 4.6 verzichtet. Für die Verbesserung der Ergebnisse eines ANN mit rohen EEG-Daten (zum Beispiel 5.2 auf Seite 30) sollte die Normierung nach Gleichung 4.6 eingesetzt werden.

4.3 Channel-Mapping

Am Max-Planck-Institut existiert das Problem, daß viele Versuche mit unterschiedlichen Konfigurationen gefahren werden. Dies äußert sich in der Anordnung und Zahl der verwendeten Elektroden, wie auch in der Art der Elektrodenableitung. Um hier für das Training, aber auch für den späteren Einsatz als Zurückweisungsprogramm einen Zugriff auf vergleichbare Datensätze zu haben, wird ein Channel-Mapping auf das Internationale 10-20-System (s. h. 2.1.2 auf Seite 7) benutzt. Das Channel-Mapping wird durch das Programm `cntconv.c` bei der Normalisierung mit erledigt.

Zu jeder *cnt*-Datei muß eine *chn*-Datei existieren. Diese beschreibt zu jeder Elektrodenbezeichnung nach dem Internationalen 10-20-System die möglichst gut übereinstimmende Elektrode der *cnt*-Datei. Einen Auszug aus so einer Datei findet man in Listing 4.1.

```
eogv=EOGV
a1=A1
a2=A2
t6=TO2
```

Listing 4.1: Auszug *chn*-Datei

4.4 Generierung künstlicher Testdaten

Für die objektive Prüfung der Eignung der verschiedenen Lösungsansätze, zur Vorauswahl und insbesondere zur Überprüfung der konkreten Implementierungen, ist es unverzichtbar Testdatensätze erzeugen zu können. Diese sollten möglichst nahe an die Problemstellung herankommen, andererseits aber auch Fehler in der Programmierung aufdecken. Zu diesem Zweck wird das Programm `gentestdata.c` verwendet. Es generiert zwei verschiedene Funktionen, einen Sinus:

$$f_1(x) = \sin x \frac{\pi}{300} \tag{4.7}$$

und eine Sägezahnkurve:

$$f_2(x) = \frac{x \mod \frac{f}{6}}{\frac{f}{6}} \tag{4.8}$$

mit $f = 250$ Hz. Im resultierenden File von 300000 Samples werden die Samplepunkte zufällig aus den zwei erzeugten Funktionen ausgewählt, wobei mindestens 30 aufeinanderfolgende Samplepunkte zu einer Funktion gehören. Der Funktion $f_1(x)$ wird die Bedeutung *kein Artefakt*, der Funktion $f_2(x)$ die Bedeutung *Artefakt* zugeordnet. Das Programm generiert passend zu den Sampledaten ein *eeg*-File und ein *rej*-File. Die Kanäle werden alle mit den gleichen Samplewerten belegt, da dies die Überprüfung der korrekten Speicheraddressierung ermöglicht und so einige Fehler in der Implementierung aufgedeckt werden konnten. Auf ein Verrauschen der Amplitudenwerte wird vorerst verzichtet, da das Problem für das Testen der ANNs so einfach wie möglich gehalten werden sollte. Die erzeugten Dateien wurden mit dem `eatr.c` und `ear.c` getestest.

4.5 Grundstruktur der ANN

4.5.1 Überblick

Das Projekt gliedert sich im Wesentlichen in drei Programme auf:

`cntconv` dekomprimiert die im *cnt*-Format vorliegenden EEG-Daten, normiert diese und schreibt sie in das eigene *eeg*-Format[5].

`eatr` (EEG artifacts training rejections) liest Datensätze im *eeg*-Format mit den dazugehörigen *rej*-Dateien und trainiert ein ANN. Das ANN wird in einstellbaren Intervallen gespeichert.

`ear` (EEG artifacts rejector) lädt ein gespeichertes ANN und kreiert aus den Daten im *eeg*-Format eine *rej*-Datei, die die vom Netz erkannten Artefakte beschreibt.

Die Grundstruktur in den Programmen `eatr` und `ear` ist analog aufgebaut und besteht aus den folgenden Modulen:

- Laden und Parsen der *eeg*-Daten (bei `eatr` auch der *rej*-Daten)
- Erzeugen und Füllen eines Zeitfensters
- ggf. Konvertieren/Transformieren der Daten
- Ausgabe bzw. Training der Zurückweisungswerte

4.5.2 cntconv

Dekomprimiert ein *cnt*-File, mappt die Kanäle mittels des dazugehörigen *chn*-File, normalisiert die EEG-Daten und schreibt einen binären (mit der Option *-b*) oder Menschen-lesbaren[6] Datenstrom nach `STDOUT`. Wird die Option *-d* verwendet, werden die Kanalmarkierungen des *cnt*-Files ausgegeben.

Der Name der Files *filename* wird ohne Suffixes angegeben, die Endungen *.cnt*, *.chn* werden automatisch ergänzt.

Das Programm wird wie folgt aufgerufen:

```
cntconv -[hbd] <filename>
```

[5] Siehe hierzu 4.1 auf Seite 20, 4.3 auf Seite 23 und 4.2 auf Seite 21
[6] Genauer, im Gnuplot-kompatiblen Textformat

4.5.3 eatr

Dieses Programm trainiert ein neues, oder geladenes neuronales Netzwerk, gibt ggf. den Lern- und Generalisierungsfehler aus, speichert in wählbaren Abständen das Netz.

Es wird wie folgt aufgerufen:

```
eatr (-h)|((-[ceilmor] [<argument>])|(-a) ... <filename> ...)
```

Die Lernrate ν wird über die Option *-l <fliesskommazahl >*, die Zahl der maximalen Lernepochen über *-m <ganzzahl >* angegeben. Mit *-e <fliesskommazahl >* legt man den Epochenfehler als Abbruchkriterium fest. Wichtig ist noch der Parameter *-i <net-filename >*, da man so ein schonmal trainiertes Netz weitertrainieren lassen kann.

Die Hilfe zu den anderen Optionen kann man mit *-h* abrufen. Auf die anderen Aufrufparameter wird an dieser Stelle nicht eingegangen, da diese in der Regel nicht benötigt werden.

Das Programm verwendet sinnvolle Standardeinstellungen, so daß man mit folgendem Aufruf gut bedient ist:

```
eatr -o <output-files>  <filename> ...)
```

Der Name der Files *filename* wird ohne Suffixes angegeben, die Endungen *.eeg* und *.rej* werden automatisch ergänzt. Gleiches gilt für *<output-files >*. Die Option *-o* erzeugt demnach die Endungen *.mqle*, *.mqge*, *.cfg* und *.net*.

Dabei wird in den *mqge-* und *mqle*-Dateien der MQGE bzw. MQLE im Menschen-lesbaren GNUPlot-Format abgespeichert. Die *net*-Datei enthält das in regelmäßigen Abständen gespeicherte und trainierte Netz. Das *cfg*-File protokolliert die Konfigurations- und Aufrufparameter.

Dem Programm können beliebig viele *<filename >* übergeben werden. Die Anzahl wird durch den verfügbaren Arbeitsspeicher begrenzt. Falls mehrere *eeg-* und *rej*-Dateien angegeben werden, so sind die Netzparameter anzupassen.

4.5.4 ear

`ear` wird mit dem trainiertem ANN und dem zu markierenden *eeg*-File geladen. Es generiert einen Datenstrom für die Standardausgabe im *rej*-Format mit den Artefaktmarkierungen.

Der Aufruf ist vergleichsweise simpel:

```
ear <net-file> <eeg-file>
```

5 Analyse der verschiedenen Lösungsansätze

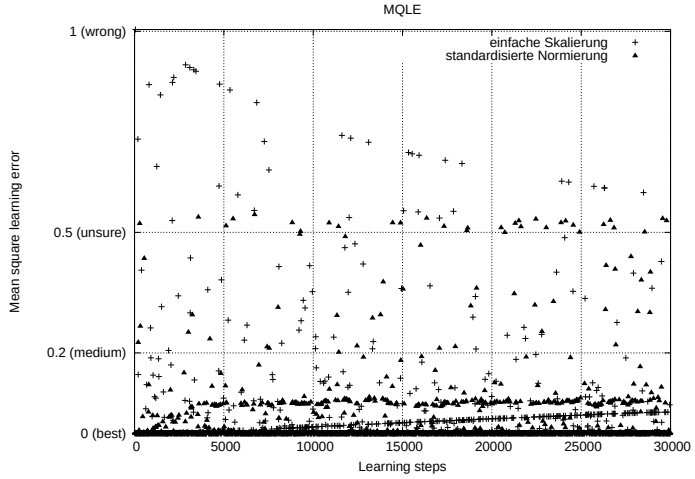
Alles was schief gehen kann
wird schief gehen.

(Murphys Gesetz)

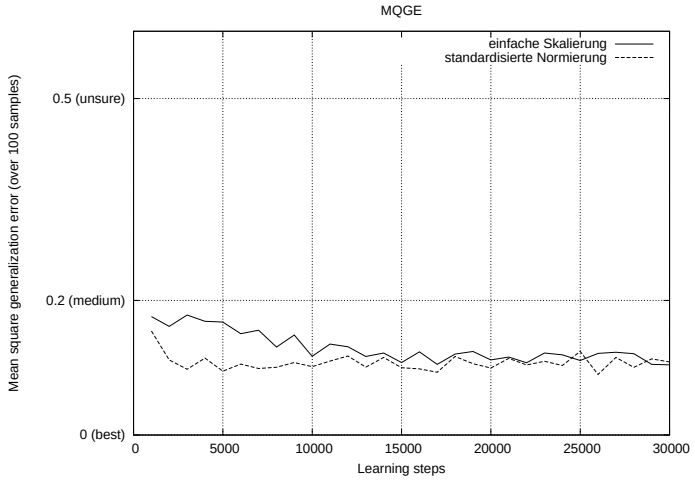
5.1 Einfluß der Normierung

Um den Einfluß der Normierung der EEG-Daten auf die Lern- und Generalisierungsqualität zu charakterisieren, wurde exemplarisch die einfache Skalierung nach Gleichung 4.2 auf Seite 21 und die standardisierte Normierung nach Gleichung 4.6 auf Seite 22 getestet.

Die Normierung nach Gleichung 4.6 auf Seite 22 benötigt ungefähr die 10fache Zeit von 4.2. Dabei erweist sich, daß bei Verwendung von 4.6 die Lern- (MQLE) und Generalisierungskurve (MQGE) des ANN (mit Präsentation der unbehandelten EEG-Daten) deutlich schneller abfällt, als bei 4.6 (s. h. Abbildung 5.1 auf der nächsten Seite).



(a) MQLE



(b) MQGE

Abbildung 5.1: Einfluß der Normierung auf MQLE und MQGE

5.2 ANN mit Rohdaten

Ein neuronales Netz sollte in der Lage sein aus der Gesamtheit der EEG-Daten diejenigen Merkmale zu extrahieren, die eine Unterscheidung in Artefakt und Nichtartefakt möglich machen. Um dem ANN nicht von vornherein Informationen vorzuenthalten wird die folgende Struktur gewählt. Wie in Abbildung 5.2 ersichtlich, wird Musterlernen mit einem gewöhnlichen *feedforward*-Netz benutzt. Dabei wird aus einer Zeitreihe jeweils zufällig ein Zeitfenster der Größe n herausgenommen und dem Netz mit der gewünschten Ausgabe präsentiert. Durch den Verzicht auf rekurrente Netze, wie Elman- oder Jordan-Netzwerke, ergeben sich zwei Vorteile. Erstens kann auf bekannte Netztypen (MLP und Lernverfahren, wie Backpropagation und damit verbundene Erfahrungen, zweitens auf Bibliotheken wie die FANN-Library zurückgegriffen und potentielle Fehlerquellen damit verringert werden. Eine Bestätigung der Wahl des ANN als MLP mit Backpropagation findet man u. a. in Bogacz u. a. (1999).

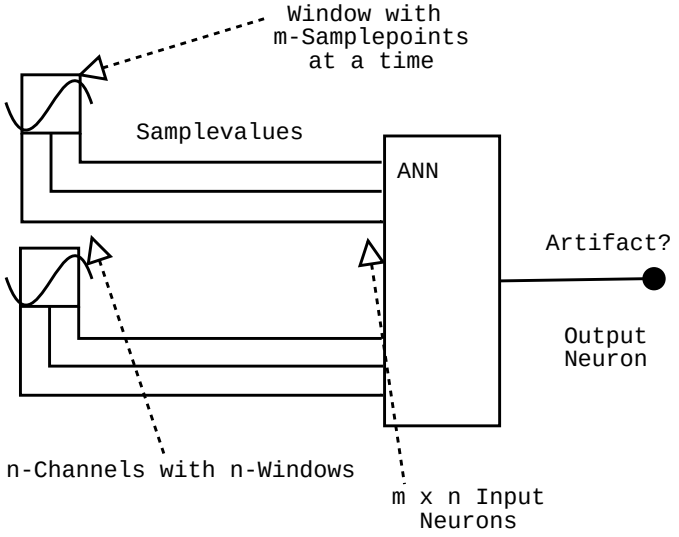


Abbildung 5.2: ANN mit unbehandelten EEG-Daten

Nach ersten Vortests (siehe 4.4 auf Seite 24) konnte gezeigt werden, daß dieser naive Ansatz in Grenzen funktioniert (Abbildungen A.1 bis A.2 auf Seiten ii–iii). Durch das Mapping der EEG-Kanäle auf das Internationale 10-20-System werden nur $c = 23$ Kanäle benutzt.

Die Festlegung der Fenstergröße basiert auf Erfahrungen der wissenschaftlichen Mitarbeiter des Max-Planck-Institutes. Als praktikabel erweist sich eine Fenstergröße von $s = 100 \ldots 300$ Samplepunkten. Diese Größe ist das Optimum zwischen den Anforderungen der möglichst geringen Komplexität des ANN auf der

einen, und möglichst sicheren Entscheidbarkeit auf der anderen Seite. Die Wahl auf $s = 256$ Samplepunkte entspricht dieser Optimalitätsforderung und wird desweiteren verwendet.

Für das vorliegende ANN ergibt sich somit eine Gesamtzahl von $cs = 256 * 23 = 5888$ an Eingangsneuronen.

Das ANN besteht aus drei Schichten, einer Eingangsschicht mit 5888 Eingangsneuronen, einer verdeckten Schicht und einer Ausgangsschicht mit nur einem Neuron. Nach verschiedenen Tests hat sich die Zahl von 50 Neuronen in der verdeckten Schicht bewährt, so daß das Netz pro Lernschritt bis zu $(5888 + 1) * 50 + (50 + 1) * 1 = 294501$ Verbindungen updaten muß.

5.3 ANN mit Rohdaten, downsampled

Im Wesentlichen entspricht der folgende Aufbau (Abbildung 5.3) dem von 5.2 auf Seite 30. Um das Lernen zu beschleunigen und die Komplexität des ANN zu verringern werden die EEG-Daten in ihrer zeitlichen Auflösung heruntergerechnet. Diese Vorgehensweise ist unkritisch, da die Nutzdaten, als auch typische Artefak-

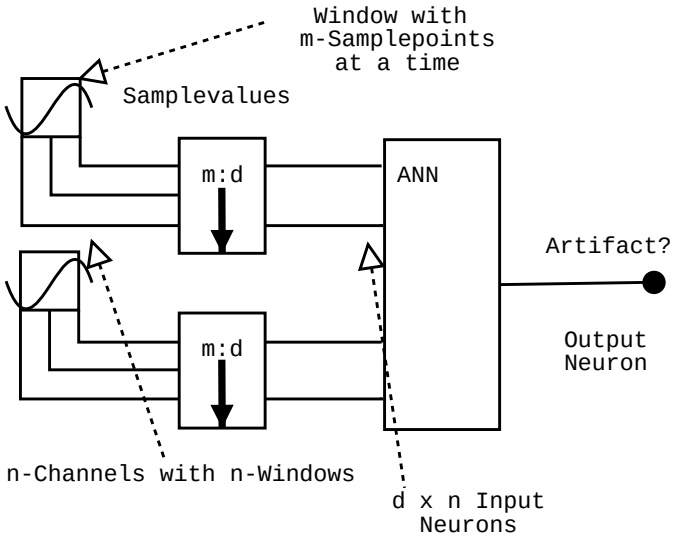


Abbildung 5.3: ANN mit heruntergesampelten EEG-Daten

te deutlich im unteren Frequenzbereich liegen (5-50 Hz). Die EEG-Aufzeichnung erfolgt üblicherweise mit einer Abtastfrequenz von 250 Hz, so daß eine Downsamplingrate von 1:4 oder 1:8 möglich ist[1].

Das Downsampling erfolgt beim 1:m durch Mittelwertsbildung der acht nebeneinander liegenden Werte:

$$x_{j\frac{1}{m}} = \frac{\sum_{i=1}^{m} x_{i+mj}}{m} \tag{5.1}$$

.

Durch das Downsampling ergibt sich eine Reduktion der Eingangsneuronen um den Faktor m, mit $m = 8$ zu $cs = \frac{256}{8} * 23 = 736$ Eingangsneuronen, und damit zu $(736 + 1) * 50 + (50 + 1) * 1 = 36901$ Verbindungen.

[1] Bei einem Downsampling von 1:8 würde die Abtastfrequenz 32 Hz betragen und damit Frequenzen bis 16 Hz nach Shannon-Theorem erfassen. Allerdings müssen hier die Frequenzen nicht rekonstruiert werden, so daß auch Artefakte mit höherer Grundfrequenz erfaßt werden. Die maximale Frequenz von regulären 'Gehirnwellen' liegt aber bei maximal 16-20 Hz.

5.4 ANN mit Rohdaten und summierten Elektroden

Eine andere Möglichkeit die Komplexität des Netzes zu verringern und die Lernzeit zu verkürzen besteht darin, die räumliche Verteilung der EEG-Daten zu vernachlässigen. Dies führt zur Summierung eines Elektrodensamples über alle Kanäle, wie in Abbildung 5.4 dargestellt. Diese Variante sollte im Gegensatz zu

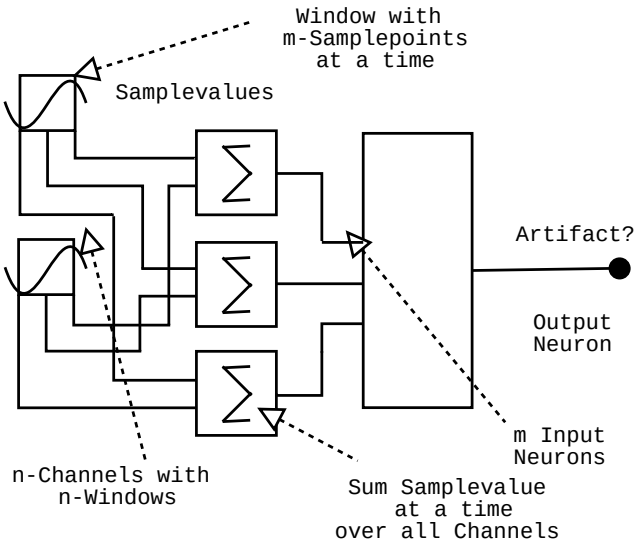


Abbildung 5.4: ANN mit EEG-Daten und summierten Elektroden

den vorigen Varianten Elektrodenartefakte erkennen können, da dem ANN jetzt die Möglichkeit der Generalisierung von einem Kanal auf die anderen offensteht. Kritisch an diesem Aufbau könnten die Konsequenzen aus dem zentralen Grenzwerttheorem sein, da dieses besagt, daß die Summe voneinander unabhängiger Zufallsprozesse[2] eine Gaußverteilung bildet.

Die Anzahl der Eingangsneuronen verringert sich dann zu 256, so daß das ANN pro Lernschritt bis zu $(256 + 1) * 50 + (50 + 1) * 1 = 12091$ Verbindungen berechnen muß.

Das gute Abschneiden dieser Variante bei den Vortests ist allerdings nicht auf die Wirksamkeit der Elektrodensummation, sondern vielmehr auf die Art der generierten Testdaten zurückzuführen, da diese, wie in 4.4 auf Seite 24 erläutert, über alle Kanäle die gleiche Funktionswerte nutzen. Nichtsdestotrotz wird diese Variante verfolgt, da sich die Komplexität des ANN damit drastisch verringert.

[2] Die EEG-Daten sind zueinander stark korreliert. Allerdings kann das mit den Artefakten anders aussehen.

5.5 ANN mit sFFT-Koeffizienten

In der klassischen EEG-Signalverarbeitung (Novák u. a., 2001) wird oft mit fouriertransformierten EEG-Daten gearbeitet. Mit der folgenden Variante wird dem Rechnung getragen, indem auf die Daten pro Fenster eine FFT angewandt wird. Trotzdem sich die Komplexität des ANN hierdurch nicht reduziert, könnten ff. Punkte sich als vorteilhaft erweisen. So sind die normalen 'Gehirnwellen' meist in einem Bereich von 3-16 Hz angesiedelt. Elektrodendrifts liegen in Bereichen von unter 1 Hz, Störspannungen meist im Bereich von 50 Hz, Muskelartefakte sind meist hochfrequent. Augenbewegungen zeigen bestimmte spektrale Merkmale. Wegen diesen Punkten könnte eine Betrachtung der Daten im Fourierraum für das Netz geeigneter sein. Auch tritt das Rauschen der EEG-Daten im Zeitbereich hier deutlicher in den Hintergrund.

Die Zahl *cs* der Eingangsneuronen beträgt $cs = 5888$, die Zahl der Verbindungen ist demnach $(5888 + 1) * 50 + (50 + 1) * 1 = 294501$.

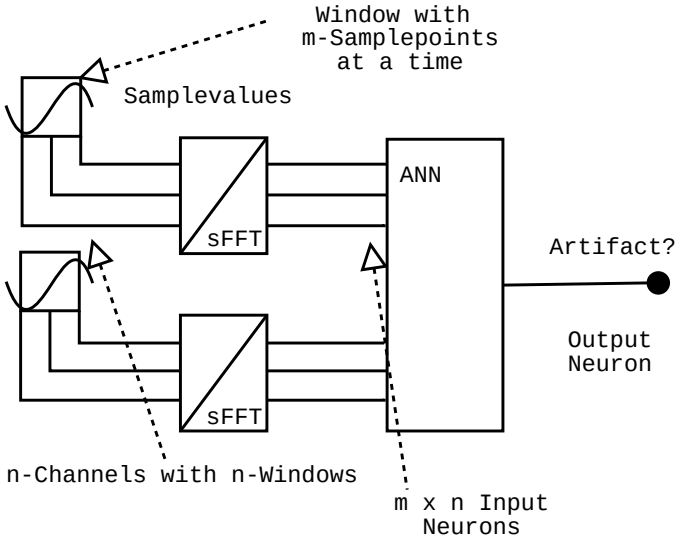


Abbildung 5.5: ANN mit Gabor-transformierten EEG-Daten

Die Implementierung basiert auf der Version 2 der *Fastest Fourier Transform in the West*-Bibliothek (FFTW). Die FFTW stellt Routinen für die schnelle Fouriertransformation unter C zur Verfügung.

5.6 ANN mit statistischen Parametern

Durch die Arbeit von Ksiezyk u. a. (1998) angeregt, wird nach Transformationen gesucht, die sowohl die Komplexität des ANN reduzieren, das Rauschen in den Daten vermindern, als auch die Eigenschaften der EEG-Daten gut genug für eine signifikante Unterscheidung von Artefakten beschreiben. Durch Cooper u. a. (1984) und die Hintergründe zur ICA (Hennig (1998), Makeig u. a. (1996) Jung u. a. (1998)) motiviert, sollten die statistischen Merkmale Mittelwert, Bereich, Standardabweichung, Schiefe und Wölbung hinreichend die EEG-Daten beschreiben können (siehe auch 2.3.1 bis 2.3.7 auf Seiten 14–16).

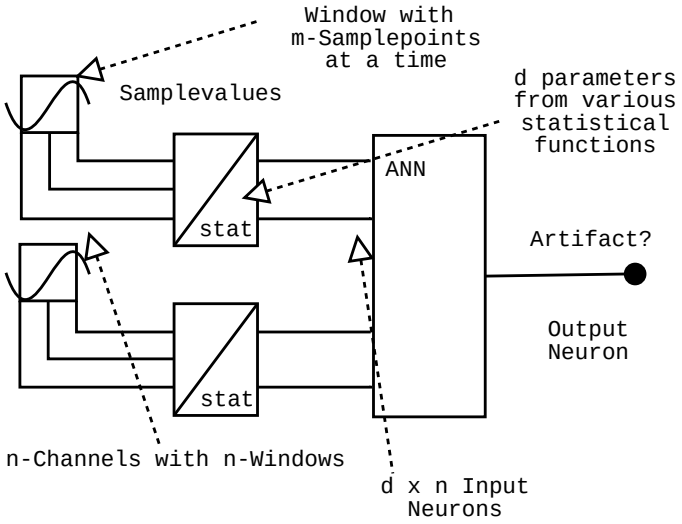


Abbildung 5.6: ANN mit statistischer Beschreibung der EEG-Daten

Um ein Maß für die Lokalität der Abweichungen (zum Beispiel vom Mittelwert) zu haben und damit in Grenzen dem ANN Trends sichtbar zu machen, werden die statistischen Parameter für zwei Fenstergrößen berechnet, für $x_{1..n}$ und für $x_{1..\frac{n}{2}}$.

Dadurch ergeben sich pro Kanal 10 Kennwerte, so daß das ANN über $23 * 10 = 230$ Eingangsneuronen und $(230 + 1) * 50 + (50 + 1) * 1 = 11601$ Verbindungen verfügt.

5.7 Implementationsdetails

5.7.1 Zufallsfunktion

Die Wahl der Zufallsfunktion ist wichtig für die möglichst zufällige Auswahl der Trainingsdaten für das ANN. Um sicherzustellen, daß die Zufallsfunktion gute Eigenschaften aufweist, wurde statt der direkten Verwendung der *libc*-Funktion `rand()` eine eigene Implementierung verwendet:

```
/* linear kongruent modulo pseudo-random generator */
unsigned int mrandomize(unsigned int max) {
    int i;
    i=(unsigned long)rand()*934567+rand();
    return((unsigned int) (abs(i)%max));
}
```

Listing 5.1: Pseudozufallsgenerator

Dabei wurde auf einen Pseudozufallsgenerator nach dem Prinzip der linearen Kongruenz gesetzt:

$$f(x_r, x_s) = |x_r * p + x_s| \mod m \tag{5.2}$$

Dabei sind x_r und x_s Zahlen einer Pseudozufallsfunktion, p ist eine konstante Primzahl, m gibt die obere Grenze des gewünschten Zufallszahlenbereichs an.

5.7.2 Sicherstellung Artefakt/Nichtartefakt-Verhältnis

Entscheidend für die Lernqualität erweist sich die Einhaltung des Artefakt/Nichtartefakt-Verhältnisses bei der Präsentation der Trainingsdaten. Mit dem Listing 5.3 auf Seite 39 wird ein Verhältnis von 1:1 eingehalten. Als artefaktbehaftet wurde das Zeitfenster betrachtet, welches mehr als 20% als Artefakt gekennzeichnete Samplepunkte enthält.

Dieser Wert wurde empirisch ermittelt und entspricht annähernd der halben Zeitdauer eines typischen EEG-Artefakts von $t = 0,5$s. Nach der Präsentation eines nach dieser Definition artefaktbehafteten Zeitfensters wird dem ANN dann ein nicht artefaktbehaftetes Zeitfenster als Eingabe präsentiert.

Durch die wechselnde Präsentation von artefaktfreien und -behafteten Daten wird vermieden, daß das Netz nicht lernt. Bei einem typischen Experiment entfallen

rund 10% der Samplepunkte auf den Artefaktbereich einer EEG-Aufzeichnung. Wählt man die Eingangsdaten zufällig, aber ohne obige Sicherstellung des Verhältnisses von 1:1, so stagnieren ausnahmslos alle ANN-Implementationen und lernen nur das Verhältnis zwischen Anzahl der Artefaktsamples und Nichtartefaktsamples, nicht aber die Generalisierung der Unterscheidung derselben.

5.7.3 Reduktion der Generalisierungsfehler bei Artefaktbestimmung

Trotz des guten Generalisierungsverhaltens der ANNs treten bei nicht gelernten Eingabedaten Generalisierungsfehler auf. Um die Häufigkeit dieser Fehler zu reduzieren, werden im Wesentlichen drei Verfahren eingesetzt.

Die ersten beiden Verfahren arbeiten eng zusammen. So wird auf der Trainingsseite (`eatr`) statt einer harten Klassifizierung des Zeitfensters, bei der ein Zeitfenster als artefaktbehaftet gilt, wenn es mindestens einen als Artefakt markierten Samplepunkt enthält, eine weiche Klassifizierung eingeführt. Diese wird durch einfache Bestimmung der Häufigkeit artefaktmarkierter Samplepunkte im jeweils aktuellen Zeitfenster vorgenommen, siehe auch Listing 5.3 auf Seite 39. Diese weiche Klassifizierung sorgt dafür, daß das ANN mit Daten aus Artefaktrandbereichen besser klarkommt. Diese Randbereiche sind kritisch, da die erfassten Trainingsdaten manuell markiert wurden, und harte Grenzen zu Artefakten im EEG-Bild in der Regel nicht existieren.

Bei einem trainierten Netz bekommt man am Ausgangsneuron damit die Häufigkeit, wieviele Samples des aktuellen Zeitfensters vom ANN als artefaktbehaftet eingeschätzt werden. Da bei der Nutzung des trainierten ANNs das Zeit-

```
    /* calculate the average artifactness of actual window
       */
    for (taps=0; taps<MAXTAPS; taps++) { /* for every
       samplepoint in actual window */
        tmpoutput[0]+=net.infiles[selected_infile].rej_
            buffer[k+taps]; /* sum binary state, 0 - means
            nonartifact, 1 - means artifact */
    }
5   tmpoutput[0]/=(float)MAXTAPS; /* average=sum/windowsize
        */
```

Listing 5.2: Ausschnitt, Bestimmung Grad der Artefaktbehaftetheit

fenster gleitend über die Eingangsdaten geschoben wird, kann man eine Verbesserung der Generalisierungsqualität durch Mittelung der Netzvorhersagen (am Ausgangsneuron) erzielen. Dieses zweite Verfahren ist im Listing 5.4 auf der nächsten Seite dargestellt. Dabei wird eine rekursive Implementierung der Mittelwertsbildung über k-letzte[3] Werte benutzt:

$$\widehat{x_{i+k}} = \frac{\widehat{x_{i+k-1}}k - x_{i-1} + x_i}{k} \tag{5.3}$$

Dieses Verfahren hat den Nachteil, daß es k-Samples zum Einschwingen benötigt, ein günstiger Wert ist $k = 10$.

Um die Artefakte zu kennzeichnen wird ein Schwellenwert eingeführt. Für spätere Implementierungen sollte das Netz nicht hart zwischen Artefakt und Nichtartefakt unterscheiden müssen, sondern besser eine Markierung *Artefakt*, *Nichtartefakt* und *unsicher* erlauben, um die Qualität der Markierungen besser einschätzen zu können.

Das dritte Verfahren nach Listing 5.5 auf Seite 40 stellt sicher, daß markierte und nichtmarkierte Abschnitte mindestens 30 Samples lang sind.

[3] in Listing AVGWINDOW

```
/* next line secure artifacts will be presented at same
    ratio as non-artifacts */
if (((tmpoutput[0]>0.2f)&&(artifactness==0))|| /* the
   value should be the half of a typical artifact. If
   the typical artifact-length==0,5 sec and
   Samplefrequency = 250 Hz and the MAXTAPS=250, then
   the value shuld be about 0.25 */
    ((tmpoutput[0]<=0.2f)&&(artifactness==1))){
    artifactness=1 ^ artifactness;
    testoutput=fann_run(ann, cnvinput);
    tmperror=0.0f;
    tmperror+= ((tmpoutput[0]-testoutput[0])*(tmpoutput
       [0]- testoutput[0])); /* generalization */
    generalization_error+=(tmperror);
    fann_train(ann, cnvinput, tmpoutput); /* train the
       network with the data */
} else { /* do nothing */}
```

Listing 5.3: Ausschnitt, Sicherstellung Artefakt-/non-Artefakt Verhältnis

```
/* window[] is an array of size AVGWINDOW */
windowsum=windowsum-window[(windowindex+1)%AVGWINDOW]+
   tmpoutput[0]; /* tmpoutput[0] is ANN-output */
window[windowindex%AVGWINDOW]= tmpoutput[0];
if (windowsum >= AVGTHRESHOLD*(AVGWINDOW-1)) {
    /* ... is an artifact */
} else /* ... not an artifact */
```

Listing 5.4: Mittelung Netzvorhersage

```
    if (tmpoutput[0]>=0.50f)
        artifact_count++;
    else nonartifact_count++;
    if ((artifact_count > 10)&& (was_an_artifact==0)){
        nonartifact_count=0;
        was_an_artifact=1;
        /* ... it is an artifact */
    }
    else if ((nonartifact_count > 10)&& (was_an_artifact==
       1)){
        artifact_count=0;
        was_an_artifact=0;
        /* ... it is not an artifact */
    }
```

Listing 5.5: Filterung Netzvorhersage

6 Diskussion

quam temere in nosmet legem
sancimus iniquam.

(Horaz, SATURAE)

6.1 Zusammenfassung der untersuchten ANN

In Tabelle 6.1 sind die Eigenschaften der untersuchten ANN kurz zusammengefaßt. Deutlich erkennt man, daß die Lernzeit von der Komplexität des neuronalen Netzes bestimmt ist. Der Einfluß der Normierung auf alle Netzwerktypen wurde mit untersucht. So zeigt sich, daß die einfache Normierung[1] nach Gleichung 4.2 auf Seite 21 in der Regel genügt. Einzig bei den Netzwerken mit unbehandelter EEG-Eingabe und mit Fouriertransformation erweist sich der Vorteil der Normierung nach 4.6 auf Seite 22[2], so daß im folgenden nur noch die Ergebnisse der einfachen Normalisierung Berücksichtigung finden.

Tabelle 6.1: Vergleich der untersuchten ANN

Netztyp	Anzahl Neuronen (Eingangs-neuronen)	Anzahl Verbind-ungen	Lerndauer in s	$MQGE_{400}$ ($MQGE_s$)
ANN, unbehandelt	5941 (5888)	294501	16083	0,134 (0,123)
ANN, downgesampelt	789 (736)	36901	3339	0,146 (0,129)
ANN, summierte Elektroden	309 (256)	12091	1949	0,127 (0,134)
ANN, FFT	5941 (5888)	294501	17417	0,131 (0,101)
ANN, statistische Kenndaten	283 (230)	11601	2862	0,083 (0,089)

Lernrate $\nu = 0,0001$, 10000000 Lernschritte, AMD K6-2, 500MHz mit 512 MB

[1] In der Tabelle mit $MQGE_{400}$ ausgewiesen
[2] In Tabelle mit $MQGE_s$ markiert

Alle Netzwerkvarianten werden mit zwei *cnt*-Files in 10000000 Lernschritten trainiert, die sorgfältig von Hand markiert wurden. Sie enthalten größtenteils Augenartefakte, vereinzelt aber auch Elektroden- und Muskelartefakte. Die beiden Files mit insgesamt 1017584 Samplepunkte weisen ein Artefaktverhältnis von 0,075 auf. Die Lernrate beträgt $\nu = 0,0001$. Die Ausführungszeit ist für einen AMD K6-2, 500MHz mit 512 MB RAM angegeben. Der in der Tabelle beschriebene MQGE entspricht dem niedrigsten der letzten 10 Werte[3].

In den Abbildungen A.3 auf Seite iv und A.4 auf Seite v zeigt sich das Lern- un Generalisierungsverhalten in detaillierter Form. So kann man am MQGE ablesen, daß für vorgenannte Zahl an Trainingssamples von knapp einer Million bei dem Netzwerk mit statistischer Vorbehandlung der EEG-Daten durchaus die Zahl von zwei Millionen Lernschritten ausreichend ist. Aus diesem Grund werden in den oben genannten Abbildungen der besseren Übersicht halber die Verläufe der MQLE und MQGE bis 2 Millionen Lernschritten dargestellt.

Interessanterweise hat laut Abbildungen A.3(a) und A.3(b) das Downsampling keinen Einfluß auf die Lernkurve. Deutlich erkennt man, wie das Netz sich an die Problemstellung anpaßt. Einzig in A.3(e) erkennt man, daß sich nach circa 700000 Lernschritten das ANN auf das Problem eingestellt hat. Die Aussage, daß die anderen Netze nicht lernen, kann daraus nicht abgeleitet werden, da diese Netze deutlich komplexer aufgebaut sind und einer längeren Trainingsphase[4] und gegebenenfalls einer angepaßteren Lernrate bedürfen.

In Abbildung A.4(e) auf Seite v erkennt man, daß die Netzvariante mit statistischen Parametern einen deutlichen Abfall in der Generalisierungskurve und damit einen schnellen Fortschritt beim Training bewirkt.

Aufgrund des guten Abschneidens mit den Daten des Vortests und der guten Eigenschaften, wie:

- geringe Komplexität
- schnelles Training
- geringer Aufwand bei Normierung
- niedrigster Generalisierungsfehler

bei der Verwendung von realen EEG-Daten wird desweiteren das ANN mit statistischer Vorbehandlung als zu bevorzugender Kandidat für die Artefakterkennung empfohlen.

[3] bezogen auf die Einträge der *mqge*-Dateien, der MQGE wird aller 1000 Lernschritte gespeichert. So wird beispielsweise der MQGE in der Tabelle aus den niedrigsten MQGE-Werten zwischen dem 9990000-sten und 10000000-sten ausgewählt.

[4] In der Tat benötigen die anderen Varianten in der Größenordnung 10^9 Lernschritte

6.2 Einfluß der Netzparameter auf die Lern- und Generalisierungsqualität

6.2.1 Einfluß der Neuronenzahl in der verdeckten Schicht

Die Zahl der Neuronen in der verdeckten Schicht ist im vorliegenden Programm mit 50 festgelegt, da vermutet werden kann, daß dies für die unterschiedlichen Artefakttypen und Erscheinungsformen ein Optimum zwischen Lerngeschwindigkeit auf der einen und Generalisierungsfähigkeit auf der anderen Seite darstellt. Um den Einfluß dieser Neuronenzahl auf die Generalisierungs- und Lernfähigkeit festzustellen, werden die MQLE- und MQGE-Kurven des ANNs mit statistischer Vorverarbeitung gemäß Abbildungen A.5 bis A.6 auf Seiten vi–vii verglichen. Wie man erkennen kann, variieren die Kurven unmerklich. Im MQGE mit 5 Neuronen kann man eine höhere Varianz der Fehlerpunkte erkennen. Auch der Bruch der Lernkurve fängt bei der Variante mit 5 oder 25 Neuronen später an, was auf unzureichende Neuronen hindeuten kann. Trotz allem führen alle Netze zu einem gleichermaßen gelagerten Generalisierungsfehler nach zwei Millionen Lernschritten. Auch wenn demnach eine Implementierung mit fünf versteckten Neuronen möglich und wegen der geringeren Netzkomplexität dementsprechend auch schneller wäre, wird die Anzahl von 50 beibehalten. Einerseits können im Rahmen dieser Arbeit nicht alle Artefakttypen untersucht werden und das Netz würde so noch Reserven für intensiviertes Training bieten, andererseits bewirkt diese Zahl auch noch kein Overlearning des ANN.

6.2.2 Einfluß des Lernfaktors

Wie in Abbildungen A.7 bis A.8 auf Seiten viii–ix ersichtlich, hat der Lernfaktor wie zu erwarten einen starken Einfluß nicht nur auf die Dauer der Trainingsphase, sondern auch auf die Qualität der Generalisierung. So zeigt sich, daß Lernfaktor von $\nu = 0,00001$ ausreicht. Lernfaktoren kleiner $\nu = 0,00001$ bewirken keine Verbesserung.

6.3 Qualitative Einschätzung

Das ANN mit statistischer Vorverarbeitung der EEG-Daten zeigt anhand der Lern- und Generalisierungskurven ein gutes Verhalten, so daß für die Artefaktklassifizierung prinzipiell künstliche Neuronale Netze einsetzbar sind. Um diese Einschätzung zu untermauern, wird die von einem trainierten Netz erzeugte

rej-Datei und das dazugehörige *eeg*- bzw. *cnt*-File manuell auf *false positives*[5] u. a. Fehlklassifizierungen kontrolliert. Das Programm benötigte 954 s für die Erstellung eines *rej*-Files für eine dem Netz nicht zum Training präsentierte *eeg*-Datei mit 4115822 Samples auf einem AMD K6-2 500 MHz mit 512 MB RAM. Das Netz wurde mit 513406 Samples in 50000000 Lernschritten bei einer Lernrate von $\nu = 0,0001$ trainiert und weißt 50 verdeckte Neuronen auf.

Dabei wurden die einzeln auftretenden Augenartefakte korrekt klassifiziert. Gleiches galt für große Elektrodenstörungen, die typisch während des Versuchsbeginns und -endes auftreten. Muskelartefakte wurden nicht immer korrekt eingeordnet. Die Ursache dafür kann in dem geringen Vorkommen in den Trainingsdaten liegen. Nicht erkannt wurden Elektrodendrifts. Da diese meist nur auf einem Channel beschränkt sind und in den Trainingsdaten nicht für alle Kanäle präsentiert wurden, kann das ANN diese nicht auf andere Kanäle verallgemeinern. Auch generiert das Netz ab und an Zurückweisungsepochen von 5 bis 20 Samples. Diese *false positives* können aber durch Verbesserung der Algorithmen in 5.7.3 auf Seite 37 vermindert werden.

Zusammenfassend können neuronale Netze für die Artefakterkennung zwar eingesetzt werden, verlangen aber nach qualitativ hochwertig klassifizierten Trainingsdaten. Für die schnelle Klassifizierung von *eye-blinks*-Artefakten kann das vorliegende Netz problemlos eingesetzt werden.

6.4 Probleme Implementation

6.4.1 Dateiverwaltung und Datenpufferung

Die aktuelle Implementierung kann zur Zeit je nach Arbeitsspeicher zwischen vier und zehn *eeg*-Dateien für die Trainingsphase verwalten. Diese Beschränkung liegt darin begründet, daß für die zufällige Auswahl der Trainingssamples, sowohl auf die Daten des *eeg*-, als auch auf die Daten des dazugehörigen *rej*-Files zurückgegriffen werden muß. Um die Komplexität und damit die Fehleranfälligkeit des `eatr`-Programms niedrig zu halten, werden diese Daten komplett im Speicher gepuffert.

[5] Gemeint sind damit die fälschlicherweise vom ANN als Artefakt erkannten Muster

6.4.2 Anzahl der Lernschritte und Wahl der Lernrate

Die Auswahl der Lernschritte und der Lernrate für das Training sind eng miteinander gekoppelt. So ist bei der Anwendung von `eatr` zu beachten, daß die Trainingsmenge in Relation zur Sampleanzahl der verwendeten *cnt*- bzw. *eeg*-Dateien liegt. Die richtige Wahl der Lernrate, sowie der Trainingsmenge kann sinnvoll nur über die von `eatr` über die Option *-o <filename >* bereitgestellten *mqle*- bzw. *mqge*-Dateien überprüft werden. Die beiden Dateien stellen den Verlauf des MQGE und des MQLE im menschenlesbaren *Gnuplot*-Format[6] dar.

6.4.3 Geschwindigkeit Lernprozess

Die Geschwindigkeit des Lernprozesses wird im wesentlichen von der Berechnung der statistischen Parameter bestimmt. Die Ursache liegt in der Behandlung der Daten. Diese erfolgt nach der zufälligen Auswahl eines Samplefensters und vor der ANN-Präsentation, so daß u. U. Werte mehrmals in die Berechnung einbezogen werden, wenn sich diese Eingangsfenster überlappen.

6.4.4 Geschwindigkeit Arbeitsprozess

Auch hier gilt, daß die Arbeitsgeschwindigkeit durch die Ermittlung der statistischen Kennwerte bestimmt ist (s. h. auch 6.4.3).

6.4.5 Channel-Mapping

Das Channel-Mapping in der vorliegenden Implementierung sollte mit den Problemen aufräumen, die durch die unterschiedlichen Elektrodenkonfigurationen entstehen.

Leider erweist sich die Umsetzung auf das Internationale 10-20-System als suboptimal. Vielfach wurden in den Versuchen EEG-Aufzeichnungen mit Elektroden aufgenommen, die dort nicht existieren. Noch schlimmer wirkt sich das Fehlen bestimmter Elektrodenregionen aus. So existieren *cnt*-Files die keine Kanäle in der Region des *lobus frontalis* enthalten. Ein Mapping führt bei solchen Konfigurationen zu unsinnigen Ergebnissen.

[6] Textdatei, die erste Spalte enthält den Lernschritt, die zweite, durch Leerzeichen getrennt, den MQLE bzw. MQGE.

Auch wird durch das Programm bisher keine Auswertung der Ableitung vorgenommen. Für die korrekte Arbeit des ANN ist es notwendig, daß die Trainingsbeispiele, als auch die Anwendungsdaten die gleichen Ableitungssysteme, z. B. gegenüber der *nasion*-Elektrode, *A1* oder *A2* verwenden.

6.4.6 Alternativ untersuchte ANN-Struktur

Eine Alternative zu den in 5.2 bis 5.6 auf Seiten 30–35 vorgestellten ANN-Typen, bzw. zu der Grundstruktur (4.5 auf Seite 25) ist die Verwendung folgender Struktur.

Statt einem Ausgangsneuron, daß den Artefaktanteil im Eingangsfenster repräsentiert, wird parallel zu diesem dem ANN ein Ausgangsfenster mit den Originalmarkierungen gegenübergestellt.

Leider zeigt sich schon mit den Vortests, daß die Generalisierungsfähigkeit deutlich schlechter, die Netzkomplexität und damit die Lerngeschwindigkeit höher ist. Das Problem scheint bei diesem Typus wieder die genaue, besser ungenaue Markierung der Artefaktgrenzen zu sein, die das Netz mehr durcheinander bringen als ihm helfen. Nichtsdestotrotz ist dieser Typ besser im Verhalten, als der Vorläufer der jetzt verwendeten Variante mit einem Ausgangsneuron. Bei diesem galt ein Fenster schon als artefaktbehaftet, wenn auch nur ein Samplepunkt des Eingangsfensters artefaktbehaftet war. Siehe hierzu auch 5.7.3 auf Seite 37.

7 Empfehlungen

> ,,Zweiundvierzig!" kreischte Luunquoal los. ,,Ist das alles, nach siebeneinhalb Millionen Jahren Denkarbeit?"
>
> *(Douglas Adams, HITCHHIKER's GUIDE)*

7.1 Implementative Verbesserungen

7.1.1 Dateiverwaltung und Datenpufferung

Um die Anzahl der für das Training zu verwaltenden *eeg*-Dateien zu erhöhen und den Speicherbedarf zu senken, bieten sich zwei Varianten an. Auf der einen Seite könnte jedes Trainingssample über eine *SEEK*-Operation nach zufälliger Auswahl explizit von aus einer *eeg*-Datei[1] in den Arbeitsspeicher geladen werden. Allerdings sind diese Zugriffsoperationen teuer, so daß diese Variante nur mit zusätzlichen Caching-Strategien verfolgt werden sollte.

Ein anderer Ansatz, der eine solche Cachingstrategie verfolgt, wäre, statt wie bisher die Dateien komplett zu puffern, nur die zufällig ausgewählten Samples zu laden und zu puffern. Dazu würde in einem Schritt ein Array von Zufallswerten angelegt, im zweiten diesen Zufallswerten Samples der verschiedenen *eeg*-Files zugeordnet. Abschliessend wird eine Sortierung pro Datei vorgenommen und die entsprechenden Daten in einen Puffer eingelesen, der für das Training verwendet wird.

7.1.2 Erhöhung der Arbeits- und Trainingsgeschwindigkeit

Wie in 6.4.3 bis 6.4.4 auf Seite 45 dargestellt, ist der begrenzende Faktor nicht die Berechnung der Backpropagation, sondern die Berechnung der statistischen Kenndaten. Eine Erhöhung der Geschwindigkeit würde dann erfolgen, wenn die

[1] und einer *rej*-Datei

Konvertierung der Daten gleich in einem Schritt erledigt, dann ausgewählt und dem Netz präsentiert würde, anstatt wie bisher, in jedem Lernschritt die Berechnung durchzuführen.

Analog dazu verfährt das Programm `cntconv`. So wäre es durchaus denkbar, statt nur die Normierung, auch diesen Schritt der Vorverarbeitung durch `cntconv` vornehmen zu lassen.

Eine deutliche Erhöhung der Arbeits- und Trainingsgeschwindigkeit ergibt sich, wenn die Komplexität des ANNs reduziert würde. Dieser Vorteil begründet sich in der quadratischen Zunahme des Berechnungskomplexität der Backpropagation mit der Zahl der Eingänge.

7.2 Alternative ANN-Ansätze

Würde eine Klassifizierung der EEG-Daten pro Kanal und pro Artefakttyp vorliegen, so könnte das Netz in Teilkomponenten zerlegt werden. So wäre es denkbar, jeweils ein einfaches ANN[2] pro Kanal für die Artefaktvorhersage zu verwenden und ein Entscheidernetzwerk einzusetzen, daß für das Gesamtfile die Artefaktmarkierung vornimmt. Vorteile einer solchen Lösung wären, neben der Reduzierung der Komplexität, die eindeutigere Klassifikation, schnelleres und genaueres Training und die bessere Kontrolle der Ausgabe.

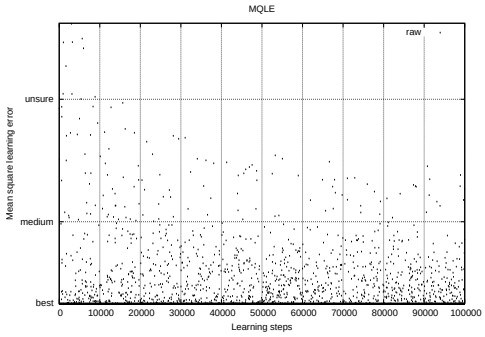
Die Wahl eines MLP mit Backpropagation erwies sich als problematisch. Eine intelligentere Implementierung könnte mit *cascaded correlation networks* arbeiten oder alternative Lernverfahren verwenden, die nicht so anfällig für Hängenbleiben in lokalen Minima sind.

Auf die Variante, einen Prädiktor für die EEG-Daten zu verwenden, wurde in 1.4 auf Seite 4 schon hingewiesen. Der Vorteil wäre sicherlich, daß man die EEG-Datensätze nicht mehr aufwendig von Hand für das Training klassifizieren muß, insbesondere eine Markierung auf Artefakttyp und Kanal würde wegfallen. Nein, man nimmt die *guten* Daten, die schon für die Versuchsauswertungen im Rahmen der ERP-Analyse verwendet werden und vergleicht die Vorhersage des Netzes mit den Realdaten. Weichen beide stärker voneinander ab, so gilt dieser Abschnitt als artefaktbehaftet. Ob diese Lösungsvariante praktikabel einsetzbar ist, sollte Gegenstand weiterführender Untersuchungen sein.

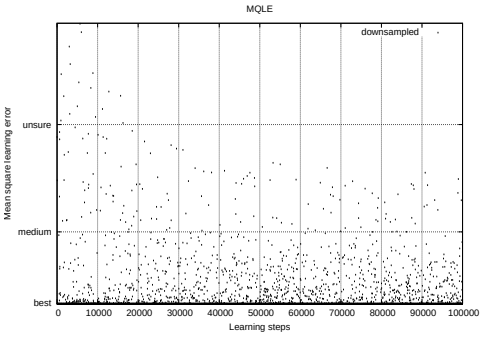
Eine andere Möglichkeit wäre die Verwendung von Kohonen-Maps. Mit diesen, oder auch der ICA könnte nach Kriterien gesucht werden, die typisch für artefaktbehaftete und artefaktfreie EEG-Samples sind.

[2] oder mehrere, auf bestimmte Artefakte spezialisierte

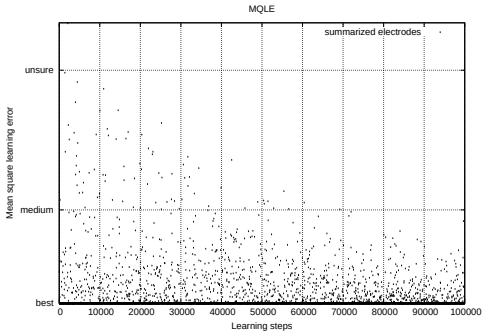
A Abbildungen



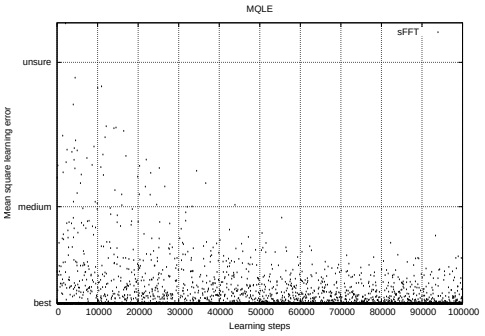
(a) ANN, roh



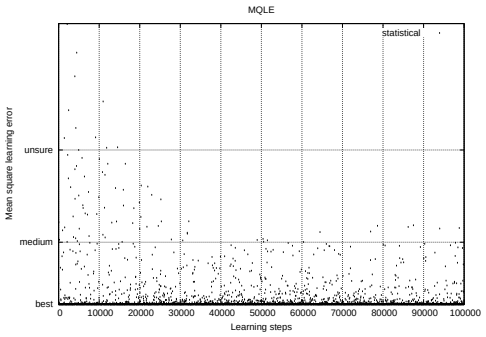
(b) ANN, downsample



(c) ANN, sum electrodes

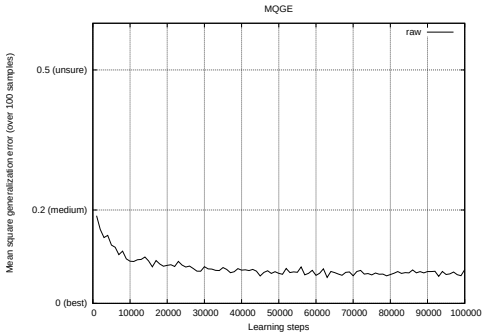


(d) ANN, sFFT

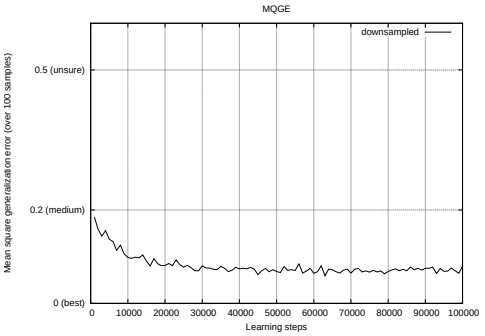


(e) ANN, statistical

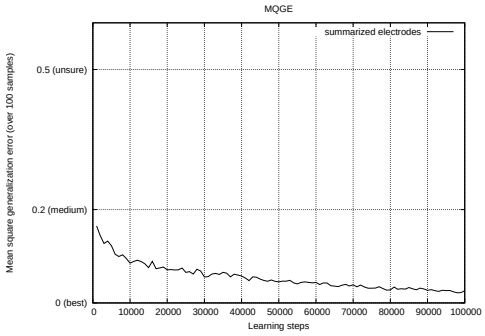
Abbildung A.1: MQLE, ANNs mit generierten Testdaten



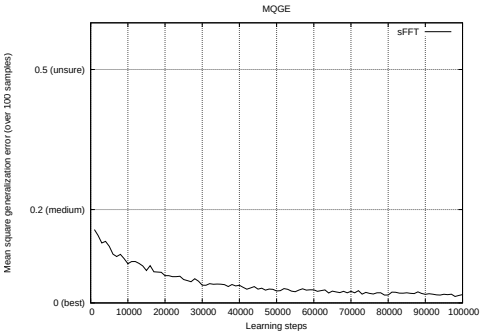
(a) ANN, roh



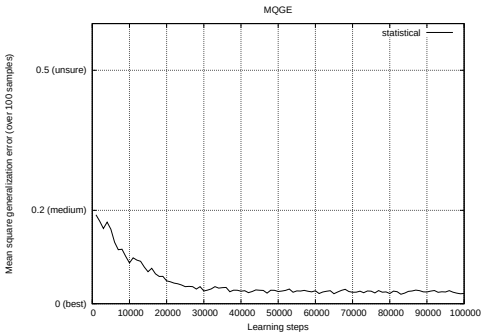
(b) ANN, downsample



(c) ANN, sum electrodes

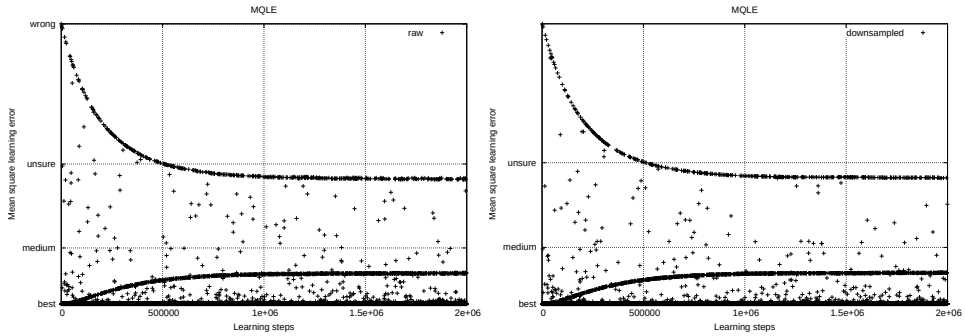


(d) ANN, sFFT



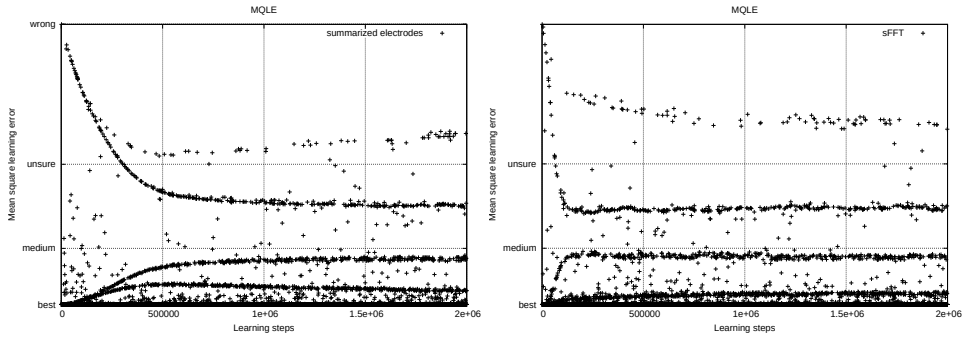
(e) ANN, statistical

Abbildung A.2: MQGE, ANNs mit generierten Testdaten



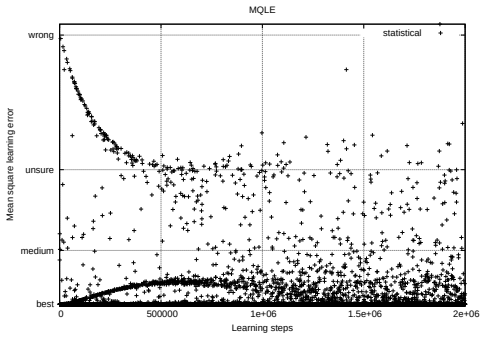
(a) ANN, roh

(b) ANN, downsample



(c) ANN, sum electrodes

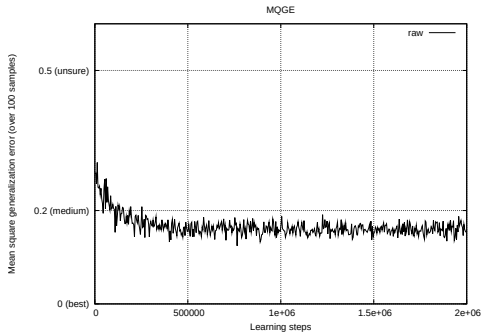
(d) ANN, sFFT



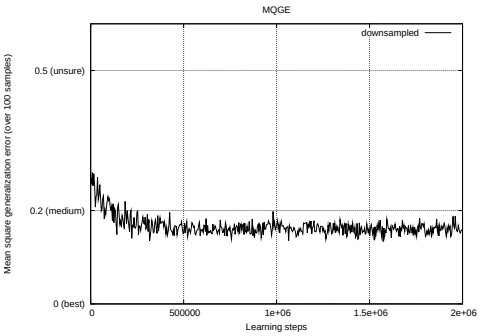
(e) ANN, statistical

Lernrate $\nu = 0,0001$, `vp40s.cnt` und `vp41s.cnt`, 10000000 Lernschritte, ersten 2000000 dargestellt, einfache Normierung

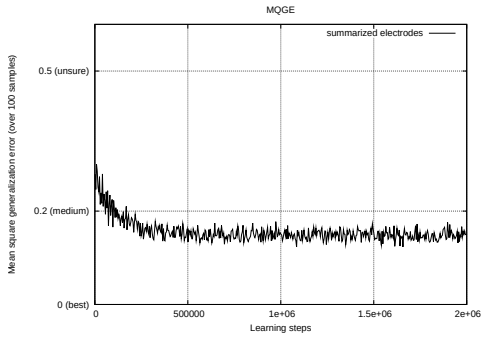
Abbildung A.3: MQLE, ANNs mit realen EEG-Daten



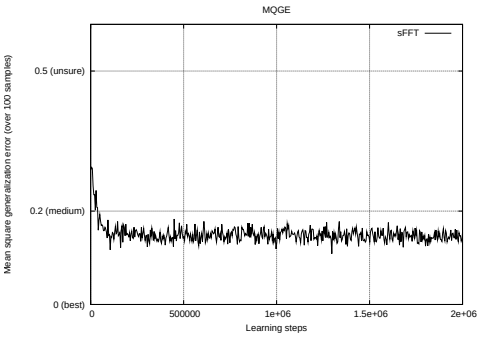
(a) ANN, roh



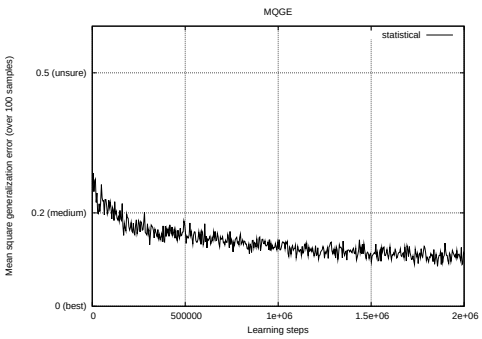
(b) ANN, downsample



(c) ANN, sum electrodes



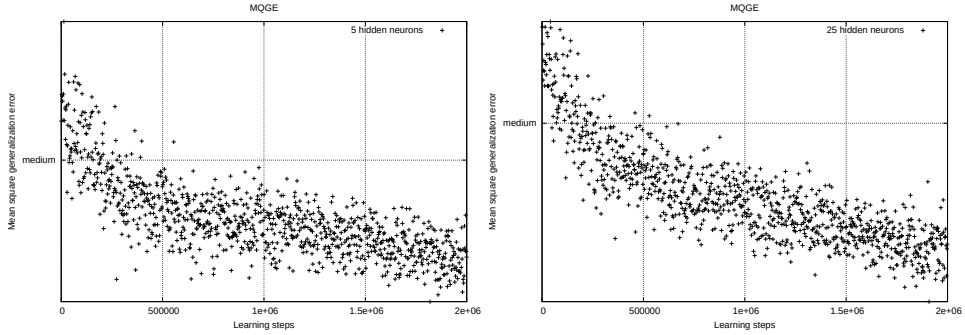
(d) ANN, sFFT



(e) ANN, statistical

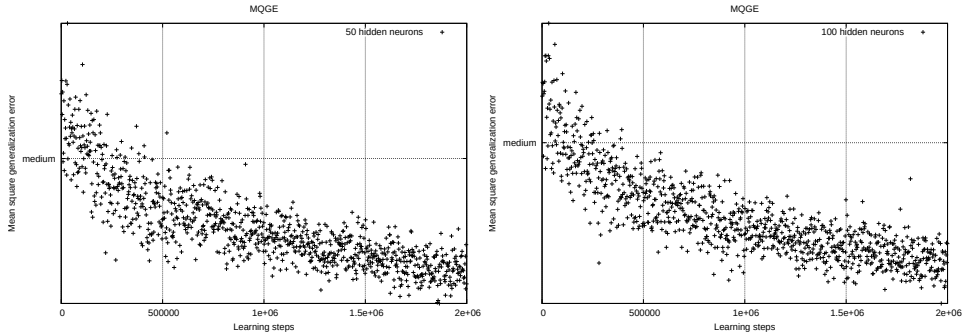
Lernrate $\nu = 0,0001$, `vp40s.cnt` und `vp41s.cnt`, 10000000 Lernschritte, ersten 2000000 dargestellt, einfache Normierung

Abbildung A.4: MQGE, ANNs mit realen EEG-Daten



(a) ANN, 5 hidden neurons

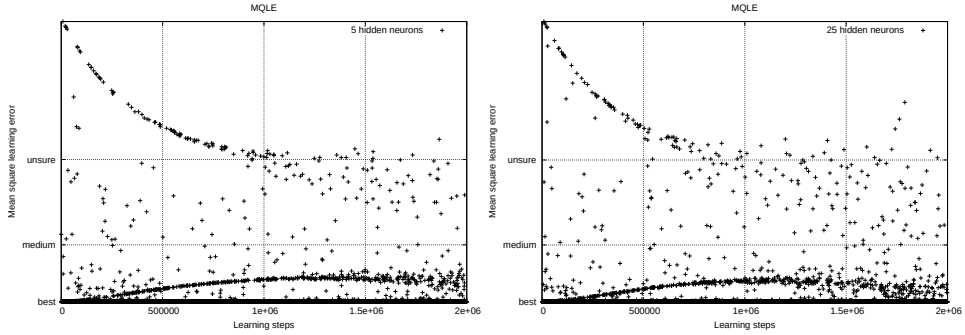
(b) ANN, 25 hidden neurons



(c) ANN, 50 hidden neurons

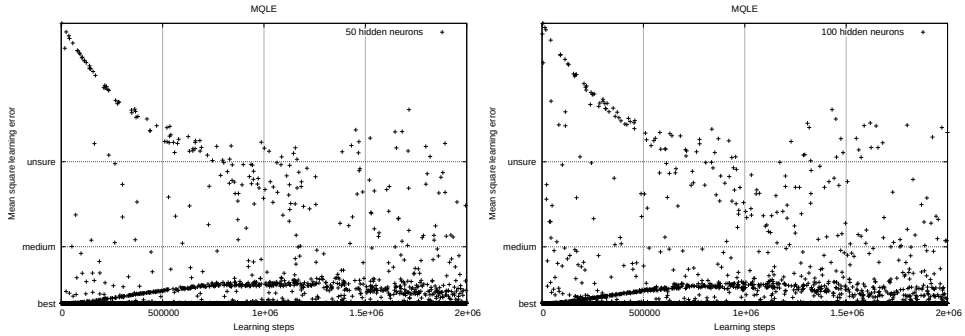
(d) ANN, 100 hidden neurons

Abbildung A.5: Einfluß Anzahl verdeckte Neuronen auf MQGE



(a) ANN, 5 hidden neurons

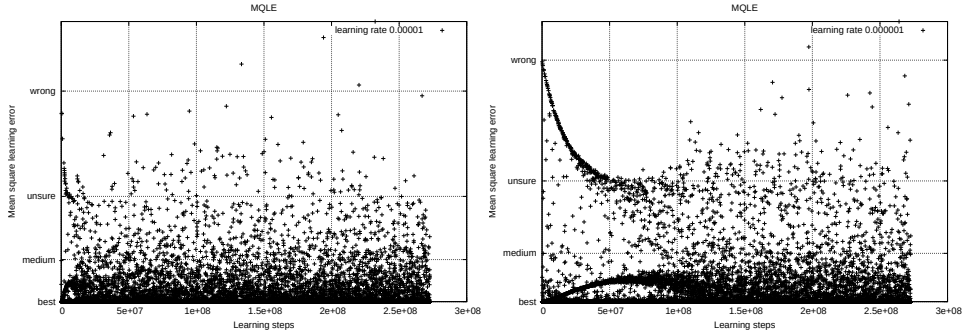
(b) ANN, 25 hidden neurons



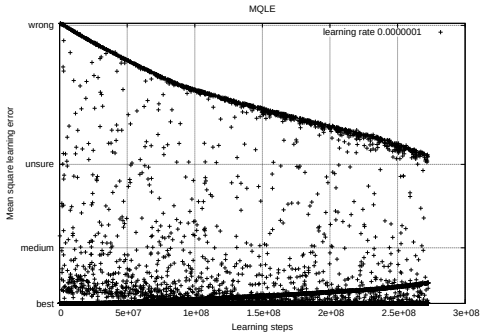
(c) ANN, 50 hidden neurons

(d) ANN, 100 hidden neurons

Abbildung A.6: Einfluß Anzahl verdeckte Neuronen auf MQLE

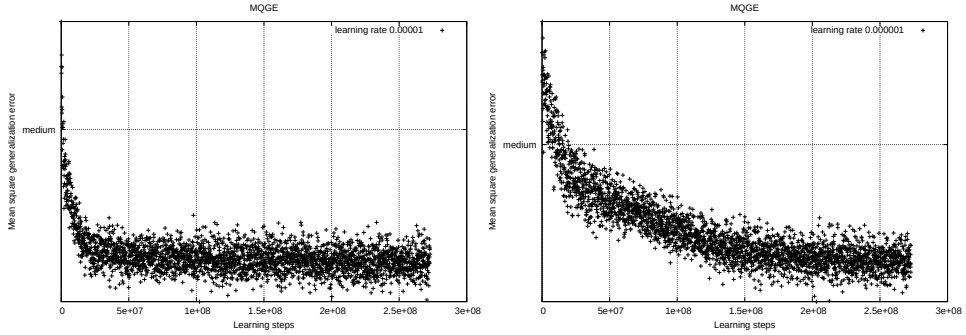


(a) ANN, $\nu = 0.00001$ (b) ANN, $\nu = 0.000001$

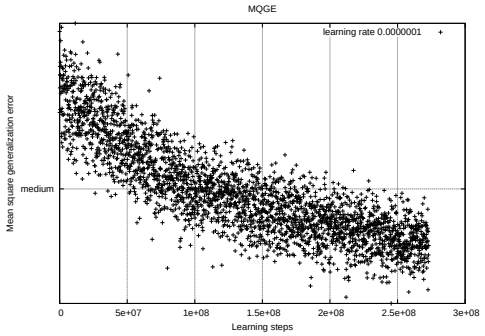


(c) ANN, $\nu = 0.0000001$

Abbildung A.7: Einfluß des Lernfaktors ν auf MQLE



(a) ANN, $\nu = 0.00001$ (b) ANN, $\nu = 0.000001$



(c) ANN, $\nu = 0.0000001$

Abbildung A.8: Einfluß des Lernfaktors ν auf MQGE

B Inhalt der beiliegenden Compact Disk

Zu dieser Arbeit gehört eine CD, die den aktuellen Quellcode der im Rahmen dieser Diplomarbeit enstanden Programme, sowie die API-DokumentationRomeyke (2004) und diese Arbeit im PDF-Format enthält.

Die Programme unterliegen der GNU General Public License[1]. Für die Kompilierung wird die GNU Compiler Colletion (GCC) verwendet, andere Compilerumgebungen wurden nicht getestet.

[1]Soweit nicht anders vermerkt. Auf die Lizenzbedingungen wird im Quellcode verwiesen

C Hilfsmittel, Software

Im Rahmen dieser Diplomarbeit wurde das TEX-Satzsystem unter Verwendung der LATEX-Makros KomaScript, NatBib, Listings u. a. für den Textsatz des vorliegenden Dokuments eingesetzt.

Als Betriebssystem wurde für die Programmierung und Dokumentationserstellung *GNU Linux* der *Debian*-Distribution verwendet. Als Editor fand *jEdit* Anwendung. *Perl* und *Bash* waren der Kleber, der alles zusammenhielt. *Gnuplot*, *XMGrace* und *Xd3D* wurden für die Visualisierung der Daten herangezogen.

Die im Rahmen dieser Arbeit entstandenen C-Programme wurden mit Hilfe der *GNU Compiler Collection – GCC* compiliert, mit *splint* auf Fehlerminimiertheit getestet, mit *GNU Make* zusammengestellt und mit *cvs* und *gcvs* wurden die einzelnen Versionen verwaltet. Das Programm *xcnt* und die *opencnt*-Bibliothek wurden für den Zugriff auf die EEG-Daten benutzt. Die *FANN*-Bibliothek erleichterte das Erstellen eines MLP-ANN, die *FFTW*-Bibliothek ermöglichte die schnelle Fouriertransformation der Datensätze.

Die Dokumentation der API und C-Quelltexte erfolgte mit *doxygen*. Für die Abbildungen wurden *Ghostscript*, *ps2pdf* und *dia* eingesetzt. Für die Verwaltung der Bibliographie wurden *bibtex* und *gbib* genutzt.

Literaturverzeichnis

[Barn 2004] BARN, Dictionary: *Dictionary Barn – A medical dictionary*. Internet `http://www.dictionarybarn.com/`. 2004 C

[Bartsch 1999] BARTSCH, Hans-Jochen: *Taschenbuch Mathematischer Formeln*. 8. Fachbuchverlag Leipzig, 1999 2.3.6, 2.3.7, 4.2, C

[Bell 2000] BELL, Anthony J.: *Unsupervised Adaptive Filtering*. Bd. 1. Kap. Information Theory, Independent-Component Analysis, and Applications, S. 237–263, John Wiley & Sons Inc, 2000 1.3

[Bogacz u. a. 1999] BOGACZ, Rafal ; MARKOWSKA-KACZMAR, Urszula ; KOZIK, Andrzej: Blinking Artefact Recognition in EEG Signal by Neural Network. In: *Proceedings of the Fourth Conference on Neural Networks and Their Applications*, Polish Neural Network Society, nov 1999, S. 502–507. – URL `http://www.cs.bris.ac.uk/Publications/Papers/2000061.pdf` 1.3, 5.2

[Bourke 2004] BOURKE, Paul: *NeuroScan EEG File Formats*. Internet `http://astronomy.swin.edu.au/~pbourke/dataformats/neuroscan/`. 05 2004. – Fileformat *.cnt, SCAN version 4 1

[Callan 2003] CALLAN, Robert: *Neuronale Netze im Klartext*. Pearson Studium, 2003 2.2.2

[Cooper u. a. 1984] COOPER, Ray ; OSSELTON, John W. ; SHAW, John C.: *Elektroenzephalographie – Technik u. Methoden*. 3. Stuttgart : Gustav Fischer Verlag, 1984 1.3, 5.6

[Delorme und Makeig 2004] DELORME, Arnaud ; MAKEIG, Scott: *EEGLAB*. Internet `http://www.sccn.ucsd.edu/eeglab`. 2004. – 1.2, 1.3

[Dlabka 2004] DLABKA, Michael: *Skript zur Vorlesung 'Neuronale Netze und Fuzzy-Systeme', Deutsche Telekom – Fachhochschule Leipzig*. 2004. – Vorlesung WS 2003/2004 10, 4.2

[Durka u. a. 1996] DURKA, Piotr ; KSIEZYK, Rafal ; BLINOWSKA, Katarzyna: Neural Networks and Wavelet Analysis in EEG Artefact Recognition. In: *II Kon-*

ferencja Sieci Neuronowe i Ich Zastosowania Szczyrk 30 IV - 4 V (Veranst.), 1996, S. 6. – Internet `http://brain.fuw.edu.pl/~durka/pd_paper.html` 1.3

[Ebe und Homma 1992] EBE, Mitsuru ; HOMMA, Isako: *Leitfaden für die EG-Praxis – Ein Bildkompendium*. Gustav Fischer Verlag, 1992 1.3

[Frigo und Johnson 2004] FRIGO, Matteo ; JOHNSON, Steven G.: *FFTW Library*. Internet `http://www.fftw.org`. 03 2004 C

[Hennig 1998] HENNIG, Carsten: ICA–Kurs / Technische Universität Berlin. 1998. – Forschungsbericht. [1] Leicht verständliche Einführung in die Independent Component Analysis, statistische kennwerte höherer Ordnung, neuronale Ansätze und andere. 6, 5.6, C

[Inc. 2004] INC., Open Source Development N.: *FreshMeat – Web's largest index of Unix and cross-platform software*. `http://www.freshmeat.net`. 2004 3.1

[Jung u. a. 1998] JUNG, Tzyy-Ping ; HUMPHRIES, Colin ; LEE, Te-Won ; MAKEIG, Scott ; MCKEOWN, Martin J. ; IRAGUI, Vicente ; SEJNOWSKI, Terrence J.: Extended ICA Removes Artifacts from Electroencephalographic Recordings. In: *Advances in Neural Information Processing Systems* (1998), Nr. 10, S. 894–900 1.3, 5.6

[Ksiezyk u. a. 1998] KSIEZYK, Rafal ; BLINOWSKA, Katarzyna ; DURKA, Piotr ; SZELENBERGER, W. ; ANDROSIUK, W.: Neural Networks with Wavelet Preprocessing in EEG Artifact Recognition / Warsaw University – Institute of Experimental Physics and Warsaw Medical Academy – Department of Psychiatry. 1998. – Forschungsbericht. Medicon98, im Internet verfügbar `http://brain.fuw.edu.pl/~durka/papers/medicon98_arif.pdf` mit eigenen Problemen und Tests, Gute vergleichbare Ergebnisse, unbedingt verwenden 1.3, 5.6

[Ltd. 2004] LTD., Compumedics: *Neuroscan*. Internet `http://www.neuroscan.com/`. 05 2004 1

[Makeig u. a. 1996] MAKEIG, Scott ; BELL, Anthony J. ; TZYY-PING, Jung ; SEJNOWSKI, Terrence J.: Independent Component Analysis of Electroencephalographic Data. In: *Advances in Neural Information Processing Systems 8* 8 (1996), S. 145–151 1.3, 5.6

[Minsky und Papert 1988] MINSKY, M. ; PAPERT, S.: Perceptrons. In: ANDERSON, J. A. (Hrsg.) ; ROSENFELD, E. (Hrsg.): *Neurocomputing: Foundations of Research*, MIT Press, 1988, S. 161–170 2.2.1

[NeuroQuest AI 2004] NEUROQUEST AI, L.L.C.: *Logically Advanced Neural Engine*. Internet `http://www.neuroquest.com/lane/`. 05 2004. – 3.1

[1] basierend auf Diplomarbeit von Carsten Hennig, 1999

[Nissen 2003] NISSEN, Steffen: *Implementation of a Fast Artificial Neural Network Library*, Department of Computer Science, University of Copenhagen, Diplomarbeit, 10 2003. – Internet `http://fann.sourceforge.net/report/report.html` 2.2.2, 3.1

[Nissen 2004] NISSEN, Steffen: *Fast Artificial Neural Network*. Internet `http://fann.sourceforge.net/`. 05 2004 3.1, C

[Novák u. a. 2001] NOVÁK, Daniel ; LHOTSKÁ, Lenka ; ECK, Vladimír ; SORF, Milan: EEG and VEP signal Processing / Czech Technical University – Department of Cybernetics. 2001. – Forschungsbericht. Gute Einführung in Typen von EEG-Artefakten und verschiedene Signal Processing Techniken für EEG Daten, incl. ICA 1.3, 6, 2.1.2, 5.5

[Nowagk 2004] NOWAGK, Rainer: *cntopen-Library*. Internet `http://www.ant-software.nl/`. 2004. – cntopenlib 4.1

[Press u. a. 1993] PRESS, William H. ; VETTERLING, William T. ; FLANNERY, Brian P. ; TEUKOLSKY, Saul A.: *Numerical Recipes in C: The Art of Scientific Computing*. 2. Cambridge University Press, 1 1993 2.3.5

[Robert und Gaudy 2002] ROBERT, Claude ; GAUDY, Aimé: Electroencephalogram processing using neural networks. In: *Clinical Neurophysiology* (2002), Nr. 113, S. 694–701. – 1.3

[Rojas 1996] ROJAS, Raúl: *Theorie der neuronalen Netze: eine systematische Einführung*. 4. Berlin : Springer-Verlag, 1996 2.2.2

[Romeyke 2004] ROMEYKE, Andreas: *EEG Artifact Rejector API Manual*. 2004 3.2, 4.1, B

[van Rossum 2004] ROSSUM, Peter van: *Lightweight neural network*. Internet `http://lwneuralnet.sourceforge.net/`. 2004 3.1

[Sarle 2004] SARLE, Warren: *comp.ai.neural-nets FAQ*. Internet: `http://www.faqs.org/faqs/ai-faq/neural-nets`. 2004 3.1

[Schleimer 2003] SCHLEIMER, Jan-Hendrik: *Zusammenfassung – Praktikum neuronale Netze, Thema: Klassifikation von EEG-Signalen*. Internet `http://www.cis.hut.fi/~schleime/endbericht.pdf`. 2003 1.3

[Technology 2004] TECHNOLOGY, Advanced N.: *Advanced Neuro Technology*. Internet `http://www.ant-software.nl/`. 2004 4.1

[Tutrin 2004] TUTRIN: *How Brain and Consciousness work*. Internet `http://www.tutrin.com`. 06 2004. – 12

[Vuckovic u. a. 2002] VUCKOVIC, Aleksandra ; RADIVOJEVIC, Vlada ; CHEN, Andrew C. ; POPOVIC, Dejan: Automatic recognition of alertness and drowsiness from EEG by artificial neural networks. In: *Medical Engineering Physics* (2002), Nr. 24, S. 349–360 1.3

[Weber u. a. 2004] WEBER, Darren ; ARNAUD, Delorme ; HOUGH, Morgan ; OOSTENVELD, Robert ; DALAL, Sarang: *EEG and MRI Matlab Toolbox*. Internet `http://eeg.sourceforge.net/`. 2004 1.3

[Wikipedia 2004] WIKIPEDIA: *WikiPedia*. Internet `http://wikipedia.de/`. 2004. – deutschsprachige Ausgabe 2.3.7, C

[Zell 1994] ZELL, Andreas: *Simulation Neuronaler Netze*. München, Wien : R. Oldenbourg Verlag, 1994 3, 9, 2.2.2, 2.2.2, 2.2.2, 3.1, C

[Zell 2004] ZELL, Andreas: *Stuttgart neural network simulator*. Internet `http://www-ra.informatik.uni-tuebingen.de/SNNS/`. 2004 3.1

Abkürzungsverzeichnis

ANSI-C entspricht der ISO C90 Normierung, der Standardisierung der Programmiersprache C durch die *International Standardization Organization* auf den Stand von 1990

CCN Cascaded Correlation Network, ANN-Typ bei dem während der Lernphase weitere Neuronen hinzugefügt werden. CC-Netze sind dadurch stärker dem Problem angepaßt, Over-Learning wird vermieden, die Zahl der Neuronen liegt meist niedriger und die Lernzeit ist geringer als in MLP-Netzen. Für Details sei auf Zell (1994) verwiesen.

CSA Compress Spectral Analysis, Darstellungsmethode, um sFFT-transformierte Daten (Spektren) entlang einer Zeitachse darzustellen, um die Spektrumsveränderung sichtbar zu machen. s. h. **??** auf Seite ?? und C

EEG Elektroenzephalogramm, Verfahren um mit Hilfe von Oberflächenelektroden von der Kopfhaut abgegriffene elektrische Spannungsunterschiede zu messen, die ein Maß für die Beurteilung der Gehirnaktivität darstellen

EOG Elektro-Oculogramm, Messung der Augenbewegung mittels durch Dipoleffekt bewirkte Stromänderungen(Barn, 2004)

ERP event related potentials, ereignisbedingte Potentiale sind eletrische Aktivitäten des Gehirns als Reaktion auf präsentierte Stimuli

FANN *Fast Artificial Neural Network*, schnelle und gut dokumentierte ANN-Bibliothek für C-Programme, entwickelt von Steffen Nissen(Nissen, 2004)

FFT Fast Fourier Transformation, mathematische Methode um Zeitreihendaten in den Frequenzraum zu transformieren, s. h. auch Bartsch (1999) oder Wikipedia (2004)

FFTW freie *Fastest Fourier Transform in the West*-Bibliothek (Frigo und Johnson, 2004), die den Anspruch erhebt, die schnellsten Routinen für die Fouriertransformationen in der Programmiersprache C zur Verfügung zu stellen

ICA Independent Component Analysis, mathematische Methode, die zur Separierung unbekannter Quellen aus einem Mischsignal entwickelt wurde. Ein Spezialfall der ICA ist die PCA, s. h. C. Eine gute Einführung in die ICA findet man in Hennig (1998)

MLP Multilayer Perceptron, klassische Konfiguration eines ANN, s. h. Zell (1994) für Details

MPI Max-Planck-Institut

MQGE *mean square generalization error*, der mittlere quadratische Generalisierungsfehler gibt den Fehler des ANN nach n-Lernschritten an, wenn dem Netz noch nicht gelernte Daten präsentiert werden.

MQLE *mean square learning error* bzw. der mittlere quadratische Lernfehler, gibt den Fehler pro Lernschritt an.

MRI Magnetic Resonance Imaging beruht auf dem Prinzip der kernmagnetischen Resonanz (NMR): durch das Anlegen eines starken Magnetfeldes orientieren sich die Spins von Atomkernen. Zwischen den beiden nun möglichen Zuständen besteht eine geringe Energiedifferenz. Bei Einstrahlung eines elektromagnetischen Pulses mit passender Frequenz entsteht eine Resonanz, die gemessenen werden kann.

PCA Principle Component Analysis, Verfahren der Dimensionsreduzierung des Datenraumes, mit dem Ziel $r \leq n$ untereinander unkorrelierte Linearkombinationen (eben Hauptkomponenten) mit zueinander maximaler Varianz zu finden

RBF Radiale Basisfunktionsnetze, ANN mit lokal wirkenden Neuronen, s. h. Zell (1994)

Index