

SIGFIRE

Dokumentation zum Beleg im Fach Neuronale Netze

Bearbeiter:

Torsten May, htw8866

Stefan Köhler, htw8875

20. September 2000

1 Übersicht

Das Projekt “sigfire”, dessen Name für “signal Filtering and REcognition” steht, demonstriert die Erkennung (Klassifizierung) und qualitative Aufwertung (Filterung) verrauschter Signale mit Hilfe neuronaler Netze.

Der Prinzipielle Ablauf der Verarbeitungsschritte ist stets folgender: Ein von einem Generator erzeugtes Eingangssignal durchläuft eine Vorverarbeitungsstufe (Mittelwertbildung), wird durch neuronale Netze klassifiziert und in Abhängigkeit von der erkannten Zugehörigkeit zu einer bestimmten Signalklasse durch ein spezielles neuronales Netz gefiltert. Die Filterung erfolgt wahlweise nach einem von zwei verschiedenen Verfahren, wobei eines der Verfahren in einer Nachverarbeitungsstufe nochmals eine Mittelwertbildung durchführt.

Durch die geschickte Kombination von Signalerkennung, Mittelwertbildung und Signalfilterung können selbst stark verrauschte Signale nahezu auf das Niveau ihrer reinen Ausgangsform geglättet werden. Das vorliegende System verwendet dabei 11 trainierte neuronale Netze, von denen stets 4 im Einsatz sind.

Das vorliegende System verarbeitet Sinus-, Rechteck- und Sägezahnkurven. Vielfältige Einstellungsmöglichkeiten zur Laufzeit gestatten einen umfassenden Test und Vergleich der Leistungsfähigkeit verschiedener Filterverfahren.

Als Voraussetzung für die gute Qualität der gefilterten Kurven müssen die Eingangsdaten allerdings normiert sein, d.h. die Amplitude und besonders die Frequenz der Eingangsdaten dürfen nicht zu stark von einem festgelegten Normalmaß abweichen (Signalpegel -0.8 bis 0.8, 25 Samples für eine Periode).

2 Bestandteile des Systems

2.1 Generator

Der sogenannte Generator erzeugt wahlweise Sinus-, Rechteck- oder Sägezahnkurven. Rechteck- und Sägezahnsignale können in reiner Form erzeugt oder durch Fourier-Entwicklung approximiert werden. Den Kurven wird auf Wunsch zufälliges oder normalverteiltes Rauschen wählbarer Stärke überlagert. Natürlich kann auch ein leeres (evtl. verrauschtes) Signal erzeugt werden. Periodenlänge (Anzahl der Kurvenpunkte für eine volle Schwingung), Amplitude (maximaler Ausschlag der Kurve) und die Iterationstiefe bei approximierten Kurven sind frei einstellbar.

2.2 Signalerkennung

Zur Erkennung des Signaltyps wird genau eine Periode der Eingangsdaten (das entspricht 25 Samples) je einem speziell auf eines der vorhandenen Muster (Sinus, Rechteck oder Sägezahn) trainierten neuronalen Netz präsentiert.

Das einzelne Ausgabeneuron dieser Netze entscheidet die Zugehörigkeit zur jeweiligen Musterklasse:

- 1 = Eingangssignal entspricht dem Muster
- -1 = Eingangssignal entspricht nicht dem Muster

Abbildung 2.1 zeigt die Topologie der verwendeten Netze zur Signalklassifikation.

Um die Stabilität der Mustererkennung zu erhöhen, wird stets der Mittelwert aus einer wählbaren Anzahl von Netzausgaben gebildet. Übrigens befinden sich die vorliegenden Erkennungsnetze noch im Status "erster Versuch". Mit etwas Experimentierfreude z.B. bei der Netztopologie läßt sich das Ergebnis sicher noch verbessern.

Die Signalerkennung dient dazu, bei der Filterung statt eines komplexen Netzes für alle vorkommenden Signale ein spezielles auf ein konkretes Muster trainiertes Netz auszuwählen. Dadurch kann eine wesentlich bessere Verarbeitungsqualität erreicht werden.

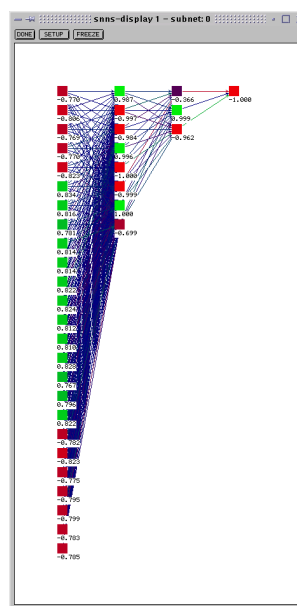


Abbildung 2.1: Signalerkennungsnetze

2.3 Signalfilterung

Durch Anwendung verschiedener Filtermethoden wird eine erhebliche qualitative Verbesserung der Eingangsdaten erzielt. "sigfire" implementiert zwei verschiedene Ansätze zur Filterung eines verrauschten Eingangssignals.

2.3.1 Autoassoziative Filterung mit komplettem Outputpattern

Eine volle Periode der Eingangsdaten (25 Samples) wird der Inputschicht eines autoassoziativen Netzes präsentiert. Solch ein Netz besitzt genau so viele Ausgangs- wie Eingangsneuronen und bildet die Eingangsbelegung auf den Ausgang ab. Dabei werden jedoch leichte Abweichungen vom Idealbild kompensiert und verrauschte Muster geglättet.

Zur Stabilisierung des Ergebnisses wird stets der Mittelwert aus mehreren Output-Patterns gebildet. Diese Mittelwertbildung ist das Kernstück des Verfahrens, wobei die Anzahl der gemittelten Netzergebnisse in diesem Schritt entscheidende Bedeutung für die Qualität des Gesamtergebnisses besitzt. Ist sie zu klein, dann wirkt die erzeugte Kurve kantig und "zittert", d.h. sie ändert von Pattern zu Pattern zum Teil stark ihre Gestalt. Wird jedoch zu viel gemittelt, so reagiert das System nur noch sehr träge auf eine Änderung des Eingangssignals.

Über den Parameter "filter passes" kann das gefilterte Signal mehrfach gefiltert werden (erneute Mittelwertbildungen eingeschlossen), was zu einer weiteren erheblichen Steigerung der Qualität führt.

Das Netz wurde wie in Abbildung 2.2 a) als feedforward-Netz umgesetzt, obwohl SNNS einen speziellen Netztyp für autoassoziative Netze bereitstellt, da den Autoren aufgrund mangelhafter Dokumentation dieses Features von SNNS nicht das notwendige Fachwissen bereitstand.

2.3.2 Filterung mit einem einzelnen Outputneuron

Aufgrund der zahlreichen Outputneuronen des autoassoziativen Filters sind ein sehr großes Netz und demnach hohe Trainingszeiten erforderlich, um einen akzeptablen Netzfehler zu erzielen. Das Netz in Abbildung 2.2 a) besitzt 2 Hidden-Schichten mit jeweils 30 und 45 Neuronen!

Daher liegt der Gedanke nahe, ähnlich wie bei der time series prediction aus einem vollständigen Pattern nur einen einzelnen Wert zu berechnen. Abbildung 2.2 b) zeigt ein solches Netz, welches mit einer deutlich geringeren Neuronenzahl auskommt.

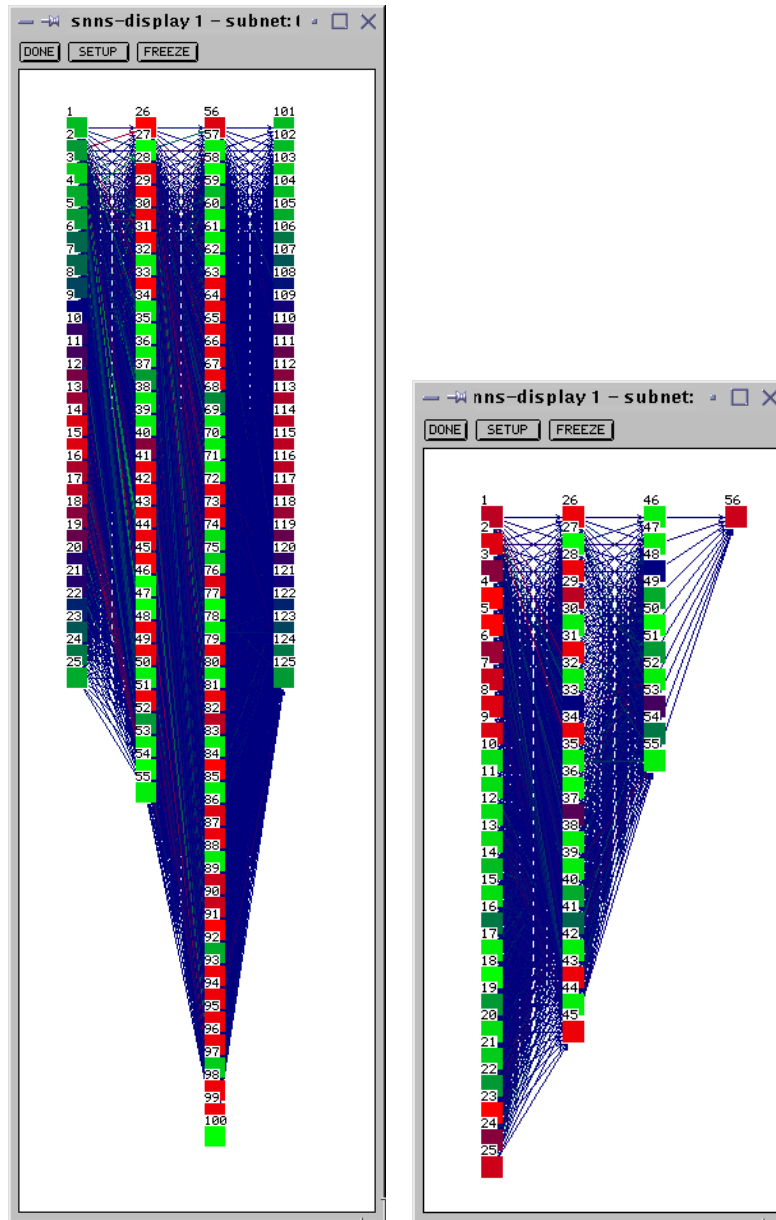
Statt des gesamten Eingangssignals wird nur das erste der 25 Samples am Ausgang abgebildet. Aufgrund der geringen Neuronenzahl kann ein gutes Lernergebnis viel schneller als bei der ersten Variante erzielt werden.

Leider hat dieses Vorgehen auch Nachteile. Zum einen wird der Output um eine volle Periode gegenüber dem Input verzögert. Zum anderen ist eine Mittelwertbildung wie in Variante 1 kaum sinnvoll, da sich die Werte innerhalb einer Periode nicht ändern (sie werden weitergeschoben) und daher nur eine *sehr* träge Anpassung an das Eingangssignal erfolgt.

2.4 Die Netze

Im System kommen für die Signalerkennung 3 und für die beiden Filtermethoden jeweils eines von 4, also insgesamt 4 von 11 trainierten neuronalen Netzen zum Einsatz. Alle Netze sind feedforward-Netze mit Vollverbindung zwischen den Schichten. Alle Neuronen benutzen den tangens hyperbolicus als Aktivierungsfunktion.

Die Netze wurden mit dem "Stuttgarter Neural Network Simulator" SNNS trainiert und mit snns2c als C-Module in das Projekt integriert (sigfire ist daher auch ohne Installation von SNNS lauffähig). Als Trainingsmethode kam Standard Backpropagation Momentum zum Einsatz.



(a) autoassoziatives Netz

(b) Filter mit individuellem Outputneuron

Abbildung 2.2: Zwei verschiedene Netztypen zur Signalfilterung

2.5 Der PatternMaker

Eines der wichtigsten Probleme beim überwachten Training neuronaler Netze ist das Zusammenstellen repräsentativer Trainingssätze. Da die einzelnen Signale in jeder Phasenlage mit verschiedenen Rauschanteilen enthalten sein sollen, entstehen schnell große Trainingssätze. Deshalb gibt es den PatternMaker, der automatisch Patternfiles für das Training mit SNNS generiert und dabei zahlreiche Einstellungsmöglichkeiten besitzt.

Der PatterMaker ist ein Kommandozeilentool und kann mit folgenden Parametern aufgerufen werden:

- h zeigt eine kurze Beschreibung der vorhandenen Optionen
- n num legt fest, wie oft eine komplette Periode der zu trainierenden Muster enthalten sein soll
- o leere Kurve (nur Rauschen) enthalten
- s Sinuskurve enthalten
- r Rechteckkurve enthalten
- t Sägezahnkurve enthalten
- a alle Kurven enthalten (entspricht -o -s -r -t)
- f statt des gesamten Musters nur ersten Punkt als desired output speichern
- e Trainingsmuster für Signalerkennung erzeugen (-1.0 oder 1.0 output)
- w normalverteiltes statt zufälligem Rauschen verwenden
- u amp setzt die Amplitude
- m max setzt den maximalen Rauschanteil (Amplitude bei zufälligem oder Standardabweichung bei normalverteiltem Rauschen wird pro Durchgang schrittweise bis zum Maximalwert erhöht)

Auf diese Weise konnte auch das große Trainingsfile für das allgemeine Erkennungsnetz sehr einfach erstellt werden. Es enthält je 250 Perioden der 4 Muster (leer, Sinus, Rechteck, Sägezahn), also $250 \times 4 \times 25 = 25000$ Trainingspatterns mit einem maximalen Rauschanteil von 0.25. Solch ein Patternfile hätte wohl kaum von Hand oder mit der Record-Funktion der graphischen Oberfläche realisiert werden können.

3 Benutzerschnittstelle

Abbildung 3.1 zeigt die Benutzerschnittstelle von sigfire, welche auf der Oberflächenbibliothek “qt” basiert und mit Hilfe QT-Architekten qtarch erstellt wurde. Die linke Seite zeigt das Eingangssignal und alle Parameter der Signalerzeugung. Die rechte Seite enthält dagegen die Ergebnisse der Signalverarbeitung und ermöglicht die Einstellung der zugehörigen Parameter. Am unteren Bildschirmrand befinden sich noch ein Paar Buttons zur Steuerung des Ablaufs.

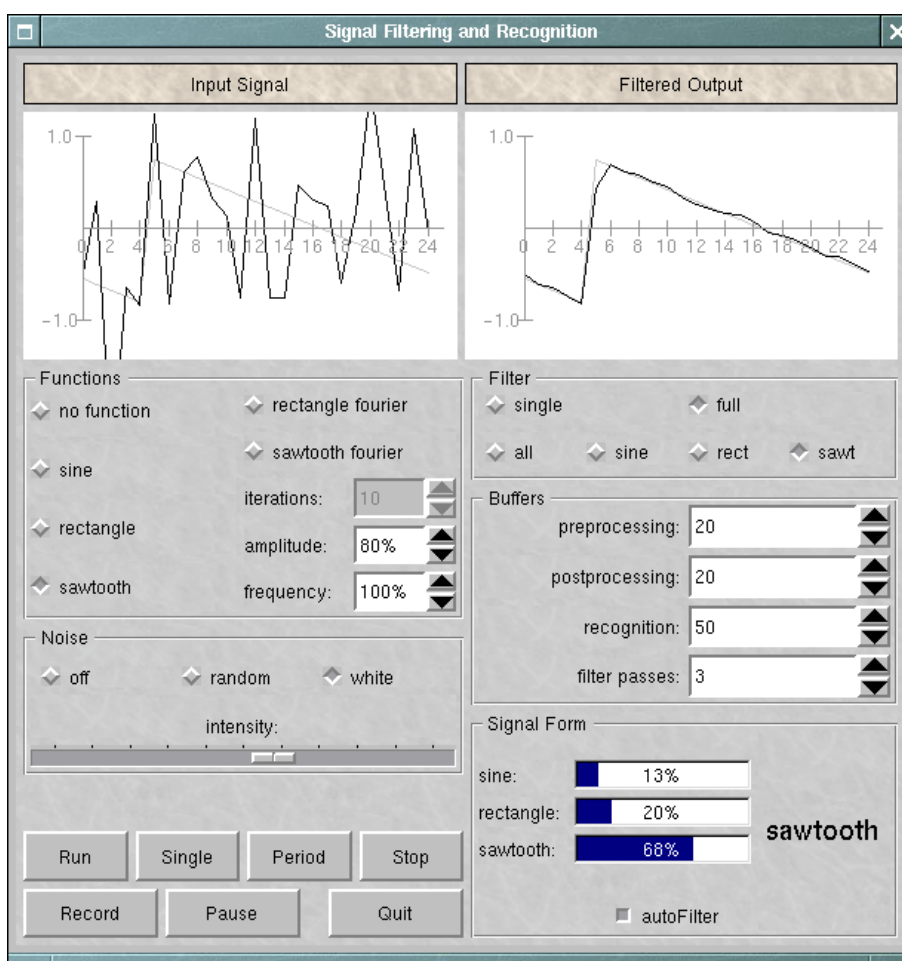


Abbildung 3.1: Benutzerschnittstelle von sigfire

Es folgt eine kurze Beschreibung der einzelnen Optionen der Bedienoberfläche.

Auf der linken Seite werden die Einstellungen für die Signalerzeugung vorgenommen:

- Functions - Einstellung der Signalform

no function kein Signal

no function kein Signal

sine Sinuskurve erzeugen

rectangle Rechteckkurve erzeugen

sawtooth Sägezahnkurve erzeugen

rectangle fourier Approximation einer Rechteckkurve erzeugen

sawtooth fourier Approximation einer Sägezahnkurve erzeugen

iterations Anzahl der Iterationsschritte bei rectangle fourier oder sawtooth fourier

amplitude Maximalausschlag, 100% entspricht 1.0; 80% entspricht 0.8 und ist die Normaleinstellung

frequency erlaubt ein Variieren Frequenz

- Noise - Einstellung des Rauschens

off kein Rauschen

random zufälliges Rauschen

white normalverteiltes Rauschen

intensity Amplitude bei zufälligem, Standardabweichung bei normalverteiltem Rauschen

Die rechte Seite ermöglicht die Konfiguration der Signalverarbeitung:

- Filter - Festlegen der Filtermethode

single Netze mit nur einem Ausgabeneuron verwenden

full Netze mit vollständiger Ausgabe (25 Neuronen) verwenden

all mit allen vorkommenden Mustern trainierte Netze verwenden

sine speziell auf Sinuskurven trainierte Netze verwenden

rect speziell auf Rechteckkurven trainierte Netze verwenden

sawt speziell auf Sägezahnkurven trainierte Netze verwenden

- Buffers - Festlegen der Mittelwertbildung

preprocessing Anzahl der zu mittelnden Patterns *vor* der Filterung, dadurch werden stochastische Abweichungen bereits etwas kompensiert

postprocessing Anzahl der zu mittelnden Netzausgaben (nur bei Filtermethode "full")

recognition Anzahl der zu mittelnden Erkennungsergebnisse

filter passes Anzahl der Filterstufen (jeweils Filter + Mittelwertbildung), bei 0 wird das Eingangssignal ohne Filterung direkt an den Ausgang gereicht

- Signal Form - Ergebnisse der Signalerkennung

autoFilter bei Erkennung eines Signaltyps automatisch auf dieses Signal spezialisierte Filternetze aktivieren

Zum Abschluß folgt nun noch eine Beschreibung der Buttons:

Run timergesteuerte kontinuierliche Erzeugung des Eingangssignals

Single Erzeugung eines einzelnen neues Samples und Weiterrücken der Kurve um einen Punkt

Period Durchlauf genau einer Periode

Stop timergesteuerten Ablauf anhalten

Record Patterfile aus aktuellen Daten interaktiv erzeugen (bestenfalls für Verifikationsdatensätze geeignet, da zu unflexibel verglichen mit dem PatternMaker)

Pause unterbricht Aufnahme eines Patternfiles mit Record

Quit beendet das Programm