NADA

Numerisk analys och datalogi

SANS
NADA
KTH
100 44 Stockholm

Studies of Artificial Neural Systems
Department of Numerical Analysis
and Computing Science
Royal Institute of Technology
S-100 44 Stockholm, Sweden

Data Mining and Classification with Credibility using a Bayesian Confidence Propagation Neural Network

Roland Orre
orre@nada.kth.se

http://www.nada.kth.se/~orre

Akademisk avhandling
som med tillstånd av Kungliga Tekniska Högskolan i Stockholm framlägges till
offentlig granskning för avläggande av teknisk licenciatexamen i Datalogi.

TRITA-NA-P2002?? ISSN 1101-2250 ISRN KTH/NA/P–2002/??-SE ISBN 91-7170-273-3

The LaTeX typesetting system has been used throughout the thesis, also for those papers that were typeset by the publisher.

Stockholm 2002 KTH Högskoletryckeriet

ISBN 91-7170-273-3

To Ottilia and Sebastian
my beloved children

That which is truly beautiful and worthwhile in life
is generally hidden away in mines of secrecy.
– *Paramahansa Yogananda*

Abstract

The aim of this thesis is to describe how a neural network, here named BCPNN (Bayesian Confidence Propagation Neural Network), which can be identified by rewriting of Bayes' rule, can be used for data mining and classifications with credibility intervals in some applications.

The aim of this thesis is to describe how a statistically based neural network technology the BCPNN (Bayesian Confidence Propagation Neural Networks) can be used within two different applications, data mining in a huge database and modelling of an industrial process. BCPNN has previously been successfully used within classification tasks like fault diagnosis, pattern recognition and hierarchical clustering analysis.

BCPNN is a neural network model reminding somewhat about Bayesian descision trees which are being used within artificial intelligence systems. As a neural network the BCPNN is rather different from backprop (BP) and other gradient methods. The learning process in BCPNN is based upon calculations of probabilities and dependencies which is often a more or less straight forward process compared to the usually time consuming iterative gradient methods. The interpretation of weight values in a BCPNN is also rather easy compared to interpretation of the weight values within a network which is trained by gradient methods.

When we say process modelling here, this refers to function approximation. A function in the general sense may be considered a spatio-temporal outcome of a spatio-temporal input. Function approximation in this sense is somewhat more complex than the modelling we do in this thesis, as we don't deal with time in those paper where we discuss process modelling. To give a glimpse of the BCPNN being able to deal also with time there are two papers included where we deal with some temporal aspects of BCPNN.

The most important results found in this thesis can be summarized in the following: *We show* how a Bayesian Neural Network can be extended to model the uncertainties in the collected statistics to produce outcomes as distributions from two different aspects: uncertainties induced by sampling, which is useful for data mining; uncertainties due to input data distributions, which is useful for process modelling. *We show* how complex dependencies can be found within large data sets but still avoiding combinatoric explosion. *We show* how these techniques have been turned into a useful tool for real world applications within the drug safety area in particular. *We compare* some results of the BCPNN technique with the well established non linear regression technique, BP (back prop networks), for processing modelling, showing that the BCPNN performs at least equally well, but provides extra information about uncertainties of produced outcomes. *We present* a simple but working method for doing automatic temporal segmentation of data sequences. *We indicate* some aspects of temporal tasks for which a predictive Bayesian neural network may be useful. Showing how the connection matrix can be reduced due to regularities in the data.

Key words: Artificial neural network, Bayesian neural network, data mining, adverse drug reaction signalling, process modelling

Acknowledgements

Although the acknowledgements part will always be incomplete, I will nevertheless, thank those people that I spontaneously think of on the way up to my writing of this thesis. When I wrote this I was a graduate student at the research group for *Studies of Artificial Neural Systems* (SANS) at the department of *Numerical Analysis and Computing Science* (NADA) at *Royal Institute of Technology* (KTH) in Stockholm. During the first part of this thesis work I was still working part time at Asea Brown Boweri (ABB).

First I would want to thank *Anders Lansner*, my supervisor and head of the SANS group. The task of being a supervisor is not an easy one as there are many irons in the fire at the same time and you need to be supportive, coaching and inspirative until someone can stand on their own feet. Anders has done a great job here, which I am very grateful for and I also want to thank him for inviting me to the research group in the first place. Thanks Anders!

Research work when being a graduate student at SANS is very joyful due to all the inspiring people that you have around you who all take part of creating a nice and inspiring athmosphere.

Innermost thanks to *Mikael Djurfeldt* for always being a good friend with whome I like to discuss all kinds of issues, technical, musical, spiritual as well as mathematical. Mikael also does a great job in being one of the world's developers of *guile*, the scheme language in which I've done all my experiments and developments lately. I also want to thank him for bringing me to the SANS-seminar in 1988, which caused me to switch my interest from expert systems and logically based AI systems towards neural networks and also made me get in contact with Anders Lansner.

Enchanting thanks to *Anders Holst* for his humble attitude and being a good play-ground for ideas and also for his ability to visualize even complicated mathematical issues in an easy way.

Next I would want to say a great thank to *Erik Fransén* and *Tom Wadden* who were my pleasant room mates at SANS for some years. Tom is always a great encouraging friend with a natural attitude towards life. I also want to thank him for improving my english and his work of reading some of my english manuscripts through. Erik has a delicate sense of humour that I appreciate and I really enjoyed the years in the choir *Blandade Röster* where Erik and I were singing. Erik also gave me an eternal wisdom which I recall now and then when I intend to save something that may become useful. He told me about his grand mother's attic where he found a box carefully labeled: "Snören för korta att användas" ("Strings to short to be used")...

Delightful special thanks to *Rogelio Melhado De Freitas* who took the initiative to bring UMC (*Uppsala Monitoring Centre*) together with the SANS group to create a new drug signalling system.

Lustrous thanks to *Hans Tråvén* who held the first SANS-seminar I attended, "Temporal Processing in Artificial Neural Networks" causing a *déjà vu* to make me realize that this is what I've been wanted to be doing all my life. He also introduced me to probabilistic mixture modelling. Thanks to *Örjan Ekeberg* who always has good ideas and is delightful to discuss with. Also thanks to *Jeanette Hellgren-Kotaleski* who is always positive and who I like to talk with, and to *Weimin Li* an always happy friend who learned me how to cook Chinese dumplings and to use a Chinese vegetable chopper. Thanks to *Joanna Tyrcha* a mathematician and a good friend who I dare to ask also the questions I dare to ask no one else. Thanks to *Jesper Tegnér* a pleasant positive mixture of a biologist and a mathematician. Thanks to *Fredrik Ullén* an excellent musician and a positive friend. I will always remember the *Transputer* cake I got on my 35th birthday. A surpassingly tasteful creation in marzipan and chocholate made by Fredrik, Mikael and Peter Lönnerberg. Fredrik had designed the truly realistic semi-sized brain in pink marzipan, Mikael and Peter the Transputer.

Yes, I am also grateful that *Anders Sandberg* helped me realize that I've always been a trans-humanist, without me knowing the name for it before. Anders has a good knowledge about almost anything and is luscious to discuss *e.g.* future technological potentials with. Thanks to *Björn Levin* for a lot of ideas about a lot of things, always a tie and he also appreciates the same *Monty Pyton* humour as I do. Thanks to *Hans Liljenström* for introducing me to several philosophical issues and to give me an insight about stochastic resonance. Thanks to *Per Hammarlund* for hinting me towards the *scm* interpreter of the scheme language which become the software basis for my first years of research, until I switched to *guile*. Thanks to *Tomas Wilhelmsson* always polite and helpful, also a formerly *Amiga* enthusiast as me. I also want to express a great thank to *Sten Grillner*, *Peter Wallén*, *Lennart Brodin* and all the people at the Neurophysiology department at Karolinska Insitute who brilliantly provided me with a fine knowledge about the nervous system.

Thanks a lot to *Guillemette Duchateau*, *Xiangbao Wu*, *Martin Wikström*, *Magnus Stensmo*, *Rosario Curbelo*, *Sverker Sikström*, *Anders Fagergren*, *Rafaella Valigi*, *Helen Fransson*, *Maria Ullström* and *Andreas Bergström*, as well as *Björn Randevik*, *John Holena*, *Hiroyoki Kataoka*, *Markus Svensén*, *Jordan Zlatev* and *Russell MacPherson*, you have all been wonderful.

Heartily thanks to *Lena Nachmansson* and *Pernilla Östlund* for being nice, kind and supportive and pleasant to discuss with. Thanks also to *Christer Westerholm* and all helpful people at NADA.

A great and cheerful acknowledge to *prof. Ralph Edwards*, *Sten Olsson*, *Marie Lindquist*, *Andrew Bate* and all the people at the Uppsala Monitoring Centre, who make our cooperative research work into a pleasent experience, I'm glad to work with you. Ralph with an insightful knowledge and humour; Sten, nice and pleasant to cooperate with; Marie, critical but positive and joyful; Andrew, always asking those tricky questions which help me find the way out of the fog. You at UMC are all great research partners. Thanks a lot!

Now I would also want to say thanks to *Håkan Loman* at STORA and *Vikram Kaul* at STFI. The initial study on prediction of paper quality using BP was funded by STORA. This work led to the development of a mixture modelling Bayesian neural network, which was funded by STFI.

Kindly thanks for the funding provided by the National Swedish Board for Technical Development (STU) under grant 89-01826P and also the support from Digital Equipment Corporation's European External Research Program (EERP) which is hereby gratefully acknowledged. These grants funded the works on temporal association and temporal segmentation.

Savoury thanks to *Staffan Ahlinder* at ABB Automation who supported my ideas about AI and expert systems, which later on led me to the insight that it was non linear systems with focus on neural networks and Bayesian statistics in particular that were my cup of tea.

Felicitous thanks to *Bertil Lundqvist*, *Leif Eriksson*, *Lars Cederblad*, *Murari Saha* and all the other people at ABB Relays who made this into a nice place to work at. Although I was working mostly part time at the research department there I did anyway appreciate this place very much. I also appreciate all kind of stimulating support I got.

Obeisant thanks to *Richard M Stallman* at MIT AI lab and all the other people who make a tremendeous effort in their work for the FSF (*Free Software Foundation*) and the GNU project. FSF has not only provided the highest quality software, FSF has also provided the humanity with the important insight that base software itself need to be free and not to be proprietary, to make it possible to easy exchange knowledge and information. Most of my research and development has been performed using GNU software. Lately mostly on GNU/Linux systems.

Ravishing thanks to *Linus Torvalds* and Transmeta and the internet development team that has made Linux into the high quality, extremely reliable, operative system kernel, it is today. Linus did what most computer science students used to dream of, to implement an operative system kernel, which is now being ported to more hardware architechtures than any other OS.

Awardful thanks to *Aubrey Jaffer* at MIT AI lab who provided the efficient *scm* interpreter which was later transmogrified into *guile* by FSF. This is the high level description language I use.

Lovely thanks to *Leila Orre* being a great life companion under many years and to provide us with two wonderful kids, Sebastian and Ottilia, all sharing a great respect for my visions and my ideas. I wish that Sebastian and Ottilia also get the opportunity to work with what they best like.

Lovely thanks also to *Kristina Berger* for being a nice and friendly life companion during some pleasant years. Thanks also to her nice daughters Hanna and Karen.

Faithful thanks to my sisters Solveig and Ruth who have always supported my ideas. Thanks also to all my relatives and other friends which were not forgotten, I'm thinking about you.

Uttermost I would want to thank both my parents *Gustav and Ingrid Orre* who provided me with a lot of love and a lot of spirit to prepare me to work for the important goals in life.

Now I just want to say to you all so far: A friendly thanks for all fun!

Preface, a historical perspective

When we collect data samples of the world to obtain information and hopefully knowledge about it we hope that this knowledge will make it possible for us to create a better world and to make ourselves and our products perform better. The samples we collect are usually fit into some statistical model for data analysis and possibly prediction of future outcomes. For a long time such statistical models were based upon what is often referred to as the classical statistical theory. The work we present here is based upon the ideas presented by Thomas Bayes and Marquis de Laplace. Bayes ideas [Bay63] were originally published postumely in 1763 by Richard Price, two years after his death. Bayes ideas were successfully used by Laplace who improved their mathematical appearance and showed with some elegant examples on the power of this new probability theory. For example, he used the probability theory to estimate the mass of Saturn. He actually computed the probability density function (pdf) for the mass of Saturn given data, *i.e.* $P(M|data, B)$, where B is the background information, like laws of mechanics. He also stated that "it is a bet of 11,000 to 1 that the error of this result is not 1/100 of its value". He would have won it! After 150 years later ackumulated data the estimations of the mass only changed the original estimate by 0.63%. Laplace discussed his ideas in a popular form in his *Essai philosophique sur les probabilités* 1814 (here referred to as [Lap14]) and finally published his analytical work *Théorie analytique des probabilités* within his collected work *Théorie* 1820. The probability methods soon become popular but unfortunately Bayes and Laplace works were later allegedly discredited and mostly forgotten until they were rediscovered by Jeffreys 1939 [Jeffreys, 1939] and has since then become inreasingly more and more popular. Although Laplace did a great work turning Bayes ideas into useful mathematical stringency the theory is nowadays called *Bayesian probability theory* to honour Bayes which came up with the original ideas.

In the work we present here we use a neural network method based upon this Bayesian probability theory for the data analysis (data mining) and process modelling (also prediction and classification). We here refer to this network as BCPNN *Bayesian Confidence Propagation Neural Network.*

Using classical statistical theory the data mining work, *i.e.* to find dependencies and relations within the data, and process modelling, has been done with different approaches and strategies. To find dependencies between variables, classical methods like the χ^2-test were developed and for process modelling have different regression models been used. A popular non linear regression method used lately is the error backpropagation network which is here referred to as BP.

In the BCPNN both the data analysis and the process modelling are done within the same framework using the same network for both purposes. We show here how such a network deal with variable dependencies and how we can put a belief value not only on the outcome of a variable but also on how significant this belief is. We also show how a predicted outcome also gets a confidence measure when belief values are propagated through this network.

In data mining, where we want to find dependencies between variables in a data base, the measure of the confidence of this dependency is important when the number of samples giving indications about it is small. This confidence not only makes it a useful tool for data mining but also for prediction and classification, especially when we need to make low risk decisions.

Contributions

This thesis consists of two parts. The first part is a summary and overview of the included papers with introductions to the problem areas including discussions about some of the results obtained. The second part consists of six supplements embodying the following papers appearing in non chronological order. They will be referred to in the text by their Roman numerals:

I ORRE R., LANSNER A.,BATE A. AND LINDQUIST M. (1998). Bayesian neural networks with confidence estimations applied to data mining - , *Submitted*

II BATE A., LINDQUIST M., EDWARDS I. R., OLSSON S., ORRE R., LANSNER A., AND FREITAS R. M. D. (1998). A bayesian neural network method for adverse drug reaction signal generation. *European Journal of Clinical Pharmacology* , *In press*

III ORRE R. AND LANSNER A. (1996). Pulp quality modelling using bayesian mixture density neural networks. *Journal of Systems Engineering* **6**: 128–136.

IV ORRE R. AND LANSNER A. (1990). A method for temporal association in bayesian networks. Tech. Rep. TRITA-NA-P9018, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden.

V ORRE R. AND LANSNER A. (1992a). A bayesian network for temporal segmentation. In *Artificial Neural Networks, Proceedings ICANN-92*, pp. 1081–1084. Royal Institute of Technology, Stockholm, Sweden, 1992.

VI ORRE R. AND LANSNER A. (1992b). A study of process modelling using artificial neural networks. Tech. Rep. TRITA-NA-P9239, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden.

Contents

Summary

1 Introduction

This thesis deals mainly with the subjects data mining and process modelling using *Bayesian Confidence Propagation Neural Networks* (BCPNN). Seen from a complexity level of the functionality of the two tasks the process modelling issue is the most complex one. From this point of view it is natural to start with the task about data mining, where an understanding of the subject will help in understanding the task about process modelling, where a general modelling will involve temporal processing as its ultimate goal.

Therefore, the thesis starts with a paper describing the theory on how weights and confidence measures are generated in BCPNN Supplement I . The important contribution of this paper is that it shows how we deal with uncertainties of the weights in the BCPNN network and how this uncertainty is useful when doing data mining.

The paper in Supplement II describes a successful application where the data mining technique described in Supplement I is used for data mining a huge data base, looking for adverse drug reaction signals.

The paper in Supplement III tells how the BCPNN network is used for modelling pulp quality in an industrial paper manufacturing process. The process modelling which is presented here mainly deals with the problem of real valued function approximation with confidence interval estimations. It should be stressed that what this network does is in principle an association of an input density function with an output density, which makes this method more powerful than any pure regression technique. The uncertainties which are dealt with here are only the uncertainties of the data samples, not the uncertainties of the weights within the network itself, which is addressed with in the previous paper Supplement I .

Now when we have made it through the chapters describing how the weights in the network are calculated and belief values are propagated we go into some temporal modelling issues. The paper in Supplement IV gives an idea of one way the BCPNN network can be used to handle the task of spatio-temporal function approximation which is an important issue for any kind of general process modelling involving both prediction and control as well as perceptive temporal tasks like speech understanding or the more speculative ones like prediction of future interest rates.

In the next paper in Supplement V we make an attempt to cope with the subject of self organizing temporal segmentation, which is an important issue when dealing with temporal problems. A temporal segment can be seen as a piece of sequential information that is often seen together and thus may have a certain meaning. The temporal segmentation problem can in some sense be seen as a temporal equivalent to the data mining problem as described in Supplement I .

The last paper in Supplement VI is included as a reference application for the process modelling task presented in Supplement III . In this work is instead a non linear regression model, the BP (error back propagation network) is used on mainly the same data set which was used in Supplement III . It deals with the subject of designing a BP network for best possible performance. It also discusses different methods to partition data into training and test sets.

It may be noted that, although the presentation order of the papers in this thesis is almost related to the complexity level of the task that we try to solve, the order is almost reverse from the chronological order of the production of the papers as well as from the level of my own understanding of the problem. The papers are instead presented in a more or less logical order, except for the last one. The reason to put this paper last is that it is merely a reference and is not focusing on the main subject of the thesis, which is to use BCPNN networks for the same problem.

2 On Data Mining

Data mining in one sense can be seen as the whole process of turning data into information and knowledge, thus involving searching for patterns, investigate possible representational forms, performing classifications, clustering, regression, function approximation, building inductive trees, finding logical rules describing the dependencies within the data set etc. [Fayyad *et al.*, 1966]. We do, however, live in a world where probability theory is necessary to understand all possible phenomenon in nature. Quantum physics, chemistry, economy, medicine as well as information processing in the brain. In a way there are no hard rules for anything, every possible process can be modelled from a probabilistic perspective.

Marquis de Laplace said "Strictly speaking it may even be said that nearly all our knowledge is problematical; and in the small number of things which we are able to know with certainty, even in the mathematical sciences themselves, the principal means for ascertaining truth - induction and analogy - are based on probabilities" [Lap14]. He indicated that the entire system of human knowledge is connected with probability theory. Therefore, we have here restricted the meaning of *data mining* merely to be the search for conditional dependencies and patterns within the data set.

From a Bayesian point of view, knowing these dependencies, or more precisely, to know the joint distributions for all events we are interested in is enough. Neither clustering or classification has nothing more than a subjective meaning. In nature there are no classes, no clusters, there are *similarities* but to define that something is similar we have to first define a distance measure according some metric in the space we are interested in, which is often a *Hamming distance* [Kanerva, 1988], an *Euclidean distance* or when we reason about probabilities it is, however, most often a graded [0..1] feature probability space [Kohonen *et al.*, 1991].

Nevertheless, there are a lot of practical reasons to cluster and classify data. It is fundamentally for us to organize data into classes to be able to make use of the information therein. We should never forget although, that clustering and classification are graded hierarchical processes that are just done for practical and subjective reasons.

Ok, now to the real subject of data mining with probabilities. First we have to find out what we mean by a probability. Assume that we, for simplicity has a binary variable, like a coin with head and tail. We throw this and count the number of occurrences for head=c_1 and the number of occurances for tail=c_0. This is a typical Bernoulli trial. By the classical definition the probability p_1, which is then the probability to get head, for an event to occur is the ratio:

$$p_1 = \lim_{c_0+c_1 \to \infty} \frac{c_1}{c_0 + c_1} \tag{1}$$

From a practical point of view, a probability is something that can never be measured exactly. To really find out the probability for a certain event we would have to repeat an experiment an infinite amount of times, which is really not encouraging. Then for instance to speak about a probability for something that has never occurred would be meaningless. When doing data mining we also want to know how significant a certain connection found is. We want to find some *quality measure* of how sure we are that there really is a dependency when we have found evidence of one. For certain applications we may also want to do that a early as possible with as small data sets as possible, for instance to be able to give a warning signal when we have found a relation which would be considered important.

Now, when using Bayes ideas there is hope. The Bayesian technique to calculate probabilities starts with a prior distribution. Each new sample generates a posterior distribution, which is used as the prior for the next sample. We assume that the probability p we are looking for is just an unknown parameter with a distribution θ, then when we have a model for this distribution we can find the unknown distribution by integrating over the unknown parameter and normalize. It is, however, unclear exactly how Bayes himself did this calculation. It is traditionally accepted that Bayes assumed that the distribution θ was a priori uniform in $[0, 1]$ [Geisser, 1993], hence the posterior density of θ is:

$$p(\theta|n) \propto \theta^n (1-\theta)^{N-n}$$

We do not know for sure if Bayes calculated the predictive probability for the next sample himself. Anyway did Price, who posthumously communicated Bayes work, do this. He found the probability for a binary variable to be:

$$P(X_{[N+1]} = 1) = \frac{n+1}{N+2}$$

which is the everyday Bayes formula used to calculate simple probabilities for binary variables where N is the total number of samples and n is *e.g.* the number of ones obtained from a Bernoulli trial. The ratio $1/2$ here indicates that we started with a uniform density.

In Supplement I we show rather detailed how this calculation is being done and we also describe how we use the same technique to calculate the variance of the posterior distribution for each new sample as well, which works exactly for single and joint probabilities. We also tell somewhat about how the formula generalizes to multiple valued variables and how to guess an a priori distribution in some specific case.

These probabilities and their variances can be calculated analytically exact but for the more complicated expressions as the weights in the Bayesian neural network, which we also refer to as the *information component* (IC) and the predicted posterior probabilities we have, so far, used the Gaussian approximation formula for the variance of a function. During the progress of this work we did, however found that for the (IC), which is logarithmic, this expression can be done exactly as well [Koski and Orre, 1998].

In the first presented paper on data mining Supplement I we start by giving a brief description of the Bayesian neural network and then we focus on the principles for estimation of the different probability measures needed, as well as the variances of their distributions.

We do, however, also deal with other data mining issues in Supplement I like finding the most dependent combinations of variables in a huge data set. Generally this is an NP-complete problem (can *not* be solved within *polynomial* time due to the huge amount of combinations to be searched through. Despite the size we may still look for combinations of variables up to a certain complexity level. To search for all possible combinations up to a certain order, would of course easily turn into an undo-able task due to combinatoric explosion. However, when the cases to be investigated show a sparse occurrence of variable values, which is the case with the data base of adverse drug reactions we are working with in Supplement I and Supplement II , then this can be done efficiently up to a rather high order of combinations as the combinatoric complexity of the problem is kept down to a reasonable level. We use a sparse matrix technique here, that only creates those input/output units that actually occur in data, then creates the connections needed when combining the so found variables in all possible combinations. The sparse matrix technique for looking for combinations is also briefly described in Supplement I .

1 The problem addressed

The main problem addressed in Supplement II is fundamentally to search for new unexpected dependencies between variables in the data base and produce lists of the most likely dependency connections found and report these to be treated by human experts. The fundamental issue here is to reduce the amount of information to be treated by the experts, to reduce the number of reports from about 35000 to about 100. The BCPNN software sees a large matrix of data, where each row in the matrix corresponds to one case report. The potential size of this matrix is rather large as for instance the number of reported drugs so far is over 12000 and the number of reported adverse drug reactions is almost 2000.

The BCPNN software we are developing is being used in a data mining project in cooperation with Uppsala Monitoring Centre for Adverse drug reactions, the WHO Collaborative Centre for International Drug Monitoring. This is an international centre which maintains a huge data base *INTDIS* over adverse drug reactions from all over the world. INTDIS contains anonymous case reports of suspected adverse drug reactions. For each report there may be up to around 70 different variables telling about the patient age, country, sex, drugs taken, drug amounts, adverse drug reactions, suspected drugs, etc. There is also information about drug substance contents. It is the largest database of this sort in the world, at the time of writing it contains about two million reports, and about 35 000 new reports are added quarterly. There are 47 national centers which collect reports in their own country and send them to Uppsala Monitoring Centre. The task of trying to find new drug-ADR signals has been carried out by an expert panel, but with such a large volume of material the task is daunting. We have instead developed a flexible, automated BCPNN-based procedure to find new signals with known probability difference from the background data.

2 Implementation Issues

The INTDIS database of adverse drug reactions is a relational data base organized as a set of SQL tables, now containing 73 variables. A data base of this size put certain requirements on the software accessing it. During the design of the system we divided the solution into different modules each concerned with different parts of the problem. The main modules were the data base accessor, data preprocessor, layer module, ANS scan module, a calculation and analysis module and a module for interactive visualization of query results. Our original idea was to use a standard data base tool for the data base accesses and use a preprocessor to transform the output from this data base accessor into a form suitable for our neural network software. We contacted a manufacturer of a well known data base software and were allowed to try this program package to use as a data generator to the neural network software. Unfortunately, after a great effort had been put in getting the data base accessor to produce data in the way we wanted, it showed that this could not be done. This way of doing data generation was too slow in a round-about way. Both the data generation and the preprocessing of the data required too much computer resources to be useful. The main problem was that for an efficient training of the neural networks we had to access the data as objects, and not in the relational data base way. Therefore we made our own accessor that could be optimized for its task, to scan the data base efficiently and transform the data into a suitable form to be presented for the BCPNN networks. The data preprocessor was no longer needed and the accessor could now directly transfer data from SQL tables to the ANN layers. We did no longer need a huge intermediate storage of data either. We gained *several hundred times* in speed up and the training times now become reasonable on a unix (Sun Sparc) work station.

The training time, when we also collect statistics about the variables as matrixes of dependencies vary a lot depending on how many variables are used and to what extent variables are combined into complex units. A simple query takes a couple of hours and a complicated one may take 7-8 hours. For a while we tried to run the software on a super-computer to decrease the training times. This was a parallel executable MIMD machine with a lot of CPUs and a lot of memory in each CPU. We did, however, find that the waiting queues to this machine was to long to make this efficient so we continued working on the work station solution. At the moment we are running most of the searches and analysis on a dual CPU 300 MHz Pentium II machine equipped with 512 Mb of memory [1] , which runs the GNU/Linux operative SMP (*synchronous multi processing*) system. Scheme [WC91] is used as high level scripting language, in this case a development version of *Guile*. All low level speedy processing is done in ANSI-C/C++.

3 Process Modelling

Imagine that you are freezing and you want to know what temperature it is around you to find out whether you are getting sick or it is just cold in the room. You happen to find a thermometer but accidentaly the one you found just gives you degrees Fahrenheit and as you are from Sweden you have no idea what these values mean. Then you remember that the references for Celsius temperature is melting ice and boiling water. With some experiments with ice from the fridge and some boiling tea water you have soon solved the problem and now have a mapping from Celsius to Fahrenheit. You may satisfy with just two measurements but you may also be an ambitious person that don't trust a single measurement and get enthusiastic when you found your pocket calculator. Now you can do some repetitive experiments and some linear regression. You will soon find out that the functional mapping between the two scales is $C \approx 0.55 * F - 18$. When you later find a book in your bookshelf telling you that the conversion rate is $C = (F - 32) \cdot 5/9$ you will be proud of your self.

Now imagine that you have an industrial process, like a pulp to paper manufacturing process. In the same way as with the thermometer above, which you didn't know the meaning of, you get a lot of measurement values telling you different values. What you really want to know is the paper quality outcome measured in some easy to understand values as tear, tensile etc. With the same technique as above using some multi-variate regression technique you may find some nice polynomial expressions telling you the functional relations between measured value and outcome values.

After a while you realize that it would be nice to know some more. You have found that the so predicted outcome is not always correct. You suspect that the reason may be that there are some *unknown variables* that you may not know. It may also be that in some regions of the space the functional relation is much more complicated than you could

[1] for computer geeks I can tell that it is built into an attache case which also has place for a lap as X-term

foresee when you created your multi variate polynomial expression, maybe the polynomial order was not high enough? On the other hand you believe that you could not do better with a more complicated polynomial because you didn't have that much data to use for the regression because those samples were expensive to make. If you make the expression more complicated you may risque to get over-fitting of the polynomial. Instead it would be nice to get some idea about when you can expect a high accuracy and when you can expect a low accuracy.

Now you get the brilliant idea that instead of mapping the values you map the uncertainties of the values instead. You remember some more from your course in statistics and imagines that each measured value comes from a certain probability density function, as does the whole data set. You realize that the whole density for the data set is rather complicated but you think that it would be possible to see the whole distribution as composed of a large set of small Gaussian density functions. You take a set of these Gaussians and let them fit to a training data set as good as possible. Then what is as good as possible in this case? Well, you have heard about something called the maximum likelihood, *i.e.* just place these Gaussians, in a way that maximizes the probability that your data has come from those Gaussian probability density functions (or processes).

Now when you get a new measured value you can calculate the probabilities that the sample is coming from any of these Gaussian. You then do a linear associative mapping to get the probabilities for the Gaussian in output space to be active. Now you want an actual value as output, not a probability density function, but for that purpose you calculate the joint expectation value for those output Gaussian you have activated, which is very easy in this case, it is the same as their center of mass. And, finally, now you can integrate over these output density functions and calculate a confidence interval for your predicted outcome.

A very general problem considers the issue of functions, values in one space, mapping to values in another space. This may be as easy as a linear conversion of temperature $C = (F-32)\cdot 5/9$ or classifying a Bach fugue being as written by Bach as well as recognizing grandmothers face in a crowd of people. Any mapping from one space to another which gives the same result each time and does not contain a memory or some internal states, can be considered a function.

In most regression models used for function approximation like the multilayer BP network, the output is a real value. In the mixture density model presented in Supplement III here the output is instead an approximation of the density function for a response variable conditioned by a certain explanatory variable value. This density function gives information about the a posteriori density which, in addition to the predicted value, can also give e.g. a confidence interval (assuming uni-modality). Observe that this works for multi-modal outcomes too, but then you may be interested in several smaller confidence intervals, not one huge. Both the input and output variables density functions are modelled my mixing Gaussian density functions. The Gaussian used are called RBF (*radial basis functions*). It is, however, some regression going on here, the adaptation of the component density functions used is similar to *kernel regression*, even though the criterion is somewhat different from the on used in multivariate regression techniques.

Advantages of doing function approximation with the mixture density model, compared with a regression technique as BP are:

1. The RBF representations of variable spaces are built unsupervised, which means that cheap unlabeled examples can be used.

2. The generalization, can be dynamically improved due to the regularization capabilities of the RBF:s which decrease the requirement of cross validation.

3. Design of the network is almost parameter free as the relation between number of training examples and preferred maximum number of RBF:s is simple.

4. The supervised training part is quick and gives good results using a one shot learning process.

5. The predicted mixture density for response variables gives, besides the ability to estimate a confidence interval, also the ability to detect ambiguous output values, i.e. a multi modal density function.

6. A missing value in an input sample vector is still useful as this only leads to a less specific conditioned a priori density in the missing dimension.

In the presented work we has concentrated on the function approximation problem and have applied this on process modeling . We have tested methods for filling in of missing input values and to do both training and prediction with data containing missing values. The tested method used for handling missing data gave no improvement during training but we obtained useful function values when data was missing during prediction.

It may be noted that spatio temporal patterns can still be modelled with a functional mapping network by adding time delays and, if necessary, state memories.

4 Temporal Modelling

When we deal with many real world problems we will find that there are a lot of them where the care for time is essential. In economics we want to do forecastings of interests and prices and it is most often so that to be able to forecast the next value in a series of values we need to know the behavior of the values for some period of time lately. In process industry the most processes are dynamical in some sense, there may be parts of the process which involves states, the process may have a kind of memory and will not select the next state purely on the instantaneous value, instead the next value vill be a function of which states the process has visited lately and there may be charging/discharging events going on, which may give a complicated temporal behavior.

The fundamental problem to be dealt with is learning/recognition of simple sequences. Assume that we have some kind of temporal associative memory where we can store sequences like:

```
``One sentence of letters''
``Another sentence of letters''
``A third sentence of characters''
```

Now we can imagine that we are able to recall these sequences by giving, *e.g.* the start of each sentence. It may be clear that given *e.g.* the following starting "tags":

```
``On....................''
``An........................''
``A t..........................''
```

we will have no problem of recalling up to:

``One sentence of ?......''
``Another sentence of ?.....''
``A third sentence of ?........''

Now, the continuation of these sequences will depend on the properties of the network. If the network was able to remember rather long sequences then it may manage to come up with the correct continuations here, but otherwise it may instead choose the most probable one ``letters''. This kind of problems is dealt with in Supplement IV .

Now, let us imagine that you are going to build a system for automatic speech recognition, but you don't want to tell the system in detail about all the words and delimiters yourself. As you are certainly aware of there is no delimiters in natural speech, you have to find these yourself. So, instead of telling the system what to learn and what to look for we just present large chunks of data for it and let the system decide on its own how to best find where a phoneme or word starts. Yes, both phonemes and words share the same problem. As all classification tasks it is an hierarchical issue. At one low level we have phonemes, at another higher level we have words. The issue of finding these parts of the patterns is the segmentation problem. The segmentation problem is fundamental in pattern recognition. We wanted to find a simple method that could perform automatic segmentation on any temporal sequence, as the one referred above contains at least two classification levels we tried this method on a toy problem instead. In this case we used words from a computer dictionary that were put together in random sequences of words.

The goal with the work which is presented in Supplement IV thus was to detect higher level items in an unlabeled sequence of data. Given data with a sequential/temporal behavior this shows up as the temporal chunking problem which may be illustrated by the example:

`thisisacontinuousstreamofdatawhichispossibletoreadwithoutseparators`

Here, we want unfamiliar lists of familiar items (characters) presented sequentially to be recognized as new items (words). In the first place just the characters are familiar. When we have seen different lists several times we will also recognize the words as familiar items. The method presented here detects segmentation points between words. Conceptually this means that we have grouped a sequence of elementary items into a new, composite item. The proposed method uses a Bayesian learning scheme in the form of a temporal associative memory earlier investigated where the relaxation scheme is modified with a few extra parameters, a pairwise correlation threshold and a pairwise conditional probability threshold.

The method we investigated was able to find start and end positions of words in an unlabeled continuous stream of characters. The robustness against noise during both learning and recall was studied. We got clear indications that the level of the threshold for the correlation[2] (*information component*) was the most useful way to automatically detect segments with this method.

5 On Regression Modelling

An important issue in process industry is the prediction of certain response variables from a process given a proper subset of available explanatory variables for that process. This

[2]later on we used the name *information component* (IC) for this measure to avoid ambiguity problems

prediction may be desired for several reasons. An actual response variable from a process may be hard or time consuming to measure. It is also desired to check the result of a change in some of the explanatory variables for a process before actually doing it.

One part of this project was done as a cooperative work between SANS and the paper industry (STORA Teknik AB). It was studied how a set of laboratory data, the result of time consuming laboratory experiments, could be predicted from a set of explanatory variables for a process. A method was developed which searched for the minimal architecture that was able to predict a certain laboratory value. The reason for trying to find a small architecture was that the data set available for training may be too small to allow for networks with a high dimensionality. The method also arranges the inputs according to the impact they have on prediction performance. This makes it possible to further reduce the network dimensionality and increase prediction quality by removing irrelevant parameters. The model will continually adapt to changes in the process behavior.

We found that in such a case like this, where the amount of data was rather limited, it was preferable to make one optimized network (*i.e.* non linear regression model) for each variable to model. As another result of this work we got valuable experience dealing with real data where we also developed a lot of methods for analysis, cross validation in BP-networks and automatic presentation of results graphically and in table form.

6 Conclusions

The most important results and findings of this thesis can be summarized in the following points:

- We show how a Bayesian Neural Network can be extended to model the uncertainties in collected statistics to produce outcomes as distributions from two different aspects: uncertainties induced by sampling, which is useful for data mining; uncertainties due to input data distributions, which is useful for process modelling.
- We show how complex dependencies can be found within large data sets but still avoiding combinatoric explosion.
- We show how these techniques have been turned into a useful tool for real world applications within the drug safety area in particular.
- We compare some results of the BCPNN technique with the well established non linear regression technique, BP (back prop networks), for processing modelling, showing that the BCPNN performs at least equally well, but provides extra information about uncertainties of produced outcomes.
- We present a simple but working method for doing automatic temporal segmentation of data sequences.
- We indicate some aspects of temporal tasks for which a a predictive Bayesian neural network may be useful. Showing how the connection matrix of the network can be reduced due to regularities in the data.

The work presented in this thesis has given us several useful methods and experiences. We now have a working method in development which is adapted towards real world application usage of these Bayesian ANN methods. This research has, in particular, given us a method for data mining, classification and prediction where huge amounts of data is involved. The application we address with this method will be a help in drug safety to perform quick and efficient analysis of adverse drug reaction reports. Although, the method is inherently general and can, as have been shown here, be applied to many different application areas and problem types.

Bibliography

[Abu-Mostafa, 1989] Abu-Mostafa Y. (1989). The vapnik-chervonenkis dimension: Information versus complexity in learning. *Neural Computation* **1**: 312–317.

[Barnard and Casasent, 1989] Barnard E. and Casasent D. (1989). A comparison between criterion functions for linear classifiers, with an application to neural nets. *IEEE Trans on Systems, Man and Cybernetics* **19**: 1030–1041.

[Bate *et al.*, 1998] Bate A., Lindquist M., Edwards I. R., Olsson S., Orre R., Lansner A., and Freitas R. M. D. (1998). A bayesian neural network method for adverse drug reaction signal generation. *European Journal of Clinical Pharmacology* , in press.

[Bayes, 1763] Bayes T. (1763). An essay towards solving a problem in the doctrine of chances. *Biometrika (reprint 1958 of orig art in Philos. Trans. R. Soc. London 53, pp. 370-418)* **45**: 296–315.

[Benaim, 1994] Benaim M. (1994). On functional approximation with normalized gaussian units. *Neural Computation* **6**: 319–333.

[Bernado and Smith, 1994] Bernado J. M. and Smith A. F. (1994). *Bayesian Theory*. John Wiley & Sons, Chichester.

[Bishop, 1994] Bishop C. M. (1994). Mixture density networks. Tech. Rep. NCRG/4288, Department of Computer Science, Aston University.

[Doutriaux and Zipser, 1990] Doutriaux A. and Zipser D. (1990). Unsupervised discovery of speech segments using recurrent networks. In Touretzky D., Elman J., Sejnowski T., and Hinton G. (eds.), *Proceedings of the 1990 Connectionist Models Summer School*, pp. 52–61. Morgan Kaufmann, San Mateo.

[Fayyad *et al.*, 1966] Fayyad U. M., Piatetsky-Shapiro G., Smyth P., and Uthurusamy R. (eds.). (1966). *Advances in Knowledge Discovery and Data Mining*. MIT press, Menlo Park.

[Geisser, 1993] Geisser S. (1993). *Predictive Inference: An Introduction*. Chapman & Hall, New York.

[Ghahramani, 1994] Ghahramani Z. (1994). Solving inverse problems using an em approach to density estimation. In Mozer, Smolensky, Touretzky, Elman, and Weigend (eds.), *Proceedings of the 1993 Connectionist Models Summer School*, pp. 316–323. Erlbaum Associates, Hillsdale 1994.

[Good, 1950] Good I. J. (1950). *Probability and the Weighing of Evidence.* Charles Griffin, London.

[Grillner *et al.*, 1987] Grillner S., Wallén P., Dale N., Brodin L., Buchanan J., and Hill R. (1987). Transmitters, membrane properties and network circuity in the control of locomotion in lamprey. *Trends in Neuroscience* **10**: 34–41.

[Grossberg, 1984] Grossberg S. (1984). Unitization, automaticity, temporal order and word recognition. *Cognition and Brain Theory* **7**: 263–283.

[Heckerman, 1997] Heckerman D. (1997). Bayesian networks for data mining. *Data Minining and Knowledge Discovery* **1**: 79–119.

[Henann N.E., 1986] Henann N.E. M. J. (1986). Suprofen - induced acute renal failure. *Drug Intell Clin Pharm* **20**: 860–862.

[Hertz *et al.*, 1991] Hertz J., Krogh A., and Palmer R. G. (1991). *Introduction to the Theory of Neural Computation*, volume 1. Addison-Wesley, Redwood City.

[Holst, 1997] Holst A. (1997). *Using Bayesian Neural Networks for Classification Tasks.* PhD thesis TRITA-NA-P9708, Royal Institute of Technology, Stockholm, Sweden, Dept. of Numerical Analysis and Computing Science.

[Holst and Lansner, 1993a] Holst A. and Lansner A. (1993a). The Bayesian neural network model and its extensions. Tech. Rep. TRITA-NA-P9325, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden.

[Holst and Lansner, 1993b] Holst A. and Lansner A. (1993b). A flexible and fault tolerant query-reply system based on a Bayesian neural network. *International Journal of Neural Systems* **4**: 257–267.

[Holst and Lansner, 1995] Holst A. and Lansner A. (1995). A higher order bayesian neural network for classification and diagnosis. In *Applied Decision Technologies: Computational Learning and Probabilistic Reasoning*, pp. 251–260. London, England, April 3-5.

[Hopfield and Tank, 1987] Hopfield J. and Tank D. W. (1987). Neural computation by concentrating information in time. *Proc. Natl. Acad. Sci. USA,* **84**: 1869–1900.

[Jeffreys, 1939] Jeffreys H. (1939). *Theory of probability.* Clarendon Press, Oxford.

[Jordan, 1986] Jordan M. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pp. 531–546. Lawrence Erlbaum, Hillsdale, Amherst 1986.

[Kandel and Schwartz, 1985] Kandel E. R. and Schwartz J. H. (1985). *Principles of Neural Science.* Elsevier Science Publishing Co, Inc, New York. 2 edition.

[Kanerva, 1988] Kanerva P. (1988). *Sparse Distributed Memory.* MIT Press, Cambridge.

[Kohonen, 1988] Kohonen T. (1988). *Self-Organization and Associative Memory.* Springer-Verlag, Berlin. 2 edition.

[Kohonen *et al.*, 1991] Kohonen T., Mäkisara K., Simula O., and Kangas J. (eds.). (1991). *Artificial Neural Networks*, Elsevier science publishing company B.V., Espoo, Finland. Proceedings of ICANN-91.

[Kolassa, 1997] Kolassa J. E. (1997). *Series Approximation Methods in Statisticss.* Springer-Verlag, New York.

[Kononenko, 1989] Kononenko I. (1989). Bayesian neural networks. *Biol. Cybernetics* **61**: 361–370.

[Koski and Orre, 1998] Koski T. and Orre R. (1998). Statistics of the information component in bayesian neural networks. Tech. Rep. TRITA-NA-9806, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden.

[Lang J.K., 1988] Lang J.K. H. G. (1988). A time-delay neural network architecture for speech recognition. Tech. Rep. CMU-CS-88-152, Dept. of Computing Science, Carnegie-Mellon University, PA.

[Lansner, 1991] Lansner A. (1991). A recurrent Bayesian ann capable of extracting prototypes from unlabeled and noisy examples. In *Artificial Neural Networks, Proceedings ICANN-91*, pp. 247–254. Royal Institute of Technology, Stockholm, Sweden, 1991.

[Lansner and Ekeberg, 1985a] Lansner A. and Ekeberg Ö. (1985a). Reliability and speed of recall in an associative network. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **7**: 490–498.

[Lansner and Ekeberg, 1985b] Lansner A. and Ekeberg Ö. (1985b). Reliability and speed of recall in an associative network. *IEEE Trans on Pattern Analysis and Machine Intelligence* **7**: 490–498.

[Lansner and Ekeberg, 1987] Lansner A. and Ekeberg Ö. (1987). An associative network solving the “4-bit adder problem”. In *Proceedings of the IEEE First Annual International Conference on Neural Networks*, pp. II–549. San Diego, USA.

[Lansner and Ekeberg, 1989a] Lansner A. and Ekeberg Ö. (1989a). A one-layer feedback artificial neural network with a bayesian learning rule. *Int. J. Neural Systems* **1**: 77–87.

[Lansner and Ekeberg, 1989b] Lansner A. and Ekeberg Ö. (1989b). A one-layer feedback, artificial neural network with a Bayesian learning rule. *International Journal of Neural Systems* **1**: 77–87, Also extended abstract in Proceedings from the Nordic symposium on Neural Computing, April 17–18, Hanasaari Culture Center, Espoo, Finland.

[Lansner *et al.*, 1989] Lansner A., Ekeberg O., Tråvén H., Brodin L., Wallén P., Stensmo M., and Grillner S. (1989). Simulation of the experimentally established segmental, supraspinal and sensory circuitry underlying locomotion in lamprey. *Soc. Neurosciene Abstr* **15**: 1049.

[Lansner and Holst, 1996] Lansner A. and Holst A. (1996). A higher order Bayesian neural network with spiking units. *Int. J. Neural Systems* **7**: 115–128.

[Laplace, 1814] Laplace P. S. (1814). *A Philosophical Essay on Probabilities.* Trans 1995 from the 5th ed 1825 (orig 1814) by Andrew Dale. Orig ”Essai Philisophique sur les Probabilités”, Dover Publications, New York.

[Leighton, 1991] Leighton R. R. (1991). The aspirin/migraines software tools, users’s manual, v5.0. Tech. Rep. MP-91W000050, The MITRE Corporation.

[Maass, 1992] Maass W. (1992). Bounds for the computational power and learning complexity of analog neural nets. Extended Abstract.

[MacKay, 1992] MacKay D. J. (1992). A practical Bayesian framework for backprop networks. *Neural Computation* **4**: 448–472.

[MacKay, 1994] MacKay D. J. (1994). Bayesian neural networks and density networks. Proc. of Workshop on Neutron Scattering Scattering Data Analysis 1994.

[Massone L, 1989] Massone L B. E. (1989). A neural network model for limb trajectory formation. may be submitted, currently don’t know.

[McClelland *et al.*, 1986] McClelland J., Rumelhart D., and the PDP Research Group. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 2. MIT Press, Cambridge.

[McClelland *et al.*, 1988] McClelland J., Rumelhart D., and the PDP Research Group. (1988). *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs and Excercises.* MIT Press, Cambridge.

[Miller *et al.*, 1990] Miller W. T., Sutton R. S., and Werbos P. J. (1990). *Neural Networks for Control.* MIT Press, Cambridge.

[Nelson and Ilingworth, 1991] Nelson M. M. and Ilingworth W. (1991). *A Practical Guide to Neural Nets.* Addison-Wesley, Massachusetts.

[NeuralWare, 1990] NeuralWare I. (1990). *Neural Computing.* Penn Center West, Pennsylvania.

[Nielsen, 1990] Nielsen R. H. (1990). *NeuroComputing.* Addison-Wesley, Massachusetts.

[Olsson S, 1985] Olsson S Biriell C B. G. (1985). Photosensitivity during treatment with azapropazone. *Br Med J* 939.

[Orre and Lansner, 1992] Orre R. and Lansner A. (1992). A study of process modelling using artificial neural networks. Tech. Rep. TRITA-NA-P9239, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden.

[Orre and Lansner, 1994] Orre R. and Lansner A. (1994). Function approximation by prediction of a posteriori density with bayesian ann:s. Tech. Rep. TRITA-NA-P9413, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden.

[Orre and Lansner, 1996] Orre R. and Lansner A. (1996). Pulp quality modelling using bayesian mixture density neural networks. *Journal of Systems Engineering* **6**: 128–136.

[Parsons, 1987] Parsons T. W. (1987). *Voice and Speech Processing.* McGraw-Hill Book Company, New York.

[Pearl, 1988] Pearl J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo.

[Rosenblatt, 1962] Rosenblatt F. (1962). *Principles of Neurodynamics.* Spartan, New York.

[Rumelhart *et al.*, 1986] Rumelhart D., McClelland J., and the PDP Research Group. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge.

[Schmiduber, 1991] Schmiduber J. (1991). Neural sequence chunkers. Tech. Rep. FKI-148-91, Institut für Informatik, Technische Universität München.

[Shannon, 1951] Shannon C. (1951). Prediction and entropy of printed english. *Bell System Technical Journal* **30**: 50–64.

[Tråvén, 1988] Tråvén H. (1988). Temporal associative memory. Tech. Rep. TRITA-NA-P8802, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden.

[Tråvén, 1991] Tråvén H. G. (1991). A neural network approach to statistical pattern classification by "semiparametric" estimation of probability density. *IEEE Transactions on Neural Networks* **2**: 366–118.

[Tråvén, 1993a] Tråvén H. G. (1993a). Invariance constraints for improving generalization in probabilistic neural networks. In *IEEE ICNN'93*, pp. 1348–1353.

[Tråvén, 1993b] Tråvén H. G. (1993b). *On Pattern Recognition Applications of Artificial Neural Networks.* PhD thesis TRITA-NA-P9318, Royal Institute of Technology, Stockholm, Sweden, Dept. of Numerical Analysis and Computing Science.

[W. Clinger, 1991] W. Clinger J. R. e. a. (1991). Revised^4 report on the algorithmic language scheme.

[Waibel *et al.*, 1989] Waibel A., Hanazawa T., Hinton G., Shikano K., and Lang K. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing* **37**: 328–339.

Supplement I

Bayesian Theory of Data Mining

BAYESIAN NEURAL NETWORKS WITH CONFIDENCE ESTIMATIONS APPLIED TO DATA MINING

Roland Orre, Anders Lansner

SANS, Dept. of Computer Science, Royal Institute of Technology, S-100 44 Stockholm, Sweden

Andrew Bate, Marie Lindquist

WHO Collaborating Centre for International Drug Monitoring,
Uppsala Monitoring Centre (UMC), S-753 20 Uppsala, Sweden

SUMMARY

An international database of case reports, each one describing a possible case of adverse drug reactions, is maintained by the UMC. There is a large set of reports, arriving in quarterly year batches. Each report is here seen as a row in a data matrix, and it consists of a number of variables, like drugs used, adverse reactions, and other patient data. The problem is to extract significant apparent dependencies to report back to medical research and practice. This is done by comparing estimated frequencies of combinations of variables with the frequencies that would be predicted assuming there were no dependencies. The estimates of significance are obtained with a Bayesian approach via the variance of posterior probability distributions. The posterior is obtained by fusing a prior distribution with a batch of data, and it is also the prior used when the next batch arrives. As an example, when estimating a probability of a binomial distribution with frequency counts, it is convenient to use a Beta distribution as prior. So, using $Beta(a_0, a_1)$ as prior, obtaining a new batch with counts c_0 and c_1, the posterior will be a new Beta distribution with easily computed parameters: $Beta(a_0 + c_0, a_1 + c_1)$. For the *non-informative* choice of a uniform prior $Beta(1, 1)$, this gives us the famous Laplace rule of succesion. When considering dependencies between n variables, it is natural to use Dirichlet distributions of dimension 2^{n-1} as priors and (since they are conjugate to the multinomial distributions) posteriors. When deciding whether the joint probabilities of events is different from what would follow from the independence assumption, the '*information component*' $\log(P_{ij}/(P_i P_j))$ plays a crucial role, and one main technical contribution reported here is an efficient method to estimate this measure, as well as the variance of its posterior distribution, for large data matrices. We also present an efficient way of searching for higher order combinations in this type of data avoiding the effects of combinatoric explosion. Following established practice in the area of Bayesian neural networks, we use $E(P(A|D))$, $V(P(A|D))$, etc to denote the mean and variance of the posterior distributions of $P(A|D)$.

1 INTRODUCTION

The Bayesian neural network we use here is a feed forward network [1] where the learning and inference rules are based upon Bayes rule [Bay63] ,[Lap14] for conditional probabilities, together with an assumption of independence between inputs gives, for an outcome A conditioned by D, being composed of n *independent* subevents d_i

$$P(A|D) = P(A)\frac{P(D|A)}{P(D)} \propto P(A)\frac{P(d_1|A)}{P(d_1)}\frac{P(d_i|A)}{P(d_i)}\cdots\frac{P(d_n|A)}{P(d_n)} \tag{1}$$

The outcome A may in general be a continuous distribution, but here we deal with discrete outcomes. In the following A means the set of mutually exclusive outcomes $a_1, a_j, \ldots, a_m$. We use the symbol "A" as this often represents *adverse drug reactions* or a combinations thereof in our application. The input data d_i most often represents drugs or combinations of drugs.

Such a network is fundamentally a naive Bayesian classifier [Goo50] but it has earlier been extended with higher order units that deals with classification and diagnosis also for tasks involving dependent inputs [LH96], where it was denoted BCPNN (Bayesian Confidence Propagation Neural Network). Here we extend the latter by calculating also the variance

$$V\left(\frac{P(d_i|A)}{P(d_i)}\right) \tag{2}$$

which is a particularly useful measure, for instance, when we do data mining on small data sets, where the uncertainty may be large, and $V(P(A|D))$ as well, which gives us a confidence measure of a prediction or classification. The Bayesian feed forward neural networks remind somewhat about Bayesian Belief Networks [Pea88] and they can theoretically be transformed into each other [HL95]. The main difference is that in the latter only the dependent variables are dealt with in each node, whereas in the neural network model both dependent and independent variables are treated in parallel and the result is propagated through a few layers only.

1 Bayesian Inference in BCPNN

Let event D in (1) consist of n *independent* sub-events $d_1, d_2 \ldots d_n$ and replace $P(d_i|A)$ with $\frac{P(d_i,A)}{P(d_i)}$ (definition): which gives the *posterior* probability $P(A|d_1, d_i \ldots d_n)$:

$$P(A|D) = P(A|d_1, d_i \ldots d_n) \propto P(A)\frac{P(d_1, A)}{P(d_1)P(A)}\frac{P(d_i, A)}{P(d_i)P(A)}\cdots\frac{P(d_n, A)}{P(d_n)P(A)} \tag{3}$$

The "$\propto$" in (3) is there because the output has to be normalized, to make mutually exclusive probabilities sum to one [2]. The next step is to partition the input space into independent subspaces, which will be used as "hidden" units. In the most extreme case we could have a unique coding for every possible combination of input variable values, but this would be very wasteful on the network size and cause extremely poor generalization. One thing which has showed to be generally useful ([LH96],[HL95]) is to code all types of

[1] is has also been used as a recurrent Hopfield-like network useful for pattern completion [LE85],[LE89],[Kon89]

[2] For the recurrent model thresholding instead of scaling has also been used [LE89]

variables; binary, discrete and real valued; into "hypercolumns", *i.e.* separate input layers for each variable coding mutually exclusive events. A binary variable is thus represented by one "on-unit" and one "off-unit". Therefore, let us now represent each event by d_i and its *complementary* event $\bar{d}_i$, where π_{d_i} is the ($\pi_{d_i} \geq 0$, $\pi_{d_i} + \pi_{\bar{d}_i} = 1$) belief on event d_i:

$$P(A|D) \propto P(A) \prod_i \left(\frac{P(d_i, A)}{P(d_i)P(A)} \pi_{d_i} + \frac{P(\bar{d}_i, A)}{P(\bar{d}_i)P(A)} \pi_{\bar{d}_i} \right) \tag{4}$$

More generally we would code each subspace representing the event d_i into K_i *mutually exclusive* ($\sum_{k=1}^{k=K_i} d_i^k = 1$) sub-events:

$$P(A|D) \propto P(A) \prod_i \left(\frac{P(d_i^1, A)}{P(d_i^1)P(A)} \pi_{d_i^1} + \frac{P(d_i^k, A)}{P(d_i^k)P(A)} \pi_{d_i^k} + \ldots + \frac{P(d_i^{K_i}, A)}{P(d_i^K)P(A)} \pi_{d_i^{K_i}} \right) \tag{5}$$

A *feed forward* neural network-like expression for (5) thus becomes:

$$P(A|D) \propto P(A) \prod_i \sum_k \frac{P(d_i^k, A)}{P(d_i^k)P(A)} \pi_{d_i^k} \tag{6}$$

or

$$P(A|D) \propto \exp \left[\log P(A) + \sum_i \log \left[\sum_k \frac{P(d_i^k, A)}{P(d_i^k)P(A)} \pi_{d_i^k} \right] \right] \tag{7}$$

where (7) may often be preferable of practical reasons and due to better noise residuals. For discrete belief values ($\pi_{d_i^k} \in \{0, 1\}$) we may use this simplified form:

$$P(A|D) {\approx} {\propto} \exp \left[\log P(A) + \sum_i \sum_k \log \left[\frac{P(d_i^k, A)}{P(d_i^k)P(A)} \right] \pi_{d_i^k} \right] \tag{8}$$

In the last expression (8) we would recognize "exp" as the *transfer function* and "$\log P(A)$" as a *bias* term from many artificial neural network architectures. The corresponding weight value is then either $\log \frac{P(d_i^k, A)}{P(d_i^k)P(A)}$ (8) or just straight $\frac{P(d_i^k, A)}{P(d_i^k)P(A)}$ (6,7). Which equation to prefer depends on the application. For precise mixture modelling [OL96] should preferably (6) or (7) be used, due to the better accuracy. The last one (8) has been used a lot in *e.g.* recurrent networks [LE89] and also in classification [HL95]. In the data mining application described here and in [BLE$^+$98] we use the logarithmic form, mainly because this has a nice connection with information theory, especially mutual information [Pea88]. Of that reason we refer to the term $\log \frac{P(d_i^k, A)}{P(d_i^k)P(A)}$ as *information component* as it is a measure of the information that migrates from one state of a variable to one state of another variable. Mutual information in its discrete form can then be regarded as a weighted sum of *information components*:

$$I(X;Y) = \sum_x \sum_y P(x, y) \log \frac{P(x, y)}{P(x)P(y)} \tag{9}$$

In the rest of this paper we use the following definitions of the weights and information component (observe that we most often skip index k in the text below):

$$W_{ij} = \frac{P(d_i^k, a_j)}{P(d_i^k)P(a_j)} \tag{10}$$

$$IC_{ij} = \log W_{ij} \tag{11}$$

2 METHOD TO ESTIMATE PROBABILITIES AND UNCERTAINTIES

The estimation of probabilities and their distributions is fundamental in our work. Therefore we start by estimating the probability for a single event. For simplicity we assume the variables we use to be binary, with outcomes 0 and 1. Now let us estimate p_1, the probability of the outcome 1, from c_1 which is the counter for the number of ones out of $C = c_0 + c_1$ trials. For a binomial distribution the probability to get c_1 ones is:

$$P(c_1|p_1, C) = \binom{C}{c_1} p_1^{c_1}(1-p_1)^{c_0} \tag{12}$$

In the classical perspective we would get the Maximum Likelihood estimate of p_1 by differentiating this distribution vs p_1 and find the maximum for $\frac{d}{dp_1}P(c_1|p_1) = 0$ as:

$$\begin{aligned} c_1(1-p_1) &= c_0 p_1 \\ \hat{p_1} &= \frac{c_1}{c_0 + c_1} = \frac{c_1}{C} \end{aligned} \tag{13}$$

This classical estimate does, however, not give well behaving estimates of p_1 for small counter values and does not tell us anything about the significance of an estimated probability. To overcome this we use the Bayesian method to assert an *a priori* probability distribution for the variable, which is refined when more *information i.e.* samples, become available. We consider p_1 to be drawn from a conjugate family of distributions, which we assert as the prior distribution. A convenient prior which is much used if we do not expect the input to be a multi modal mixture [BS94],[Hec97] is the Beta distribution, here described by the *hyperparameters* α_1 and α_0

$$P(p_1) = \frac{\Gamma(\alpha_1 + \alpha_0)}{\Gamma(\alpha_1)\Gamma(\alpha_0)} p_1^{\alpha_1 - 1}(1-p_1)^{\alpha_0 - 1} \tag{14}$$

which gives a posterior for p_1, given the counters c_1 and c_0, which is also a Beta [BS94]:

$$P(p_1|c_1, c_0) = \frac{\Gamma(C + \alpha_1 + \alpha_0)}{\Gamma(c_1 + \alpha_1)\Gamma(c_0 + \alpha_0)} p_1^{c_1 + \alpha_1 - 1}(1-p_1)^{c_0 + \alpha_0 - 1} \tag{15}$$

The expectancy value $\hat{p_1} = E(p_1)$ we get by integration and normalization, where the reduction makes the Γs disappear:

$$E(p_1) = \frac{\int_0^1 p_1 \cdot {p_1}^{c_1 + \alpha_1 - 1}(1-p_1)^{c_0 + \alpha_0 - 1} dp}{\int_0^1 {p_1}^{c_1 + \alpha_1 - 1}(1-p_1)^{c_0 + \alpha_0 - 1} dp} \tag{16}$$

The solution to this [where $\beta(x, y) = \Gamma(x)\Gamma(y)/\Gamma(x + y)$] is:

$$E(p_1) = \frac{\beta(c_1 + 1 + \alpha_1, C - c_1 + \alpha_0)}{\beta(c_1 + \alpha_1, C - c_1 + \alpha_0)} \tag{17}$$

which simplified gives the following ($\alpha = \alpha_1 + \alpha_0$) for $\hat{p_1}$:

$$\hat{p_1} = E(p_1) = \frac{c_1 + \alpha_1}{C + \alpha} \tag{18}$$

In the same way we find the variance estimation ($\hat{\sigma_p^2} = V(p) = E(p^2) - E(p)^2$) as:

$$V(p_1) = \frac{\int_0^1 {p_1}^2 \cdot {p_1}^{c_1+\alpha_1-1}(1-p_1)^{c_0+\alpha_0-1}dp}{\int_0^1 {p_1}^{c_1+\alpha_1-1}(1-p_1)^{c_0+\alpha_0-1}dp} - \frac{\left(\int_0^1 p_1 \cdot {p_1}^{c_1+\alpha_1-1}(1-p_1)^{c_0+\alpha_0-1}dp\right)^2}{\left(\int_0^1 {p_1}^{c_1+\alpha_1-1}\,(1-p_1)^{c_0+\alpha_0-1}\,dp\right)^2} \quad (19)$$

The estimate of $V(p_1)$ thus becomes

$$V(p_1) = \frac{(c_1+\alpha_1)\,(C-c_1+\alpha-\alpha_1)}{(C+\alpha)^2\,(1+C+\alpha)} \quad (20)$$

1 Joint Probabilities

For the joint probability p_{ij}, which has four different outcomes, we assert a 3-dimensional Dirichlet-distribution of p_{11}, p_{10} and p_{01} ($p_{00} = 1 - p_{11} - p_{10} - p_{01}$) as prior in the hyperparameters $\gamma_{11}, \gamma_{10}, \gamma_{01}, \gamma_{00}$. Consider *e.g.* $P(p_{11})$:

$$\begin{aligned} P(p_{11}) &= Di(p_{11}|\gamma_{11},\gamma_{10},\gamma_{01},\gamma_{00}) \quad (21)\\ &= \frac{\Gamma(\gamma_{11}+\gamma_{10}+\gamma_{01}+\gamma_{00})}{\Gamma(\gamma_{11})\Gamma(\gamma_{10})\Gamma(\gamma_{01})\Gamma(\gamma_{00})} {p_{11}}^{\gamma_{11}-1}{p_{10}}^{\gamma_{10}-1}{p_{01}}^{\gamma_{01}-1}(1-p_{11}-p_{10}-p_{01})^{\gamma_{00}-1} \end{aligned}$$

The marginal distributions to Dirichlet are also Dirichlet but in this case they reduce to a one dimensional Dirichlet which is a Beta (14). The posterior distribution given the counters $c_{11}, c_{10}, c_{01}, c_{00}$ is also a Dirichlet distribution [BS94]:

$$P(p_{11}|c_{11},c_{10},c_{01},c_{00}) = Di(p_{11}|c_{11}+\gamma_{11},c_{10}+\gamma_{10},c_{01}+\gamma_{01},c_{00}+\gamma_{00})$$

The expectation value $E(p_{11})$ thus becomes:

$$E(p_{11}) = \frac{\int_0^1\int_0^1\int_0^1 p_{11} Di(p_{11}|c_{11}+\gamma_{11},c_{10}+\gamma_{10},c_{01}+\gamma_{01},c_{00}+\gamma_{00})dp_{01}dp_{10}dp_{11}}{\int_0^1\int_0^1\int_0^1 Di(p_{11}|c_{11}+\gamma_{11},c_{10}+\gamma_{10},c_{01}+\gamma_{01},c_{00}+\gamma_{00})dp_{01}dp_{10}dp_{11}} \quad (22)$$

The evaluation of this integral involves some hypergeometric functions and is a bit cumbersome and we skip the details here. These expectation values can also be looked up in a statistical textbook like [BS94]. Whatever we do we end up with the following:

$$E(p_{11}) = \frac{c_{11}+\gamma_{11}}{c_{11}+\gamma_{11}+c_{10}+\gamma_{10}+c_{01}+\gamma_{01}+c_{00}+\gamma_{00}} = \frac{c_{11}+\gamma_{11}}{C+\gamma} \quad (23)$$

and for the variance (observe similarity with (18,20)):

$$\begin{aligned} V(p_{11}) &= \frac{E(p_{11})(1-E(p_{11}))}{1+c_{11}+\gamma_{11}+c_{10}+\gamma_{10}+c_{01}+\gamma_{01}+c_{00}+\gamma_{00}} \quad (24)\\ &= \frac{(c_{11}+\gamma_{11})(C+\gamma-c_{11}-\gamma_{11})}{(C+\gamma)^2(1+C+\gamma)} \end{aligned}$$

2 Weights and Information Components

In our first attempt to find the expectation values for the weights $[E(W_{ij}) = E(\frac{p_{ij}}{p_i p_j})]$ and their variances we tried the same approach as above by using the integral:

$$\int_0^1 \int_0^1 \int_0^1 \frac{{p_{11}}^{\gamma_{11}-1}{p_{10}}^{\gamma_{10}-1}{p_{01}}^{\gamma_{01}-1}(1-p_{11}-p_{10}-p_{01})^{\gamma_{00}-1}}{(p_{11}+p_{10})^{\gamma_{11}+\gamma_{10}-2}(p_{11}+p_{01})^{\gamma_{11}+\gamma_{01}-2}} dp_{11} dp_{10} dp_{01} \quad (25)$$

We could, however, not find any closed form solution to this, which would still not have taken into account any cross dependencies. Instead we have used the following simplified expression, utilizing (18),(23) above, α and β are number of mutually exclusive events in each class for variables i and j respectively:

$$E(W_{ij}) \quad \approx \quad \frac{E(\hat{p}_{ij})}{E(\hat{p}_i)E(\hat{p}_j)} = \frac{(c_{ij}+\gamma_{ij})(C+\alpha)(C+\beta)}{(C+\gamma)(c_i+\alpha_i)(c_j+\beta_j)} \quad (26)$$

For the specific case of the IC_{ij} this can, however be calculated exactly, due to:

$$E(IC_{ij}) = E(\log \frac{p_{ij}}{p_i p_j}) = E(\log p_{ij}) - E(\log p_i) - E(\log p_j) \quad (27)$$

and it can be shown [KO98] that when p is Beta(a,b) distributed, then

$$E \log p = \frac{b}{a(a+b)} - b \cdot \sum_{n=1}^{\infty} \frac{1}{(a+n)\cdot(a+b+n)} \quad (28)$$

In the application work we present here we have, however, used the following simplified form for the expectation value $E(IC_{ij})$

$$E(IC_{ij}) \quad \approx \quad \log E(W_{ij}) \approx \log \frac{E(\hat{p}_{ij})}{E(\hat{p}_i)E(\hat{p}_j)} \quad (29)$$

The variance for the weight $[V(W_{ij}) = E(W_{ij}^2) - E(W_{ij})^2]$ is harder to estimate. So far we have used the Gauss' approximation for the variance of a function *i.e.* $V[g(X_1,\ldots,X_k)] \approx \sum_{i=k}^{k} V(X_i)(\frac{\partial g}{\partial \mu_i})^2$, where we have skipped the covariant terms. We assume symmetrical distributions, therefore we set $\mu_i = E(X_i)$. The variance for the weight $V(W_{ij})$ then is:

$$V(W_{ij}) \quad \approx \quad \frac{V(p_{ij})}{{p_i}^2{p_j}^2} + \frac{{p_{ij}}^2 V(p_i)}{{p_i}^4{p_j}^2} + \frac{{p_{ij}}^2 V(p_j)}{{p_i}^2{p_j}^4} \quad (30)$$

$$= \quad \frac{(C+\alpha)^2(C+\beta)^2(c_{ij}+\gamma_{ij})}{(C+\gamma)^2(c_i+\alpha_i)^2(c_j+\beta_j)^2} \cdot \quad (31)$$

$$\cdot \left[\frac{(C-c_{ij}+\gamma-\gamma_{ij})}{(1+C+\gamma)} + \frac{(c_{ij}+\gamma_{ij})(C-c_i+\alpha-\alpha_i)}{(c_i+\alpha_i)(C+\alpha+1)} + \frac{(c_{ij}+\gamma_{ij})(C-c_j+\beta-\beta_j)}{(c_j+\beta_i)(C+\beta+1)}\right]$$

For the information component IC_{ij} we can, due to the properties of the log-function as in (27) and in [KO98] write the variance $V(IC_{ij})$ as an exact expression (here including covariant terms):

$$\begin{aligned} V(IC_{ij}) \quad = \quad & V(\log p_{ij}) + V(\log p_{ij}) + V(\log p_{ij}) \quad (32) \\ & -2cov(\log p_{ij}, \log p_i) - 2cov(\log p_{ij}, \log p_j) + 2cov(\log p_i, \log p_j) \end{aligned}$$

and it was proved [KO98] that for p being Beta(a,b) distributed, then:

$$V(\log p) = \sum_{n=0}^{\infty} \frac{b^2+2ab+2bn}{(a+n)\cdot(a+b+n)^2} \tag{33}$$

For the $V(IC_{ij})$ we are still using the simplier approach with Gauss' approximation also here until our software can collect the covariant terms, *i.e.* :

$$V(IC_{ij}) \approx V(\hat{p}_{ij})\left(\frac{1}{\hat{p}_{ij}}\right)^2 + V(\hat{p}_i)\left(\frac{-1}{\hat{p}_i}\right)^2 + V(\hat{p}_j)\left(\frac{-1}{\hat{p}_j}\right)^2 \tag{34}$$

Here we measure the IC_{ij} in bits (i.e. use $\log_2$), which gives the following explicit expression:

$$V(IC_{ij}) \approx \frac{\frac{C-c_{ij}+\gamma-\gamma_{ij}}{(c_{ij}+\gamma_{ij})\,(1+C+\gamma)} + \frac{C-c_i+\alpha-\alpha_i}{(c_i+\alpha_i)(1+C+\alpha)} + \frac{C-c_j+\beta-\beta_i}{(c_j+\beta_j)(1+C+\beta)}}{(\log 2)^2} \tag{35}$$

3 Variance of Conditioned Posterior Distribution

Now we end up by calculating the variance also for $V(P(A|D))$, which is a product of sums according to (6) but to make the approximation of this easier to calculate we use the logarithmic version (7) here too. To calculate an approximate variance for this expression is straight forward by using the Gaussian approximation for a sum of *independent* variables, i.e. $V(\sum_i c_i \cdot X_i) \approx \sum_i c_i^2 \cdot V(X_i)$. We do, however, have to care for the proportionality due to the normalization. Let the class A consist of m *mutually exclusive* subclasses $a_1, a_j \ldots a_m$ and let κ be the scaling factor for each subclass, from (6) we let

$$\begin{aligned} \theta(a_j|D) &= P(a_j)\prod_i \sum_k \frac{P(d_i^k, a_j)}{P(d_i^k)P(a_j)}\pi_{d_i^k} && (36)\\ \kappa &= \frac{1}{\sum_k \theta(a_j|D)} && (37)\\ P(a_j|D) &= \kappa\theta(a_j|D) && (38) \end{aligned}$$

then

$$\begin{aligned} V(\theta(a_j|D)) &= V\left[\exp(\log(\theta(a_j|D)))\right] && (39)\\ &\approx \exp\left(E(\log(\theta(a_j|D)))\right)^2 \cdot V(\log(\theta(a_j|D))) && (40)\\ &= E(\theta(a_j|D))^2 \cdot V(\log(\theta(a_j|D))) && (41) \end{aligned}$$

and

$$\begin{aligned} V(\log\theta(a_j|D)) &\approx V(\log P(a_j)) + V\left(\sum_i \log \sum_k \frac{P(d_i^k, a_j)}{P(d_i^k)P(a_j)}\pi_{d_i^k}\right) && (42)\\ &\approx \frac{V(P(a_j))}{E(P(a_j))^2} + \sum_i V\left(\log \sum_k \frac{P(d_i^k, a_j)}{P(d_i^k)P(a_j)}\pi_{d_i^k}\right) && (43) \end{aligned}$$

now set $\left[W_{ij}^k = \frac{P(d_i^k, a_j)}{P(d_i^k)P(a_j)}\right]$ and assume the terms in the sum (43) are independent:

$$V(\log\theta(a_j|D)) \approx \frac{V(P(a_j))}{E(P(a_j))^2} + \sum_i V\left(\log\sum_k W_{ij}^k \pi_{d_i^k}\right) \tag{44}$$

$$\approx \frac{V(P(a_j))}{E(P(a_j))^2} + \sum_i \frac{V\left(\sum_k W_{ij}^k \pi_{d_i^k}\right)}{E\left(\sum_k W_{ij}^k \pi_{d_i^k}\right)^2} \tag{45}$$

$$\approx \frac{V(P(a_j))}{E(P(a_j))^2} + \sum_i \frac{\sum_k \pi_{d_i^k}^2 \cdot V\left(W_{ij}^k\right)}{E\left(\sum_k W_{ij}^k \pi_{d_i^k}\right)^2} \tag{46}$$

Then we end up with the following approximation for $V(P(a_j|D)$ from (41,43,46)

$$V(P(a_j|D) \approx \kappa^2 E(\theta(a_j|D))^2 \left(\frac{V(P(a_j))}{E(P(a_j))^2} + \sum_i \frac{\sum_k \pi_{d_i^k}^2 \cdot V\left(W_{ij}^k\right)}{E\left(\sum_j W_{ij}^k \pi_{d_i^k}\right)^2}\right) \tag{47}$$

For practical reasons we may still want to use logarithmic probabilities $\log(P(a_j|D)$ in the output. Then, according to (7,36,37)

$$\log P(a_j|D) = \log\theta(a_j|D) - \log\sum_j \theta(a_j|D) \tag{48}$$

$$= \log P(a_j) + \left[\sum_i \log\sum_k W_{ij}^k \pi_{d_i^k}\right] + \log\kappa \tag{49}$$

From (37, 46)

$$V(\log P(a_j|D)) \approx V(\log\theta(a_j|D) - \log\kappa) \tag{50}$$

$$\approx \frac{V(P(a_j))}{E(P(a_j))^2} + \sum_i \frac{\sum_k \pi_{d_i^k}^2 \cdot V\left(W_{ij}^k\right)}{E\left(\sum_k W_{ij}^k \pi_{d_i^k}\right)^2} + V(\log\kappa) \tag{51}$$

$$V(\kappa) \approx \frac{1}{(\sum\theta(a_j|D))^4} \tag{52}$$

$$V(\log\kappa) \approx \frac{V(\kappa)}{\kappa^2} = \frac{1}{(\sum\theta(a_j|D))^2} \tag{53}$$

$$V(\log P(a_j|D)) \approx \frac{V(P(a_j))}{E(P(a_j))^2} + \sum_i \frac{\sum_k \pi_{d_i^k}^2 \cdot V\left(W_{ij}^k\right)}{E\left(\sum_k W_{ij}^k \pi_{d_i^k}\right)^2} + \frac{1}{(\sum\theta(a_j|D))^2} \tag{54}$$

4 Selecting Proper Priors

The priors for p_i and p_j are not critical as the convergence to the "real probability" is rather quick [BS94]. The most simple prior for a binary variable to use here is $\alpha_1 = \alpha_0 = 1$, *i.e.* a non-informative (sometimes called *ignorant prior*) [BS94], which corresponds to an a priori assumption about equal probability distribution. We should, however, be aware that for a k-nary variable (as in 5), we need another prior. We could then assert $\alpha_i = 1, \alpha = k * \alpha_i$.

The prior for p_{ij} is more critical because we want the weight values to behave well for small counter values. Here with well we mean that no data samples are equal to independence, *i.e.*

$$\lim_{c_i,c_j,c_{ij}\to 0} IC_{ij} = \log \frac{\hat{p}_{ij}}{\hat{p}_i \hat{p}_j} \approx 0 \tag{55}$$

we choose the prior for p_{ij} so that:

$$\lim_{c_{ij},C\to 0} IC_{ij} = \log \frac{\hat{p}_{ij}}{\hat{p}_i \hat{p}_j} = \log \frac{\frac{c_{ij}+\gamma_{ij}}{C+\gamma}}{\hat{p}_i \hat{p}_j} \approx 0 \tag{56}$$

Then we can initialize γ_{ij} to some small number (*e.g.* 1) , as

$$\gamma_{ij} = 1 \tag{57}$$

$$\gamma = \frac{\gamma_{ij}}{\hat{p}_i \hat{p}_j} \tag{58}$$

In figure 1 we see an example of how this looks when starting with the ignorant prior without any samples and then how the posterior distribution get more and more narrow when we add a few samples. In the figure it is also shown how the estimated prior for the joint distribution P_{ij} look like for some of the first samples.

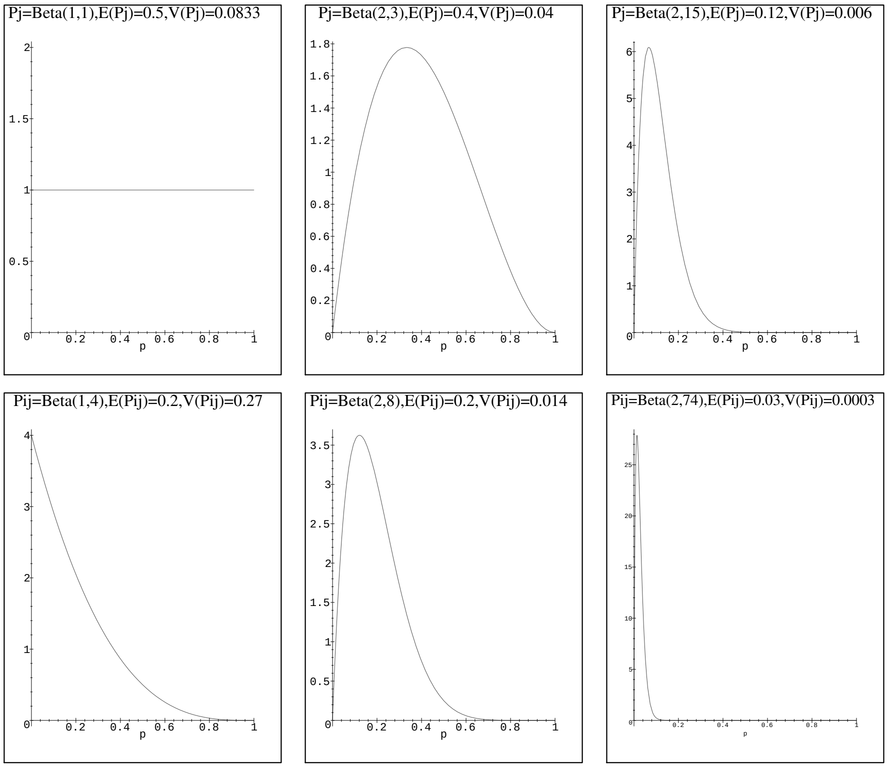


Figure 1: Some examples of priors for P_i and P_{ij}. The **upper diagrams** show priors for P_i when $\alpha_0 = \alpha_1 = 1$; $(c_0 = 1, c_1 = 0)$; $(c_0 = 2, c_1 = 1)$ and $(c_0 = 14, c_1 = 1)$ respectively. The **lower diagrams** show the corresponding estimated priors for P_{11} when $\gamma_{11} = 1$; $c_{11} = 0, c_{11} = 1, c_{11} = 1$ and c_0, c_1 when both i, j are as in the diagrams above.

3 OTHER METHOD RELATED ISSUES

1 Variable Value Coding

The coding of binary and discrete variables into a *neural layer* representation is rather straight forward. For a binary variable, x we input the values $[x, \bar{x}]$, which is also the simplest example of a *hypercolumn*, *i.e.* a *neural* layer with mutually exclusive input units. When the variable value is missing we input the a priori probabilities instead: $[p_x, p_{\bar{x}}]$. Any discrete variable, whose values are mutually exclusive is coded in this way. In general we may input a normalized mixture of belief values instead of the a priori probabilities for missing values. Real valued variables are coded using such a mixture of belief values, which here represents the degree of membership to a set of Radial Basis Functions (RBFs) [Ben94]. The placement of these RBFs is usually done using the EM algorithm [Trå93].

2 Dependent Variables

Variables which we find to be very dependent on each other we handle by coding the combination of these variables into a separate subspace, a *hypercolumn*, *i.e.* a *neural* layer with mutually exclusive combinations. As an example, assume that we find that the binary variables X and Y are dependent of each other, then we make a hypercolumn with 4 units representing $\{\bar{x}\bar{y}, \bar{x}y, x\bar{y}, xy\}$. When the number of combinations get large this coding is inconvenient, then a *reduced* coding is used, where only the combinations or *features* that actually occur in training data will be coded. Considering real valued variables there is not much difference. The RBF units themselves may combine an arbitrary number of dimensions into, what is basically, a normalized mixture of belief values as one hypercolumn. The RBF coding is done "before" the treatment of discrete variables, which is necessary to be able to combine real valued variables with discrete variables. To find dependencies between variables we use the following methods (ϕ is a threshold):

- check for strong pairwise mutual information between all pairs of subspaces. The procedure is repeated to find higher order combinations as long as:

$$\phi_{MI} < \sum_{xy} p_{xy} \log \frac{p_{xy}}{p_x p_y} \tag{59}$$

- check all variable combinations up to a certain complexity level in one shot. To decide what combinations to save we use the Kullback-Leibler distance between the joint distribution and the marginal distributions. We save those subspaces where:

$$\phi_{KL3\ldots} < \sum_{xyz\ldots} p_{xyz\ldots} \log \frac{p_{xyz\ldots}}{p_x p_y p_z \cdots} \tag{60}$$

- as in previous item: check all combinations of variables up to a certain complexity level in one shot. Save those combinations only, in a sparse manner, where the *information component* between input and output layer is above a certain threshold θ_{IC}

$$\phi_{IC} < \log \frac{p_{xy}}{p_x p_y} \tag{61}$$

A comment on the thresholds ϕ_{MI} , ϕ_{KL} and ϕ_{IC} above: At the moment these kind of thresholds are considered to be design parameters in the BCPNN network. We have no automatic method generating them so far.

3 Sparse Matrix Technique

When working with this huge WHO data base, INTDIS, of adverse drug reactions we first used full matrixes. We found this to be inefficient because a typical full connection matrix, containing 20-50 millions of connection elements did often contain a non zero value in 1-2% of the positions only. Therefore we started using a sparse matrix technique, which reduced both the required computer time as well as memory requirements drastically. The technique is "double sparse" *i.e.* it allows us to create not only matrix elements dynamically, but also the "neurons" in the input/output layers dynamically.

Thanks to this sparse technique and the organization of the database in reports, were only a very small subset of all possible combinations can occur on each, we do not need to do the search for dependent variables fully incrementally. We can decide beforehand how high complexity level of combinations we want to investigate.

4 RESULTS AND EXAMPLES

1 Signal Generation

In the data base application we want to generate an early warning signal when a certain dependency between a drug or a set of drugs vs adverse drug reactions is detected. The procedure is to look for significant differences in weight values between input and output variables when the last quarterly batch of reports is added to the data base. To be able to do this in a feasible way considered computing time and memory utilisation we used a sparse matrix coding of the connection matrix. The procedure was tested on some well known signals like the drug Suprofen causing back pain and the drug Azapropazone causing photosensitivity reaction. Results from these time scans can be seen in figure 2...5 respectively.

The diagram in figure 2 shows how the IC *(information component)* for Suprofen vs back pain varies over the years 1983 to 1990. The bars around the IC curve show, for each quarterly year on the x-axis, a 95 % estimated confidence interval for the IC. The diagram in figure 3 shows how the cumulative probability function $P(IC > 0)$ develops over time. An article of the drug Suprofen causing back pain was published in 1985 [OS85]. From the diagrams in figure 2 and figure 3 we can see indications of an association between the drug and the adverse reaction with rather high certainty, around 80 % after the first quarter 1984, which rises to around 97% in the middle of 1984,

For the Azapropazone case there was a paper published in 1986 of this drug causing photosensitivity reaction [HN86]. From the diagram in figure 4 showing IC for Azapropazone vs photosensitivity reaction we do see, however, a clear indication of this association already in 1975. In figure 5 we see the prior probabilities for the drug and the adverse reaction. Observe that the scale for the probabilities in the right diagram is logarithmic. We also see the posterior probability for the adverse reaction given the drug. As we can see is $P(A|D) >> P(A)$, which clearly indicates a conditioned dependency between these. All three probabilities are shown with 95% confidence intervals, but the prior probabilities are much more narrow than the conditioned posterior probability because there are less samples in the joint distribution.

Of course it may not be enough to just look upon ICs between individual drugs and individual adverse drug reactions. In many cases an adverse drug reaction or a combinations of several adverse drug reactions *i.e.* a *syndrome* may be correlated with certain combinations of drugs and other variables, like patient's age, for instance.

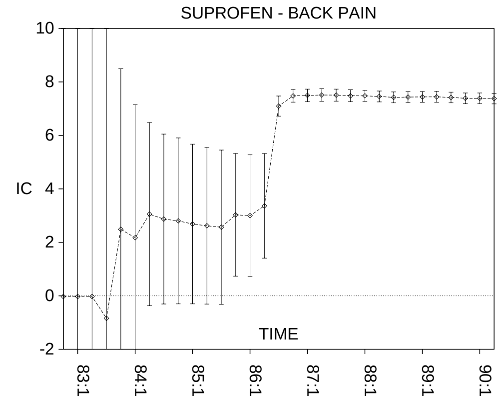


Figure 2: A well known signal: the drug Suprofen causing back pain. The diagram shows the IC (*information component*) for the drug-ADR association. The error bars show a 95% estimated confidence interval.

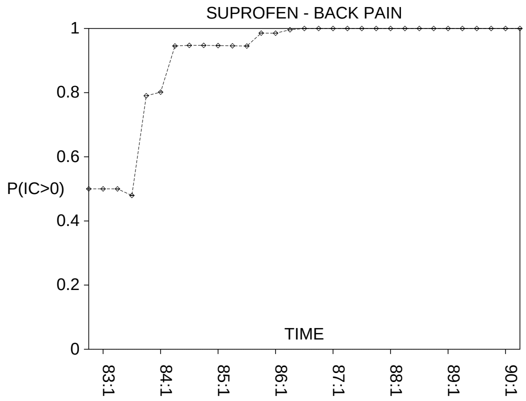


Figure 3: Suprofen back pain: The diagram shows how the prob. $P(IC > 0)$ develops over the years, we see that we have a clear indication of this association with 80% certainty already after the first quarter 1983.

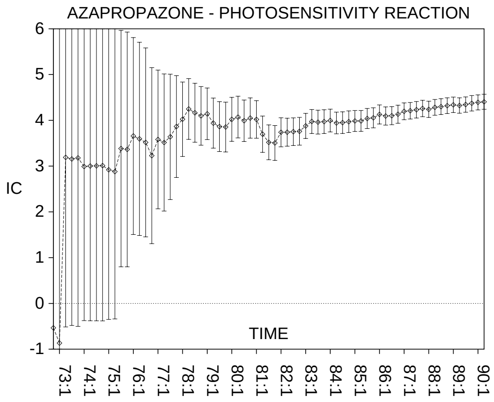


Figure 4: The development from the year 1973 to 1990 of the information component for the drug Azapropazone vs the photosensitivity reaction with 95% confidence interval.

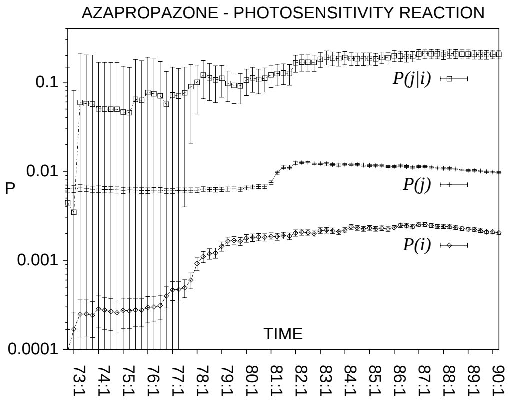


Figure 5: Prior prob for Azapropazone $P(i)$ and for photosensitivity reaction $P(j)$ as well as the conditioned posterior $P(j|i) = P(photosens|Azapropazone)$ with 95% conf. int. Logarithmic scale.

2 Digoxin versus Age and Rash

The following experiments aim to demonstrate that IC analysis can be used to study the relationship between combinations of variables in the database, including, but not being restricted to, drug adverse reaction association pairs. To establish this, the relationship between the drug Digoxin and the patient's age was examined by investigating the IC between Digoxin and different age intervals for the adverse reaction rash.

The BCPNN network was here set up to generate counters in a slightly different manner than previously, concerning the C-value in particular. C would normally be the total number of reports, but in two of these experiments (in figure 7 and 9) it has been set as

C=total number of reports within a specific age group. The age groups are here 10 year intervals. The c_i is the counter for the drug or for the drug combined with age. The occurances of the drug being reported as suspected drug (figure 7) or other drug (figure 9) are counted separately. c_j= the number of reports for the adverse reaction or the adverse reaction combined with age group, and c_{ij}=counts the intersection between c_i and c_j in the normal way.

In figure 6 we see a time scan of the IC for Digoxin versus the adverse reaction Rash from 1967 to 1997, when the IC has stabilized at a level of -2. The diagram in figure 7 shows the IC for the end of 1997, but here displayed separately for different age groups. In this diagram (figure 7) we see, for each age interval, that there was a negative IC between Digoxin and rash. The association was most negative for age range $30 \cdots 40$, although in general there seemed to be a trend towards lower ICs for higher ages, *i.e.* less probability for Digoxin to be the suspected drug for causing Rash in elderly patients. However, the confidence intervals are rather large and the trend is therefore unreliable. We can also see that the uncertainty in IC is higher for younger patients, which may be explained by the diagram in figure 8.

In figure 8 we see how the IC for Digoxin vs age varies with the age of the patient. The diagram of IC vs age in figure 8 shows a clear trend of increasing IC with age. From a minimum of IC=-4 for $20 \cdots 30$ year olds ($c_{ij} = 34$) to a maximum value of IC=3 ($c_{ij} = 244$) for the age group of 90+ year olds. The highest c_{ij} value was for $70 \cdots 80$ year old patients where ($c_{ij} = 2228$) (IC=1.7). The standard deviations are small for all IC values due to the large number of reports of Digoxin in the database (7370).

In figure 9 we look upon IC between Digoxin and Rash within different age groups when Digoxin was not the suspected drug but was reported as *other* drug. In the same way as for the results where Digoxin was the suspected drug most Digoxin-rash associations had negative ICs , however there was a definite trend of increasing positive IC for increasing age range, such that for age groups $70 \ldots 80$, $80 \ldots 90$ and 90+ there was a definite positive association between Digoxin when recorded as other and rash.

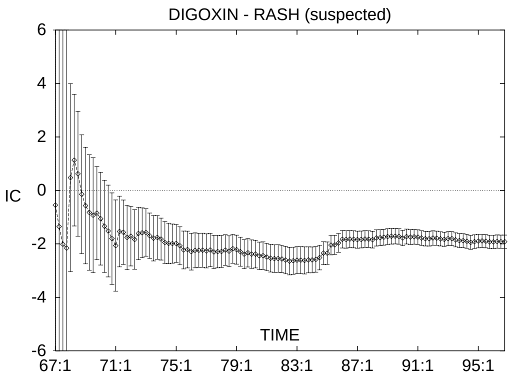


Figure 6: A time scan of IC for the drug Digoxin vs the ADR rash from the year 1967 to 1997. At 1997 the IC has stabilized around a level of -2.

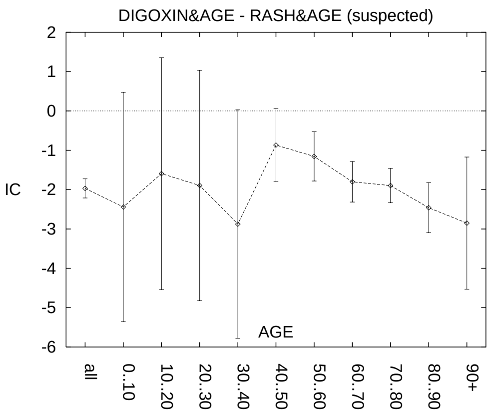


Figure 7: IC between Digoxin and rash for the last quarter 1997 displayed separately for each age group, with ten year intervals. The age group "all" sums all intervals.

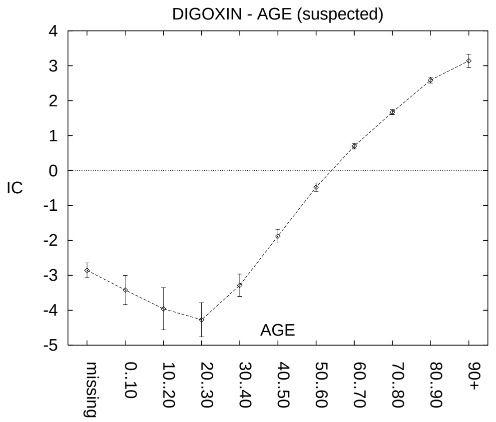


Figure 8: Here we see how the IC for Digoxin vs age varies with age, indicating a higher prob. to find elderly patients taking the drug Digoxin.

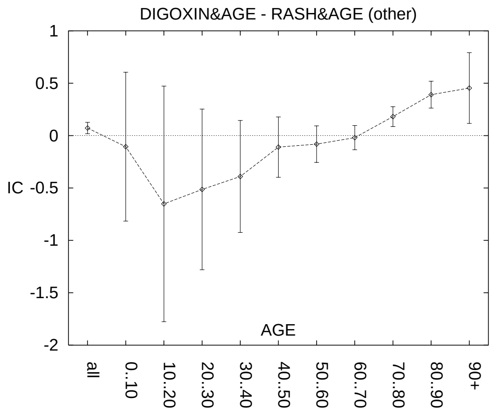


Figure 9: The IC between Digoxin and rash displayed separately for each age group. This concerns when Digoxin was reported as "other drug" *i.e.* not suspected.

2.1 Discussion

The contrast between Digoxin-rash profiles over age for Digoxin=suspected (or interactive) drug and Digoxin = other drug was striking. This is probably because if elderly patients are taking Digoxin they are more likely to be taking other drugs concomitantly, than a younger person taking Digoxin. Therefore the occurrence of rash as an adverse reaction is more likely to be attributed to another drug, for elderly patients as compared to younger patients. The aim of these experiments was to examine the potential of the BCPNN methodology for looking at associations when 3 different variables are considered together. The software includes methods to search for dependent variable combinations of any order, which is described below.

5 SEARCH FOR DEPENDENT VARIABLES

1 Goal Description

Here we give an example of using this method to data mine the database to asses the validity of our complex variable method and to provide and indication of this method being effective in finding combined variable effects, in this case a syndrome. This also shows that the method is computationally tractable. We considered a known adverse drug reaction syndrome complex association: the Neuroleptic Malignant Syndrome which is frequently reported in the WHO database. The syndrome itself is a combination of several symptoms which themselves can be reported as individual adverse drug reactions: Creatine Phosphokinase Increased, Fever, Death and Hypertonia. Therefore combinations of these adverse drug reactions will be reported together with this drug relatively frequently. We are interested therefore in the strength of associations between combinations of the adverse reactions with Haloperidol, and how these associations compare to the strength of associations between the combinations of the adverse reactions among themselves. Ultimately whether it is possible to pick the syndrome out.

2 Setup of the Experiment

We aimed to investigate all single, pair and triplet combinations of all adverse reactions in the database and then compare each combination with the drug Haloperidol. For this purpose we used the *sparse matrix* (see 3) method which we considered to be the tractable method in this case. One conventional method to use here would be to first scan the data base checking all ADRs against all ADRs, then do a selection of what ADRs pairs to be considered related by some thresholding, then scan the data base again checking these ADR pairs versus all ADRs again. Do a new selection by thresholding and then check these triplets versus the Drug. This kind of approach certainly works in finding general feature detectors as it would, in most cases, find combinations where the Kullback-Leibler distance between the adverse reactions

$$\sum_{123} p(ADR_{123}) \log \frac{p(ADR_{123})}{p(ADR_1)p(ADR_2)p(ADR_3)} \tag{62}$$

would be quite large, but not necessarily find those where

$$p(ADR_{123}|drug) >> p(ADR_{123}) \tag{63}$$

which are the reactions we really want to find. The specification given the sparse BCPNN was to partition the drugs into the classes "Haloperidol", "other drug" as input layer and make all possible combinations of adverse reactions in the output layer, *i.e.* the output layer will represent a subset of the powerset of all ADRs on each report. The subset we used here included combinations up to three ADRs As a side effect this gives us the possibility to check *e.g.* the KL-distance (62) for all ADRs found in the data base at once, which gave us a tremendous speedup compared with the original matrix approach, which took several days on a Sun UltraSparc. The actual search needed only about 7 hour of computing time on the same UltraSparc. At the same time we are guaranteed to not miss any combination in the investigation of $p(ADR_{123}|drug) >> p(ADR_{123})$.

3 Results

We generated lists of the associations according the following table:

drug	ADR-comb	# ADR comb	# $IC > 0$
Haloperidol	single-ADR	1700	281
Haloperidol	double-ADR	35000	4019
Haloperidol	triple-ADR	550000	5388

where the column "# ADR comb" tells us how many combinations that were found in total. The column "# $IC > 0$" tells us how many of these had a positive IC. These one with a positive IC were then sorted on the level of the IC, *i.e.* the strength of the association between the drug and the ADR-combination.

As could be expected the Neuroleptic Malignant Syndrome (NMS) were on the top of all these lists. In the pairs and triplets list this syndrome were also found to be strongly associated with some of the other symptoms which are included in the symptom picture of the NMS: Creatine Phosphokinase Increased, Fever, Death and Hypertonia. The reason for this is that the syndrome is not so strictly defined as being composed of these symptoms only, it is enough that the patient has a couple of these symptoms to be diagnosed as NMS.

We also found that all of the included symptoms were high on all the three lists. For the single ADR list all the four symptomatic ADRs were among the highest 200 IC values. For the list with ADR pairs these four ADRs were also included among highest 200 IC values and three of these combinations were in the top ten. For the list with triple ADRs all combinations with these foure ADRs were among the highest 400 and three combinations were in the top ten.

6 DISCUSSION

We have made extensive use of the Gaussian approximation formula for variances of functions here. Further on, we have assumed independence between the variables and thus also skipped the covariance terms just to make it as simple as possible to start with. These approximation formulas also work best for Gaussian distributions, but here we have used Beta and Dirichlet distributions as our actual model distributions. We have to be aware that this may be a coarse approximation in many cases, especially when the conditional independence between variables is not fulfilled. As long as we care for these things and make sure that dependent variables are coded in what we call "hypercolumns", *i.e.* , by partitioning the input space into mutually exclusive regions, then it seems as this simple approach is useful.

Then we may ask if we can do this better? In the beginning we found it very hard to do exact calculations of especially the variance of the IC. It was encouraging to find, which was shown in section 2 and also in [KO98], that in case of the logarithmic IC we could express the solutions in exact analytical forms. These will be considered in the fortgoing work, when the software has been adapted to collect the covariant terms as well.

To be able to propagate propabilities and to calculate variances of posterior output distributions conditioned on a set of inputs we have so far used the non logaritmic W_{ij}, whose variance now may be better approximated. We yet don't know if it is reasonable to find an exact expression for the conditioned output probability distribution, or simplified, its variance. Otherwise we could probably do fairly good by numerical integration to find these variances, which would, however, require a tremendous computational power. It can surely be done and we will look upon improving, *e.g.* expectation values over joint distributions by numerical methods.

The goal here, was, however to find a more or less neural network like solution to the variance problem. There are statistical methods being developed that may do better in approximating the variances like "saddle point approximations for statistical series" [Kol97]. Such methods may still become simple enough to treat them in a neural network like manner. We do not, however, consider this a "holy" condition that the solution has to be a neural network as long as it works for our purpose, even if the neural network approach is both computationally and architecturally efficient. Our goal is merely to be able to propagate complete distributions one day, like the work being done within the Bayesian network area, where Gibbs sampling is used for that purpose, like in the program BUGS. This kind of technique is, however, at the moment, too computationally extensive to be used in our data mining application.

We are happy to see that our approach help in giving earlier and more efficient signalling of suspected adverse drug reactions. This application is dealt with in more detail in [BLE+98]. The method is now (spring 1998) taken into usage for filtering of adverse drug reaction reports to produce warning signals on new unknown reactions when they have become significant.

7 ACKNOWLEDGEMENTS

We gratefully acknowledge the reporting of spontaneous adverse drug reaction case reports from National Centres to the WHO Collaborating Centre for International Drug Monitoring, which make up the database described in this paper. Rogelio Melhado Defretias was the person that initialized the contact between WHO Uppsala monitoring centre and Royal Institute of Technology. Thanks to Anders Holst who supplied a lot of patience with us discussing the issues herein. He also provided the original idea of how to do the variance calculations using Gauss formula.

References

[Bay63] Thomas Bayes. An essay towards solving a problem in the doctrine of chances. *Biometrika (reprint 1958 of orig art in Philos. Trans. R. Soc. London 53, pp. 370-418)*, 45:296–315, 1763.

[Ben94] Michel Benaim. On functional approximation with normalized gaussian units. *Neural Computation*, 6:319–333, 1994.

[BLE+98] Andrew Bate, Marie Lindquist, I. Ralph Edwards, Sten Olsson, Roland Orre, Anders Lansner, and Rogelio Melhado De Freitas. A bayesian neural network method for adverse drug reaction signal generation. *European Journal of Clinical Pharmacology*, 1998. in press.

[BS94] José M. Bernado and Adrian F.M. Smith. *Bayesian Theory*. John Wiley & Sons, Chichester, 1994.

[Goo50] I. J. Good. *Probability and the Weighing of Evidence*. Charles Griffin, London, 1950.

[Hec97] David Heckerman. Bayesian networks for data mining. *Data Minining and Knowledge Discovery*, 1:79–119, 1997.

[HL95] Anders Holst and Anders Lansner. A higher order bayesian neural network for classification and diagnosis. In *Applied Decision Technologies: Computational Learning and Probabilistic Reasoning*, pages 251–260, 1995. London, England, April 3-5.

[HN86] Morales J.R. Henann N.E. Suprofen - induced acute renal failure. *Drug Intell Clin Pharm*, 20(11):860–862, 1986.

[Hol97] Anders Holst. *Using Bayesian Neural Networks for Classification Tasks*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, Dept. of Numerical Analysis and Computing Science, 1997.

[KO98] Timo Koski and Roland Orre. Statistics of the information component in bayesian neural networks. Tech. Rep. TRITA-NA-9806, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden, 1998.

[Kol97] John E. Kolassa. *Series Approximation Methods in Statisticss*. Springer-Verlag, New York, 1997.

[Kon89] Igor Kononenko. Bayesian neural networks. *Biol. Cybernetics*, 61:361–370, 1989.

[Lap14] Pierre Simon Laplace. *A Philosophical Essay on Probabilities.* Trans 1995 from the 5th ed 1825 (orig 1814) by Andrew Dale. Orig "Essai Philisophique sur les Probabilités", Dover Publications, New York, 1814.

[LE85] Anders Lansner and Örjan Ekeberg. Reliability and speed of recall in an associative network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):490–498, July 1985.

[LE89] Anders Lansner and Örjan Ekeberg. A one-layer feedback artificial neural network with a bayesian learning rule. *Int. J. Neural Systems*, 1(1):77–87, 1989.

[LH96] Anders Lansner and Anders Holst. A higher order Bayesian neural network with spiking units. *Int. J. Neural Systems*, 7(2):115–128, May 1996.

[OL96] Roland Orre and Anders Lansner. Pulp quality modelling using bayesian mixture density neural networks. *Journal of Systems Engineering*, 6:128–136, 1996.

[OS85] Boman G Olsson S, Biriell C. Photosensitivity during treatment with azapropazone. *Br Med J*, page 939, 1985.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, 1988.

[Trå93] Hans G.C. Tråvén. *On Pattern Recognition Applications of Artificial Neural Networks.* PhD thesis, Royal Institute of Technology, Stockholm, Sweden, Dept. of Numerical Analysis and Computing Science, 1993.

Supplement II

A Data Mining Application

A Bayesian neural network method for adverse drug reaction signal generation

Andrew Bate, Marie Lindquist, I Ralph Edwards, Sten Olsson,
The WHO Collaborating Centre for International Drug Monitoring, Uppsala, Sweden

Roland Orre, Anders Lansner, Rogelio Melhado De Freitas.
SANS, NADA, Royal Institute of Technology, Stockholm, Sweden

Author for correspondence: Prof I. Ralph Edwards
WHO Collaborating Centre for International Drug Monitoring
Stora Torget 3, S-753 20 Uppsala, Sweden.
`ralph.edwards@who.pharmasoft.se`

Abstract

Objective: The database of adverse drug reactions (ADRs) which is held by the Uppsala Monitoring Centre on behalf of the 47 countries of the WHO Collaborating Programme for International Drug Monitoring contains nearly two million reports. It is the largest database of this sort in the world and about 35 000 new reports are added quarterly. The task of trying to find new drug-ADR signals has been carried out by an expert panel, but with such a large volume of material the task is daunting. We have developed a flexible, automated procedure to find new signals with known probability difference from the background data.
Method: Data mining, using various computational approaches, has been applied in a variety of disciplines. A Bayesian Confidence Propagation Neural Network (BCPNN) has been developed which can manage large data sets, is robust in handling incomplete data, and may be used with complex variables. Using information theory, such a tool is ideal for finding drug-adverse reaction (ADR) combinations with other variables, which are highly associated compared to the generality of the stored data, or a section of the stored data. The method is transparent for easy checking and flexible for different kinds of search.
Results: Using the BCPNN some time scan examples are given which show the power of the technique to find signals early (captopril-cough) and to avoid false positives where a common drug and adverse reactions occur in the database (digoxin-acne; digoxin rash). A routine application of the BCPNN to a quarterly update is also tested showing that 1004 suspected drug-ADR combinations reached the 97.5 % confidence level of difference from the generality. Of these, 307 were potentially serious ADRs, and of these 53 related to new drugs. Twelve of the latter were not recorded in the CD editions of The Physicians Desk Reference or Martindale's Extra Pharmacopoea or appeared in Reactions Weekly online.
Conclusion: The results indicate that the BCPNN can be used in the detection of significant signals from the data set of the WHO Programme on International Drug Monitoring. The BCPNN will be an extremely useful adjunct to the expert assessment of very large numbers of spontaneously reported adverse drug reactions.

Key words: Bayesian, neural, network, adverse, drug, reactions.

1 Introduction

It is in the very nature of drugs that they will cause adverse reactions. However, the incidence rates of specific adverse drug reactions vary considerably from drug to drug. In the same way there will always exist certain high risk groups of adverse drug reactions (ADRs) with specific drugs. The WHO Database is the largest international database of case reports of spontaneous reporting of suspected ADRs. This database, held by the Uppsala Monitoring Centre (UMC), now contains nearly two million reports of adverse drug reactions. One of the main responsibilities of the UMC is to produce signals, according to the accepted WHO definition: 'Reported information on a possible causal relationship between an adverse event and a drug, the relationship being unknown or incompletely documented previously. Usually more than one report is required to generate a signal depending on the seriousness of the event and the quality of the information'[1]. The current procedure of signal generation is as follows: On a quarterly basis lists of potential drug-ADR problems are generated on new reports received at the centre. A panel of experts are then sent data on these associations, and asked to comment on them. From these comments a final list of signals is generated which is then circulated to the National Pharmacovigilance Centres as well as the expert panel. It is then the responsibility of individual National Centres to react to each signal as they see fit [2]. There are obvious limitations in the current system. Experts are only able to consider a finite amount of data in the time available and the data considered could be incorrect or more likely incomplete. Also the experts' assessments are based on judgement as well as prior knowledge, which creates a bias towards discovering signals in those drug-ADR associations that are already suspected, or have been highlighted for other reasons [3]. Many other approaches have been made to the problem of optimising the signal generation process and have been well reviewed by B gaud et al [4]. However, it has been clear for many years that an automated signalling system would improve the current system considerably [5], but the size of the database has made it impossible to consider all possible drug-ADR combinations in a routine automated way. We needed a system with large computational power to consider all possible links in the database of nearly two million records, each with, currently, 49 fields. The advances in information technology, in combination with the well-established theory of Bayesian statistics, have allowed us to develop a data mining system based a Bayesian neural network. This method helps to minimise the limitations of the current system as all drug-ADR combinations are considered in an unbiased manner. Strong associations between specific drugs and specific ADRs will be highlighted. Signals, which will have been generated without either external prompting or prejudgement, can then be investigated further.

2 Methods

1 The WHO Database

The WHO database consists of nearly two million individual case reports of suspected adverse drug reactions for specific, but anonymous, patients. These reports are provided by doctors and other health professionals throughout the world. They contain administrative data, patient data, ADR data, medication data and additional information. In total there are 49 different fields, although not all fields are filled in on each case report. For example only approximately 10 % of case reports received at the centre have the fields for onset, treatment, indication, outcome, dose, age and sex all filled in. The drug information states the drugs taken by the patient, their respective quantities and duration of use. Drugs specifically suspected of causing the reported adverse reaction are also indicated as S (suspected) or I (interacting). Concomitant medication is recorded as O (other).

2 The Bayesian Confidence Propagation Neural Network

Neural Networks [6] are built from biologically inspired computing elements, neurons, which are coupled into networks. These neurons are simple, but when used in combinations, they can perform complex tasks like pattern recognition and diagnosis [7]. Each neuron receives an external input as well as several inputs from other neurons, each with an attributed weight in the network. The combination of all the inputs and their respective weights to a specific neuron when summed together and added to a bias value, generates a single output. This output then acts as one of the inputs for the other neurons in the network. The network we use is called a Bayesian Confidence Propagation Neural Network (BCPNN)[8], it is a feed forward Neural Network where learning and inference are done using the principles of Bayes Law. For the work presented in this paper we use it as a one layer model [9], although it can be extended to a multilayer network [10]. Such a multilayer network will be required in further investigations of combinations of several variables in the WHO database and has already been successfully applied to areas like diagnosis [11], expert systems [12] and data analysis in paper and pulp manufacturing [13].

The main advantages with BCPNN, as for many other Neural Network architectures, are that they are self organizing and suitable for implementation on parallel computers. They also provide an efficient computational model which performs well on sequential machines. Another advantage with BCPNN is the simple interpretation of the weights as probabilistic entities. The information stored as the weights in the BCPNN is used here for quantifying drug-ADR dependencies.

This Bayesian neural network has the computational power to consider all links, and the ability to highlight potential signals. The network is transparent, in that it is easy to see what has been calculated and robust, in that valid, relevant results can still be generated despite missing data. This is extremely advantageous as most reports in the database contain some empty fields. The results are reproducible, making validation and checking simple. The network is easy to train; it only takes one pass across the data which makes it highly time efficient. Searches through the database are done quickly and efficiently using a sparse matrix method. This method utilises the fact that a relatively small proportion of all possible drug-adverse reaction combinations are actually non-zero in the database.

3 The Bayesian approach to signal generation

In the WHO database all adverse reactions are reported with a specific drug or set of drugs. A drug is therefore associated with, that is, occurs on the same report as, all ADRs a certain number of times between zero and C (where C is the total number of reports in the database). The number of times a specific drug-ADR combination (c_{ij}) occurs in the database is clearly dependant on the number of times the drug itself is reported throughout the database, as well as the total number of reports of that ADR. The absolute value of c_{ij} is, in itself, far from ideal in predicting the strength of association of a drug-ADR combination. We are in essence looking for values of c_{ij} which are higher than we would expect from the values of both c_i (the number of reports of a specific drug in the database) and c_j (the number of reports of a specific adverse drug reaction in the database).

For any individual report in the database, there is a certain probability that a specific ADR is listed on it - the 'prior probability'. If that case report has a specific drug on it, the probability of the ADR now being present could be different - the 'posterior probability'. If the posterior probability is higher than the prior probability - then the presence of the drug on the report has enhanced the chance of the ADR being present, and the drug-ADR pair are present together in the database more often than expected.
Bayes Law states:

$$P(A|D) = \frac{P(A,D)}{P(D)}$$

This can be rewritten in the form:

$$P(A|D) = P(A)\frac{P(A,D)}{P(A)P(D)}$$

where $P(A/D)$ is posterior probability, the probability of a specific adverse reaction being present on a report given the information that a specific drug is listed on it; $P(A)$ is prior probability, the probability that a specific adverse reaction is present on a report; $P(D)$ is prior probability, the probability that a drug is present on a report; $P(A,D)$ is coincident probability, the probability that both a specific drug and an adverse reaction are present on the same report. Thus the prior probability and the posterior probability are related by a symmetrical factor $[P(A,D)/P(A)P(D)]$.

Mutual Information, as defined in Information Theory [14], measures the amount of information we get about one variable (X) when we have information about the state of another variable (Y), that is measures the strength of association between two variables X and Y:

$$I(X,Y) = \sum_x \sum_y P(x,y) \log \frac{P(x,y)}{P(x)P(y)}$$

where x represents a specific state of the variable X, and y represents a specific state of the variable Y. In Information Theory all measures of information are logarithmic, so that information from independent events are additive.

If we consider the variables Captopril and coughing in the database, both variables are binary in that both can have one of two states, that is either to be present on a report or not. There are four possible combinations of the states of the two variables, which when all are combined give the Mutual Information for Captopril and coughing. The Information Component (IC) is the strength of the association between a specific state in each of two variables and is the logarithmic form of the symmetrical factor relating the prior and posterior probability stated previously:

$$IC = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

Thus there are four ICs which refer to the different combinations of the variables Captopril and coughing.

In this paper we are only interested in the strength of the IC between specific drugs and adverse drug reactions present on the same report, not the other three possible ICs for this combination of binary variables. For the rest of this paper the IC will specifically refer to this particular combination of states of the drug and adverse reaction, which adds to the robustness and simplicity of the method as only positive reports of drug or adverse reaction need be counted (as well as the total number of reports). When the IC is positive for a drug-ADR association, this implies that the drug-ADR pair are more strongly associated than expected, compared to the c_i and c_j values and the rest of the database, the reverse applies to negative values, and IC values close to zero represent independence between the drug and ADR - that is the prior and posterior probabilities are the same, additional information about the drug does not change the probability of the ADR being present on a specific case report. We therefore intend to search the database for positive values of IC.

In the BCPNN the weight between a neuron in the adverse reaction layer and a neuron in the drug layer is equal to the IC for that specific drug adverse reaction combination. There is a 'finite' probability of any drug being reported with, that is suspected of being associated with, any ADR. This probability may be extremely small so it may never have occurred much less been reported. The pharmacovigilant community, by increasing awareness and reporting rates, is trying to obtain, by a variety of methods, as accurate an estimate of this probability as possible. The IC that we calculate is a measure of the strength of association of a drug adverse reaction combination, as the IC is only calculated on a finite number of reports it is merely an estimate of the real 'IC', the more reports we have the more accurate this estimate becomes. In our database the numbers of reports of individual drugs, ADRs, and drug-ADR combinations vary enormously. However the higher the values of c_i, c_j and c_{ij} the more accurate an estimate of the IC we have. Thus for every drug-ADR combination, we determine an interval estimate of the IC as a measure of certainty of the value of the IC. The combination of the absolute value of the IC and its interval estimate gives us an estimate of the probability of a specific association between a drug and ADR based on the spontaneous drug reports in our database. Having highlighted the association it can then be investigated further using the current signalling procedure in place at the Uppsala Monitoring Centre.

The Bayesian approach is based on the following: an estimation of a prior probability is made. This estimate is then improved when some new information is received by calculating a posterior probability based on both the prior probability estimate and the new information. This process is then repeated and the estimate will be constantly improved as more information is obtained.

As we do not know the 'real' probability of $p(A)$, $p(D)$ or $p(A,D)$, we assert a beta distribution [15] for each probability. From these distributions we calculate the 'expectation values' and variances of the beta distribution of each variable. The expectation value of each beta distribution is the estimate of the probability. As the counters c_i, c_j, c_{ij} and C increase we calculate new beta distributions for $p(A)$, $p(D)$ and $p(A,D)$, based on the prior distributions and these new counters, and therefore new expectation values and variances for each of the three. These distributions become narrower as we obtain more information (ie the variance always decreases). As the counters increase in value, the previous posterior distributions become the new prior distributions, and a new set of posterior distributions can be calculated.

This Bayesian approach allows us to estimate the probabilities, and hence the IC, even for low counter values. The calculation of the variance for $p(A)$, $p(D)$ and $p(A,D)$ as well as the IC provides an indication of the certainty of these probability estimates.

The Gaussian approximation of calculating the variance of a function of many variables allows us to calculate the variance of the information component ($V(IC)$) from the variances of $p(A)$, $p(D)$ and $p(A,D)$. Using this method, the variance of the IC is calculated by:

$$V(IC) \approx \left(\frac{1}{\log 2}\right)^2 \left[\frac{C - c_{ij} + \gamma - \gamma_{ij}}{(c_{ij} + \gamma_{ij})\,(1 + C + \gamma)} + \frac{C - c_i + \alpha - \alpha_i}{(c_i + \alpha_i)(1 + C + \alpha)} + \frac{C - c_j + \beta - \beta_i}{(c_j + \beta_j)(1 + C + \beta)}\right]$$

where α_1 and α_0 are factors in the beta distribution of $p(A)$ and $p(D)$ and γ_{11} and γ are the corresponding factors for the joint probability $p(A,D)$ [15]. Both pairs of factors reflect our beliefs in the probabilities given by the prior beta distributions. An a priori assumption is made of equal probability distribution for $p(A)$ and $p(D)$, as any probability is as likely as any other without further information, in a beta distribution this corresponds to the constants α_1 and α_0 (where $\alpha = \alpha_1 + \alpha_0$) being defined as: $\alpha = \alpha_1 + \alpha_0 = 1$. γ_{11} and γ define the joint beta distribution $p(A,D)$.

We set $\gamma_{11} = 1$ and define γ (Orre et Lansner personal communication) by:

$$\gamma = \frac{\gamma_{11}}{p(A)p(D)}$$

such that the IC tends to zero as c_{ij} and C tend to zero, because we assume an independent relationship between a drug-ADR when we have no reports of the drug or the ADR.

4 Implementation of BCPNN on the WHO database

All experiments done using the BCPNN followed the same procedure: To calculate the ICs between all drugs and all adverse reactions in the database we needed to calculate c_i, c_j and c_{ij} for all possible combinations. Thus we specified that we wanted to associate all drugs recorded as 'suspected' or 'interacting' with all adverse reactions, by giving a layer specification to the ANN software, such that one layer contained neurons representing drugs in the database and another layer contained neurons representing adverse reactions. The relevant adverse reactions and drugs were then specified and a counter generated for each. The network was trained by reading from the database and updating these statistical counters. Training and learning in the network occurred during the same run over the whole database and a matrix was generated containing the c_i, c_j and c_{ij} values for the specified drug-ADR combinations, which could then be further analysed. This matrix was limited in size by the use of the sparse matrix method, which creates the statistical counters as they are required, hence only non-zero counters occur in the database and run times are shortened considerably. Having generated the counters, the ICs and their corresponding 95 % confidence limits are then calculated.

In order to generate signals on the basis of ICs and their associated interval estimates, it was important to demonstrate that the ICs increased in value over time for a signal as the data on the particular association increased. Several time scan experiments were done by specifying a particular drug-ADR combination, and calculating the ICs and confidence limits at quarterly time intervals.

The results were graphed. From these time scans three examples were chosen to illustrate:

- When a signal of high probability would have been generated on current information compared with the world literature reports cited in MEDLINE (captopril and coughing).
- The behaviour of false positive signals over time for both a drug-ADR association that has a low c_{ij} value (digoxin and acne) and a drug-ADR association that has a high c_{ij} value (digoxin and rash). Digoxin is one of the most commonly reported drugs in the database. However, digoxin is reported exceptionally rarely with the common adverse reaction 'acne', whereas digoxin and the commonly reported ADR rash are frequently reported together

In a routine operation for finding signals, we intend to do quarterly updates. All ADRs and drugs that occur in the latest quarterly production will be selected, and the effect of these newly received case reports on drug-ADR associations throughout the database will be examined. To test this we selected a test set of case reports, approximately 36000 from the end part of 1995, to represent a new quarter of reports being received at the centre (all later reports were excluded completely from this test), and selected all drugs and ADRs that occurred in this list by doing the search on the whole database, up to but excluding this 'test' quarter, we then repeated the scan having added the 'new' data set. The ICs and their associated 95 % confidence limits were then compared before and after the new information was added. The criterion for a signal was drug-ADR combinations where the lower 95 % confidence limit of the IC changed from a negative to a positive value on addition of the test quarter. That is all drug-ADR combinations where the probability of a relationship between the drug-ADR based on the spontaneous reports in the database changed from below 0.975 to above, on addition of the test quarter.

3 Results

1 Example of early signal detection

Figure 1 shows the time scan obtained when a run was done of captopril (reported as 'suspected' or 'interacting' drug) and coughing from the first quarter of 1979 to the first quarter of 1996. The IC increases considerably in value over time, as the number of reports of the drug-ADR (c_{ij}) association increases, as also the total numbers of reports of the drug (c_i) and ADR (c_j) increase, so the interval estimate of the IC decreases, that is our estimate of the 'real' IC becomes more precise. The combination of these two effects is that the lower 95 percent confidence limit increases in value markedly towards and above zero. As can be seen on Figure 1 the lower 95 percent confidence limit crosses above zero at time 81/3, that is once the reports from the third quarter of 1981 had been added. At this time there were three reports of this association in the database. When this lower 95 percent confidence limit is equal to zero, This is a very strong association statistically which demands to be investigated pharmacologically, clinically and epidemiologically. An isolated literature report of this now well known signal was published in Dutch in July 1983 [16], but the signal was not widely reported in the literature until 1986. It should be noted that the confidence interval estimate of an IC estimate becomes smaller as we base the estimate on more samples, and as the IC value stabilises. A decrease in the confidence interval of ICs as time passes is a property of all drug-ADR time scans. Thus a stabilised positive value

IC for a drug-ADR association will imply an ever increasing likelihood of a real signal as further reports are added to the database.

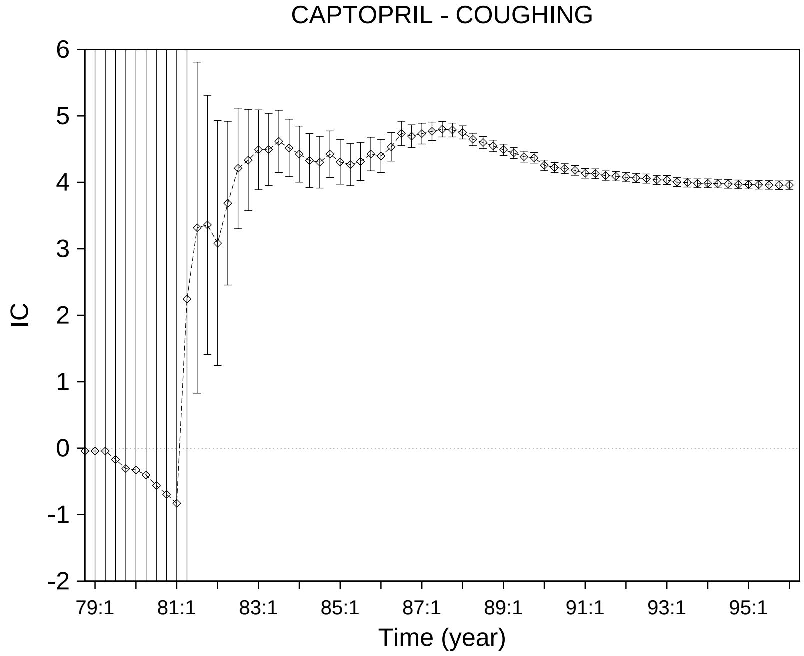


Figure 1: The change in IC between 1979 and 1996 for the association captopril-coughing. The IC is plotted at quarterly intervals with 95 percent confidence limits shown.

2 Examples of false-positive signal avoidance

1) The time scan of digoxin (suspected or interacting) against acne from the first quarter of 1967 to the start of 1996 is shown in figure 2. The IC decreases throughout the time scan, as no reports are received of this association - apart for a slight increase after quarter 1988/4 when the combination was reported once. As the associated 95 percent confidence interval diminishes with the numbers of reports of drug and ADR increase, this makes our IC estimate better and demonstrates the diminishing possibility of a causal relationship between digoxin and acne.

2) When a time scan of digoxin and rash is observed (Figure 3) the IC increases initially (1968), indicating a trend to a possible association (but with a large 95 percent confidence interval), but then decreases markedly towards a distinct negative value. Also the 95 percent confidence interval decreases rapidly because of the rate of increase of the number of the reports of digoxin, rash and digoxin-rash association. This definite negative IC represents a situation where although digoxin and rash are reported together often (high c_{ij}), yet relative to the values of c_i and c_j they are not. Thus this association does not stand out in our database as being more common than the generality. The probability of rash being reported on a specific case report with an unidentified drug is not increased if the drug is digoxin. This means that in our database there is no unexpectedly strong association between digoxin and rash. Therefore, this would not be signalled on our criteria.

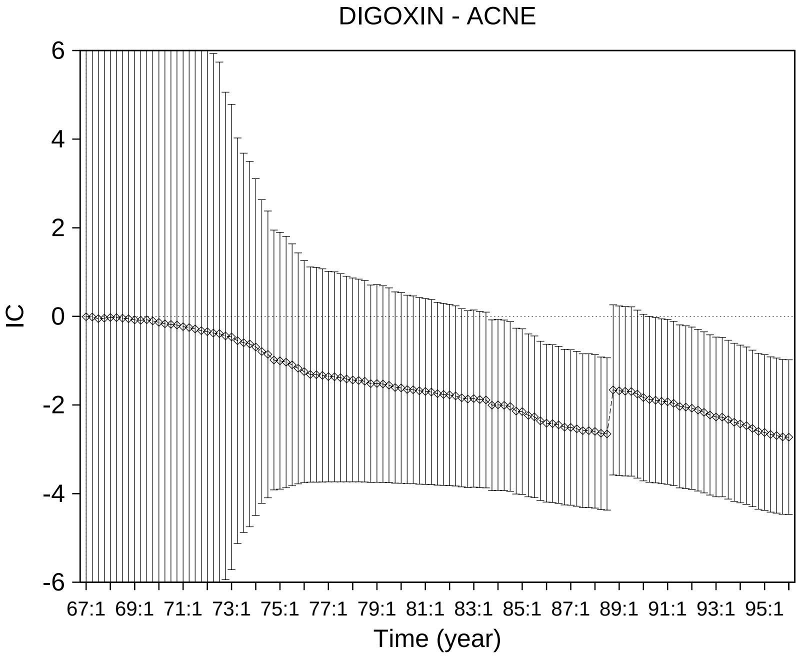


Figure 2: The change in IC between 1968 and 1996 for the association digoxin-acne. The IC is plotted at quarterly intervals with 95 percent confidence limits shown.

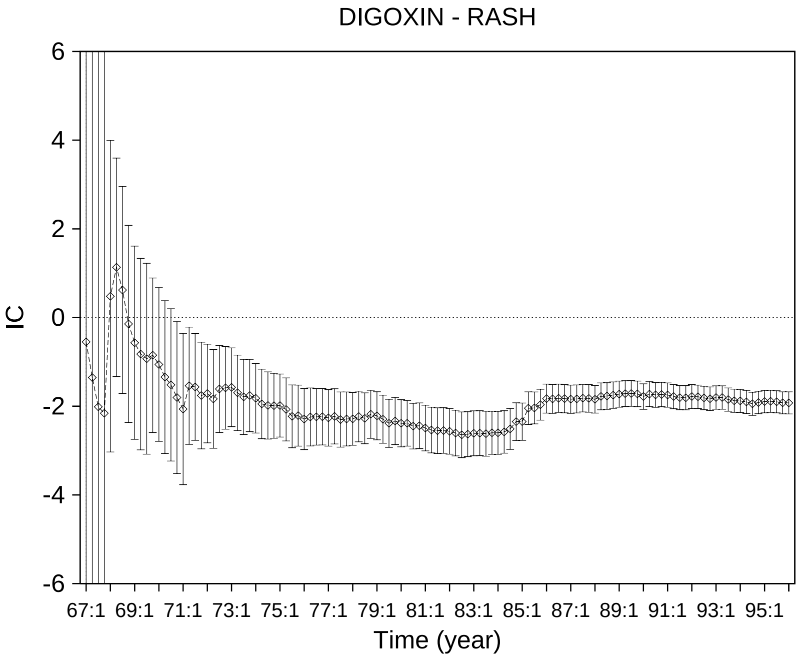


Figure 3: The change in IC between 1968 and 1996 for the association digoxin-rash. The IC is plotted at quarterly intervals with 95 percent confidence limits shown.

3 Quarterly update test

In order to simulate a regular quarterly screening of the database, an examination of the signalling criteria was done using historic data from a 3-month batch of 36000 reports from the start of 1996, using the procedure described above. In this way a list was generated containing 1004 different drug-ADR associations, where the lower 95 percent confidence limit of the IC estimate crossed above zero as a consequence of the addition of the 'test' quarter. This list contained 307 associations of ADR terms in the WHO 'Critical Terms List'. A 'critical' term is defined as referring to or possibly being indicative of a serious disease state, and have been regarded as particularly important to follow up. A serious disease is one that may be fatal, life-threatening, causing or prolonging inpatient hospitalisation, or resulting in persistent or significant disability or incapacity. Of these 307 associations, 53 associations were of 'new' drugs, that is drugs first reported since the start of 1990. These 53 associations were compared with entries for the relevant drugs in the latest editions of Martindale (Micromedex Healthcare Series Volume 93) and the Physicians Desk Reference Drug Interactions, Side Effects, Indications, Contraindications System TM, May 1997 (PDR), and Reactions online (ADIS Press). Of these 53 associations there were 12 that were not recorded in Martindale, PDR, or Reactions at the time of writing, (table 1).

Whilst the procedure is close to the routine previously used to find important signals, no clinical assessment has been made to exclude confounding drugs or disease (which may account for the relationship between fluvastatin and myocardial infarction, for example). On the other hand some judgement has been made over the selection of the final 12 associations on the basis that the reported term in the database was notably different from any described in the literature sources used. It is noteworthy that, whilst the association of alendronic acid and oesophagitis was well referenced, the more serious ulceration was a topic of concern and debate at the last annual meeting of the national centres participating in the WHO Programme for International Drug Monitoring, October 1997, Geneva.

Drug	Adverse reaction	Comments*
Clarithromycin	Larynismus	-
	Renal failure acute	-
Losartan	Cardiac failure	-
	Pulmonary oedema	-
	Peripheral ishaemia	-
Alfuzosin	Coma	-
	Angina pectoris	*Martindale* refers to prazosin for adverse effects; chest pain mentioned for prazosin
Sertraline	Arrhythmia	-
Fluvastatin	Myocardial infarction	-
Venlafaxine	Delirium	*Reactions* mentions hypomania in August 1997
Measles,mumps	Paralysis	-
and rubella vaccine		-
Alendronic acid	Oesophageal ulceration	Oesophagitis listed in *Martindale*,one case of oesophageal ulcer reported in *Reactions* May 1997

* In the absence of comments, there has been no mention of the listed reactions in the reference

Table 1: Signals on new drugs (record in the WHO database after 1990) and WHO "Critical Terms" identified in test run on data from the end of 1995.

4 Discussion

Since the instigation of the WHO drug monitoring programme the potential benefits and need for an automatic signalling system were envisaged [5]. The BCPNN, based on Bayesian statistics and neural network architecture and methods, allows us to find and quantify relationships between two or more data fields, such as a drug and ADR, that differ significantly from the background of inter-relationships in the database. We are thus able to highlight potential signals by the behaviour of a selected IC over time, and have demonstrated that it will be possible to find such relationships at earlier stages in the drugs life than at present. As such it is a suitable tool for the signalling of adverse drug reactions.

The limitations of spontaneous reporting of adverse drug reactions, from varying degrees of under reporting, to delays in reporting, to misreporting, to incomplete information, are well understood [17]. Despite the limitations spontaneous reporting has established itself as an effective tool in drug monitoring, once these limitations are appreciated [18]. The intention behind a signal is to bring a particular drug-adverse reaction to the attention of the pharmacovigilant community as quickly as possible on the basis of spontaneous ADR data, so that it can be investigated in more detail to maximise drug safety. A signal is not 'right' or 'wrong'. It is merely a suggestion of a possible problem, aimed at highlighting potential drug problems not discovered in clinical trials. On the basis of our proposed signalling production criteria, we would have produced a signal relating captopril with coughing with considerable confidence in the strength of the reported relationship before the third quarter of 1981. That would have been two years in advance of its first mention in the literature. We have also shown that false-positive signals can be avoided.

The quarterly update demonstrated the BCPNN system's ability to highlight potential signals from a large amount of data and verified that computationally we are able to search for IC values through the whole database. This new system is to be used in conjunction with the current signalling process. All drug-ADR pairs will be considered in the quarterly updates. The quantitative certainty of substantial difference from the background of reports in the database attributed to some drug-association will highlight them for the benefit of clinical reviewers. This will emphasise the need for further work on the reported signal. However, a high IC, like strong statistical correlation, does not imply that there is a direct causal relationship between eg. A drug and a reaction, it merely suggests the possibility of one.

The environment of our database is dynamic. Many factors may influence the database: new drugs are frequently added to the database, new countries start reporting, drug uses and advertising approaches will change. Thus there is a shifting background of associations mean that ICs and their distributions will change over time as the database gradually evolves, irrespective of additional data affecting any two fields. The problem of getting early and useful ADR signals from 2 million case records is like finding the proverbial 'needle in a haystack'. Data mining is like a magnet in providing a powerful tool for finding signals. Then the whole database becomes the control, so that any new association highlighted can be contrasted, with a determined level of significance, against the background of all reported information. In this case the 'haystack' size becomes an advantage in providing a stable norm for adverse drug report experience.

This BCPNN methodology will continue to be developed: Further investigation of drug-ADR associations will be possible by examining the behaviour of the combined IC of more than 2 fields, such as ADR and drug with other reported fields like age, gender and drug indication, to arbitrary complexity. Also, other categories of 'C' than the total database could be used eg. All reports on antibiotics or all reports on females.

All searches of the database so far have been for drug-ADR associations where the drug has been reported as being 'suspected', that is reports where the drug has been recorded as concomitant medication have been excluded. In future work investigation will be made on the impact of the drug causality on the possibility of a signal, since clinical/pharmacological preconceptions in drug causality should be avoided in determining new signals, as they reflect biased assessment of the drug-ADR association. There may be drugs which are not known to cause an adverse reaction, and are inappropriately encoded as 'other' drugs. However many false signals would occur, many for those drugs used frequently in combination with drugs known to cause specific ADRs.

As described above the test quarterly update was thresholded using a positive value of the lower 95 % confidence limit of the IC as the criteria for a reasonably certain signal - further investigation will be done to verify or improve this thresholding level. Although the existing BCPNN is robust in situations where there is missing data, some improvements can be made by inference calculations for the incomplete fields. This could improve the sensitivity of the system. This new methodology is intended to enhance, not replace, the systems that are currently used to detect signals. The value of experts in the field cannot be overestimated, as the qualitative risk-benefit assessment of potential signals is an essential step in the process of their detection and evaluation.

References

1. Delamothe T (1992) Reporting adverse drug reactions. Br Med J 304: 465
2. Fucik H, Edwards IR (1996) Impact and credibility of the WHO adverse reaction signals. Drug Information Journal 30: 461-464
3. Meyboom RHB, Egberts ACG, Edwards IR, Hekster YA, de Koning FHP, Gribnau FWJ (1997) Principles of Signal Detection in Pharmacovigilance. Drug Safety 16(6): 355-365
4. See: Methodological approaches in pharmacoepidemiology: application to spontaneous reporting (1993). B gaud B, Chaslerie A, Fourrier A, Haramburu F, Miremont G (eds.) Elsevier Science Publishers BV, Amsterdam, The Netherlands.
5. Finney DJ (1974) Systematic signalling of adverse reactions to drugs. Meth Inform Med 13(1): 1-10
6. Cross SS, Harrison RF, Kennedy RL (1995) Introduction to neural networks. Lancet 346: 1075-1079
7. Hertz J, Krough A, Palmer RG (1991) Introduction to the theory of neural computation. Addison-Wesley, Redwood City, California
8. Lansner A, Holst A (1996) A higher order bayesian neural network with spiking units. Int J Neural Syst 7(2): 115-128
9. Lansner A, Ekeberg (1989) A one-layer feedback artificial neural network with a bayesian learning rule. Int J Neural Syst 1(1): 77-87
10. Holst A (1997) The use of a Bayesian neural network model for classification tasks [dissertation]. Stockholm: Royal Institute of Technology
11. Holst A, Lansner A (1996) A higher order neural network for classification and diagnosis. In: Gammerman A (ed) Computational Learning and Probabilistic Reasoning. John Wiley & Sons Ltd, Chichester, pp199-209
12. Holst A, Lansner A (1993) A flexible and fault tolerant query-reply system based on a Bayesian neural network. Int J Neural Syst 4(3): 257-267
13. Orre R, Lansner A (1996) Pulp quality modelling using bayesian mixture density neural networks. J Syst Eng 6: 128-136
14. Pearl J (1988) Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann Publishers, Inc., San Francisco
15. Heckerman D (1997) Bayesian networks for data mining. Data Mining and Knowledge Discovery 1: 79-119
16. Knoben JM (1983) Prikkelhoest door gebruik van captopril. Nederlands Tijdschrift voor Geneeskunde 127(29): 1306
17. ten Ham M (1992) WHO's role in international ADR monitoring. Post Marketing Surveillance 5: 223-230
18. Edwards IR, Wiholm B-E, Martinez C (1996) Concepts in Risk-Benefit Assessment. Drug Safety 15(1): 1-7

Supplement III

Modelling With Probabilities

Pulp Quality Modelling Using Bayesian Mixture Density Neural Networks

Roland Orre and Anders Lansner

SANS, Dept. of Numerical Analysis and Computing Science
Royal Institute of Technology, S–100 44 Stockholm, Sweden
E–mail: orre@sans.kth.se, FAX: +46–8–790 09 30

Abstract

We model a part of a process in pulp to paper production using Bayesian mixture density networks. A set of parameters measuring paper quality is predicted from a set of process values. In most *regression* models, the response output is a real value but in this mixture density model the output is an approximation of the density function for a response variable conditioned by an explanatory variable value, i.e., $f_Y(y|X = x)$. This density function gives information about the confidence interval for the predicted value as well as modality of the density. The representation is Gaussian RBFs *(Radial Basis Functions)*, which model the *a priori* density for each variable space, using the stochastic EM (*Expectation Maximization*) algorithm for calculation of positions and variances. Bayesian associative connections are used to generate the response variable *a posteriori* density. We found that this method, with only two design parameters, performs comparably well with backpropagation on the same data.

keywords mixture density neural network function approximation

1 Introduction

The fundamental problem we look upon here is function approximation from a set of explanatory (X) and response (Y) variables. The purpose is to model a process, which is assumed to be determined by these variables. We do not handle any *temporal* behaviour of the process here. In the initial phase of this project, which was done in cooperation with STORA Teknik AB, we used feedforward networks trained with the error backpropagation (BP) algorithm [7]. In the continuation of this project, which was done in cooperation with STFI *(Swedish Pulp and Paper Research Institute)* we developed a mixture density model for function approximation [8].

Mixture density networks have been used for, *e.g.* classification of speech segments and satellite image pixels [10] and classification of globins in protein sequences [6]. Function approximation has been done by, *e.g.* predicting the *a posteriori* density [1] or using the EM-algorithm directly [2]. The method of predicting the *a posteriori* density using a Bayesian associator as hidden layer has not been much used, as far as we know, but has earlier been suggested by, *e.g.* [3] and [6]. The prediction of the *density function* for the response value gives a way to detect ambiguous response values as well as to get a quality measurement of the prediction. In figure 1 a sketch of the density method we propose here is presented.

We use a stochastic EM-algorithm [9] for the RBF-units and a BCPNN (Bayesian Confidence Propagation Neural Network), which have earlier been successfully used for pattern completion [2] and classification [4], to associate an explanatory conditioned density with a response density function.

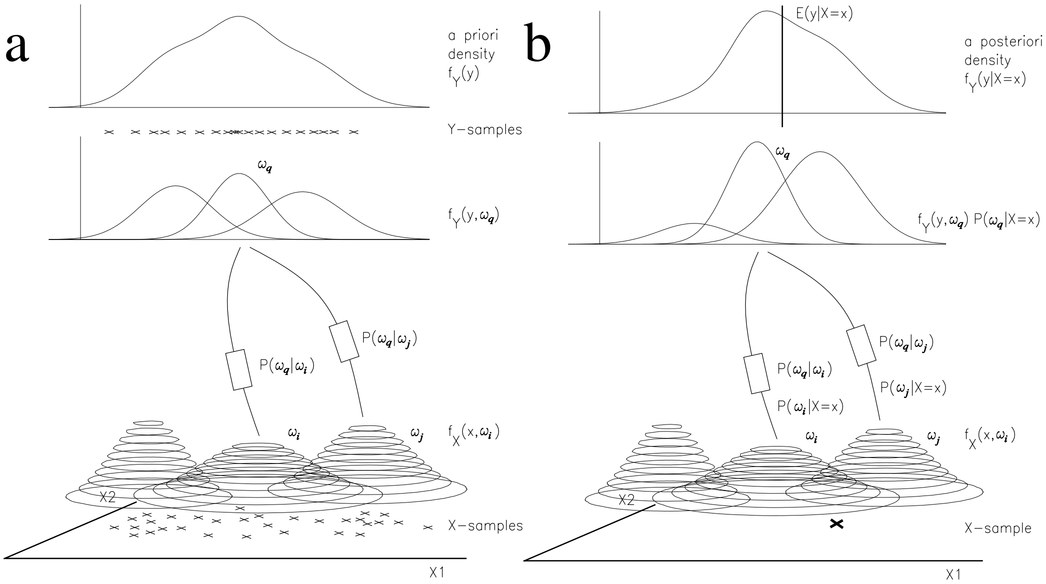


Figure 1: An overview of the density method. **a:** Training phase: We let a set of Gaussian density functions (ω_i) model the density of a set of data samples in explanatory (X) and response (Y) variable spaces. When an X-sample is drawn from the process ω_i in explanatory space there is a certain probability $P(\omega_q \mid \omega_i)$ that a Y-sample is drawn from the process ω_q in response space. **b:** Recall phase: When a certain X-sample is input we get a response probability $P(\omega_i|X=x)$ from each of the explanatory *processes*. These probabilities are propagated to the response space and cause the response variable density to be conditioned by the explanatory value $X=x$.

2 Data and Experiment Setup

The data we have been working with here is a set of process input parameters being automatically measured and a set of process outputs being manually measured in lab. In table 1 are the names and ranges for these variables listed. There are three different pulp types where the input variables *%pulp type1, %pulp type2* and *%pulp type3* are percentages of their contents in the pulp mix. There are two fiber length classes *% middle* and *% long* were the measure is percentage of the length class versus other fiber length classes. The *drain-time* is a measure of how long time it takes for standardized piece of pulp to drain. The *drain-speed* is a measure of how quick the water flows out of the pulp. The variables *%medium, %long, drain speed* and *drain time* are being automatically measured with an interval of about one hour. The manually measured output values (y) being predicted, here *tear* and *tensile*, are measured by lab experiments about once a day.

In figure 2 we see the experimental setup used for prediction of these outputs. The pulp types and fiber classes are fed to the input layer. From the first hidden layer with RBFs we get the probabilities $P(\omega_i|X = x)$, *i.e.* probabilities for the x-values to belong to one of the "classes" ω_i. The weights $(P(\omega_q|\omega_i)$ between the hidden RBF layer and the density output layer associates a class ω_i in the input layer with a class ω_q in the output layer. A summation in the density output units gives the probabilities $P(\omega_q|X = x)$, which is the probability for a certain density function ω_q to be the generating response process when a certain explanatory sample value x is fed into the input layer.

The output (y) is calculated as the center of mass for the active output densities. To be able to estimate a confidence interval for this output there is also an integration (formula 16) being performed, which is not shown in figure 2. In the following two chapters is a brief theoretical description of the *a priori* and *a posteriori* density estimations given.

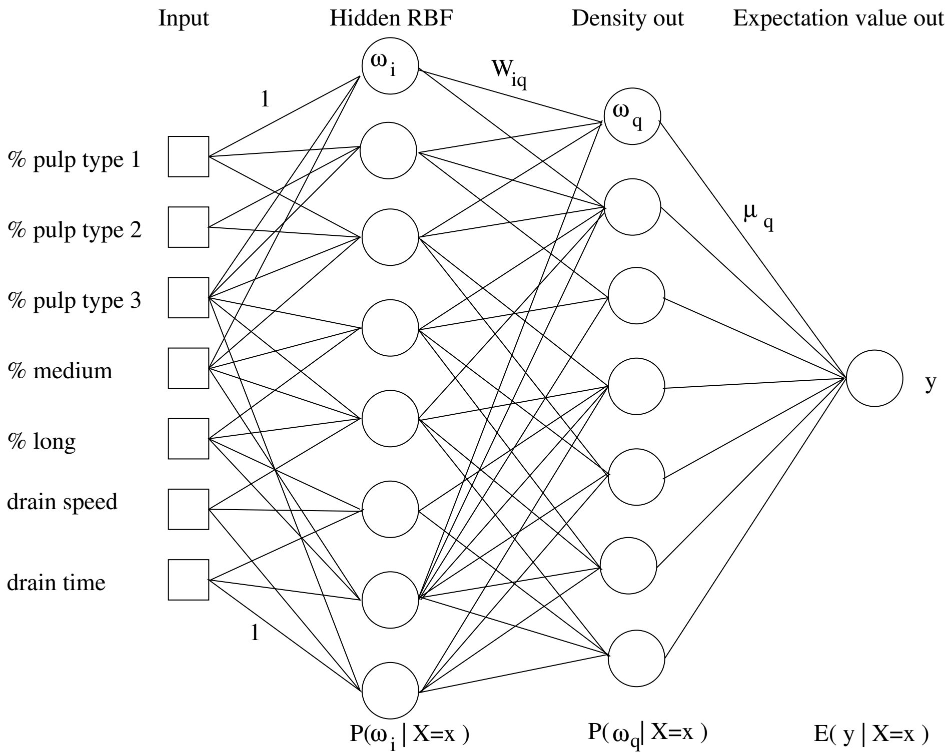


Figure 2: Setup of experiment. At left we input the process parameters, at right we output the response variable (e.g. y=tear or y=tensile) value. The input units just distribute the explanatory values to all hidden RBFs. Outputs from the hidden RBF units are $P(\omega_i|X = x)$, i.e. probability that value x belongs to class ω_i. The weight values between the hidden RBFs and the density output layer are $P(\omega_q|\omega_i) = \frac{P(\omega_i \& \omega_q)}{P(\omega_i)P(\omega_q)}$. For a formal description see (15) .

parameter	range		
% pulp-type1	0	$\cdots$	100
% pulp-type2	0	$\cdots$	100
% pulp-type3	0	$\cdots$	100
% middle	22.2	$\cdots$	30.9
% long	51	$\cdots$	75
drain-time	29.2	$\cdots$	45.9
drain-speed	255	$\cdots$	367
tear	11.6	$\cdots$	16.5
tensile	60.8	$\cdots$	84.7

Table 1: The set of variables used listed with their active ranges. The ones above the line are inputs, *i.e. explanatory* variables. The *response* variables *tear* and *tensile* below the line are the ones being predicted.

3 A Priori Density Approximation

The *a priori* density functions for *explanatory* ($f_X(x)$) and *response* ($f_Y(y)$) variables are composed of component densities ω_i (1).

$$f(x) = \sum_{i=1}^{n} P(\omega_i)f(x|\omega_i) = \sum_{i=1}^{n} \alpha_i\varphi(x,\theta_i) = [e.g.Gaussian], = \sum_{i=1}^{n} P(\omega_i)N(x,\mu_i,\sigma_i^2) (1)$$

Each component i is characterized by a set of parameters, which for the Gaussian case would be a *covariance matrix* C or a *variance* σ_i^2 for one dimensional or symmetrical densities, a *center value* μ_i and a *probability* $P(\omega_i)$. These parameters are here estimated by the *EM algorithm*. We have a set of N samples $\{x_1,\ldots,x_n\}$ which is drawn from a mixture, $f(x)$, of n density functions (1). By applying Bayes rule about conditioned probability on the density functions we get an expression for the probability that a certain X-value x was generated from the component ω_i (2). Then search the parameters α_i and θ_i which maximizes the log-likelihood (L) of the samples under the constraint that the probabilities α_i sum to 1, which can be solved by using the Lagrange multiplier method.

$$P(\omega_i|x) = \frac{P(\omega_i)P(x|\omega_i)}{P(x)} = \frac{\alpha_i\varphi(x,\theta_i)}{\sum_{j=1}^{n}\alpha_i\varphi(x,\theta_i)} \quad , \quad \log L = \sum_{k=1}^{N}\log f(x_k), \tag{2}$$

If we assume Gaussian component densities we get expressions for estimates of the center values μ_i and covariance matrices C_i (3) as a set of non linear equations which can be solved numerically. (C_i is the covariance matrix for component i and d is the number of dimensions for the variable)

$$\hat{\mu}_i = \frac{\sum_{k=1}^{N}\hat{P}(\omega_i|x_k)x_k}{\sum_{k=1}^{N}\hat{P}(\omega_i|x_k)} \quad , \quad \hat{C}_i = \frac{\sum_{k=1}^{N}\hat{P}(\omega_i|x_k)(x_k-\mu_i)(x_k-\mu_i)^T}{\sum_{k=1}^{N}\hat{P}(\omega_i|x_k)}. \tag{3}$$

A stochastic variant of the EM-algorithm [9], has been used here. Both μ_i and σ_i^2 for N samples of a form which can be rewritten into a recursive expression (4)

$$\theta_N = \frac{\sum_{k=1}^{N} P(\omega|x_k)\theta(x_k)}{\sum_{k=1}^{N} P(\omega|x_k)} \qquad \begin{aligned} \theta_{N+1} &= \frac{\sum_{k=1}^{N+1} P(\omega|x_k)\theta(x_k)}{\sum_{k=1}^{N+1} P(\omega|x_k)} \\ &= \frac{P(\omega|x_{N+1})\theta(x_{N+1}) + \sum_{k=1}^{N} P(\omega|x_k)\theta(x_k)}{\sum_{k=1}^{N+1} P(\omega|x_k)} \\ &= \frac{P(\omega|x_{N+1})\theta(x_{N+1}) + \sum_{k=1}^{N+1} P(\omega|x_k)\theta_N - P(\omega|x_{N+1})}{\sum_{k=1}^{N+1} P(\omega|x_k)} \end{aligned} \tag{4}$$

and simplified (5), where the step size η causes a competitive update among the units. To get a smooth start we scale down the step further by δ.

$$\theta_{N+1} = \theta_N + \eta_{N+1}(\theta(x_{N+1}) - \theta_N) \quad , \quad \eta_{N+1} = \frac{P(\omega|x_{N+1})}{\sum_{k=1}^{N+1} P(\omega|x_k)}\,\delta \tag{5}$$

The denominator for η (5) can be replaced with a moving average, caring mostly for the L most recent samples $D_{N+1} = (1-1/L)D_N + P(\omega|x_{N+1})$. To further simplify things we may assume symmetric density functions, then we don't need the covariance matrix. In the final incremental update expressions for center value (6) and variance (8) we use an intermediate estimate of the next value. For the center value (6) this is just the next sample and for the variance (7) it is the squared distance over the number of dimensions.

$$\mu_{N+1} = \mu_N + \eta_{N+1}(x_{N+1} - \mu_N) \tag{6}$$
$$\hat{\sigma}^2_{N+1} = (x_{N+1} - \mu_N)^T(x_{N+1} - \mu_N)/d \tag{7}$$
$$\sigma^2_{N+1} = \sigma^2_N + \eta_{N+1}(\hat{\sigma}^2_{N+1} - \sigma^2_N) \tag{8}$$

This is an "almost" parameter free algorithm. It needs an initial placement μ_i and variance σ_i^2 for the units but none is "critical". It is a good strategy to start updating the variances when the positions have somewhat begun to stabilize. As an example we can, in figure 3, see how a set of 40 RBFs have adapted to 676 data points forming a square. There is also an example with two variables of pulp data in the same figure. In the aspect of function approximation we will get a high resolution where there is a lot of samples. The placement of the PDFs is not necessarily unique, there may exist several solutions which maximize the likelihood (2).

4 A Posteriori Density Generation

After having found a model of the *a priori* density function of a variable we want to find the *a posteriori* density function, $f_Y(y \mid X = x)$, for a response variable conditioned by a certain explanatory variable value. We index the explanatory density components with i and the response density components with q.

$$f_X(x) = \sum_{i=1}^{n} P(\omega_i) f(x|\omega_i), f_Y(y) = \sum_{q=1}^{m} P(\omega_q) f(y|\omega_q) \tag{9}$$

By applying Bayes rule $[p(q|i) = p(q)p(i|q)/p(i)]$ to a component density we get an expression for the probability of an X-value being generated from the component ω_i, where

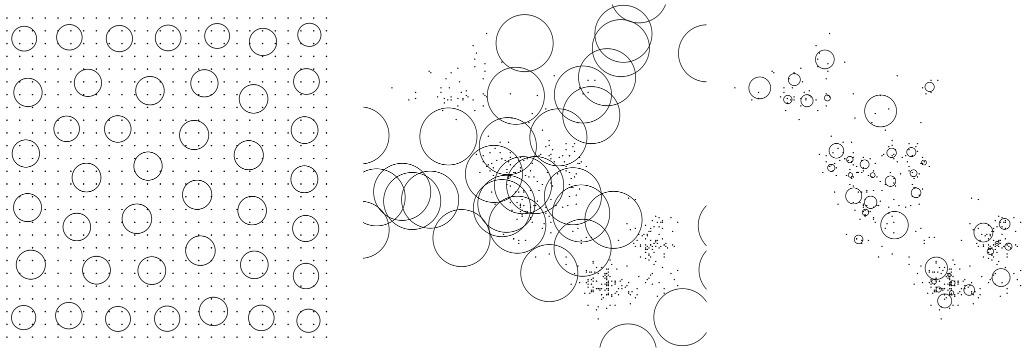


Figure 3: **Left:** 41 units are adapted to a square distribution. We see the RBF units with their σ plotted as circles and the data samples as dots. **Middle:** 40 units randomly initialized. **Right:** After about 200 iterations the units had become stable in this case.

we can look upon $f(x)$ as a proportionality factor for a certain X-value and thus, use normalization over all component probabilities for a certain X (10).

$$P(\omega_i|x) = \frac{P(\omega_i)f(x|\omega_i)}{f_X(x)} \propto P(\omega_i)f(x|\omega_i), \sum_{i=1}^{n} P(\omega_i|X{=}x) = 1 \tag{10}$$

Then we want to express the response variable density function conditioned by a certain X-value ($f_Y(y|X{=}x)$) as a relation between component densities of explanatory variables (ω_i) and response variables (ω_q) (9). We start by rewriting the probability of a response variable component density (9) being conditioned by an explanatory variable value.

$$f_Y(y|X{=}x) = \sum_q f_Y(y|\omega_q)P(\omega_q|X{=}x) \tag{11}$$

We then write the probability for a response variable density component conditioned by an explanatory variable value as a probability relation between explanatory and response components and explanatory component probabilities conditioned by the same explanatory value.

The probabilities for ω_q will only depend on the X-value through the probabilities for ω_i. When the $P(\omega_i|X{=}x)$ are "almost" mutually exclusive we can use "theorem about total probability" (12) and thereafter Bayes rule (13). The definition of conditioned probability $[p(b|a) = p(a \cap b)/p(a))]$ under the assumption that ω_q and ω_i are independent gives the expression (14).

$$\begin{aligned} P(\omega_q|X = x) &= \sum_i P(\omega_q|\omega_i)P(\omega_i|X = x) && (12) \\ &= \sum_i P(\omega_q)\frac{P(\omega_i|\omega_q)}{P(\omega_i)}P(\omega_i|X = x) && (13) \\ &= P(\omega_q)\sum_i \frac{P(\omega_q \& \omega_i)}{P(\omega_q)P(\omega_i)}P(\omega_i|X = x) && (14) \end{aligned}$$

We can now combine (11) and (14) to get the expression for $f_Y(y|X{=}x)$ (15). As was stated in (10) the $P(\omega_i|X{=}x)$ is just a version of $P(\omega_i)f(X{=}x|\omega_i)$ scaled so that their sum is normalized to 1. The component density $f(x|\omega_i)$ may be the normal distribution density function $N(x, \mu_i, \sigma_i^2)$.

$$f_Y(y|X{=}x) \quad = \quad \sum_q f_{Y_q}(y|\omega_q)P(\omega_q)\sum_i \underbrace{\frac{P(\omega_q \& \omega_i)}{P(\omega_q)P(\omega_i)}}_{W_{iq}} P(\omega_i|X{=}x) \tag{15}$$

The expression which is marked W_{iq} is similar to the expression which is used for weight calculation in a one layer Bayesian network for binary pattern recognition [2]. Finally we want a predicted value as output, which is the expectation value of the response value density function. This can be calculated by integrating the density function (16) to give "a center of mass". In the case with, e.g, Gaussian component densities, which were used here, we need not do an integration for this. We may just sum the center values μ_q weighted by their probabilities (17). We also want some measure of the prediction *quality*. By integrating (18) the density function we may estimate a confidence interval (19) for the prediction.

$$E(y|X{=}x) = \int f_Y(y|X{=}x)\, y\, dy \tag{16}$$

$$F_Y(\gamma) = \int_{-\infty}^{\gamma} PD_Y(y|X{=}x)\, dy \tag{18}$$

$$E(y|X{=}x) = \sum_q P(\omega_q|X{=}x)\, \mu_q \tag{17}$$

$$\left.\begin{array}{lcl} 0.025 & < & F_Y(y_1) \\ 0.975 & > & F_Y(y_2) \end{array}\right\} \Rightarrow y_1 \le Y_{95\%} \le y_2 \tag{19}$$

5 Response Value Prediction

For response value error calculations we use the standard deviation for the difference between actual and desired output as a percentage of the used value range $(0 \ldots 1)$. $\delta_i = y_{out} - y_{desired}$

$$\sigma_{err} = \sqrt{\frac{\sum_{i=1}^{n} \left(\delta_i - \frac{\sum_{i=1}^{n} \delta_i}{n}\right)^2}{n-1}}$$

In figure 4, we see an example on how the function $y = 0.5 \sin 10x$ is approximated with 60 explanatory units and 40 response being trained with 960 samples. We added some normal distributed noise with a standard deviation of 0.07 around the nominal function value. In figure 5, where we used 150 samples for the training set and 50 samples for the test set, we see how the performance on training and test set respectively on the same problem as above varies when the number of explanatory and response units are varied.

This mixture density method is actually symmetrical for predictions $X \to Y$ and $Y \to X$, which is illustrated in figure 6 where we first recall Y from X, in the normal way, and then X from Y. As the function $y = \sin(x)$ is not bijective the inverse mapping is multi valued which results in a multi modal density function.

In figure 7 we see how the estimation of expectation values and the confidence intervals improves as the number of samples increases for the example above. This is shown with regularization, described below, for the test set and without regularization for the training set.

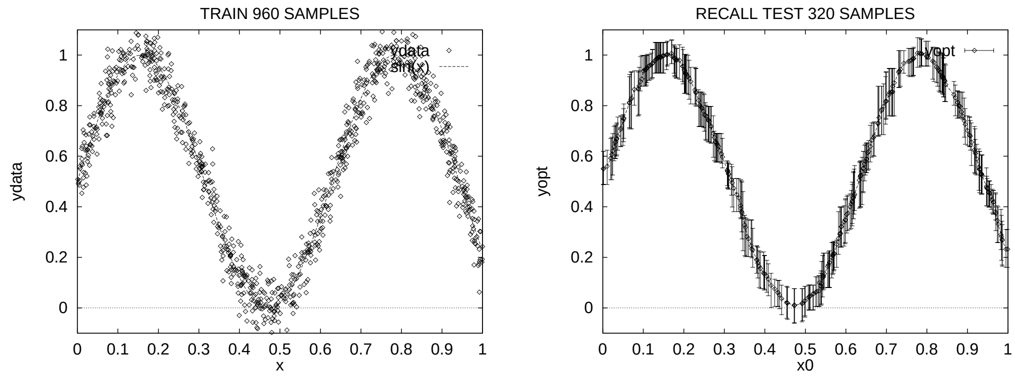


Figure 4: **Left:** Training data consisting of 960 samples with a normal distributed noise ($\sigma = 0.07$). **Right:** Recalled y-value from 320 samples with a network using 60 explanatory units and 40 response units. The error bars show one predicted standard deviation.

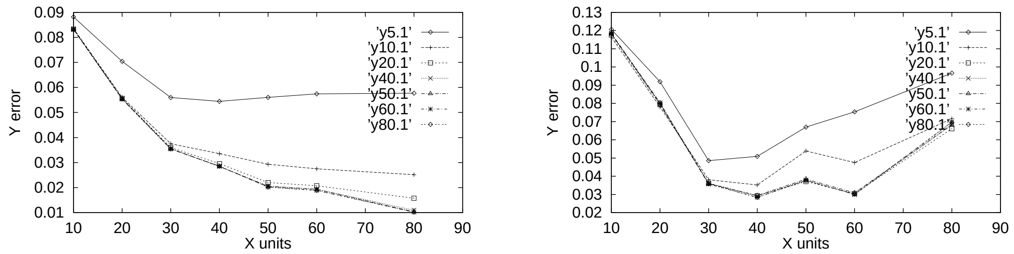


Figure 5: The performance for training set (left) and test set (right) when we vary the number of units in explanatory (X) and response (Y) layers. On the x-axis is the number of explanatory units. The Y units are shown by different curves in the same diagram.

1 Regularization, a Way to Improve Generalization

One way to improve the generalization performance when the RBF-model has been trained with a sample set which is too small or whose density is not representative for the whole set is to increase the "fuzziness" in the system by scaling up the variances. The sample will then be classified as possibly being generated from several close PDFs instead of just the few closest ones. A method which has proven useful is to increase the variances until the distance between the input sample vector and the expectation value of the PDFs which represent this value is minimized, figure 8.

6 Results: Predictions of *tear* and *tensile*

In figure 9 we see a pair of diagrams for the predictions of the two laboratory measured pulp response variables *tear* and *tensile* on a test set. Here is also an estimation of a 67 % confidence interval for the predicted values shown as error bars. Considering these intervals they are reasonable sized as they have about the same size as the measurement errors of the variables [7]. In some cases, like sample 9 for *tear* and sample 5 for *tensile* we may get a very large confidence interval due to the fact that the test sample value is far from any RBF

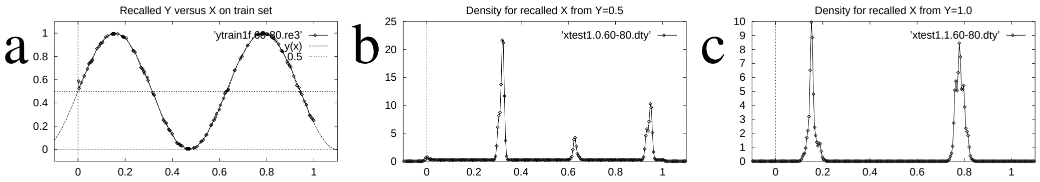


Figure 6: **a:** A recalled sine function as Y=f(X). **b:** Recalled density for X when input on Y is 0.5 (arcsin(Y)). **c:** Same as b when Y input is 1.

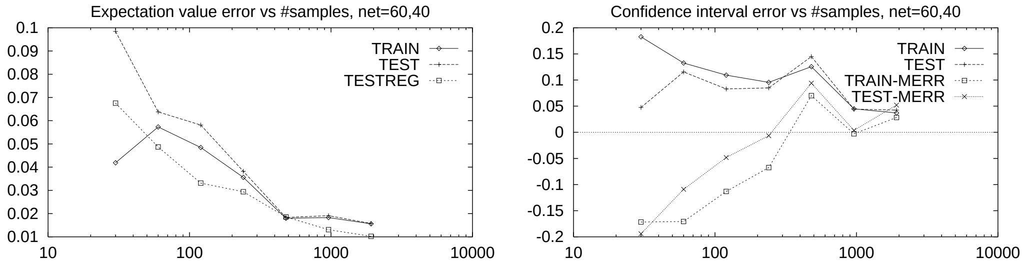


Figure 7: **Left:** The prediction performance of the expectation value improves for a specific network when the number of samples used for training increases. In this case we could perform better on the test set with regularization than on the training set without regularization most of the time. **Right:** The confidence interval estimation improvement as the number of samples increases. The two upper curves "TRAIN" and "TEST" show the σ_{err}. "TRAIN-MERR" and "TEST-MERR" show the average error. The confidence intervals for test set were calculated after regularization.

center, thus activating most units resulting in a predicted value close to the expectation value of the *a priori* density for the response variable.

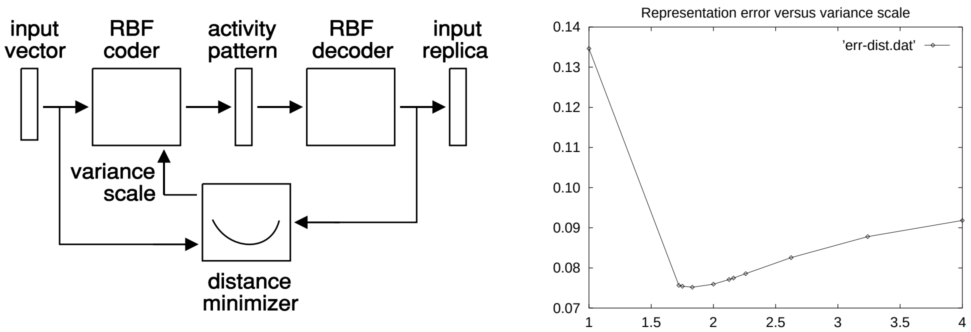


Figure 8: **Left:** Dynamic regularization by adapting the explanatory sample "fitness" by scaling the variances, thus improving the generalization capability. **Right:** Representation error vs. variance scaling (golden search used as distance minimizer).

In table 2 and table 3 we see *tear*, and *tensile* predicted from the six explanatory variables shown in table 1. Table 2 shows the performance on training sets and table 3 shows the performance on test sets. We can see how the prediction performance varies when the number of units in explanatory and response layer is increased from 10 to 80. The data sets in these tests was randomly partitioned into 75 % training and 25 % test data.

As a comparison with the results in tables (2 3) the best BP results obtained on test sets for these variables earlier [7] (search through many architectures) was 11.7 % for *tear* and 10.8 % for *tensile*. This indicates that this mixture density method performs equally well as BP considering the predicted values when the optimal architecture for a certain data set is used.

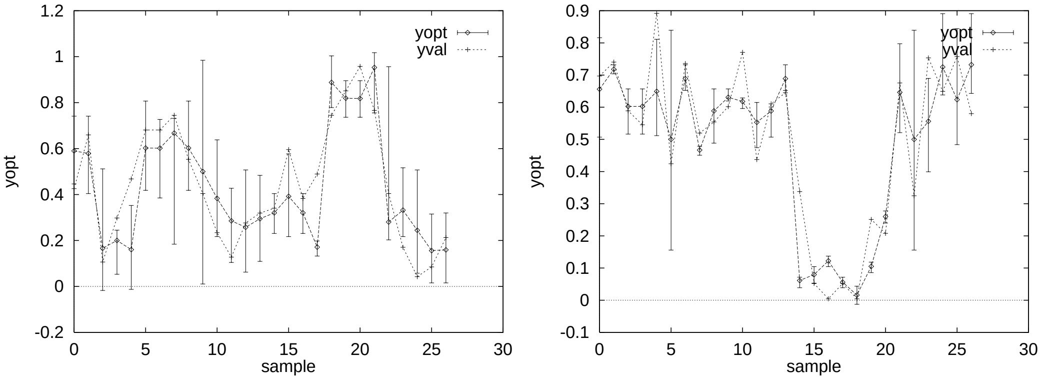


Figure 9: Two test sets with prediction outputs for *tear* and *tensile* with estimated 67% confidence intervals (*one standard deviation*) plotted as error bars. The predicted values are marked with ◇:s (with bars) and the process output values are marked with +:s. In some cases like sample 9 for *tear* and sample 5 for *tensile* we see a large confidence interval estimation due to the input vector being far from all the RBF units.

tear-TRAIN		tensile-TRAIN	
size	σ_{err}	size	σ_{err}
80-80	8.01	80-20	7.77
80-10	8.43	80-80	8.09
80-20	8.51	40-20	8.22
80-40	8.94	80-40	8.56
40-40	9.33	40-80	9.13
40-20	9.70	80-10	9.20
40-10	9.91	40-40	9.28
40-80	10.21	40-10	9.53
20-20	10.45	20-10	9.73
20-10	11.56	20-80	10.38
20-80	11.75	20-40	10.47
20-40	11.83	20-20	10.51
10-20	15.15	10-20	11.69
10-10	15.83	10-80	12.50
10-40	15.99	10-40	12.56
10-80	16.86	10-10	13.46

Table 2: Prediction performance on training data for *tear* and *tensile* for different sizes of explanatory and response layer, sorted by performance. A size “80-10” means 80 input RBFs and 10 output RBFs.

tear-TEST		tensile-TEST	
size	σ_{err}	size	σ_{err}
40-10	11.86	20-80	9.89
40-20	13.81	20-40	9.90
20-10	15.29	20-20	10.09
40-80	15.34	20-10	12.02
20-40	15.63	10-10	12.94
10-80	16.56	40-80	13.09
40-40	18.02	10-80	13.59
20-80	18.06	40-40	14.78
10-10	18.10	10-40	14.82
80-20	18.45	40-20	15.47
80-10	18.47	80-40	15.74
10-40	18.55	40-10	15.88
20-20	18.80	10-20	16.98
80-80	18.88	80-20	17.81
10-20	20.31	80-10	21.75
80-40	20.98	80-80	23.00

Table 3: Prediction performance on test data for *tear* and *tensile*. The best test result using BP we obtained earlier was 11.7% for *tear* (two hidden layers with 10 units in each) and 10.8% for *tensile* (same configuration).

7 Discussion

Advantages of using the mixture density model, compared with *e.g.* using MLP with BP are the following: 1) The RBF representations of variable spaces are built *un-supervised*, which is why expensive labeled examples are not needed at that moment. 2) The generalization can be dynamically improved due to the *regularization* capabilities of the RBFs, which decreases the requirement of cross validation. 3) The *design* (selection of architecture) of the network is rather easy and need not be done in a supervised way as the number of RBF units relates to the density and representation of the variable in each space. 4) The supervised training part can be done as a *one shot* quick process as this is just to collect statistics. 5) The predicted mixture density for response variables gives, besides the ability to estimate a confidence interval, also the ability to detect ambiguous output values, which will show up as a multi modal density function. 6) A missing value in an input sample vector is still usable as this only leads to a less specific conditioned *a priori* density in the missing dimension.

8 Acknowledgement

We hereby gratefully acknowledge STFI *Swedish Pulp and Paper Research Institute* and the Swedish Research Council for Engineering Sciences (TFR) under grant TFR 93-672 for sponsoring this project. We also gratefully acknowledge STORA Teknik AB for supplying us with process data.

References

[1] Chris M. Bishop. Mixture density networks. Tech. Rep. NCRG/4288, Department of Computer Science, Aston University, 1994.

[2] Zoubin Ghahramani. Solving inverse problems using an em approach to density estimation. In Mozer, Smolensky, Touretzky, Elman, and Weigend, editors, *Proceedings of the 1993 Connectionist Models Summer School*, pages 316–323, Hillsdale 1994, 1994. Erlbaum Associates.

[3] Anders Holst and Anders Lansner. The Bayesian neural network model and its extensions. Tech. Rep. TRITA-NA-P9325, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden, 1993.

[4] Anders Holst and Anders Lansner. A flexible and fault tolerant query-reply system based on a Bayesian neural network. *International Journal of Neural Systems*, 4(3):257–267, 1993.

[5] Anders Lansner and Örjan Ekeberg. A one-layer feedback, artificial neural network with a Bayesian learning rule. *International Journal of Neural Systems*, 1(1):77–87, 1989. Also extended abstract in Proceedings from the Nordic symposium on Neural Computing, April 17–18, Hanasaari Culture Center, Espoo, Finland.

[6] David J.C. MacKay. Bayesian neural networks and density networks. Proc. of Workshop on Neutron Scattering Scattering Data Analysis 1994, 1994.

[7] Roland Orre and Anders Lansner. A study of process modelling using artificial neural networks. Tech. Rep. TRITA-NA-P9239, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden, 1992.

[8] Roland Orre and Anders Lansner. Function approximation by prediction of a posteriori density with bayesian ann:s. Tech. Rep. TRITA-NA-P9413, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden, 1994.

[9] Hans G.C. Tråvén. A neural network approach to statistical pattern classification by "semiparametric" estimation of probability density. *IEEE Transactions on Neural Networks*, 2(3):366–118, 1991.

[10] Hans G.C. Tråvén. Invariance constraints for improving generalization in probabilistic neural networks. In *IEEE ICNN'93*, pages 1348–1353, 1993.

Supplement IV

Temporal Association

A Method for Temporal Association in Bayesian Networks

Roland Orre, Anders Lansner

Studies of Artificial Neural Systems
Dept. of Numerical Analysis and Computer Science
Royal Institute of Technology
SANS, NADA, KTH, S-100 44 Stockholm, Sweden

Abstract

A method for sequential pattern recognition and prediction in Bayesian networks is investigated. The basic approach in this method is to add stimulus delay lines to an associative network, thus converting temporal structure to a spatial one. Some methods to avoid very large connection matrices are studied. Results show that it is possible to efficiently store sequences in a network where the connection matrix is strongly reduced.

1 Background

The goal of the present work was to develop a useful model for temporal pattern recognition and generation using a neural network based on Bayesian learning. The temporal network model will further serve as a part of an ANS (Artificial Neural System). It provides a way to do simulation studies of problems that need to be treated in a temporal manner.

Considering certain aspects of information processing performed by biological neuronal networks such as recognition and motor control, it is quite reasonable to assume that the treatment of temporal patterns is a fundamental property of the nervous system. In the case of auditory recognition the cochlea performs something quite similar to an electro-mechanical discrete Fourier transform of the sound input. The mechanical pressure variations are converted to a time varying pattern of intensities for the decoded frequencies. Certain sounds are only discriminable due to their temporal properties. For instance, when a recorded piano is played backwards it will certainly not be recognized as a piano. The same effect is seen on higher level audio perception, such as listening to melodies or speech understanding. As another example we could consider motoric efference. When generating motor actions such as speech and locomotion, both a precise timing and precise patterns of muscle activation sequences are essential for an accurate result. Yet most of the research in the field of artificial neural networks has dealt with static pattern recognition and classification problems. Of course it is fundamental to the science of neural networks to have a good knowledge and understanding of how to design static classifiers and associative memories. On the other hand, for studies of higher level behavioral aspects of biological systems and for application of neural network solutions to real world problems, it is important to have network models that processes temporal patterns with reasonable performance.

2 Definition of the temporal problem

Is there such a thing as an ideal temporal associative memory? If so, then how could that be defined? A temporal memory could be imagined to remember spatio-temporal patterns. Experience from static associative networks tells us, however, that patterns stored in a network should not be dependent on intermediate values for single neurons. Such a network will be too sensitive to disturbances as noise and faulty neurons. If we want continuous values we use groups of neurons instead. A reasonable limit then would be to treat only sequences of patterns, whose prototypes are binary, and to use graded output from units as belief propagation. In any case, graded values and spatio-temporal patterns can be approximately achieved by different coding techniques such as interval coding and population coding.

One approach to define a sequential memory could be to extend a definition of a static memory. Consider the following definition of an ideal static autoassociative memory for binary patterns (Kohonen 1988):

(i) *An ideal autoassociative memory is a system which holds copies of distinct input signal sets* $x^{(p)}, p = 1, 2, \ldots, k$ *in its internal state, and produces the copy of a particular set* $x^{(r)} = (\xi_1^{(r)}, \xi_2^{(r)}, \ldots, \xi_n^{(r)})$, $r \in 1, 2, \ldots, k$ *to the outputs, whenever (in the recall mode) the inputs are excited by a set of signals* $x = (\xi_1, \xi_2, \ldots, \xi_n)$ *in which a specified subset of the values* ξ_i *matches with the corresponding subset of* $\xi_i^{(r)}$.

This could serve as the basis for a definition of an "ideal" sequential memory. The following definition has been the basis for the present work:

(ii) *An ideal sequential associative memory holds copies of sequences of instantaneous patterns, defined as in (i),*
$$x^{(p,s)}, p = 1, 2, ..., k; s = s_0, s_1, ..., s_n$$
in its internal state, where s is an implicit state number, and produces a copy
$$X^{(r,s)} = x^{(r0,s0)}, ..., x^{(ri,si)}, ..., x^{(rj,sj)}, ..., x^{(rm,sm)}$$
of a particular stored sequence of instantaneous patterns, whenever, in recall mode, the network is stimulated with a sequence of instantaneous patterns
$$Y^{(u,s)} = y^{(u_0,s_0)} ... y^{(u_i,s_i)} ... y_{(u_j,s_j)}$$
where, in a specified subset of the sequence $Y_{(u,s)}$ *each member* $y_{(ui,si)}$ *matches a specified subset of each member* $x_{(ri,si)}$ *according to (i), in the sequence* $X_{(r,s)}$. *An ideal temporal associative memory could then be defined based on (ii) where the relative time between different instantaneous patterns and the absolute recall speed is also considered.*

(iii) *An ideal temporal associative memory holds copies of sequences of instantaneous patterns, where each instantaneous pattern is a set of signals,*
$$x^{(p,t)}, p = 1, 2, ..., k; t = t_0, t_1, ..., t_n$$
in its internal state, where t is an implicit time stamp, and produces a copy
$$X^{(r,\tau)} = x^{(r_0,\tau_0)}, ..., x^{(r_i,\tau_i)}, ..., x^{(r_j,\tau_j)}, ..., x^{(r_m,\tau_m)}$$
of a particular stored temporal sequence of instantaneous patterns
$$X^{(r,t)} = x^{(r_0,t_0)}, ..., x^{(r_i,t_i)}, ..., x^{(r_j,t_j)}, ..., x^{(r_m,t_m)}$$
whenever, in recall mode, the network is stimulated with a temporal sequence of instantaneous patterns
$$Y^{(u,t)} = y^{(u_0,\tau_0)} ... y^{(u_i,\tau_i)} ... y^{(u_j,\tau_j)}$$

where, in a specified subset of the sequence $Y^{(u,t)}$ each member $y^{(ui,ti)}$ matches a specified subset of each member $x^{(r_i,t_i)}$, according to (i), in the sequence $X^{(r,t)}$ in monotonic order, and

$$t_0 - t_i = T \cdot (\tau_0 - \tau_i), t_0 - t_j = T \cdot (\tau_0 - \tau_j), ...; T \in R.$$

In most circumstances, however, we are not interested in reversed recalls. Thus we may limit the produced sequence of sets to those where T > 0, i.e. a sequence would only be recalled in the same order as it was stored. As a further restriction, in the current study we have focused on sequential memories according to definition (ii). Thus it is only the order of instantaneous patterns in a sequence that is essential. In the following examples some sequences; S1, S2 and S3; are given. Each instantaneous pattern in these is a character, or, as in examples e2 and e3 a pair of characters. Assuming that we have an ideal associative memory available, a sequential memory may then be implemented under one of two trivial constraints. Constraint 1, each instantaneous pattern is unique. Two sequences, S1 and S2, are stored in the associative memory, such as:

(e1) `S1 = 142857142857...`
`S2 = 093093...`

Here any of the sequences S1 or S2, both infinitely long, may be uniquely produced if a stimulus pattern y(u,t) matches any element of the pattern sets 1,2,4,5,7,8 or 0,3,9. Constraint 2, we have the possibility to store time or equivalent contextual information together with each instantaneous pattern x(p,t),(Rosenblatt 1962):

(es) `S1 = (a,0),(b,1),(a,2),(c,3),(a,4),(d,5)`
`S2 = (e,0),(c,1),(e,2),(b,3),(e,4),(b,5)`

Each instantaneous pattern, in this case, is made unique by giving it an explicit time tag, but any pattern may occur just once in a specific context or at a specific time. If the sequences S1 and S2 in (e2) above are learned and a third sequence, S3, such as:

(e3) `S3 = (c,0),(d,1),(a,2),(d,3),(c,4),(a,5)`

is added, the sequences S1 and S3 could not be unambiguously recalled because the instantaneous pattern (a,2) is no longer unique among the sequences S1, S2 and S3 because it occurs both in S1 and S3. Another drawback of making each instantaneous pattern unique is that it becomes hard for the network to make generalizations. Let, for example, a temporal network learn the following sequences S1 and S2:

(e4) `S1 = ababcde`
`S2 = bcabcab`

When storing such sequences as S1 and S2, in which an instantaneous pattern "a" is always followed by a "b", it is possible to generalize about this. A constraint that makes each instantaneous pattern to be unique, i.e. where a network may only use pattern information from the previous timestep, will be hard to fulfill. The basic problem in temporal networks is how to deal with the history. A prediction must normally use data older than t minus 1. The property of a set of sequences telling how old the data must be for one to deal with it, in order to specify each continuation uniquely, is here called context length.

Definition:

> *The context length of a sequence is the maximum age of the data required to specify each continuation uniquely.*

For example: The sequence "142857142857..." (fraction part of 1/7) has context length 1 but the word "mathematics" has context length 4. Now, consider the differences between the properties needed for sequential pattern classifiers and sequential pattern generators. For classification purposes, e.g. phoneme recognition, the problem consists mainly of detection of specific features in the input stream, features that are often invariant to some properties of the input signal. One property to pay attention to in recognition circumstances is whether automatic segmentation of the input stream has to be managed. Such as:

```
"CANYOUREADTHIS" vs "CAN YOU READ THIS"
"SEGMENTSMAYBEAPROBLEM" vs "SEGMENTS MAY BE A PROBLEM"
```

This is a well known problem in, for example, recognition of continuous speech. There are certain languages, like Finnish, where it may be less accentuated due to consequent stress of the first syllable of each word. The stress of a word is, however, not yet a property to be treated in an efficient manner in automatic speech recognition. (Elenius K., personal communication). For pattern generation purposes we may pay attention to how the sequence is activated and how tolerant the generated sequence has to be against errors in the triggering stimulus. For low level pattern generators in biological motor systems, it has been shown (Grillner et al 1987) that basic spinal pattern generators may be driven by a tonic stimulus. This has also been shown in simulated models of spinal pattern generators (Lansner et al 1989). Figure 1 shows a model of the Lamprey swim generator that has been simulated.

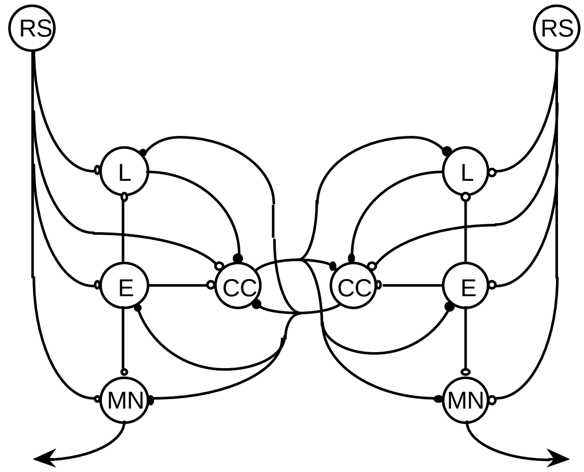


Figure 1: The spinal swimming rhythm generating network of a Lamprey. "E" are excitatory interneurons that drive the motorneurons. "CC" are interneurons that inhibit the opposite side. "L" are lateral interneurons that terminate activity on the active side. "MN" are motorneurons and "RS" are reticulospinal neurons. The reticulospinal neurons are driving the spinal network. Filled circles designate inhibitory synapses and unfilled circles excitatory synapses.

For temporal motor control mechanisms we do not yet know what kind of activating patterns are used. It is however reasonable to believe that the activating pattern sequences are quite short. In fact, the start condition could be just a single pattern, like a goal coordinate in a reaching movement. At a certain level in the system this may reflect how the movement is initialized. In the following we focus on the problem of generating long sequences from short activating sequences.

3 Different methods for temporal sequential networks

We introduce by giving a short review of some previously studied methods for sequential pattern recognition and completion. This is in no way a complete review. It is an illustration of some methods which differ from the approach in the present work. A state machine built round an associative memory is studied by Kohonen. Outputs from the associative memory are fed back through delay lines to the network. Delayed replicas of the outputs will be associated with incoming stimuli (Kohonen 1988). Methods and theories from adaptive signal processing that are used for prediction of stochastic processes are also applicable to neural network models. The weights may be calculated using the least-mean-square (LMS) algorithm or, e.g., Kalman-filter algorithms (Trvn 1988). In another model, that is called short term memory (STM), each unit remembers a small history of its input signal by letting the signal pass a convolute giving an exponential decay (Trvn 1988). Each signal passes a STM-loop. There are several STM-loops in parallel and the outputs from each STM-loop are weighted into a decision network that selects output patterns that are most close to the valid patterns. The Hidden Markov Model is a kind of state machine where the weights are interpreted as transition probabilities. A HMM may be trained using the forward-backward algorithm (Waibel et al 1987). A Jordan network is a way of implementing sequential association in back propagation networks by adding feedback and a set of recurrent state units. (Jordan 1986), (Massone, Bizzi 1989). Another way to use back-propagation for temporal processing is to use "Time Delayed Neural Networks (TDNN's)". TDNN is often used to designate multilayer backpropagation networks where each unit has multiple weights with different delays. These units may however be used with other learning rules (Waibel et al 1987), (Lang, Hinton 1988). The sequential or temporal networks mentioned above may be used either for pattern recognition or pattern generation. Most of them have in common that they may be seen as implementing a type of predictor

$$\hat{p}(t) = F(s(t), \hat{p}(t-1), \ldots, \hat{p}(t-k)) \tag{1}$$

where the estimated pattern at time t is solely a function of the stimulus at time t and k steps of predicted pattern history. This is, of course, not true for models where units with different input delays, as the TDNN units, are used to sample the stimuli. The other extreme in that case is

$$\hat{p}(t-c) = F(s(t), s(t-1), \ldots, s(t-k)) \tag{2}$$

where the estimated pattern at time t-c is solely a function of the input stimuli.The constant c is normally chosen between 0 and k. For classification purposes the estimated pattern may be a decision, like hyphenate vs not hyphenate. A variant of the latter is

$$\hat{p}(t, t-1, \ldots, t-k) = F(s(t), s(t-1), \ldots, s(t-k)) \tag{3}$$

where a sequence may be recognized as a whole. A network model for recognition of temporal patterns that corresponds rather well to this predictor principle has been studied (Tank, Hopfield 1986). Stimuli are projected on a network through continuous delay functions that also make a compression of information in time .

4 Methods and simulation results

In this section we present the models developed and some simulation results.

1 Networks with delayed inputs

The basic method for implementation of sequential memories studied in this work is the addition of delayed stimulus connections to an autoassociative network, i.e. temporal structure is transformed to a spatial one (figure 2). With this architecture we will get the predictor

$$\hat{p}(t, t-1, \ldots, t-k) = F(s(t), s(t-1), \ldots, s(t-k), \hat{p}(t), \hat{p}(t-1), \ldots, \hat{p}(t-k)) \quad (4)$$

where the sequence within the whole context length k is predicted from both k steps of predicted pattern history and from k+1 steps of stimulus. Assumptions: the rate of stimulus change is slow compared with the relaxation time of the network. The delay lines has equal delay characteristics.

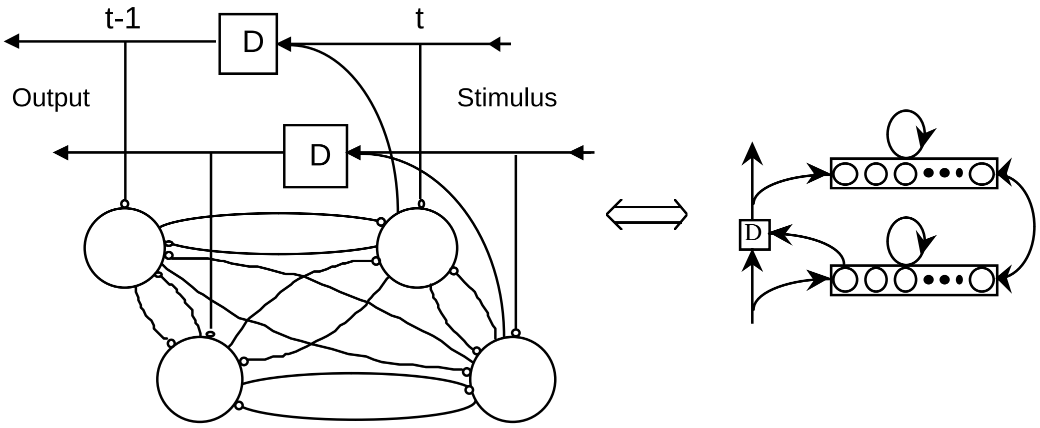


Figure 2: Picture illustrating the principle for temporal to spatial conversion. The left figure shows a fully connected associative network, here represented by 4 neuronal units. A part of the network will see a delayed replica of the input signal (t-1). Outputs from some of the units will be mixed with the input signal. The right figure shows the same in a more formalized way. A network population is represented here by a rectangular box. An oval with an arrow shows that the population is recurrent. An arc binding two network populations together means that all units in one population are projected on all units in the other population in the direction of the arrows.

The associative network model chosen is of Bayesian type (Lansner,Ekeberg 1989). There are several reasons for choosing a Bayesian network. The learning rule is fairly simple and biologically reasonable. The Bayesian criterion is also considered to be the best in comparison with other common classifiers as Perceptron (Linear), Least Mean Square and Sigmoid (Barnard, Casasent 1989). The learning problem in these types of Bayesian networks is mainly a question of collecting statistics. The weights are computed from mutually conditional probabilities (assuming independent patterns), such that:

$$W_{ij} = \ln \frac{P(j|i)}{P(j)} = \ln \frac{P(j\&i)}{P(i)P(j)} \tag{5}$$

The method used in this work for collecting relevant statistics is an incremental learning rule (Ekeberg ., personal communication). The probabilities are estimated without prior knowledge of the patterns by using exponential convolutes and thus will be good estimates for both stationary and non-stationary processes. In the notation used here Sj(n) is sample value for unit j when the n'th pattern is presented. Pij(n) is the compound probability that unit i and j are simultaneously active. t is a time constant that is chosen large enough to smooth out short term variations but short enough to follow non stationary processes.

$$\tilde{P}_j^{(n+1)} = \tilde{P}_j^{(n)} + \frac{S_j^{(n)} - \tilde{P}_j^{(n)}}{\tau} \qquad \tilde{P}_{ij}^{(n+1)} = \tilde{P}_{ij}^{(n)} + \frac{S_i^{(n)} \cdot S_j^{(n)} - \tilde{P}_{ij}^{(n)}}{\tau} \tag{6}$$

A problem with these fast and simple learning rules in one layer nets is that probabilities are assumed to be pairwise independent. This means that certain patterns are not distinguishable. It is possible to deal with dependent patterns by using multilevel nets. E.g. a one level Bayesian network could be enhanced by complex nodes (Lansner, Ekeberg 1987), i.e. "interneurons", working as feature detectors, that are using a different and more complicated learning rule. In this work there has been no attention paid to this possibility. Here, only one layer nets with incremental Bayesian learning are considered. We assume that the problem of independence has been solved before the input is given to the Temporal Associative Network (TAN). In a network where just one delay step is used,as shown in figure 2, it is possible to store sequences with a context length of one. To recall sequences in this network, output from the part of the network that is stimulated without delay is fed back and mixed with the delayed stimulus. To manage longer context lengths the most obvious thing would be to add more stimulus delay lines thus spreading the temporal information over a larger network, figure 3. The outputs from the subnets is coupled to the delay lines and from there propagated. This is refered to here as output feedback. The degree of output feedback is not critical but some tests showed rather good results when the outputs and the inputs influenced the propagated data with one half each. If the output feedback is too large the network will have hard to change a faulty decision. In the corresponding way the network will be sensitive to noise when the influence of the inputs is too high.

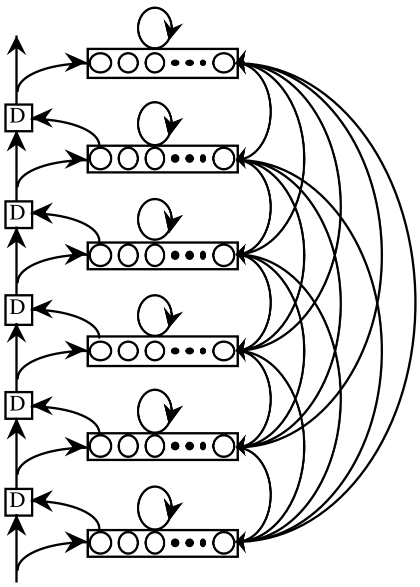


Figure 3: A network with 5 delay steps that could manage context lengths of 5. The stimuli is propagated in the direction of the vertical arrows. After the network has reached a state of relaxation it produces an output from each neuronal population (the rectangles). Outputs from the neuronal populations are fed back to the stimulus propagation line as the arrows going from top of a rectangle and leftwards show. These connections are here called output feedback .

The storage capacity (Zmax) in an associative network relates to the number of units (N), where "ln" is the natural logarithm, as (Lansner, Ekeberg 1985):

$$Z_{max} = \mathcal{O}\left((\frac{N}{\ln(N)})^2\right) \tag{7}$$

When storing sequences of patterns in a network, configured as in figure 2, that could manage a context length of one, it would be expected that the maximum number of sequences of length "l" possible to store would be

$$Seq_{max} = \mathcal{O}\left(\frac{(\frac{N}{\ln(N)})^2}{l-1}\right) \tag{8}$$

The assumption behind this is that each instantaneous pattern in the sequence is coded as a single unit. If this is the case, the first and last step of each sequence will not generate any weight change, i.e. each pattern in a sequence is associated with its follower, except, of course, for the last one. When we add more delay lines to manage contexts of length "c", as in figure 2, we would, with the same assumption as in the previous example, expect the maximum capacity to be decreased to

$$Seq_{max} = \mathcal{O}\left(\frac{(\frac{N}{\ln(N)})^2}{l+c-2}\right) \tag{9}$$

When we need to treat long context lengths, with this model, very large amounts of weights are required. The number of weights will increase with the square of the context length "c" ,where "n" is number of units in a single pattern:

$$W_{tot} = \left((c+1)\cdot n^2\right) - (c+1)\cdot n\Big) \tag{10}$$

5 Matrix reduction due to time invariant relations

When looking at the connection matrix (figure 4) one could expect some symmetries to be found due to dependencies between patterns at different timesteps. A matrix element like [t-3,t-2] that connects output from timestep (t-3) with inputs at (t-2) should be the same as the element [t-4,t-3] etc.

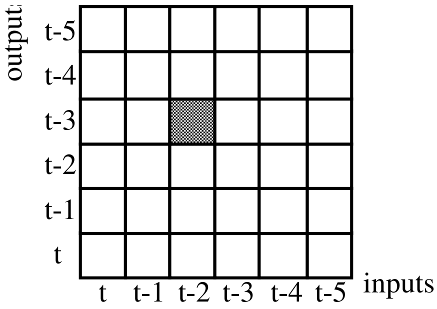


Figure 4: The connection matrix that connects different timesteps with each other. Observe that the elements of this matrix are also matrices which make the network at each timestep recurrent. The interpretation is that the outputs from timestep t-3 connects to inputs at timestep t-2 and so on.

If the symmetry principle is correct then we would get a matrix like the one in figure 5. The connection matrix will have a diagonal structure where each element, representing the set of weights projecting one population on another, is constant along a diagonal. Due to this symmetric structure the number of unique weights is reduced to increase linearly with the context length instead of growing with the square.

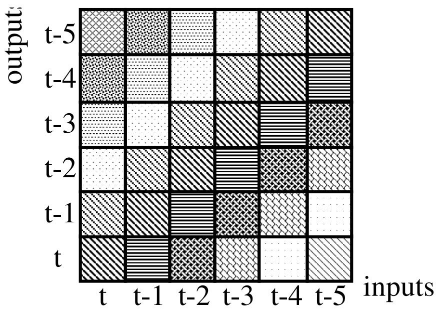


Figure 5: Due to invariance between dependencies at different timesteps the matrix will show a diagonal structure.

By utilizing the diagonal structure of the connection matrix it would be possible, in a simulated network model, to use a smart lookup of weight values. If this is possible to realize in a simulated model it is still, however, not very attractive because it is still computationally expensive and totally unplausible from a biological point of view. It would also be rather unpractical to implement this model in hardware. Perhaps it could be a creative approach to reason in the following way. The multiple sets of equal weights along the diagonals are redundant in the sense that, once a set of weights has been used in the relaxation it is possible to ignore that set at further timesteps. As a consequence, we could then try to simply remove the redundant part of the matrix as shown by figure 6. This operation would also make the number of weights linearly proportional to the context length, such that:

$$W'_{tot} = (n^2 - n) \cdot (2 \cdot c + 1) \tag{11}$$

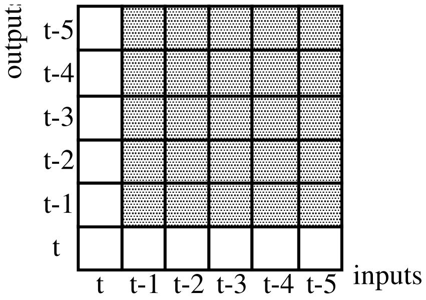


Figure 6: Assuming that the multiple occurrences of equal weight sets over the diagonals are redundant, we could simply remove them (grey). Thus the total matrix will be L-shaped and the amount of weights will grow linearly with the context length.

A network with connectivity reduced according to the hypothesis that multiple sets of time invariant weights are redundant is shown in figure 7.

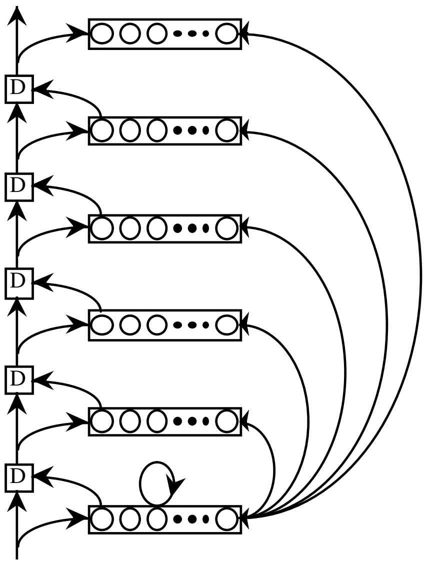


Figure 7: A TAN (Temporal Associative Network) with 5 delay steps and "redundant" connectivity removed.

To characterize the performance of the present model and to investigate the effect of removing the "redundant" part of the matrix some tests have been performed. figure 8 shows some results from tests that have been run using random sequences of characters as input. Each character has been coded as a unique active unit. By coding each character uniquely we are sure that the characters themselves are pairwise independent. Tests where distributed character coding is used have also been performed with reasonable results on the larger nets but they are not shown here. The same set of training and test data has been used for all the following tests.

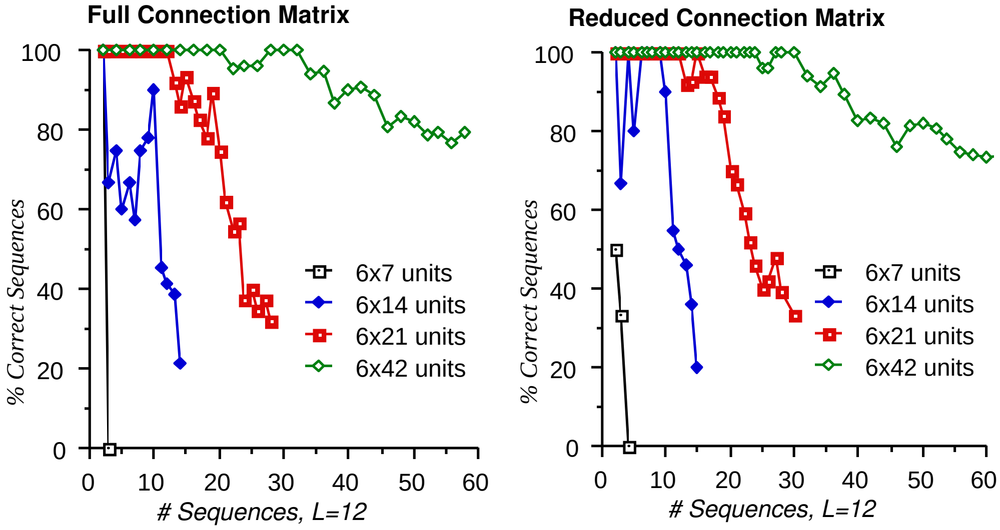


Figure 8: Comparison of behavior between a TAN with full connection matrix (left) and a TAN with reduced connection matrix (right). The same random character sequences has been used in both the tests. The vertical axis shows the percentage of sequences that has been 100 been stimulated with the first three steps of the sequences learned.

When completion tests are run on the network with full and reduced connection matrix we see, as figure 8 shows, that the nets behave almost equivalently. As a matter of fact, the nets with reduced matrices perform slightly better except for the small network with 42 units. The results above indicate that the principle for temporal to spatial conversion, that is used here, could be a useful method for implementation of sequential associative memories.

6 Left-Right context

One reason for using a network that is recurrent in the time domain, as we did in the previous section, is that this makes it possible to do pattern completions within the whole manageable context length for the net, i.e. we not only may predict the future from old data, but also tell what the past should have been, based on recent data. Earlier we defined the context length for a sequence as the maximum number of timesteps required to specify each continuation uniquely. The context in his definition could be designated "left-context". In the same way, "right-context" could be defined as the number of steps required to specify a unique history. It is, of course, totally irrelevant to speak about left and right in the time domain but, if we associate time with, for instance, sequential reading of text, which in most languages is performed from left to right, we get a useful interpretation. A more general expression would be preand postcontext. Seen from a statistical point of view pre- and post-context may be compared with pre- and post-dictors that are used to determine the correct value of a signal based upon passed history and future (Parsons 1987). Pre- and post-contexts have the same size for any sequence but this does not imply that this amount of timesteps is necessary for determination of the sequence uniquely. The context length defines the minimal number of delay steps we need to generate the sequence. There may be many parts of a sequence shorter than the context length that uniquely specifies it. Assume that a network similar to the one in figure 7, but with 10 delay steps, has learnt the following sequences whose context length is 4 :

```
S1 = MATHEMATICS
S2 = MATERIALLY
S3 = MATRICULATE
```

If the network is stimulated with e.g. "MAT" it will not be able to unambiguously choose a continuation. The network will choose one of the sequences anyway. Consider the following stimulus-recall process ("." means empty input) :

```
Stim = MAT.              Recall = MATH
Stim = MAT..             Recall = MATHE
Stim = MAT...            Recall = MATHEM
Stim = MAT...L           Recall = MATERIAL
Stim = MAT...L.          Recall = MATERIALL
Stim = MAT...L.T         Recall = MATRICULAT
Stim = MAT...L.T.        Recall = MATRICULATE
```

By stimulating the same network with `".......ICS"` it will recall `"MATHEMATICS"`, i.e., by utilizing the right-context (or post-context), a temporal network will be able to recall a whole history or "cause" when it is stimulated with a part of a sequence or change its decision when additional input is available. For a sequence generating network this is a valuable property to prevent noise in input to give errors in output. Experiments like these has been performed on human beings for the English language (Shannon 1951). It was found that the predictor and the postdictor have the same characteristics but the predictor is slightly better. It means that it is somewhat easier to guess a word when given the beginning of it than given the end. Actually, a network with temporal recurrency, like the one here described, could equally well be used to recall a sequence backwards if the delay lines were reversible. Biological evidence for reversible temporal networks are currently unknown, but for certain problem domains this could be a useful property. Take for instance a labyrinth learning network that has learnt a sequence of turns to find its way across the labyrinth. By reversing the direction of recall and reversing the direction of turn the network would find the reverse way across.

7 Feedforward classification of old data

To predict time sequences is somewhat like walking in a rather well-known landscape. Often we may know where we are and where to go by just looking in the closest environment. But, sometimes the closest environment is not enough and we have to look around for distant landmarks. When we look in the close environment we sometimes take a faulty decision, but soon we may recognize a distant environmental feature that makes us change our mind and choose a new direction. This is easy, but if we have misinterpreted a distant landmark we may go several kilometers in the wrong direction. In a sequential associative network it could, referring to the analogy above, be reasonable to assume that we may use the feedforward principle for old data, i.e. "distant landmarks" and the recurrent principle for recent data, i.e. "close environment". The network will then be robust in the recurrent part where data are highly correlated and an error in stimuli may cause great errors in the decided output. When the stimuli outputs mix is propagated to the feedforward part of the network the output decided is no longer possible to change. To investigate if the feedforward principle is feasible with the Bayesian learning rule for predictions of sequences some tests has been done on feedforward connected networks (figure 9, Left).

As can be seen from tests on a feedforward network (figure 9, Right) the capacity is about half of the capacity for a recurrent network (figure 8), with the same set of data. This is expected since there is half the number of connections and the stimulus is not noisy. A network of this type has to make correct decisions at each timestep, otherwise the faulty patterns will be propagated and be the basis for new decisions. An attempt to decrease the number of weights without loosing too much capacity or the left-right context principle is to use recurrency for recent data and feedforward connectivity for old data as figure 10 shows. The capacity is slightly less than for the fully recurrent net. The reason for the capacity to be that high here may be explained when one considers how autocorrelation functions usually behave as a function of time difference. Since most sequences probably have a rather short context length, the feedforward weights are only needed to resolve just a few ambiguities.

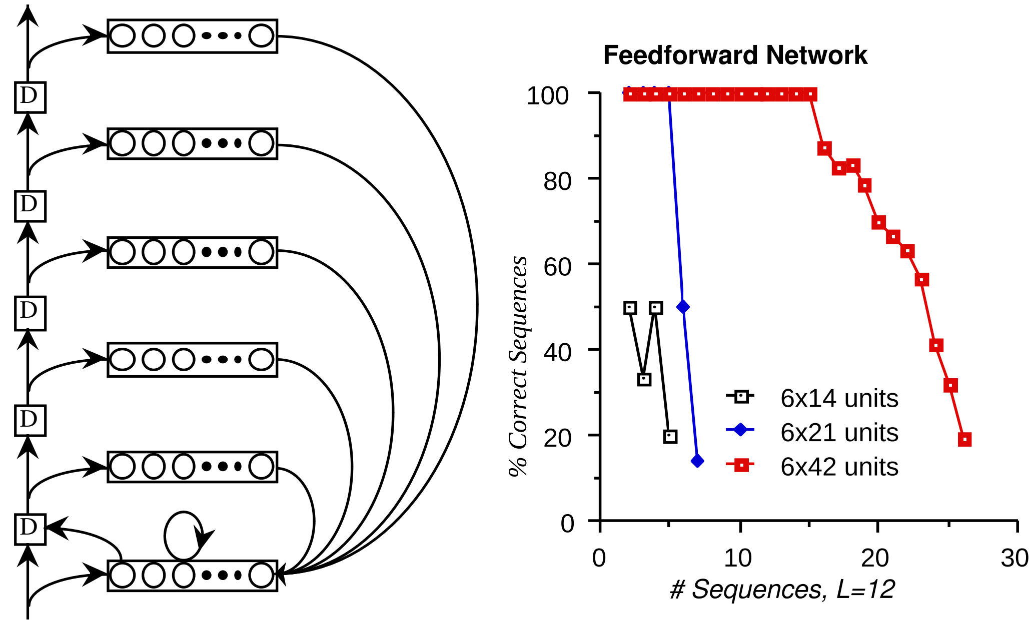


Figure 9: (Left). A temporal network with feedforward temporal connections. (Right). Results from completion tests on a feedforward temporal network.

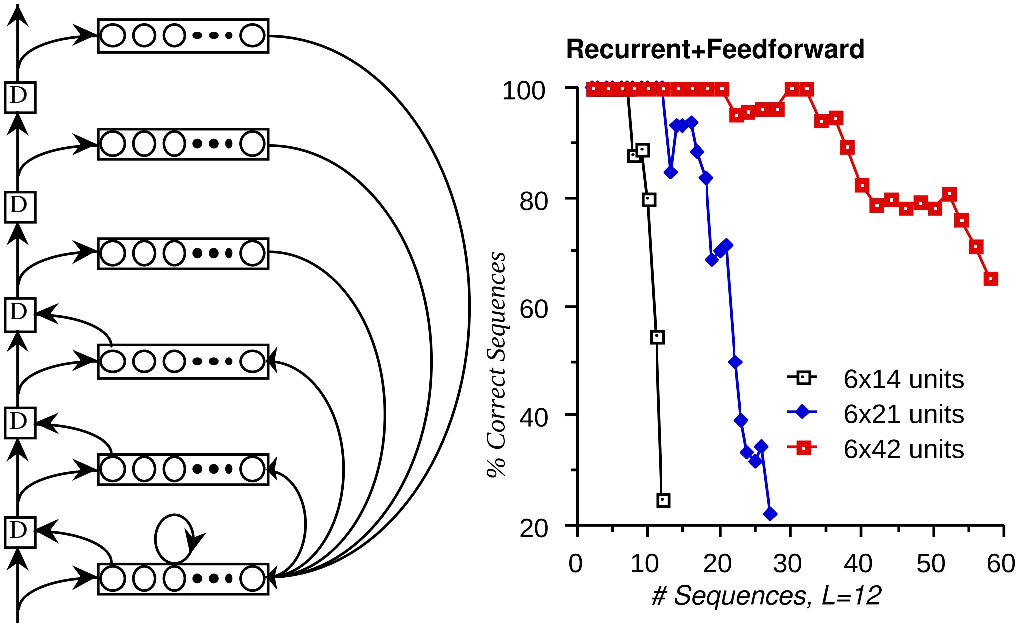


Figure 10: (Left):A temporal network where recent timesteps are recurrent and older timesteps are feedforward only. (Right): A few results obtained with the combined network. These results are only slightly worse than for the fully recurrent network.

8 Coarse summation of old data

If we consider the consequences of being able to manage very long sequences in temporal neural networks we will probably come to the conclusion that it is rather unrealistic to have a temporal resolution that is linear with time. We would get very large connection matrices even when using the symmetry and feedforward principles investigated above. Many of the weights would be unused due to small correlations between data with large time differences. As an example we can look at the following expression "the correlation is low". If we consider the temporal connections between "the" and "low" it should be quite clear that the exact time when "the" occurred compared with "low" is less important compared with "t" and "e" in "the". When looking at perception in biological systems we know that the threshold for a difference in sensation relates to the change in stimulus as (Weber's law) (Kandel 1985): k=DS/S, i.e. the threshold for experience of a change in stimulus is proportional to the size of the stimulus. In this way our sensations are proportional to the logarithm of the magnitude of the absolute stimulus. A hypothesis about temporal resolution could then be stated in the same way

$$\Delta Resolution = k \cdot \frac{\Delta T}{T} \tag{12}$$

which would give a resolution that is a logarithmic function of time distance. One way of implementing a logarithmic-like resolution in a network like the one we havedescribed here, would be to sum the history over a number of time steps that grows exponentially with the "subjective" time distance. Figure 11 shows some examples of this summation of history that we will now refer to as coarse summation.

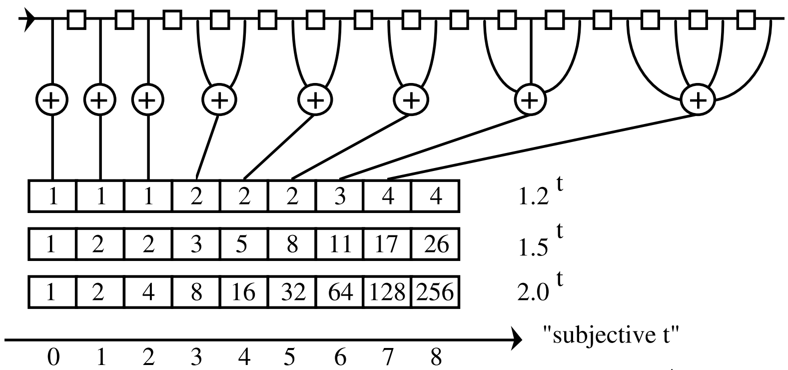


Figure 11: The principle for coarse summation of propagated stimuli/outputs to obtain a logarithmic-like time resolution. The number of sum steps is an exponential function of the "subjective" time, here with bases 1.2, 1.5 and 2.0.

In an implementation of a temporal network that uses coarse summation it is necessary to choose the base for the exponential function in accordance with the statistical properties of the data. To see if the hypothesis about logarithmic resolution could be reasonable anyway, some tests were run on a network configured as in figure 12, with 2 as a base for the exponential summation.

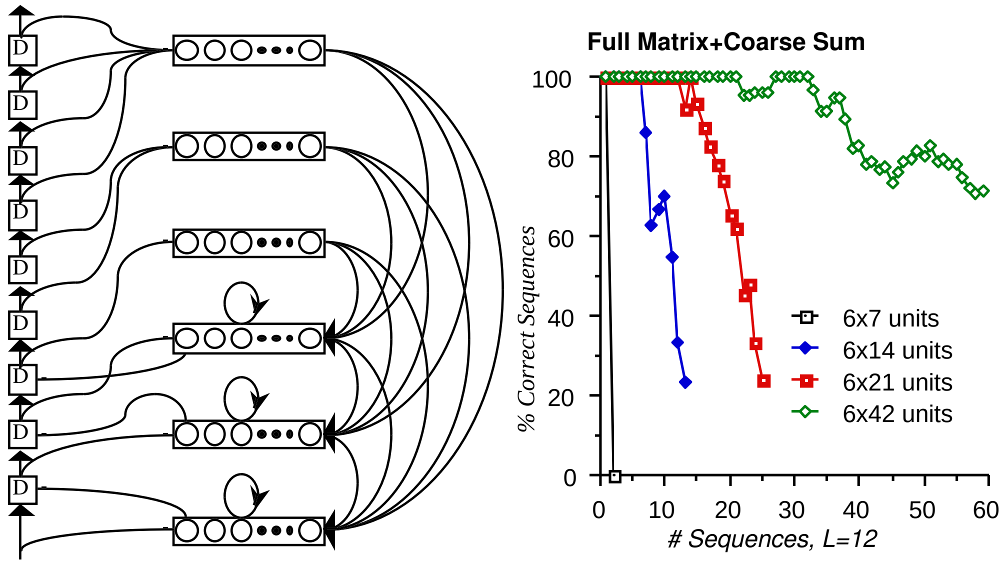


Figure 12: (Left) A temporal network that achieves a logarithmic-like resolution versus age of data. The varying resolution is achieved by coarse summation of old data. The coarse summed data is feed forward weighted into a fully recurrent network that spans over a few timesteps. (Right) Results from some tests with random sequences on some nets with coarse summation.

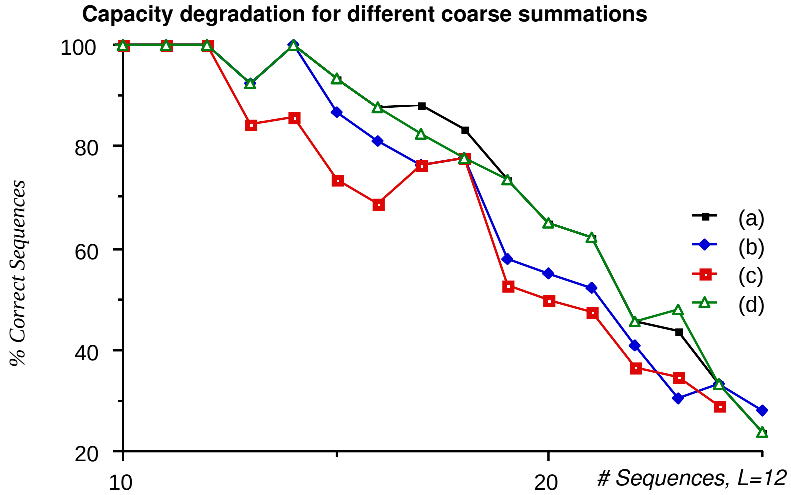


Figure 13: This diagram shows how the capacity degrades when different type of coarse summation is used. The capacity for perfect recall is, in this test, the same for all, but they show different degradation when this maximum capacity is exceeded. (a): Unit on if any input is on, i.e. like a logical OR-function. (b): Average, i.e. the sum has been divided with the number of coarse summed units. (c): A unit is on if most of the units are on, i.e. an average with a threshold. (d): The summation unit's output value is just the sum of its inputs.

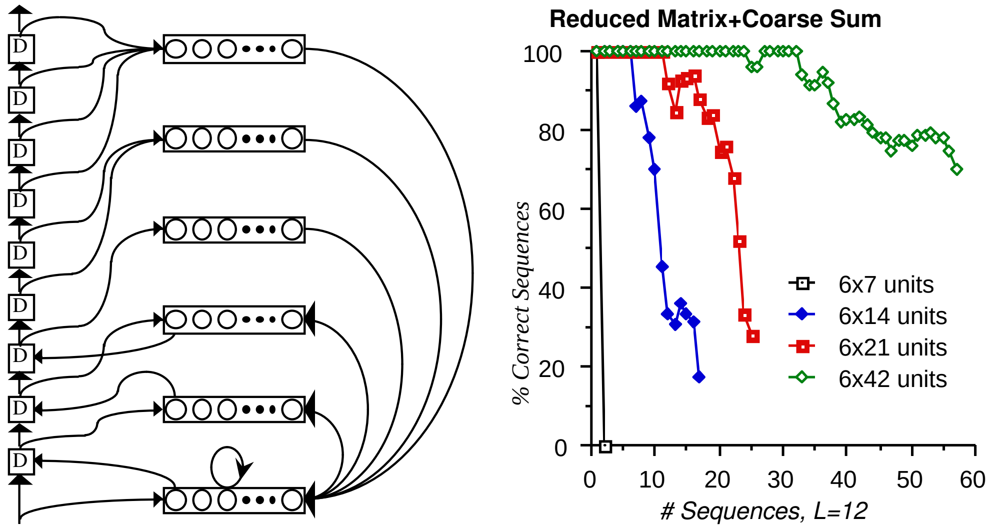


Figure 14: (Left) A temporal network with logarithmic-like time resolution where the coarse summed stimuli is only feedforward weighted into the recurrent part of a network with reduced connectivity. (Right) Results from tests with coarse summation and reduced connectivity.

A question that arises is how the stimuli has to be summed to give the best performance. If we assume that the coarse summed network works according to the same principles as the feedforward network described above, then the best summation principle would probably be one were the new weight values match the values in the feedforward network as much as possible. To achieve this, the output from a coarse summation unit could be just a simple sum of its inputs. A strange thing with this method is that the outputs of the summation units would exceed one. The output from a unit in the Bayesian network model that is used here reflects the probability for this unit to be active. We may however, see these summation units as help units that makes us collects statistics for several units at the same time. Results from a test run with some different types of coarse summation is shown in figure 13. This test indicates that the best type of summation to use is either a simple OR-function or a simple sum.

Completion tests on networks with coarse summation of old data showed quite good results when the sums where coupled to a fully connected network, figure 12 (Right). Corresponding tests on network where the recurrent part had a reduced matrix show, as can be seen in figure 14, even a slightly better result.

9 Sequence capacity for different architectures

The goal with this work was to find a useful model for temporal association in Bayesian networks to be used in simulation studies of temporal phenomena. Thus it is important to consider the model characteristics such as, capacity versus the number of units and number of weights.

To make possible such a comparison all tests have been run with the same random training and test sequences of characters with unique unit coding of characters. Each sequence in the test had the length 12. The number of neuronal populations were 6, i.e. the manageable context length would be 5, except for the coarse summed network that could possibly manage a context length of 9.

As the basic principle for temporal association investigated here is to use delayed inputs for conversion of temporal information to spatial, there may be different performances expected depending on how the spatial matrix is connected due to, for instance, time invariance effects.

The number of units in these models have been kept the same. A "unit" here designates everything which has its outputs connected to other units via weights. Some of these units are, for the feed forward nets, just used as input units, or for the coarse summed networks as stimulus summation units.To vary the number of units in a network model, the width of each pattern and the repertoire of the random sequences were varied.

Figure 15 shows the capacity plotted as number of sequences versus number of units for the different models investigated in this work. It should be observed that the theoretical capacity is based on experiences from static content addressable memories (CAM). It may be a surprise that the theoretical capacity is lower than most of the capacities measured. There is, however, nothing strange in this. What really matters is that we have the same proportionality. The theoretical capacity is based on empirical results for independent patterns with a sparse activity rate of about 1temporal examples we have both dependencies between the patterns and a varying rate of activity.

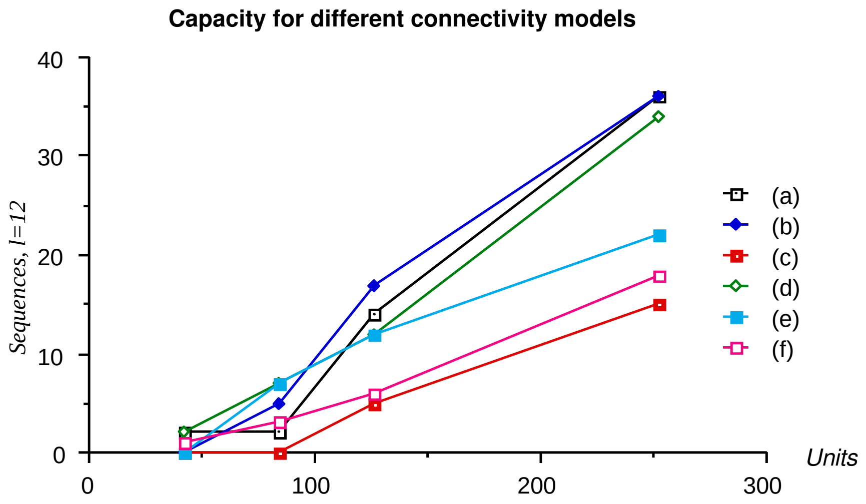


Figure 15: Capacity versus number of units for some temporal network models with different connectivity principles. (a): Fully connected recurrent network. (b): Recurrent network with reduced connectivity due to symmetry. (c): Network with feedforward temporal information only. (d): Mixed recurrent , fully connected with coarse feedforward summation (e): Mixed recurrent, reduced connectivity, with coarse feedforward summation (f): Theoretical capacity. based on static CAM results.

10 Sequence capacity versus number of weights

In the previous plot in figure 15 we looked at capacity versus number of units in each of the tested network models, but, what really makes the cost, at least, in artificial neural networks, is the number of weights. To check which model uses its weights in the most efficient manner the same data as above is used, but instead the number of recalled information bits versus the number of weights for the different models is investigated. The information content of a recalled sequence is designated as the ratio between the number of possible full length sequences and the number of possible sequences that are used as input for recall. The number of information bits in a recalled sequence is just the two-logarithm of this ratio. We get the total number of information bits (I) as the difference between the two-logarithms for the number of possible sequences and the number of possible input sequences (R) times the number of stored sequences (S).

$$I = S \cdot (\log_2(S_{possible}) - \log_2(R_{possible})) \quad (13)$$

This is based upon corresponding calculations for static CAM capacity (Lansner, Ekeberg 1985). Figure 16 shows how the recall capacity per weight differs for the different architectures. As these test indicate the weights are much better utilized when connectivity is reduced than with fully connected networks. The criterion for correct completion in this figure is that at least 95 tests above show a dip in the capacity to e.g. 90 100 figure 16 were plotted, the first dip in capacity was construed as the capacity limit.

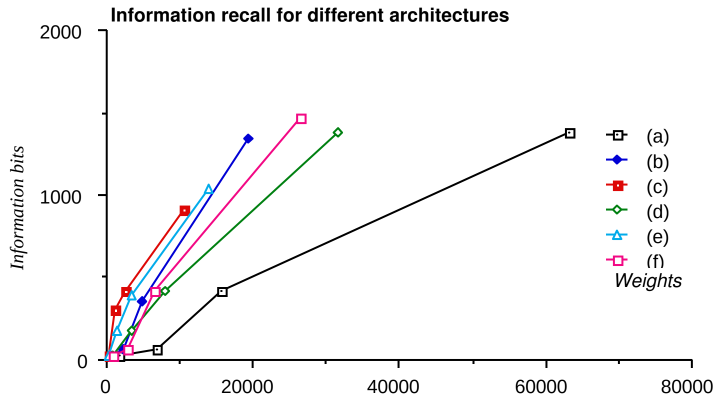


Figure 16: Capacity as recalled information per weights for some temporal network models with different connectivity principles. (a): Fully connected recurrent network. (b): Recurrent network with reduced connectivity due to symmetry. (c): Network with feedforward temporal information only. (d): Mixed recurrent, fully connected with coarse feedforward summation (e): Mixed recurrent, reduced connectivity, with coarse feedforward summation. (f): Feedforward only, as in (c), but with 10 delay steps.

11 Discussion

We may ask: Is there any special reason to choose a temporal model that spreads the temporal information across a spatial network? Yes, this method not only gives the system the possibility to draw conclusions out of the input data from the history but also to "change its mind" when new inputs show that an earlier decision was wrong. Thus, the network will be robust against noise in the input and generate correct sequences also when faulty decisions sometimes are taken. Using the feedforward principle for old data means that the network will be unable to do this.

There are some aspects of temporal association that have not been dealt with in the present work: speed of recall, recall in reverse order, time independent sequential, logical reasoning and semi-sequential processing that may run independently in parallel except for certain rendezvous. It has also been assumed that the sequences do not overlap to much for a one-layer network.

Speed of recall and recall in reverse order are both represented by the constant T given in the paragraph "Definition of the temporal problem" above. When T is less than one, the recall speed is greater than the learning speed and when T is greater than one, the recall speed is lower. Further, when T is negative the order of recall is reversed. Considering the varying speed of recall, it seems to have relevance in biological systems. When we have learned for example a sequence of movements, the speed may be varied within certain limits. A reversed order of recall may not be equally relevant in biological systems. It is common that people have problems when trying to do certain tasks in reverse order. Try for instance to rattle off the alphabet backwards!

When looking at motor systems it is rather unrealistic to imagine that the performance of an action could be reversed by just reversing the order of muscle stimulation. To reverse a movement a completely different strategy has to be taken with different muscles involved etc. This does not imply that this should be the fact in all levels of the system. It is possible that an action stored at a high level in the system, where just model coordinates about the outer world is treated, may be recalled in a reversed order.

The actual movements are then planned from these coordinates and different learned strategies would then be used in different directions. When one consider coarse summation as a method to manage long context lengths there are a few questions that may arise. (a) Is this a relevant method? (b) Is this in some sense biologically relevant?

(a) Further tests of this method on sequences with long contexts has to be done in combination with a more thorough statistical analysis. If looking at implementation of such a method it may also seem impractical to propagate and sum stimuli/outputs in the way we do here. Some function that approximates this behavior is probably to be preferred.

(b) There is currently no indication that coarse summation, as it is described here, may take place in biological systems. If such a mechanism exists it is more probable that it is based upon, for instance, concentration changes etc. in biochemical reaction systems.

All the tests in this work have been run on random sequences of characters. Further investigations have to be done on structured sequences of data, i.e. sequences where it is possible to define a grammar; as well as on sequences with distributed activity in the instantaneous patterns. Investigations have to be done with noisy start sequences and the capacity convergence for large networks has to be checked as well.

12 Conclusion

The studies done in this work indicate that efficient sequential associative memories may be built using one layer Bayesian networks, where temporal information is transformed to spatial information using stimulus delay lines. The reason for this principle to be affordable is that, due to time invariant relations between patterns, the weight matrix has a structure with multiple symmetries where "redundant" connections may be removed. The number of connections will then grow linearly with the maximum context length managed by the network. By using recurrency in the time domain, a temporal network will be able to tell the next continuation of a sequence and at the same time change its previously taken decision when new contradicting stimuli arrives. The results indicates that the best information capacity per weight is achieved when reduced connectivity is used. It can be used either for a network that is fully recurrent in the whole context length or in combination with feedforward and coarse summation to manage longer context lengths.

As a result of this work a Temporal Associative Network (TAN) software package is available. The package is written in ANSI-C and is instantiable with the following parameters: numbers of units in a pattern, number of neuronal populations, number of fully connected steps, number of steps with reduced connectivity, number of steps with feedforward connections, number of feedback steps, number of normal input steps and number of coarse summed stimuli steps.

13 Acknowledgements

Many thanks to Mikael Djurfeldt and Tom Wadden for critical reading and valuable comments on the manuscript. Funding provided by the National Swedish Board for Technical Development (STU), under grant 89-01826P is hereby gratefully acknowledged. Support from Digital Equipment Corporation's European External Research Program (EERP) is also gratefully acknowledged.

References

[Barnard and Casasent, 1989] Barnard E. and Casasent D. (1989). A comparison between criterion functions for linear classifiers, with an application to neural nets. *IEEE Trans on Systems, Man and Cybernetics* **19**: 1030–1041.

[Grillner *et al.*, 1987] Grillner S., Wallén P., Dale N., Brodin L., Buchanan J., and Hill R. (1987). Transmitters, membrane properties and network circuity in the control of locomotion in lamprey. *Trends in Neuroscience* **10**: 34–41.

[Hopfield and Tank, 1987] Hopfield J. and Tank D. W. (1987). Neural computation by concentrating information in time. *Proc. Natl. Acad. Sci. USA,* **84**: 1869–1900.

[Jordan, 1986] Jordan M. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pp. 531–546. Lawrence Erlbaum, Hillsdale, Amherst 1986.

[Kandel and Schwartz, 1985] Kandel E. R. and Schwartz J. H. (1985). *Principles of Neural Science*. Elsevier Science Publishing Co, Inc, New York. 2 edition.

[Kohonen, 1988] Kohonen T. (1988). *Self-Organization and Associative Memory*. Springer-Verlag, Berlin. 2 edition.

[Lang J.K., 1988] Lang J.K. H. G. (1988). A time-delay neural network architecture for speech recognition. Tech. Rep. CMU-CS-88-152, Dept. of Computing Science, Carnegie-Mellon University, PA.

[Lansner and Ekeberg, 1985] Lansner A. and Ekeberg Ö. (1985). Reliability and speed of recall in an associative network. *IEEE Trans on Pattern Analysis and Machine Intelligence* **7**: 490–498.

[Lansner and Ekeberg, 1987] Lansner A. and Ekeberg Ö. (1987). An associative network solving the "4-bit adder problem". In *Proceedings of the IEEE First Annual International Conference on Neural Networks*, pp. II–549. San Diego, USA.

[Lansner and Ekeberg, 1989] Lansner A. and Ekeberg Ö. (1989). A one-layer feedback, artificial neural network with a Bayesian learning rule. *International Journal of Neural Systems* **1**: 77–87, Also extended abstract in Proceedings from the Nordic symposium on Neural Computing, April 17–18, Hanasaari Culture Center, Espoo, Finland.

[Lansner *et al.*, 1989] Lansner A., Ekeberg O., Tråvén H., Brodin L., Wallén P., Stensmo M., and Grillner S. (1989). Simulation of the experimentally established segmental, supraspinal and sensory circuitry underlying locomotion in lamprey. *Soc. Neurosciene Abstr* **15**: 1049.

[Massone L, 1989] Massone L B. E. (1989). A neural network model for limb trajectory formation. may be submitted, currently don't know.

[Parsons, 1987] Parsons T. W. (1987). *Voice and Speech Processing*. McGraw-Hill Book Company, New York.

[Rosenblatt, 1962] Rosenblatt F. (1962). *Principles of Neurodynamics*. Spartan, New York.

[Shannon, 1951] Shannon C. (1951). Prediction and entropy of printed english. *Bell System Technical Journal* **30**: 50–64.

[Tråvén, 1988] Tråvén H. (1988). Temporal associative memory. Tech. Rep. TRITA-NA-P8802, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden.

[Waibel *et al.*, 1989] Waibel A., Hanazawa T., Hinton G., Shikano K., and Lang K. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing* **37**: 328–339.

Supplement V

Temporal Segmentation

A Bayesian Network for Temporal Segmentation

Roland Orre and Anders Lansner

SANS, Dept. of Numerical Analysis and Computing Science
Royal Institute of Technology, S–100 44 Stockholm, Sweden
E–mail: orre@sans.kth.se, FAX: +46–8–790 09 30

Abstract

A recurrent network which segments an unlabeled externally timed sequence of data is presented. The proposed method uses a Bayesian learning scheme earlier investigated, where the relaxation scheme is modified with a few extra parameters, a pairwise correlation threshold and a pairwise conditional probability threshold. The method studied is able to find start and end positions of words which are in an unlabeled continuous stream of characters. The robustness against noise during both learning and recall is studied.

1 INTRODUCTION

The segmentation problem is fundamental in pattern recognition. Given data with a sequential/temporal behaviour this shows up as the *temporal chunking problem* [1] which may be illustrated by the example:

$$thisisacontinuousstreamofdatathatispossibletoreadwithoutseparators$$

Here, we want unfamiliar lists of familiar items (characters) presented sequentially to be recognized as new items (words). In the first place just the characters are familiar. When we have seen different lists several times we will also recognize the words as familiar items. The method presented here detects segmentation points between words. Conceptually this means that we have grouped a sequence of elementary items into a new, composite item.

2 METHODS

1 Learning and Recall

The learning rule used for the ANN was earlier investigated [2]. The network can be characterized as a recurrent Hopfield type with graded output units. Weights, biases (1) and the transfer function (4) for the units are derived from Bayes' rule.

$$\beta_q = \log p(i) \qquad (1)$$

$$w_{iq} = \log \frac{p(i|q)}{p(i)} = \log \frac{p(q\&i)}{p(i)p(q)} = \log \frac{p_{qi}}{p_i p_q} \qquad (2)$$

$$s_q = \beta_q + \sum_{h \in A} w_{hq}\pi_h \qquad (3)$$

The neurons sum their inputs as in (3) where s_q is the support of the receiving unit q, w_{hq} is the weight from unit h, (in the set of active units A) to unit q, β_q is the bias and π_h is the output from h. The relaxation process is performed as a statistical inference where explicit time is simulated by letting a *dynamic* support E_q being charged as a "leaky integrator" of the support s_q. An output value π_q is a hypothesis of $p(q|A)$ and is obtained by exponentiating the dynamic support E_q as is shown in (4). Here we have calculated the weight values as in (5).

$$\pi_q = \begin{cases} 1 & E_q \geq 0 \\ e^{E_q} & \beta_q \leq E_q < 0 \\ 0 & E_q < \beta_q \end{cases} \quad (4) \qquad w_{iq} = \begin{cases} 0 & p_i < p_\nu \vee p_q < p_\nu \\ -log\ C_\theta & p_{iq} < p_\theta \\ -log\ C_\sigma & p(q|i) < p_\sigma \\ log \frac{p_{iq}}{p_i p_q} & otherwise \end{cases} \quad (5)$$

The threshold p_ν in (5) acts as a noise limiter and has been set to a low constant value. The p_θ is a limit for correlations between units. A high p_θ causes uncommon patterns to be forgotten while p_σ is more independent of how frequent a certain pattern segment is. C_θ and C_σ are some large numbers that are characteristic for the network size and the number of patterns stored. They are not critical but should typically be set so $\forall i \forall q : (\frac{1}{C} < \frac{p_{iq}}{p_i p_q})$ to make the inhibition a monotonic function of the correlations.

2 Temporal Coding

The architecture used to map sequential information onto the recurrent network is illustrated in figure 1. The information is learned at every position. This is like a kind of translation invariant storage of the sequence.

3 Usage

One possibly way to use the detection of segmentation points (starts and ends) is illustrated by figure 2. The output from the segmentation algorithm can be used as a "print now" signal to a network that stores or recognizes each word at a fixed position.

4 Segmentation Principle

Assume that each temporal position, as in figure 1 is represented by a neural population, where each letter is coded with one unit. Further assume that the segmentation network has seen a noise free sequence like "`NEURALJUNIORNEURALEXPERTNEURAL`". The sequence consists of n words, in this case of equal length l, where the most frequent word occurs at least c times. These words are seen by the network in all positions. The following calculations depend on how the first and last word are treated. We assume here that a frequent word starts and stops a sequence. The probability for a unit to be active is, $p_\nu^u \geq (l(n-1)+1)^{-1}$ *i.e.* , a character has occurred at least once in a unique word and $p_\nu^c \geq c(l(n-1)+1)^{-1}$ or c times in a frequent word. The pairwise unit correlation p_{iq}, is $(l(n-1)+1)^{-1}$ within a unique word and $c(l(n-1)+1)^{-1}$ within a frequent word. To detect words that occur more frequently than their combinations we could use $p_\theta^c \geq c(l(n-1)+1)^{-1}$. If we also want to detect infrequent words combined with frequent words we could threshold the normalized correlation $p(i|q)$, as is expressed by p_σ. $p(i^c|q^c) \simeq \frac{p_\theta^c}{p_\nu^c} \simeq 1$ and $p(i^u|q^u) \simeq \frac{p_\theta^u}{p_\nu^u} \simeq 1$ while $p(i^u|q^c) \simeq \frac{p_\theta^u}{p_\nu^c} \simeq \frac{1}{c}$. By setting the $p_\sigma \geq \frac{1}{c}$ in (5) we could make weights between units in frequent, and in infrequent, patterns inhibitory. When all words are about equally frequent we could set p_σ just above the greatest pairwise correlation between words. For n equally frequent words this implies that $\lim_{\frac{p_{qi}}{p_i p_q} \to 1} p_\sigma = \frac{1}{n^2}$.

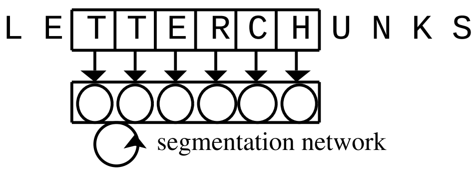


Figure 1: A continuous stream of data passes a tapped delay line that spreads the temporal information spatially over the network.

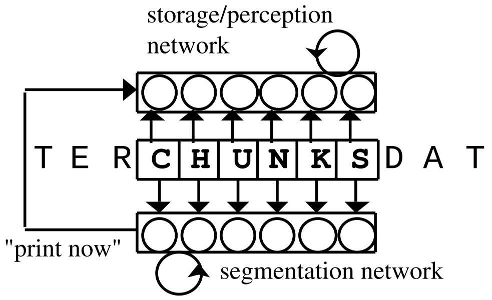


Figure 2: The segmentation network may signal "print now" to the actual storage or perception network when a segment start is found.

5 Segmentation Algorithm

A segmentation hypothesis S_τ (6), where τ is an index of the neural population that codes a character at a specific time-step, is propagated in parallel with the input sequence. S_T (8) is a sum of all S_τ. A decision about segmentation is done based on the current value of S_T and a decision threshold ξ (9). S_τ is based upon population activities A_τ close around each hypothesis point τ (6). A_τ is a normalized sum of the activity in population τ (7) where $n(\tau)$ is the number of units in this neural population.

$$S_\tau = \begin{cases} 0 & \tau = 0 \\ A_\tau & A_{\tau+1} < A_{\tau-1} - \xi \\ -A_\tau & A_{\tau+1} > A_{\tau-1} + \xi \\ 0 & otherwise \end{cases} \qquad (6)$$

$$A_\tau = \frac{1}{n(\tau)} \sum_{i=1}^{n(\tau)} \pi_{i\tau} \qquad (7)$$

$$S_T = \sum_{\tau=T-1}^{1} S_\tau \qquad (8)$$

$$SEG = \begin{cases} BEGIN & S_T \geq \xi \\ END & S_T \leq -\xi \\ NO & otherwise \end{cases} \qquad (9)$$

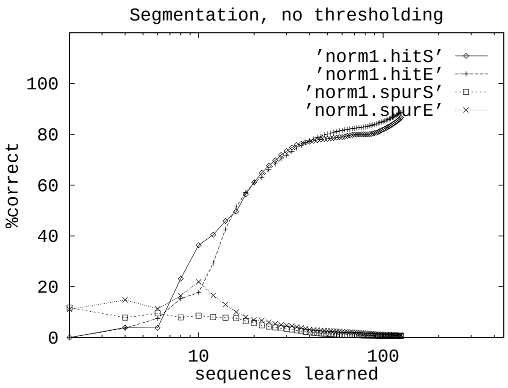


Figure 3: Random combinations of 17 words are learned and segmented. Upper curves show correct hits and lower show spurious hits.

3 EXAMPLE

Learn the sequences "NEURALJUNIOR", "NEURALEXPERT", "BAKERJUNIOR" and "BAKEREXPERT" on a network with six character populations as in figure 1. Each of the words "NEURAL", "JUNIOR", "EXPERT" and "BAKER" is twice as frequent as any of the combinations. Each sequence is learned at every position on the network. Biases and weights are calculated according to (1) and (5) and a relaxation is done for each position when the sequence is presented. In the following example a "_" means low output activity and a "+" stands for an ambiguous output hypothesis. To begin with we let the p_θ have a low value. We start by stimulating the network as in #1 below. For each time-step the network sees a 6 character long part of the sequence. After each stimulation in #1 we do a relaxation of the network and get outputs like in #2. We now do the same but before adjust p_θ to a higher value, like $p_\theta = 0.05$. We will then, after each relaxation, get the outputs in #3.

ex	t0	t1	t2	t3
#1	BAKERE	AKEREX	KEREXP	EREXPE
#2	BAKERJ	AKEREX	KEREXP	EREXPE
#3	BAKER_	AKER__	___EXP	__EXPE

The segmentation algorithm looks at the activity difference between adjacent populations. In example #3 it receives a strong indication for "R" in "ER_" to be an ending and "E" in "_EX" to be a start.

4 RESULTS

These experiments have been done by forming sequences of words picked from an English dictionary. Every next word in a sequence has been randomly selected with a certain probability. In figure 3 it is shown how the algorithm performs on a network tuned for pattern completion. The network learns an increasing amount of random sequences for a set of 17 words.

Segmentation performance, versus p_θ-level for noise free data, when just a few sequences are learned, is illustrated in figure 4. In figure 5 we can see how the p_ν-level affects the performance. When varying the p_ν-level we could not reach the same performance as when varying the p_θ-level. When the p_θ was set to the optimal level the performance could not be improved, only decreased, by adjusting p_ν. This was also the case when some words were 15 times more frequent than the least common words. This would indicate that p_θ could be used for good segmentation performance but not p_ν.

One way to do segmentation using a recurrent network is to look at the distance between the stimulus pattern and the resulting pattern after relaxation. In figure 6 we compare the segmentation on noise free data with the *ideal* case where the words are learned at a fix position (f13) versus all positions (a13). In the former case the words were learned until no further improvement in segmentation could be achieved. The words were in this case of the same length as the network.

In figure 7 we see how the segmentation quantitatively depends on p_θ when we have noise in the learning data. A noise level of 0.4 means that the probability $p(wrong\ character) = 0.4$ during stimulation for each population.

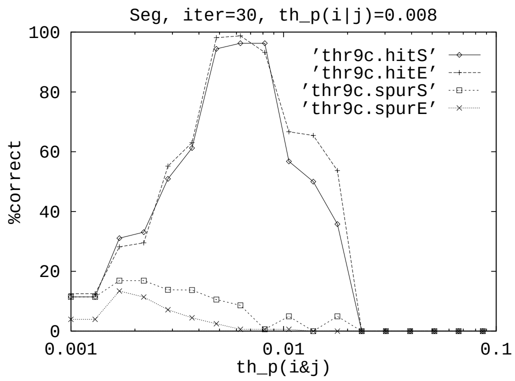


Figure 4: The segmentation performance dependency on the p_θ-level.

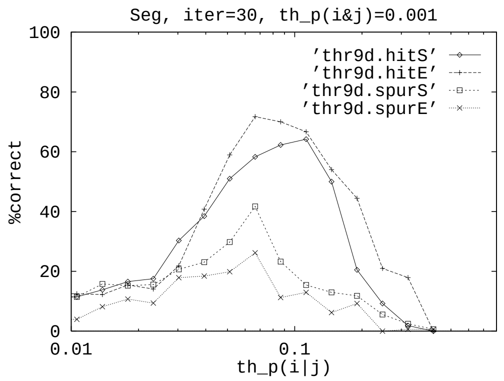


Figure 5: The segmentation performance dependency on the p_ν-level.

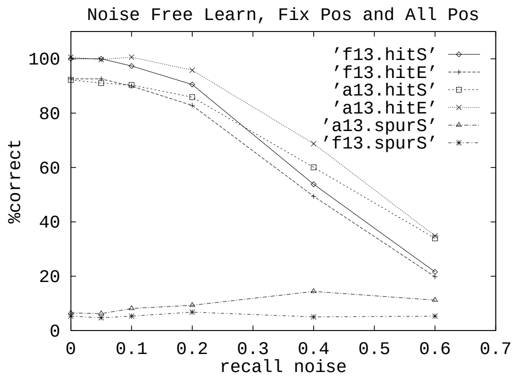


Figure 6: Segmentation when 13 words are learned at a fix position (f13) compared with when they are learned at all positions (a13).

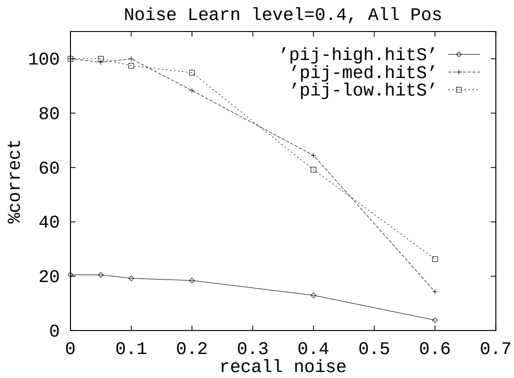


Figure 7: The performance of segmentation for three different p_θ-levels when there is noise (prob 0.4) in the training data.

References

[1] Grossberg, S. Unitization, automaticity, temporal order and word recognition. *Cognition and Brain Theory*, 7:263–283, 1984.

[2] Lansner, A. and Ekeberg, Ö. A one-layer feedback, artificial neural network with a bayesian learning rule. *Int. J. Neural Systems*, 1(1):77–87, 1989. Also extended abstract in Proceedings from the Nordic symposium on Neural Computing, April 17–18, Hanasaari Culture Center, Espoo, Finland.

[3] Schmiduber, J. Neural sequence chunkers. Tech. Rep. FKI-148-91, Institut für Informatik, Technische Universität München, 1991.

[4] Doutriaux, A. and Zipser, D. Unsupervised discovery of speech segments using recurrent networks. In Touretzky, D., Elman, J., Sejnowski, T., and Hinton, G., editors, *Proceedings of the 1990 Connectionist Models Summer School*, pages 52–61. Morgan Kaufmann, San Mateo, 1990.

[5] Lansner, A. A recurrent bayesian ann capable of extracting prototypes from unlabeled and noisy examples. In *Artificial Neural Networks, Proceedings ICANN-91*, pages 247–254. Royal Institute of Technology, Stockholm, Sweden, 1991.

Supplement VI

Regression Modelling

A Study of Process Modelling using Artificial Neural Networks

Roland Orre and Anders Lansner

SANS – Studies of Artificial Neural Systems
NADA – Dept. of Numerical Analysis and Computing Science
Royal Institute of Technology, S–100 44 Stockholm, Sweden

Abstract

We model a part of a process in pulp to paper production using feed forward connected neural networks. A set of parameters related to paper quality is predicted from a set of process values. The predicted values are results from laboratory experiments which are time consuming. We check for irrelevant inputs and we manage with training sets that are considered small. The output vector is separated into single values which are predicted on different architectures adapted to each output. A strategy that continuously adapts the process model seems to be useful. In this work the backprop learning algorithm been used.

1 INTRODUCTION

The general problem is to model a process $\boldsymbol{P}$ as vector transformations. The actual process that is modelled has as inputs a known parameter vector $\boldsymbol{x}$, an unknown parameter vector $\boldsymbol{y}$ and a random disturbance vector $\boldsymbol{d}$. The process' output vector $\boldsymbol{o}$ is thus

$$\boldsymbol{o} = \boldsymbol{P}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{d})$$

The process $\boldsymbol{P}$ may have a memory and be a function also of its parameters and output history but for simplicity we assume that it is not. The problem is to make a model that can predict the process' output vector $\boldsymbol{o}$ when only the vector $\boldsymbol{x}$ is given. The actual vector $\boldsymbol{x}$ is however, also unknown. Due to physcial measurement errors, manual input errors and approximation errors the input vector $\boldsymbol{x}$ is also an estimate. The modeled vector output thus becomes

$$\hat{\boldsymbol{o}} = \hat{\boldsymbol{P}}_{\boldsymbol{model}}(\hat{\boldsymbol{x}})$$

The specific problem in this study was to investigate how some measures of pulp quality could be predicted. These values are results from laboratory experiments and should be predicted from a set of input parameters that are obtained from quick automatic experiments and some additional information about the process. The laboratory measurements that are predicted are *csf, density, elongation, tear* and *tensile* with the emphasis on *tear* and *tensile.*

In the following will a process *signal* refer to either a process *input* or *output* value.

2 METHOD

The computational method developed for this study uses *feed-forward artificial neural networks* with the *error back-propagation* learning algorithm. This algorithm, which is most often called just *back-propagation*, will not be described here. There are several text books available on this subject. A theoretical textbook that describes this and other neural network principles is for instance "Introduction to the Theory of Neural Computation" [HKP91]. General introductions to the field are [MRtPRG86], [RMtPRG86], [MRtPRG88], [Neu90], [Nie90] and [NI91].

The *back-propagation* learning method is just one of several artificial neural network algorithms which are available today. The reasons for selecting this method here is that it is both well understood and straightforward to use. It is also, for these reasons, the method that has become the most used neural network method in the industry today.

1 Tools and Environment

The tool that is used for the back-propagation network learning and execution is "Aspirin/ MIGRAINE" release V5.0 [Lei91]. This is a set of programming tools for neural network simulations which has been developed by an internally funded research effort at the MITRE Corporation. These tools include a compiler for a neural network description language "Aspirin" and a user interface "MIGRAINE". The software is written in C and the Aspirin compiler generates C code as output that is automatically compiled and linked together with the MIGRAINE interface to create an application program that runs the specific network architecture(s) that is described in the Aspirin file.

The MITRE corporation decided to release this software March 1988 from V4.0 free of charge, publicy available. It may be freely used and modified for research and development purposes. The only requirement is that it is briefly acknowledged in any research paper or other publication where this software has made a significant contribution. If the software will be used for commercial gain the MITRE Corporation has to be contacted for further conditions of use.

The Aspirin/MIGRAINE software package is easily installed and runs on most UNIX systems. Tested systems are for instance Apollo, Convex, DecStation, IBM RS/6000, 486/386(System V), HP 9000, NeXT, News, Silicon Graphics and Sun. It also supports some coprocessor boards that can be added to a UNIX host, at the current writing these include i860 boards and iWarp Cells. During the current writing it was announced that Aspirin V6.0 is available.

For diagram generation the "GNUPLOT" V3.2 interactive plotting program has been used. This package is also free of charge and installs on all UNIX systems plus some others.

2 Method Development

The investigations performed in this report have resulted in the development of a method for preprocessing of data, doing an iterative search through different network architechtures and the removal of non significant network inputs. This code is written in the language scheme ([WC91]) and has been made so that this is the only interface needed to run the learning and prediction through MIGRAINE on a UNIX work station.

3 Specific Settings Used

To make it easier to repeat the experiments that have been done in this study, a few words about some specific control parameters, especially to the Aspirin/Migraine interface. A typical command line for learning would be "command -I 1000000 -l -d df -F net-name -t 10000 itok 0.05 -s 10000 -a 0.05 -F net-name save-name". For other parameters like inertia, default values have been used.

command This is the name of the compiled and linked MIGRAINE code for a specific declaration file. For a two layer net with seven inputs, ten hidden and one output this will, with the syntax used here, be "..bp/7_10_1".

-I 1000000 The upper limit for the number of learning iterations to run. This is not the same as *epochs*, which can be obtained by dividing the number of iterations with the size of the training set.

-t 10000 itok 0.05 Tells how often the test network will be tested and how many samples that have to be correct to see if it has reached the error limit. The "itok" has typically been set to the numbers of vectors in the training set. The error limit has typically been set to 0.05 when one want to learn the train set as good as possible within the iteration limit. For the continuous learning case the error limit has been set to a specific limit for each item to save time.

-a 0.05 The learning rate, "alpha", has typically been set lower or equal to the error limit. Alpha has typically been started on 0.1 for a limited number of iterations, like 100000, then decreased to 0.05.

3 DATA

1 Input and Output Data

The data that has been processed in this study is a set of vectors consisting of process input parameters and automatically and manually measured process outputs. The manually measured output values being predicted are *csf, density, elongation, tear* and *tensile.* Nine process inputs, most of them automatically measured, have been used. For proprietary reasons these inputs are called *par1* – *par9* here. The names and ranges of the input and output values used are shown in table 1.

2 Data Selection and Filtering

As the network learning algorithm will do a regression analysis of training data to create an input output mapping with as little error as possible, we need training data of as good quality as possible. To increase the quality some filtering of the data has to be done. There may be several reasons for filtering out data. Data may simply be missing, for instance due to some missed manual input, or data may be erroneous. In this study there has not been any possibility for any consistency check, *i.e.* , checking that each value in an input vector is reasonable considering all the other values. The only thing we could do was to remove data that obviously was outside some extreme limits. The limits for the acceptable range for each item was decided from histograms showing the distribution of each parameter. These histograms are shown in appendix A.

parameter	range		
csf	250	$\cdots$	449
density	523	$\cdots$	665
elongation	3.9	$\cdots$	6.3
tear	11.6	$\cdots$	16.5
tensile	60.8	$\cdots$	84.7
par1	0	$\cdots$	6
par2	0	$\cdots$	100
par3	0	$\cdots$	100
par4	0	$\cdots$	100
par5	29.2	$\cdots$	45.9
par6	6.3	$\cdots$	14.1
par7	22.2	$\cdots$	30.9
par8	255	$\cdots$	367
par9	51	$\cdots$	75

Table 1: The set of signals used listed with their active ranges. For two of the input values, *par2* and *par6*, there were many values missing. Of this reason these two signals were skipped from the input data set.

The total amount of data samples that have been available in this investigation is around 200 vectors from one process and around 50 vectors from another process. The set of 50 vectors was considered too small to make use of at the present time. Also the set of 200 vectors is a small set for some of the more complex nets that have been tested, why one should be cautious in drawing conclusions too far. After filtering the set of 200 vectors, by removing those vectors that obviously contained input values that were outside reasonable limits, we ended up with 110 vectors of good quality that could be used for training and testing. This may seem to be very few, but it was considered more important to have a good quality on those that were left than to risk learning of erroneous vectors. In the first experiments we tried to filter the data set selectively for each output parameter to not need to remove more samples than necessary but later on we removed all sample vectors containing some bad value.

3 Preprocessing and Scaling

In the preprocessing of data that is done before presenting it to a network, the first step consists of filtering out vectors where some of the signal values lie outside some accepted range. If too many values of a signal lie outside the accepted range then the whole column for that signal is removed and will not be used as input or output to the network. The data is then transformed to a form that is suitable as inputs to a network. In this study all signals have been construed as continuous valued data. The simpliest way to represent continuous valued variables as, for instance, *csf* above, is to let the analogue output value of a unit be proportional to the actual value. In that case only one neuron is needed to represent each value. The output value of a neuronal unit is normally, of practical reasons, limited. It is therefore necessary to scale the parameter values to fit within the limited output range. The scale has been chosen so that the parameter range is mapped to fit within an almost linear piece ($0.25 \cdots 0.75$) of the sigmoid output function of the units (figure 1). There are not the same limitations of the input signals, but for practical reasons all inputs have been scaled to lie within the same range as the outputs, i.e., $0.25 \cdots 0.75$.

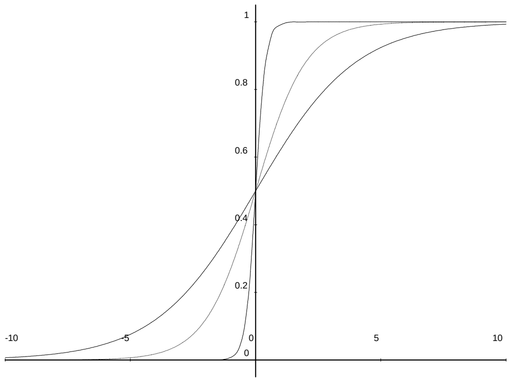


Figure 1: A sigmoid function, in this case $(\exp(-kx)+1)^{-1}$, with three different k. This kind of squashing function is the most commonly used one to make the output of a neural unit non-linear. The output of a unit will show an almost linear response for a small input range, otherwise the output is limited, often to the interval 0...1.

4 RESULTS

To be able to classify how well a network predicts a parameter one needs some measure of quality. In the following we have used the average error (merr) and the standard deviation (sigma) for the difference between actual and predicted signal. The standard deviation and the average error are both expressed in percent of the dynamic range for the predicted signal. The average error is good measurement to see if there is some some offset or systematic error in the predicted values. The standard deviation is a well established error measurement, the formula used for its calculation here is

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \frac{\sum_{i=1}^{n} x_i}{n})^2}{n-1}}$$

1 Partitioning of Input Data

Before training a network, the data set has to be splitted into a training part and a test part. This has to be done to be able to measure the *generalization* capability of the network, *i.e.* , its performance on data it has not been trained on.

For some experiments the data has been splitted in four different ways, *first, last, random* and *split.* These are illustrated in figure 2. The partition “First 75%”, for instance, says that 75% of the total amount of data has been picked consecutively, starting from the oldest sample. “Last 75%” is similar but goes backwards starting from the most recent sample. For “Random 75%” the samples have been picked according to a rectangular distribution from the whole interval. The “Split 3-1” separates the data systematically, three to the train set, one to the test set and so on. These four ways of partitioning data are called “first”, “last”, “rand” and “split” in table 2 and “fi75”, “la75”, “rn75” and “sp31” when they occur in most other tables, like in appendix B. A column head like *csf.fi75*, tells us that this is the output signal *csf* predicted with the partition “First 75%”.

Too see what difference the type of data partition can make, look at table 2, which has been extracted from the tables in appendix B. What is shown is the prediction quality averaged over all architectures for four different partitions of the data set. As the “last” partition shows the highest error rate it indicates that the first 25% of the data set would behave different than the rest. This could be interpreted as that something has happened with the process during the first sampling period. Further, if we assume that the output signals and the input parameters would change slowly, but mainly caused by other factors than the input parameters, then the best guess would probably be that the new value is just close to the old one. In such a case it could be expected that the systematic "split 3-1" partitioning would perform best. For *csf* and *tensile* we actually get the best performance with the "split" partition. The difference may, however, not necessary be significant for a conclusion. For industrial process signals are, of course, predictions about the closest future the most interresting to make. Of this reason, most of the prediction experiments have been done by using the "First 75%" partition.

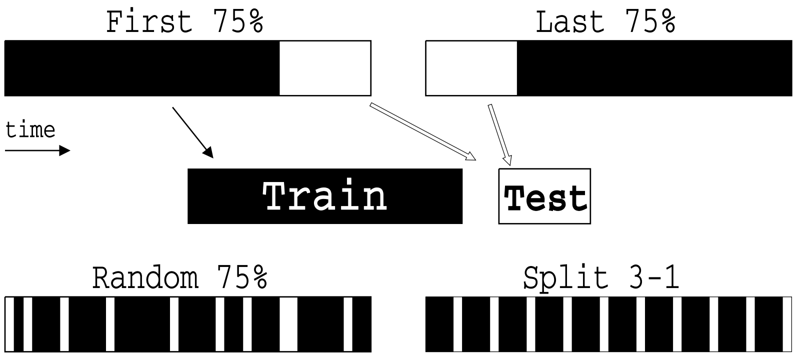


Figure 2: Four different partition types, which have been used. In all four cases have 75 % of the input data been used as training data.

	first	last	rand	split
csf	14.1	37.8	14.3	11.2
density	12.9	15.3	9.9	14.4
elongation	8.0	12.0	10.9	9.6
tear	14.0	26.6	12.3	13.9
tensile	15.7	16.5	13.2	12.9

Table 2: Prediction performance for different partititons, averaged over all tested architechtures. As "last" shows the largest error for all outputs it indicates some change in the process during the first sampling period.

2 Finding a Good architechture

The network algorithm makes the input output mapping by adapting the weights of the network to the training data. If certain criteria of the training set are fulfilled, like that each input vector corresponds to a unique output vector or value, then the mapping function of the network can be done with arbitrary precision if we choose a network which is just complex enough. If the network get too complex, however, it will probably *generalize* badly. This problem can be thought of as similar to finding a polynomial approximation to some function. If the polynomial degree is too high it will probably approximate the actual function badly due to noise in the regression input data.

If there is a big amount of training data available the network can be allowed to be rather complex. In the case studied here we tired five different architectures from a simple one layer, i.e., where the output is just a linear combination of its inputs, to a three layer architechture. Three different two layer networks have been tested where the three layer network had about the same complexity as the two layer network with 20 units in its hidden layer. See figure 3 for a clarification of this. The number of inputs have varied but the first hidden layer has had 10, 20 and 30 units. The three layer net had 10 units in both hidden layers. The syntax used in result tables is like: "**7_20_1**" or "**7_10_10_1**" where the first means 7 inputs, 20 in hidden and 1 output and the second is 7 inputs, 10 hidden, 10 hidden and 1 output.

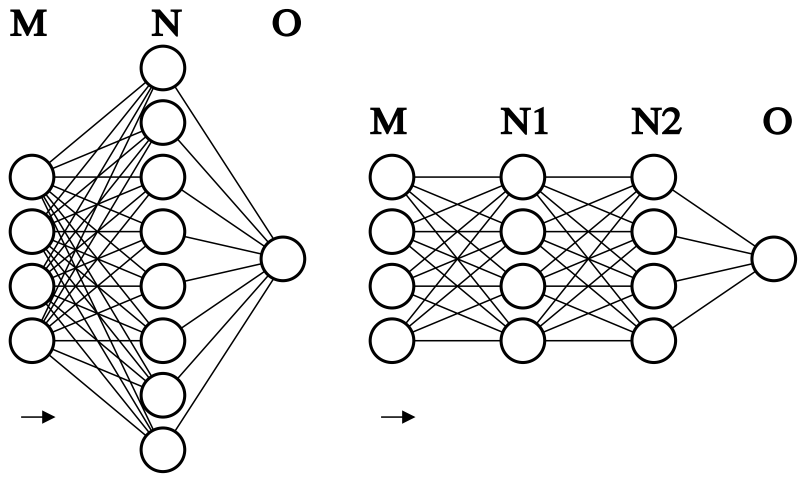


Figure 3: Two networks with about the same complexity. With the syntax used in this document to denote network architectures they would be described as M_N_O (4_8_1) and M_N1_N2_O (4_4_4_1) respectively. The number of weights in the left one is $((M+1)N)+(N+1)O = 49$ and in the right one $(M+1)N1+(N1+1)N2+(N2+1)O = 45$.

In figure 4 below, we can see how two different networks performs on a training and a test set of the parameter *elongation* (stretch in diagram). A one layer network, which corresponds to a linear system of equations, and a two layer network with 30 units in the hidden layer.

2.1 Performance on different architechtures

When doing a prediction of some output parameter we normally don't know in advance how complex the problem is. It may be linear in the sense that each output can be written as a linear combination of its inputs or it may be so complex that it needs several hidden layers with a lot of units. To get a picture of how complex the problem is for each output parameter each of them has been been run separately on several different architechtures (Appendix B). Table 3 shows the average of the standard deviations of the prediction error over different partitions for the output parameters *csf, density, elongation, tear,* and *tensile* when these are learned on different architechtures. From this table it may be a preliminary proposal that *csf* and *tear* are best predicted using a linear combination of its inputs when *elongation* seems to do best with a two layer net and *density* and *tensile* seems to make best use of the properties of the three layer network.

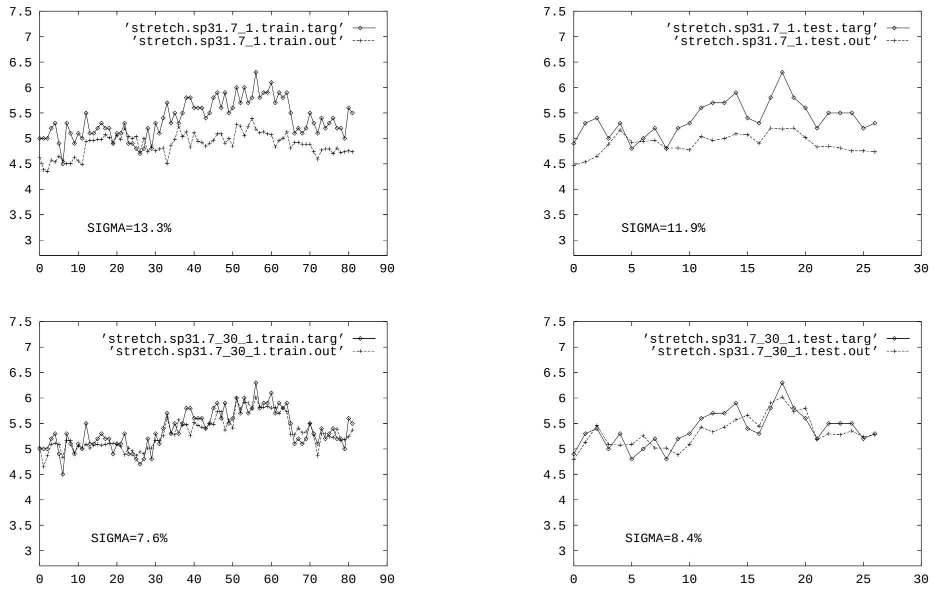


Figure 4: On the left is shown how two networks have adapted to a 75% training set for *elongation* (stretch in diag). The upper diagrams show a one layer, *i.e.* , a linear combination, network and the lower diagrams show a two layer network. The standard deviations for the errors are shown in the diagrams. On the right we see the prediction performance for the same nets on a 25% training set.

3 Requirements on Cpu-time

The *backpropagation* learning algorithm is a numerical method that is similar to *gradient descent*, *i.e.* , a way to find an optimal solution (a minimum), versus some set of adjustable parameters, to some function under some boundary conditions. In the neural network case the adjustable parameters are the weights and the function to minimize is the output error. The time to train a network is not a simple function of the number of weights, neither is it a simple function of the number of training samples. If there was just one optimal solution then the time to find the optimum could grow faster than polynomial in the number of weights. There may often, however, in a large network be multiple solutions that are equally good. Generally speaking, it takes a longer time to train a more complicated network. In the experiments performed in this study we have run the training until a specified error limit was reached, or until a maximum number of iterations has been performed. In the tables in, for instance, appendix B the table include the value “Kepo” that is the number of thousands of training epochs. One epoch is when all training samples have been used once. In figure 5 it is shown how “Kepo/hour” decreases when the number of weights increases. When the experiments have been run the iteration limit has been set so that the most complex net should come no further, which means that the less complex nets have run many more iterations than necessary. The nets that have been used here need no more than an hour to to retrain completely on a reasonable fast personal computer or a workstation. It should be noticed that a larger training set would not make the training time significantly

TRAIN set	7_1	7_10_1	7_20_1	7_30_1	7_10_10_1
csf	10.0	6.8	6.8	6.7	6.9
density	11.2	9.4	9.2	9.3	9.7
elongation	12.6	7.8	7.9	8.0	8.4
tear	12.9	10.4	10.2	9.8	9.6
tensile	12.1	9.5	9.5	9.6	9.9
TEST set					
csf	11.7	13.4	21.9	20.9	18.2
density	13.8	13.9	13.6	13.1	11.3
elongation	11.4	9.8	9.6	9.5	10.4
tear	13.4	17.5	19.5	19.2	14.3
tensile	14.4	13.7	16.0	15.7	13.0

Table 3: A more complex architechture generally performs better on the training set but may cause a bad performance on the test set. Here it seems like *csf* and *tear* just need one layer but *elongation* needs two and *density* and *tensile* needs three layers.

longer. If the set of training samples is too small compared with the network size, then the training time would be rather small also for a complex net but with a bad generalization capability.

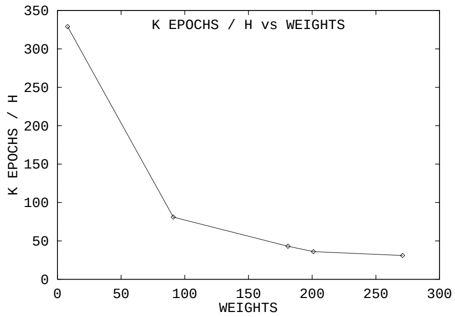


Figure 5: Number of K epochs, i.e., number of thousands epochs per hour, plotted versus the number of weights for five of the networks that have been used in this study. The timing which has been plotted here is for the networks 7_1, 7_10_1, 7_20_1, 7_10_10_1 and 7_30_1 with *stretch* as the output. It was measured on a DecStation 5100.

4 Selection of Significant Inputs

When we learn a function by examples it may be so that an output is much dependent on some of the inputs, but it can also be so that the actual function that we try to predict is not at all dependent on a specific input parameter. In the latter case when we adapt the network to the training data the algorithm will of course try to adapt the network also to those variables that the output is actually not a function of. When this network then is used on test data, these non significant inputs may act just as random disturbances.

If plotting just one variable versus another we can visually get a feeling that one input is clearly dependent on some other input as in figure 6 or that they are mostly independent as seems to be the case in figure 7. This is however, not easy to do visually when dealing with multi dimensional data.

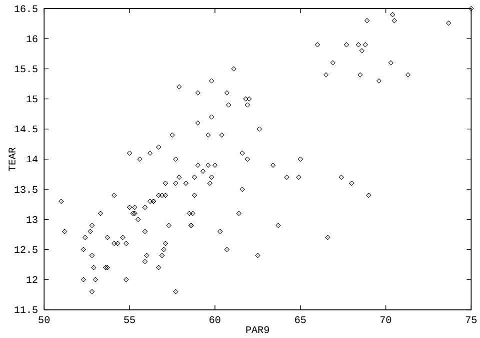


Figure 6: These two variables show a clear correlation.

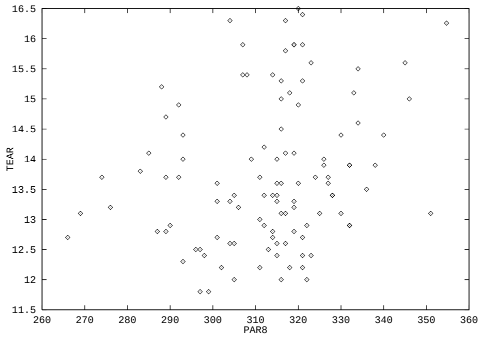


Figure 7: These two variables seem to be uncorrelated.

4.1 Input sensitivity test

The method used here to find the input parameters that have the greatest impact on an output value is to first train a network with all input parameters available and then test how much the prediction capability of the network is affected when each of the input parameters is "removed". The removal of the input has been done here by replacing the input with its average value. The tables in appendix C show how each of the parameters have been predicted from all the others. The tables are arranged from greatest impact to smallest. The table 4 below can also be found in appendix C. These two tables show how the prediction performance on the test sets for *tear* and *tensile* is affected when each of the other parameters are replaced with its average. A "—" signifies that none of the inputs were missing. An interpretation is that *tear* is mostly affected by the input *par3*, and *tensile* by the parameter *par4*. Another interpretation may be that *par1, csf, par7* and *density* may not be significant for *tear*. In the same way *density, tear* and *par3* appear to have a low impact on *tensile*. It may be noted that the error difference when an input is "removed" and when it is available in most cases here is to small to give an answer with high significance for the small input data sets that have bee used.

4.2 Removal of non significant inputs

The tables in appendix D show how the performance is affected when the inputs that in appendix C caused an improvement on the results of the test set when they were removed from the train and test sets. In some cases they have been run on more than one architechture, both the architechture as in appendix C and some other architechture where they previously showed a good result.

4.3 Removal of Lab Data inputs

In the results shown in appendix E the strategy for removal of inputs have been a little different. The inputs here do not include those that are time consuming to measure, *i.e.* , *csf, density, elongation, tear* and *tensile*. Further, if some input showed to be insignificant on the training it was removed. The tests have been performed as in appendix C.

These tests indicate that the pulp type, indicated by *par2, par3 and par4* is significant for both *tear* and *tensile*. We can see that the *par1* seem to bee of some significance for *csf, elongation, par9* and *tensile*.

tear	%sigma	%merr
par3	22.3	6.1
par9	20.0	0.7
elongation	19.2	-1.0
par4	17.7	-0.9
par8	17.4	-1.1
par5	17.3	1.3
tensile	17.2	4.3
—	17.1	-0.3
par1	16.5	-1.9
csf	16.5	-0.3
par7	16.2	6.2
density	16.2	0.1

tensile	%sigma	%merr
par4	20.7	-19.2
csf	16.1	-7.6
elongation	15.7	-7.8
par9	15.4	-7.5
par5	15.4	-7.1
par1	15.3	-6.9
par7	15.3	-5.1
par8	15.2	-7.5
—	15.2	-7.5
density	15.1	-7.1
tear	14.2	-12.7
par3	13.8	1.3

Table 4: From appendix B. Prediction performance on the test set. The properties *par1, csf, par7 and density* seem, from this test set, somewhat less important for *tear* than pulp type and fiber length. Also for *tensile* it seems as pulp type has a high significance. The error differences here are probably to small to tell that some of these input signals are irrelevant for *tear* and *tensile*.

5 Different Learning Strategies

5.1 Instant prediction

The prediction being performed in the previous examples is based on the instant input values only. The assumption is that the predicted process has no memory, *i.e.* , an output is a function of the momentarily input values only and does not depend on the history of an input or output.

As a simple example of a process that can not be predicted this way. Imagine that the process we model is the output voltage from a capacitor being charged with a current proportional to the input voltage. In such a case nothing can be said about the output values by just looking at the instantaneous input values.

5.2 Continuous learning

If the process which is modelled would change its characteristics over the time, then it could be expected that we could improve the prediction result by making the network a one step predictor. The idea is that we let the network learn the behaviour of the process during a limited time span. The process outputs are then predicted from the next input vector being measured. When the actual output vector, which was predicted, becomes available, this can be used as new training data for the network. The oldest sample is then thrown and replaced with the newest one. The figure 8 below shows how *tear* is predicted with a one layer and a two layer network using *par1, par8, par5, par9, , par4* and *par3* as inputs. The performance according to the standard deviation is about equally good with 12.0% for the **7_1** net and 12.3% for the **7_10_1** net. The latter net does, however, show a better error distribution with no offset why that one would be to prefer. In figure 9 it is shown how many learning iterations that has to be done for each new sample to reach the same error performance that was obtained on the first traning set. With "iterations" is here meant the number of separate training patterns that has to presented for the network. A low number shows that the network could be quickly retrained within the goal to keep the error level low. A high value here indicates that the network could not learn the new sample within a limited number of iterations.

In appendix F the one step predictor performance for the continuous learning case is listed for *csf, density, elongation, tear* and *tensile*. The columns "%sigma" and "%merr" there, as before, stand for percent standard deviation and percentage mean error. There is also a column "maxerr" that shows the largest error as $predicted - target$ in actual units. Diagrams for error distribution use the range of the neuron outputs. There are also diagrams for the number of iterations needed to relearn the network after each sample. In most cases when the whished error limit could not be reached quickly, the number of iterations have reached their limit. This is also a nice result in that sense that a much lower threshold on the number of the limit iterations can be set without affecting the performance.

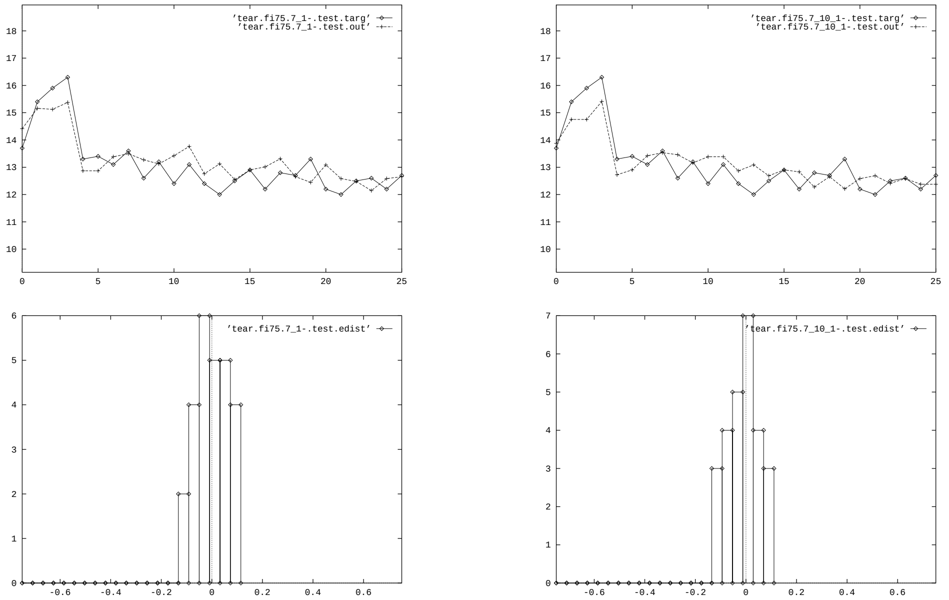


Figure 8: The property *tear* predicted with a one-layer (left) and a two-layer (right) net. They perform equally well but the two-layer net has a better error distribution with no offset, thus, the two-layer one would be preferred here.

5.3 Continuous learning, Tear and Tensile

In appendices G and H we have studied the *tear* and *tensile* in particular. In this case more samples have been used than in the previous examples. The original data set size is the same but here only those sample vectors containing bad values that would affect *tear* and *tensile* have been removed in the preprocessing. We then ended up with 179 useful sample vectors. The training approach was here a little different from the previous section.

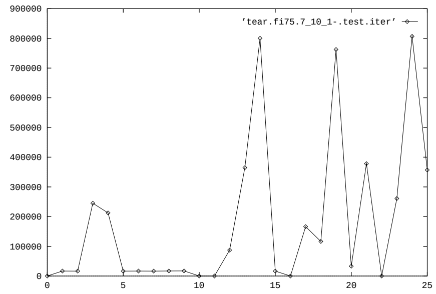


Figure 9: In the continuous learning case the number of learning iterations may vary widely for each new sample. A high iteration count here tells us that the whished error level not could be reached for a certain sample.

The network has been trained until no significant improvement could be obtained on the test set. Then, for each step forward, the network has been retrained but it has not been allowed to retrain fully. The number of iterations has been limited to about 200000. The idea behind this is to produce a kind of inertia, i.e., a small variation in the input would not cause a drastic change of the network parameters.

The same length as in previous section , i.e., 82 samples has been used as the training set for prediction of the next sample. This means 46 % of the total set. We can see from the table in appendix G that the best result here is obtained from the linear, one layer network in the case with continuous learning. The *7_10_10_1* could probably have given a better result if it had been allowed to retrain fully as is indicated by the retrain iteration counts diagram.

When we look at the result for *tensile* in appendix H we see that we also here get the best result from the one layer network with continuous learning.

There is a great prediction failure between sample number 15 to 30 for all the multilayer networks where the one layer network performs rather well. This is probably an indication of too few training samples. The *tensile* target is rather rugged as can be seen from the diagrams. The multilayer nets have tried to adapt to this shape. If the jaggy shape actually is there due to some distorsion or measurement errror then it could be expected that the more complex nets would try to adapt to this too quickly with too few training samples available.

5.4 Fixed learning, Tear and Tensile

To get a chance to compare the performance of the continuous learning case with the one where we have trained the net only on the beginning of the history for the same data as in the previous section, here 82 samples, these are shown in appendices I and J. The one layer network here seems to have no possibility to follow the tricky part between sample 15 to 30 that appear both in *tear* and *tensile*. There seems to be information about this, anyway, in the training part as all the multilayer nets react on this interval. Here one can see that the number of units in the hidden layer have a great influence on the capability to cope with the sample interval 15 to 30. One interresting result is of course that the three layer net **7_10_10_1** detects the interval perfectly but goes in the wrong direction both for *tear* and *tensile* in this interval.

5.5 Temporal prediction

The Aspirin tool which has been used here supports three forms of temporal coding. One is to have recurrent connections from delayed replicas of outputs that can be weighted in on any layer. The delay can be from one to four steps. Another form is to delay the inputs one or several steps and present these delayed inputs through weights for some layer. A third method is to do averaging over several input and present the average value through weights for the network. These methods can of course be combined. If one of these methods was going to be used here it is of course hard to say which method would be best. This would depend on the problem. The results here does not include any of these temporal codings. There are some problems with temporal coding that have not been solved. One problem is that the samples are not equidistant in time, thereby making it necessary to do some kind of extra preprocessing so that the samples become equidistant. One such way is to do linear or spline interpolation between the samples.

5.6 Limited history

In an implementation of this method the learning strategy will probably be some form of continuous learning. In such a case it may be important to address the problem of which size of the history that is preferrable to learn. The reason for this is that the process may change due to external environmental changes that are not measured. In such a case will the old history more act like disturbances.

6 Best Results

In table 5 below are the best results that were obtained in this study extracted. The table shows the signal name, the partition strategy, the learning strategy, the parameters standard deviation and its average error. The table does only include the outputs *csf, density, elongation, tear* and *tensile*. It may be noted that the table with best possible results, any partition, where the input information is restricted to quickly available parameters, shows better results than the table where the partition is restricted to the first one, without any restrictions on the input parameters. If this difference should be treated as being significant it is interresting, because, if the result depends on the partition then it would indicate that the prediction may be done better by, for instance, decreasing the size of the history etc.

The most interresting table may be the one that shows the continuous learning case. This is the case that would most correspond to the reality when doing prediction in real time, but the results may be possible to improve by, for instance, more samples over a shorter period, temporal and interval coding. See the section “FUTURE EXTENSIONS” for a discussion about this. The diagrams with test results for the continuous learning case are supplied in appendix F. There is also some statistics about the actual input values used in appendix K.

Best possible, process data only, any partition					
output	arch	part	sigma	merr	inputs
csf	7_1	sp31	6.8	-1.8	p1,p8,p5,p9,p7,p4,p3
density	7_10_10_1	la75	9.2	6.8	p1,p8,p5,p9,p7,p4,p3
elongation	7_20_1	fi75	7.1	8.2	p1,p8,p5,p9,p7,p4,p3
tear	7_1	rn75	11.7	-7.6	p1,p8,p5,p9,p7,p4,p3
tensile	7_10_10_1	la75	10.8	5.2	p1,p8,p5,p9,p7,p4,p3

Best with process data and lab data, first partition					
output	arch	part	sigma	merr	inputs
csf	8_10_1	fi75	9.0	7.9	p3,ten,p8,p7,den,p9,elo,p1
density	7_10_10_1	fi75	11.9	5.8	p3,tea,csf,p4,p7,p5,pl
elongation	7_20_1	fi75	7.1	8.2	p1,p8,p5,p9,p7,p4,p3
tear	7_1	fi75	11.9	7.5	p1,p8,p5,p9,p7,p4,p3
tensile	8_10_10_1	fi75	14.4	-4.4	p4,csf,elo,p9,p5,p1,p7,p8

Best with process data, first partition					
output	arch	part	sigma	merr	inputs
csf	7_30_1	fi75	13.5	0.1	p1,p8,p5,p9,p7,p4,p3
density	7_10_10_1	fi75	11.9	10.3	p1,p8,p5,p9,p7,p4,p3
elongation	7_20_1	fi75	7.1	8.2	p1,p8,p5,p9,p7,p4,p3
tear	7_1	fi75	11.9	7.5	p1,p8,p5,p9,p7,p4,p3
tensile	7_30_1	fi75	15.3	-6.9	p1,p8,p5,p9,p7,p4,p3

Best with process data, continuous learning					
output	arch	part	sigma	merr	inputs
csf	7_30_1	co75	11.9	-0.8	p1,p8,p5,p9,p7,p4,p3
density	7_20_1	co75	13.9	2.6	p1,p8,p5,p9,p7,p4,p3
elongation	7_20_1	co75	8.0	2.3	p1,p8,p5,p9,p7,p4,p3
tear	7_1	co75	12.0	2.8	p1,p8,p5,p9,p7,p4,p3
tensile	7_10_10_1	co75	14.3	-3.3	p1,p8,p5,p9,p7,p4,p3

Table 5: The best results that were obtained in this study. Abbreviations: den=*density*, elo=*elongation*, tea=*tear*, ten=*tensile*. *par1* – *par9* are written as *p1* – *p9*.

5 DISCUSSION OF RESULTS

The results of this study have showed that it is possible to do a prediction of certain laboratory measurements with a fairly good precision. Some important questions that arise are:

- Are the results good in comparision with other methods?
- Is the precision obtained high enough?
- Can the precision be improved?

Considering the first and second questions it is not possible to give a concise answer as we have not had results available from similar experiments with other methods on the same type and amount of data. Further, the data set underlying our investigation have most likely been too small to allow an answer with high significance.

1 Prediction Quality

We have reached a prediction quality that lies around 5-10% standard deviation, calculated relative to the dynamic range of the parameters. Under some circumstances this may be sufficient and under others it may not. For many of the parameters one should take into consideration that the actual ranges that have been used are often small compared to the absolute value of the parameters.

It must be clear, however, that no method can make a better result than what is possible due to disturbances and measurement errors in the training and test data. If the errors for these were measured as parts of the absolute values they would be much smaller than presented here. We did also observe that the prediction quality was rather much dependent on the type of partitioning into training and test set that was done as was shown in table 2. This indicates that something has happend in the process from the first to the last sample. The prediction quality can probably be increased by using more samples and by using more measurement values. More samples is always better as this allows us to use a more complex network and more measurement values, like for instance *temperatures* etc., may allow us to find other significant inputs.

The coding technique used is also important. By using, for instance interval coding, as is described in the section “FUTURE EXTENSIONS” below, it may be possible to gain some more robustness and reasonable solutions also when the data set is inconsistent and unreliable.

2 Data Availability

The data that has been available for training of the networks has been quite small compared to what common rules of thumb say. For some of the more complex nets, we should, considering these rules, have used at least ten times the amount of data that was actually available. This would be perfectly possible considering training times etc. This shortage of data has to be taken into account when the results are evaluated. Those cases where we got a poorer result with a larger hidden layer than with a smaller one, indicate that the data set was in fact too small. Some less intuitive results, such as the indication that density would not be significant for predicting tear and tensile may also be attributed to the relative lack of data.

6 FUTURE EXTENSIONS

Under the assumption that the neural network method developed and studied here is considered good enough, it may be used almost “as is”, to do prediction of industrial process parameters. The method could, however, be extended to cover a wider range of usage than the cases studied here would indicate.

1 Measurement Estimation

One of the first things we did in this study was to throw away those samples containg data that could be easily detected as not being within reasonable limits. The reason for doing so is of course that if we fed this data into the learning network it would learn an illegal input output mapping. There may be several reasons for data to be outside allowed limits. The data may either be totally missed due to forgotten manual input or outside limits due to a bad probe. The reason may also be that laboratory data is missing due to some problems. It may also be that some tests are so complicated or expensive that they are preferably not done too often.

In all the cases above in which some data is missing it may not be necessary to throw the whole sample anyway. Under the assumption that one datum can be reasonably estimated from a subset of the rest of the data samples, then the missing datum can instead be replaced with its estimation. Each parameter can have an optimal associated network that runs on a timescale suited for the cost and prediction quality for that specific parameter, thus making all parameters instantly available without explicit measurements.

2 Consistency Check

When a datum has found to be within its allowed extreme limits this does however, not guarantee that it is valid. A manually fed value may look right but may still be 20 % from the correct value. A probe may be degraded or need calibration. Assume then, that a set of networks are used for prediction of each parameter from a subset of the other parameters. For each new sample that arrives, the network performs a prediction of each of the parameters. This prediction can then be compared with the value just fed in. Data can thus be checked sample by sample, datum for datum that it does not significantly deviate from its predicted value or that it does not cause an extreme value in some other parameter. This will of course be a great help for an operator doing manual input but it will also provide a possible way to give alarms about the necessity for recalibration or replacement of degraded probes.

3 Optimization of Learning Time

The method could probably, with some effort, be extended with optimization of the learning time. The learning iterations would then be stopped when the performance on the training set could not be improved and before the performance on the test set would possibly get worse due to overlearning. This optimization may be a hard problem to solve in a general way but by using statistics about required iterations versus number of connections this could probably be done in a "good enough" way. A great advantage with this would be that the required precision would not need to be set in advance. The network would do the best possible anyway.

4 Installation of online system

The method studied in this work for parameter estimation can be installed online on a workstation or on a reasonably fast personal computer, preferrably a UNIX work statiton.

5 Interval Coding Technique

An important coding technique that has not yet been tested is to use several units to code for one value. Such interval coding technique can be utilized for both logical (on/off) as well as for analogue values.

There are several reasons that speak for interval coding. We may first look at the input values. First it may be so that the precision needed is not rectangular distributed over the whole input range. An input value range may be distributed into several groups where all the values lie in clusters with almost no values in between. With an interval coding technique the available precision can be focused on these groups. One may also want to indicate the quality of a sample. With interval coding it is possible to lower the significance for certain samples.

The same reasons why interval coding is good on the inputs is also valid on the outputs. There is however, one more important reason to have interval coding on the outputs. As was mentioned previously, a multi-layer network with an architecture complicated enough, can in principle learn an input output mapping with arbitrary precision if each input vector corresponds to a unique output vector. If, however, one input vector value can result in several different output values in the training set, then this set can not be learned completely. By instead using interval coding on the outputs the possibility to learn the training set can be increased. It is also possible that the training vectors are nice but during prediction one may get conditions when the input vector would indicate two or several contradictory values on the outputs. This situation can also be resolved with the help of interval coding.

In cases with multiple valued outputs this can be interpreted as multiple hypothesis about the values.

6 Temporal Coding

As was mentioned in section 5.5, there are some temporal representation problems that have to be solved before a temporal coding of the inputs can be done. One method is to sample the process periodically, thus making the samples equidistant in time. Another method, that was previously mentioned, could be to do some kind of interpolation between the samples. If the process contains some kind of "memory", then such coding technique may improve the results further.

7 SUMMARY

This study has focused on the problem of predicting various measures of paper quality from relevant input parameters. Typically, we have reached a prediction quality of 5-10% relative standard deviation. The results obtained do not give a complete answer to whether or not the artificial neural network technique can do better or at least equally good as other methods. It has, however, indicated that a quite good result can be obtained even though the amount of data available for training and evaluation of the networks was limited. Some ways of extending the use of this method to on-line verification and filling in of data has been proposed. A neural network based technique has an advantage in that it is self-organizing and capable of adapting to a process that may change with time.

8 Acknowledgments

STORA Teknik AB is acknowledged as being the sponsor of this project and supplier of data. We also want to thank the MITRE corporation for making their Aspirin/MIGRAINE software package available and Aubrey Jaffer at MIT AI-lab for making his scheme package *scm* available.

References

[AM89] Y.S. Abu-Mostafa. The vapnik-chervonenkis dimension: Information versus complexity in learning. *Neural Computation*, 1:312–317, 1989.

[HKP91] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation*, volume 1. Addison-Wesley, Redwood City, 1991.

[Lei91] Russel R. Leighton. The aspirin/migraines software tools, users's manual, v5.0. Tech. Rep. MP-91W000050, The MITRE Corporation, 1991.

[Maa92] Wolfgang Maass. Bounds for the computational power and learning complexity of analog neural nets. Extended Abstract, 1992.

[Mac92] David J.C. MacKay. A practical Bayesian framework for backprop networks. *Neural Computation*, 4(3):448–472, 1992.

[MRtPRG86] J.L. McClelland, D.E. Rumelhart, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 2. MIT Press, Cambridge, 1986.

[MRtPRG88] J.L. McClelland, D.E. Rumelhart, and the PDP Research Group. *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs and Excercises.* MIT Press, Cambridge, 1988.

[MSW90] W. Thomas Miller, Richard S. Sutton, and Paul J. Werbos. *Neural Networks for Control.* MIT Press, Cambridge, 1990.

[Neu90] Inc. NeuralWare. *Neural Computing.* Penn Center West, Pennsylvania, 1990.

[NI91] Marilyn McCord Nelson and W.T. Ilingworth. *A Practical Guide to Neural Nets.* Addison-Wesley, Massachusetts, 1991.

[Nie90] Robert H. Nielsen. *NeuroComputing.* Addison-Wesley, Massachusetts, 1990.

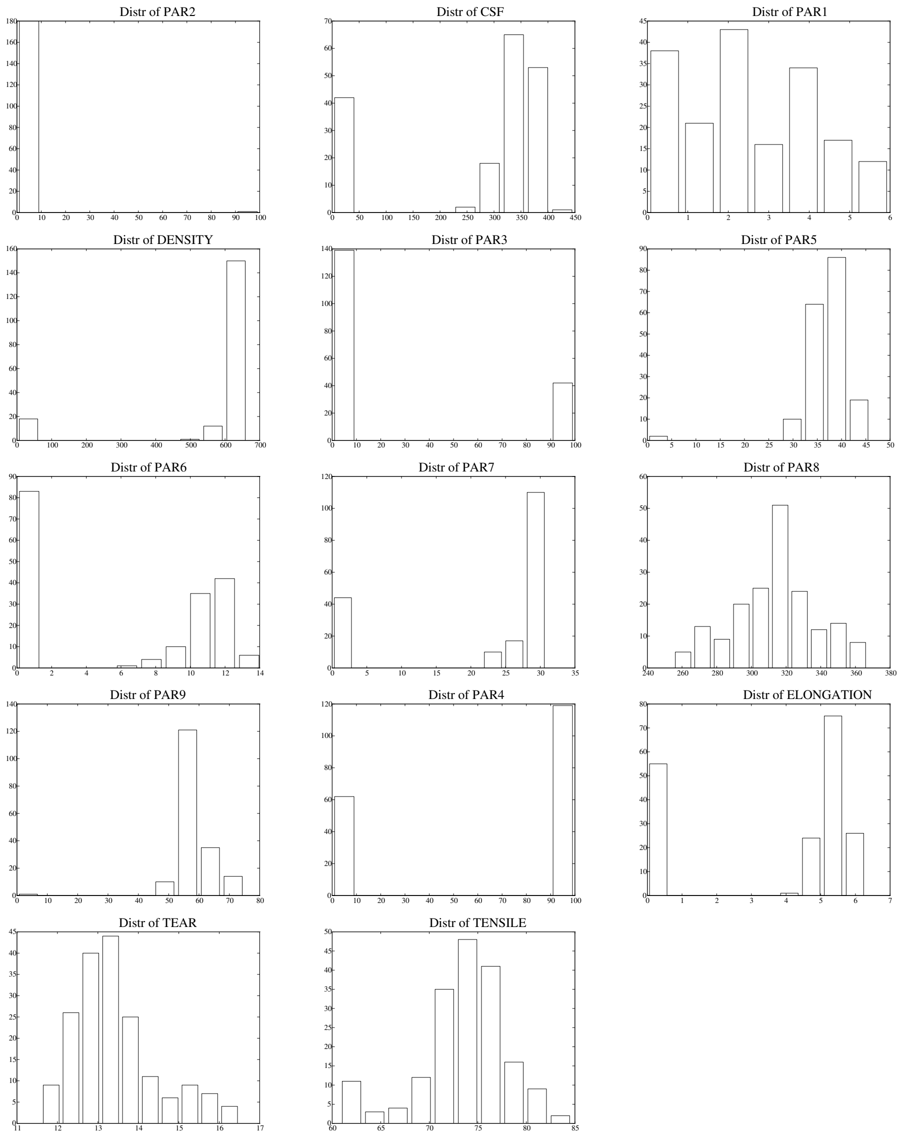
[RMtPRG86] D.E. Rumelhart, J.L. McClelland, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge, 1986.

[WC91] J. Rees et al W. Clinger. Revised^4 report on the algorithmic language scheme. 1991.

VII

Appendix

Appendix A : Histograms



These histograms show the frequences for the “ANNJ” data set. The “PAR2” has here the same value for almost all samples and for “PAR6” are half of the samples missing why these items could not be used as inputs.

Appendix B : Find Architecture

Train set

csf.fi75	%sigma	%merr	Kepo
7_1	8.7	-2.7	10.9
7_20_1	6.8	0.7	11.2
7_10_1	6.7	0.6	11.0
7_30_1	6.6	0.7	11.0
7_10_10_1	6.6	-0.3	10.9

csf.la75	%sigma	%merr	Kepo
7_1	10.1	-1.0	10.8
7_30_1	6.8	1.4	10.8
7_10_1	6.7	2.6	10.5
7_20_1	6.7	1.0	10.9
7_10_10_1	6.7	3.8	10.5

csf.rn75	%sigma	%merr	Kepo
7_1	10.3	-2.6	11.3
7_10_10_1	7.4	-3.5	11.0
7_10_1	7.0	-1.0	11.1
7_20_1	6.7	-1.8	11.3
7_30_1	6.5	-1.4	11.3

csf.sp31	%sigma	%merr	Kepo
7_1	10.9	-2.9	10.8
7_30_1	7.0	1.4	11.4
7_10_10_1	7.0	2.3	11.5
7_20_1	7.0	0.5	11.2
7_10_1	6.8	1.4	11.3

dens.fi75	%sigma	%merr	Kepo
7_1	10.0	-1.3	12.0
7_10_1	9.2	0.6	11.6
7_30_1	8.5	0.2	11.4
7_20_1	8.4	-0.7	11.5
7_10_10_1	8.4	0.1	11.7

dens.la75	%sigma	%merr	Kepo
7_1	12.1	-1.2	12.2
7_10_10_1	11.1	-0.6	11.2
7_20_1	10.3	-1.2	11.2
7_30_1	10.2	-1.0	11.2
7_10_1	10.1	-0.7	11.3

Test set

csf.fi75	%sigma	%merr
7_10_10_1	14.5	-2.4
7_1	14.4	-5.9
7_10_1	14.3	-0.6
7_20_1	13.8	0.1
7_30_1	13.5	0.1

csf.la75	%sigma	%merr
7_20_1	48.3	23.3
7_10_1	45.7	26.0
7_30_1	44.7	18.3
7_10_10_1	36.9	23.3
7_1	13.3	-14.6

csf.rn75	%sigma	%merr
7_10_10_1	16.1	-7.2
7_10_1	15.7	-6.1
7_20_1	14.4	-6.6
7_30_1	13.3	-6.3
7_1	12.2	-3.9

csf.sp31	%sigma	%merr
7_10_1	13.7	4.3
7_30_1	12.2	3.4
7_10_10_1	11.9	3.0
7_20_1	11.2	2.6
7_1	6.8	-1.8

dens.fi75	%sigma	%merr
7_10_1	14.0	9.6
7_20_1	13.2	8.2
7_1	12.8	6.8
7_30_1	12.7	9.6
7_10_10_1	11.9	10.3

dens.la75	%sigma	%merr
7_10_1	17.7	6.0
7_20_1	17.2	2.9
7_1	16.9	-2.0
7_30_1	15.7	5.8
7_10_10_1	9.2	6.8

Train set

dens.rn75	%sigma	%merr	Kepo
7_1	12.3	-12.3	12.3
7_10_10_1	10.4	-1.9	11.6
7_30_1	10.1	-0.7	11.4
7_10_1	9.9	-1.5	11.5
7_20_1	9.8	-0.4	11.3

dens.sp31	%sigma	%merr	Kepo
7_1	10.4	-10.1	11.6
7_10_10_1	8.9	-1.2	11.3
7_30_1	8.5	-1.0	11.4
7_20_1	8.4	-1.1	11.4
7_10_1	8.3	-0.9	11.3

elong.fi75	%sigma	%merr	Kepo
7_1	11.2	-3.3	11.4
7_20_1	8.0	0.7	10.8
7_10_10_1	7.7	1.1	11.2
7_30_1	7.5	1.2	11.1
7_10_1	7.4	1.2	11.2

elong.la75	%sigma	%merr	Kepo
7_1	12.5	-5.2	12.0
7_30_1	8.7	-2.4	11.2
7_20_1	8.7	-2.1	11.2
7_10_1	8.7	-1.8	11.1
7_10_10_1	8.5	-1.0	11.3

elong.rn75	%sigma	%merr	Kepo
7_1	13.4	-19.1	12.0
7_10_10_1	8.6	-2.6	11.6
7_30_1	8.4	-3.4	11.6
7_10_1	7.1	-3.0	11.3
7_20_1	7.1	-2.2	11.3

elong.sp31	%sigma	%merr	Kepo
7_1	13.3	-20.3	11.9
7_10_10_1	8.9	-2.7	11.3
7_10_1	8.2	-2.0	10.8
7_20_1	8.0	-2.0	10.9
7_30_1	7.6	-2.3	11.0

tear.fi75	%sigma	%merr	Kepo
7_1	13.2	4.9	11.4
7_10_1	10.7	0.6	11.6
7_10_10_1	10.6	1.9	11.3
7_30_1	9.9	2.0	11.9
7_20_1	9.8	1.9	11.5

Test set

dens.rn75	%sigma	%merr
7_1	10.6	-14.5
7_20_1	9.9	-2.8
7_10_10_1	9.7	-5.1
7_30_1	9.6	-3.3
7_10_1	9.5	-4.5

dens.sp31	%sigma	%merr
7_1	14.8	-7.0
7_10_1	14.5	2.0
7_30_1	14.3	1.8
7_10_10_1	14.3	1.4
7_20_1	14.3	1.9

elong.fi75	%sigma	%merr
7_10_10_1	9.4	7.6
7_1	8.3	5.9
7_10_1	7.9	7.8
7_30_1	7.3	7.7
7_20_1	7.1	8.2

elong.la75	%sigma	%merr
7_1	13.4	-1.1
7_10_10_1	12.3	-2.8
7_30_1	11.7	-4.5
7_20_1	11.3	-4.5
7_10_1	11.2	-3.8

elong.rn75	%sigma	%merr
7_1	12.0	-20.6
7_20_1	10.8	-3.7
7_10_10_1	10.6	-2.9
7_30_1	10.6	-4.0
7_10_1	10.5	-4.6

elong.sp31	%sigma	%merr
7_1	11.9	-20.3
7_10_1	9.6	-2.5
7_20_1	9.3	-2.6
7_10_10_1	9.0	-3.0
7_30_1	8.4	-2.6

tear.fi75	%sigma	%merr
7_30_1	16.0	4.8
7_10_10_1	15.3	1.5
7_20_1	14.2	4.3
7_10_1	12.7	3.8
7_1	11.9	7.5

Train set

tear.la75	%sigma	%merr	Kepo
7_1	12.3	-9.7	11.7
7_20_1	10.4	-1.1	11.6
7_10_1	10.4	-1.4	11.2
7_10_10_1	10.0	-5.4	11.8
7_30_1	9.9	-0.8	11.8

tear.rn75	%sigma	%merr	Kepo
7_1	13.1	-4.7	12.2
7_10_1	10.6	-2.6	11.9
7_20_1	10.5	-4.2	11.8
7_30_1	10.3	-1.0	11.7
7_10_10_1	9.5	-6.0	12.0

tear.sp31	%sigma	%merr	Kepo
7_1	13.1	-6.0	12.5
7_20_1	10.2	-0.7	11.9
7_10_1	9.9	-1.4	11.4
7_30_1	9.2	-0.5	11.1
7_10_10_1	8.2	-1.3	11.3

tensile.fi75	%sigma	%merr	Kepo
7_1	9.3	0.5	10.0
7_30_1	8.1	-0.4	10.9
7_10_1	8.0	0.1	10.7
7_20_1	7.9	1.9	10.6
7_10_10_1	7.7	0.3	10.8

tensile.la75	%sigma	%merr	Kepo
7_1	14.0	12.3	12.0
7_10_10_1	12.3	2.3	12.2
7_30_1	10.8	1.4	11.7
7_20_1	10.7	1.4	11.8
7_10_1	10.5	1.2	11.4

tensile.rn75	%sigma	%merr	Kepo
7_1	11.7	4.7	11.5
7_10_10_1	9.8	-0.2	11.0
7_30_1	9.6	-0.5	11.2
7_10_1	9.6	-1.1	11.3
7_20_1	9.4	-0.2	10.9

tensile.sp31	%sigma	%merr	Kepo
7_1	13.3	16.3	11.7
7_20_1	10.1	2.8	11.8
7_10_1	10.1	2.9	11.7
7_30_1	9.9	2.5	11.6
7_10_10_1	9.8	3.3	11.3

Test set

tear.la75	%sigma	%merr
7_20_1	34.8	-31.6
7_30_1	33.7	-26.4
7_10_1	30.4	-30.6
7_1	17.6	-16.0
7_10_10_1	16.5	-24.2

tear.rn75	%sigma	%merr
7_20_1	12.9	-5.1
7_30_1	12.7	-1.5
7_10_1	12.5	-4.3
7_10_10_1	11.7	-6.8
7_1	11.7	-7.6

tear.sp31	%sigma	%merr
7_20_1	14.7	-0.2
7_10_1	14.4	-0.0
7_30_1	14.3	0.6
7_10_10_1	13.6	1.1
7_1	12.4	-5.1

tensile.fi75	%sigma	%merr
7_10_10_1	15.9	-4.9
7_20_1	15.9	-5.5
7_10_1	15.8	-7.2
7_1	15.6	-5.6
7_30_1	15.3	-6.9

tensile.la75	%sigma	%merr
7_20_1	22.3	-5.7
7_30_1	21.4	-4.6
7_1	15.0	14.4
7_10_1	13.1	5.0
7_10_10_1	10.8	5.2

tensile.rn75	%sigma	%merr
7_1	13.6	7.9
7_20_1	13.4	3.9
7_30_1	13.4	3.2
7_10_1	13.1	2.7
7_10_10_1	12.6	3.5

tensile.sp31	%sigma	%merr
7_1	13.3	12.7
7_10_10_1	12.9	-0.9
7_10_1	12.9	-1.6
7_30_1	12.7	-1.9
7_20_1	12.6	-1.5

Appendix C : Find Significant Inputs

Train set

csf	%sigma	%merr	Kepo
par5	13.3	-0.5	4.7
par3	12.3	-4.7	4.7
par4	11.6	-1.9	4.7
tensile	9.3	-2.0	4.7
par8	8.6	-0.9	4.7
density	6.7	-0.3	4.7
par1	6.7	-0.4	4.7
par7	6.4	0.2	4.7
tear	6.2	-1.0	4.7
elongation	6.2	-1.2	4.7
par9	6.0	-0.8	4.7
—	5.9	-0.5	4.7

par1	%sigma	%merr	Kepo
par4	41.2	21.8	9.4
tear	38.9	2.9	9.4
tensile	38.1	7.0	9.4
par3	33.9	47.2	9.4
par7	32.2	7.4	9.4
density	31.2	-1.1	9.4
csf	30.2	-0.5	9.4
par8	28.3	-1.0	9.4
elongation	26.4	-2.3	9.4
par9	25.4	-0.3	9.4
par5	25.1	2.9	9.4
—	23.9	1.2	9.4

density	%sigma	%merr	Kepo
par3	17.2	-0.6	8.0
tensile	14.0	4.7	8.0
par4	11.5	-1.8	8.0
csf	9.1	-0.0	8.0
par8	8.9	0.2	8.0
par1	8.8	-1.6	8.0
par9	8.6	3.1	8.0
par7	8.5	-1.6	8.0
tear	8.2	1.7	8.0
elongation	8.1	3.2	8.0
par5	7.0	1.3	8.0
—	6.0	0.9	8.0

Test set

csf	%sigma	%merr
par3	18.1	-0.1
tensile	17.9	-1.1
par8	16.3	-2.1
par7	15.9	-1.2
density	15.7	-2.3
par9	15.7	-1.7
elongation	15.7	-2.0
par1	15.5	-2.1
—	15.3	-1.8
tear	14.9	-3.4
par4	12.8	-9.8
par5	10.6	1.8

par1	%sigma	%merr
density	40.8	1.4
par4	39.5	55.0
par9	38.9	8.3
elongation	38.6	5.6
—	37.4	5.0
par5	36.7	6.5
par8	35.3	11.9
tear	35.3	31.0
csf	34.9	7.1
tensile	32.4	23.5
par7	29.7	30.2
par3	29.3	50.5

density	%sigma	%merr
par3	17.3	0.8
tear	16.6	6.5
csf	15.9	8.6
par4	15.7	8.1
par7	15.4	9.7
par5	15.2	8.2
par9	14.9	8.9
—	14.7	9.5
elongation	14.6	10.0
par8	14.4	9.3
par1	14.3	7.3
tensile	12.6	14.1

Train set

elongation	%sigma	%merr	Kepo
par7	11.1	4.6	5.3
par8	10.2	0.6	5.3
par3	10.0	3.6	5.3
par4	9.6	2.9	5.3
tear	8.7	1.4	5.3
par5	8.7	1.7	5.3
tensile	8.6	1.1	5.3
par1	8.1	1.9	5.3
par9	7.7	1.5	5.3
density	7.6	1.7	5.3
csf	7.6	1.5	5.3
—	7.4	1.4	5.3

par3	%sigma	%merr	Kepo
par4	42.5	-13.7	1.5
tensile	20.1	5.0	1.5
par7	8.5	-6.8	1.5
par9	7.3	1.2	1.5
csf	7.0	0.2	1.5
tear	5.4	0.0	1.5
elongation	5.2	-0.6	1.5
par1	4.5	-0.7	1.5
par8	4.3	-0.2	1.5
density	4.1	-0.6	1.5
par5	3.8	-0.5	1.5
—	3.7	-0.3	1.5

par5	%sigma	%merr	Kepo
csf	4.1	0.5	2.2
par8	3.9	0.4	2.2
tear	3.2	0.9	2.2
par4	3.1	0.4	2.2
elongation	3.0	0.5	2.2
par3	3.0	0.5	2.2
tensile	2.9	0.5	2.2
par9	2.9	0.6	2.2
par1	2.8	0.5	2.2
—	2.8	0.6	2.2
density	2.8	0.5	2.2
par7	2.8	0.5	2.2

Test set

elongation	%sigma	%merr
par7	10.3	14.0
par8	9.3	7.9
par3	9.1	11.8
par9	9.0	5.9
tear	8.8	13.6
par4	8.6	12.8
par5	8.5	8.3
tensile	7.9	8.2
csf	7.8	9.2
—	7.8	9.3
density	7.7	9.7
par1	7.3	11.0

par3	%sigma	%merr
par4	27.6	16.0
tensile	14.3	8.8
csf	10.0	4.3
par9	9.5	2.6
par7	9.0	-0.9
tear	6.7	4.0
elongation	6.3	5.8
par8	4.9	5.1
par1	4.6	4.6
—	4.6	4.9
par5	4.3	5.0
density	3.9	4.6

par5	%sigma	%merr
par8	4.9	-1.4
par4	4.5	-0.9
tensile	4.0	-1.3
—	4.0	-1.4
par7	4.0	-1.3
par1	3.9	-1.4
elongation	3.9	-1.5
density	3.9	-1.5
par9	3.9	-1.5
tear	3.7	-2.3
par3	3.7	-1.7
csf	3.3	-1.4

Train set

par7	%sigma	%merr	Kepo
par4	49.0	37.4	8.7
par3	37.4	35.6	8.7
tensile	33.3	-5.8	8.7
tear	29.0	-5.5	8.7
par1	23.6	0.8	8.7
par8	23.6	-0.4	8.7
csf	22.2	2.5	8.7
density	21.8	0.4	8.7
elongation	20.8	-0.7	8.7
par9	20.4	4.9	8.7
par5	16.9	-0.4	8.7
—	15.0	-0.5	8.7

par8	%sigma	%merr	Kepo
par4	18.0	9.0	6.0
par3	16.8	-4.1	6.0
par5	14.0	-1.4	6.0
par9	11.0	-1.9	6.0
par7	10.7	1.2	6.0
elongation	9.9	0.1	6.0
csf	9.6	1.8	6.0
tensile	8.6	-0.4	6.0
tear	7.8	0.7	6.0
par1	7.7	3.1	6.0
density	6.4	0.6	6.0
—	5.3	1.1	6.0

par9	%sigma	%merr	Kepo
par4	6.4	-1.1	0.6
par3	4.2	0.4	0.6
par8	3.3	-0.2	0.6
tear	3.2	-0.4	0.6
tensile	3.1	-0.3	0.6
csf	3.1	-0.2	0.6
elongation	3.1	-0.2	0.6
par7	3.1	-0.1	0.6
density	3.1	-0.2	0.6
par1	3.0	-0.1	0.6
par5	3.0	-0.2	0.6
—	3.0	-0.2	0.6

Test set

par7	%sigma	%merr
tear	48.6	-53.8
par4	40.9	-22.0
elongation	37.5	-21.9
—	35.8	-21.7
par8	35.2	-17.8
par5	35.1	-18.7
density	32.6	-17.7
csf	31.9	-20.1
par1	31.4	-14.5
tensile	28.0	-8.5
par9	28.0	-7.0
par3	14.3	9.6

par8	%sigma	%merr
par4	14.5	-13.0
tear	12.4	-12.3
par9	12.0	-5.7
par7	12.0	-8.3
tensile	11.2	-8.5
par5	10.7	-5.1
par3	10.7	-3.9
—	9.7	-9.2
elongation	9.5	-9.4
par1	9.1	-7.3
density	8.5	-8.0
csf	7.8	-8.3

par9	%sigma	%merr
par4	6.6	6.4
tear	3.8	4.2
csf	3.6	3.6
tensile	3.5	3.7
density	3.3	3.5
par3	3.3	2.2
par1	3.3	3.4
par7	3.2	3.2
—	3.2	3.6
par5	3.2	3.7
elongation	3.1	3.6
par8	3.0	3.5

Train set

par4	%sigma	%merr	Kepo
par3	40.6	15.5	2.4
tensile	38.1	15.4	2.4
csf	6.6	0.5	2.4
par9	5.8	1.5	2.4
elongation	4.5	0.8	2.4
density	3.8	-0.3	2.4
tear	3.4	0.3	2.4
par7	3.4	-1.1	2.4
par8	3.4	-0.2	2.4
par5	3.0	0.1	2.4
par1	2.8	-0.1	2.4
—	2.8	-0.1	2.4

tear	%sigma	%merr	Kepo
par3	16.2	-0.3	6.6
par4	14.7	5.6	6.6
tensile	14.6	-1.0	6.6
par7	13.0	1.9	6.6
elongation	10.5	-2.6	6.6
csf	9.5	-0.9	6.6
par9	9.1	-3.4	6.6
par1	8.6	-4.1	6.6
par5	8.5	-2.1	6.6
density	8.5	0.3	6.6
par8	8.2	-1.2	6.6
—	7.6	-1.6	6.6

tensile	%sigma	%merr	Kepo
par3	19.2	-3.8	3.9
par4	18.5	-0.9	3.9
tear	10.1	-1.0	3.9
par8	8.2	-1.2	3.9
par7	7.9	-0.5	3.9
csf	7.9	-1.0	3.9
density	7.8	-0.6	3.9
elongation	7.6	-2.0	3.9
par1	7.5	-1.1	3.9
par5	7.4	-1.1	3.9
—	7.4	-1.0	3.9
par9	7.3	-1.2	3.9

Test set

par4	%sigma	%merr
tensile	31.2	15.3
par3	21.5	4.3
par9	17.3	5.2
tear	14.9	5.3
csf	13.7	4.9
elongation	13.7	4.3
par8	11.4	3.8
par1	10.4	2.8
—	10.3	2.9
par5	10.0	2.6
par7	7.4	2.7
density	6.6	1.7

tear	%sigma	%merr
par3	22.3	6.1
par9	20.0	0.7
elongation	19.2	-1.0
par4	17.7	-0.9
par8	17.4	-1.1
par5	17.3	1.3
tensile	17.2	4.3
—	17.1	-0.3
par1	16.5	-1.9
csf	16.5	-0.3
par7	16.2	6.2
density	16.2	0.1

tensile	%sigma	%merr
par4	20.7	-19.2
csf	16.1	-7.6
elongation	15.7	-7.8
par9	15.4	-7.5
par5	15.4	-7.1
par1	15.3	-6.9
par7	15.3	-5.1
par8	15.2	-7.5
—	15.2	-7.5
density	15.1	-7.1
tear	14.2	-12.7
par3	13.8	1.3

Appendix D : Non Significant Inputs Removed

Train set

csf.fi75	%sigma	%merr	Kepo
8_1	15.9	19.8	23.6
8_10_1	7.9	2.4	22.1

par1.fi75	%sigma	%merr	Kepo
4_30_1	34.5	0.4	13.1

dens.fi75	%sigma	%merr	Kepo
7_10_10_1	8.9	-1.2	22.4
7_30_1	8.9	-0.4	22.3

par3.fi75	%sigma	%merr	Kepo
8_30_1	5.3	-2.2	0.9

par5.fi75	%sigma	%merr	Kepo
3_30_1	4.1	-0.8	15.9

par7.fi75	%sigma	%merr	Kepo
3_30_1	32.9	6.3	24.3

par8.fi75	%sigma	%merr	Kepo
7_30_1	9.2	1.1	21.1

par9.fi75	%sigma	%merr	Kepo
8_30_1	3.2	-1.3	0.7

par4.fi75	%sigma	%merr	Kepo
8_30_1	2.5	-0.0	5.4

elong.fi75	%sigma	%merr	Kepo
9_20_1	6.1	-0.1	21.4
9_30_1	6.1	1.3	21.7

tear.fi75	%sigma	%merr	Kepo
7_1	12.6	6.6	11.3
7_30_1	10.1	2.4	11.3
7_10_10_1	9.4	0.9	11.4

tensile.fi75	%sigma	%merr	Kepo
8_1	12.8	-7.9	12.1
8_10_10_1	8.5	1.3	11.5
8_30_1	7.2	-0.1	10.9

Test set

csf.fi75	%sigma	%merr
8_1	13.9	23.0
8_10_1	9.0	7.9

par1.fi75	%sigma	%merr
4_30_1	25.3	18.0

dens.fi75	%sigma	%merr
7_30_1	12.3	5.1
7_10_10_1	11.9	5.8

par3.fi75	%sigma	%merr
8_30_1	3.4	3.0

par5.fi75	%sigma	%merr
3_30_1	3.9	-2.8

par7.fi75	%sigma	%merr
3_30_1	18.9	-16.4

par8.fi75	%sigma	%merr
7_30_1	17.0	-10.9

par9.fi75	%sigma	%merr
8_30_1	3.1	2.2

par4.fi75	%sigma	%merr
8_30_1	8.8	1.3

elong.fi75	%sigma	%merr
9_30_1	11.8	10.7
9_20_1	9.0	10.7

tear.fi75	%sigma	%merr
7_30_1	20.3	-0.2
7_10_10_1	18.9	-0.9
7_1	15.3	7.1

tensile.fi75	%sigma	%merr
8_1	17.0	-4.8
8_30_1	14.5	-6.1
8_10_10_1	14.4	-4.4

Appendix E : Non Significant and Lab Data Inputs Removed

Train set

csf	%sigma	%merr	Kepo
par5	12.3	-0.2	7.8
par3	9.7	1.6	7.8
par8	8.9	0.2	7.8
par7	8.4	2.3	7.8
par1	7.6	1.4	7.8
par4	7.1	-1.2	7.8
par9	6.8	0.3	7.8
—	6.6	0.1	7.8

par1	%sigma	%merr	Kepo
par4	51.8	-22.4	13.1
par3	36.1	10.2	13.1
par7	34.5	5.4	13.1
par8	33.2	-3.9	13.1
par9	33.1	-1.2	13.1
par5	32.1	-3.0	13.1
—	31.3	-3.6	13.1

density	%sigma	%merr	Kepo
par9	12.0	3.2	11.4
par4	11.5	8.4	11.4
par1	10.7	0.6	11.4
par7	9.8	5.8	11.4
par8	9.0	0.0	11.4
par3	8.5	0.7	11.4
par5	8.4	0.3	11.4
—	7.5	-0.1	11.4

elongation	%sigma	%merr	Kepo
par3	37.9	-13.9	12.7
par4	23.2	1.0	12.7
par7	22.0	-5.6	12.7
par8	10.4	-1.6	12.7
par1	8.3	-2.1	12.7
par9	8.2	-0.1	12.7
par5	8.2	-0.6	12.7
—	7.1	-0.5	12.7

par3	%sigma	%merr	Kepo
par7	44.6	-37.2	5.7
par4	40.2	-16.2	5.7
par9	17.6	6.0	5.7
par1	5.8	-1.1	5.7
—	5.6	-0.5	5.7
par8	5.4	-0.5	5.7

Test set

csf	%sigma	%merr
par1	14.9	0.1
par3	14.4	0.2
par4	13.9	-0.8
par9	13.6	-0.7
par8	13.6	-0.3
—	13.5	-0.5
par7	13.4	0.4
par5	11.5	2.5

par1	%sigma	%merr
par4	33.7	9.2
—	31.8	11.8
par9	31.0	10.2
par3	30.0	21.8
par5	29.5	14.6
par7	28.4	18.9
par8	28.2	11.5

density	%sigma	%merr
par9	16.4	8.6
par7	15.8	17.0
par3	15.6	9.0
par5	15.5	9.0
—	15.1	10.5
par8	14.8	10.3
par4	12.7	16.1
par1	12.7	6.8

elongation	%sigma	%merr
par7	17.1	14.3
par3	16.1	20.2
par4	10.7	21.7
par1	8.6	5.9
par8	8.5	3.5
—	8.2	5.7
par9	8.0	5.0
par5	7.9	5.5

par3	%sigma	%merr
par4	35.0	16.3
par9	19.9	3.8
par7	18.3	-7.4
par8	4.8	2.2
—	4.7	2.1
par1	4.7	2.2

Train set

par5	%sigma	%merr	Kepo
par8	5.4	1.5	2.2
par3	4.4	1.6	2.2
par1	4.3	1.6	2.2
par4	4.3	1.2	2.2
par9	4.3	1.6	2.2
—	4.3	1.5	2.2

par7	%sigma	%merr	Kepo
par4	43.3	26.4	9.1
par3	41.0	20.0	9.1
par8	35.2	4.0	9.1
par1	34.5	1.4	9.1
par9	32.9	3.6	9.1
par5	32.1	1.2	9.1
—	32.1	0.9	9.1

par8	%sigma	%merr	Kepo
par4	18.0	9.2	8.3
par5	15.1	5.1	8.3
par3	14.4	3.4	8.3
par7	13.7	7.3	8.3
par9	13.4	4.5	8.3
par1	13.3	11.2	8.3
density	10.7	4.3	8.3
—	9.9	5.2	8.3

par9	%sigma	%merr	Kepo
par4	8.3	-1.6	1.0
par3	5.3	0.4	1.0
par8	3.3	-0.4	1.0
par1	3.1	-0.4	1.0
par7	3.1	-0.4	1.0
par5	3.0	-0.4	1.0
—	3.0	-0.5	1.0

par4	%sigma	%merr	Kepo
par3	38.0	0.7	11.8
par9	27.6	7.9	11.8
par7	26.5	-31.4	11.8
par8	7.7	0.0	11.8
par1	7.3	-1.0	11.8
par5	7.3	-0.1	11.8
—	6.8	0.6	11.8

Test set

par5	%sigma	%merr
par8	4.5	-0.8
par4	3.7	-0.2
—	3.0	-0.8
par3	2.9	-1.1
par1	2.8	-1.1
par9	2.5	-1.2

par7	%sigma	%merr
par5	12.9	-10.6
—	10.5	-9.9
par8	10.3	-9.9
par9	10.1	-9.3
par1	8.7	-9.6
par4	6.2	-7.4
par3	4.5	-8.2

par8	%sigma	%merr
par7	18.6	-3.5
par5	15.8	0.1
—	12.8	-2.6
par4	12.1	-4.0
density	10.5	-1.7
par3	9.5	0.3
par9	9.4	2.6
par1	6.6	3.1

par9	%sigma	%merr
par4	7.9	7.2
par3	3.7	1.4
par1	3.3	3.3
par5	3.2	3.4
—	3.2	3.3
par7	3.1	3.1
par8	2.9	3.4

par4	%sigma	%merr
par9	23.0	1.7
par7	22.8	-26.8
par3	19.0	-1.8
—	11.5	9.5
par5	10.9	9.9
par8	10.2	8.7
par1	6.8	5.4

Train set

tear	%sigma	%merr	Kepo
par4	27.7	4.1	15.2
par3	20.8	-8.7	15.2
par9	19.5	-5.6	15.2
par7	13.2	-4.0	15.2
par1	13.1	-4.8	15.2
par5	12.6	-1.2	15.2
par8	11.1	1.0	15.2
—	9.7	-0.1	15.2

tensile	%sigma	%merr	Kepo
par4	24.8	4.8	13.9
par3	22.6	-4.3	13.9
par1	10.6	3.6	13.9
par7	9.5	0.2	13.9
par8	9.3	0.2	13.9
—	8.1	-0.1	13.9

Test set

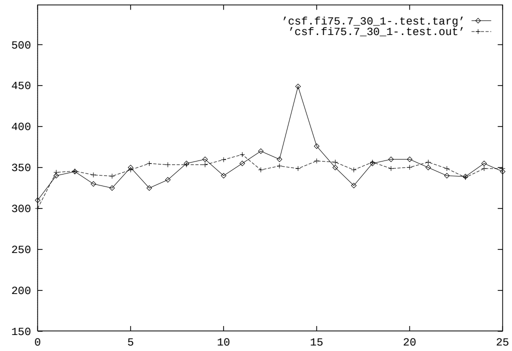
tear	%sigma	%merr
par3	30.2	1.8
par9	29.0	0.6
par4	21.2	-14.5
par7	20.3	5.1
par8	17.5	1.1
—	16.3	2.6
par1	15.5	-0.3
par5	13.4	7.9

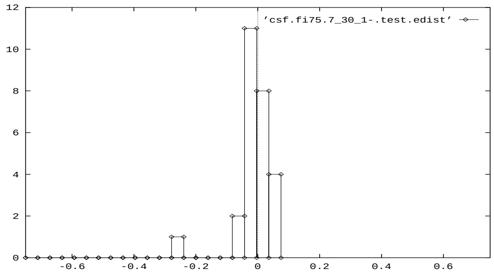
tensile	%sigma	%merr
par4	26.0	-17.4
par1	16.1	-6.1
—	15.5	-7.1
par7	15.4	-4.3
par8	15.1	-6.7
par3	13.6	3.9

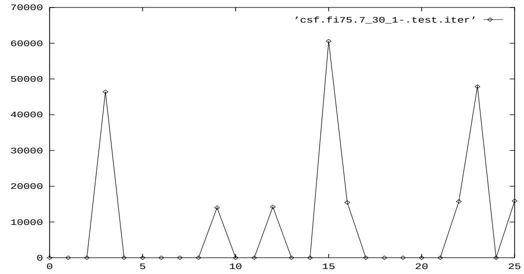
Appendix F : Continuous learning

CSF

csf			
arch	%sigma	%merr	maxerr
7_10_1	15.5	-0.8	-138.6
7_1	15.4	-3.1	-143.1
7_10_10_1	14.4	-0.5	-126.5
7_20_1	12.5	0.0	-104.9
7_30_1	**11.9**	**-0.8**	**-100.3**

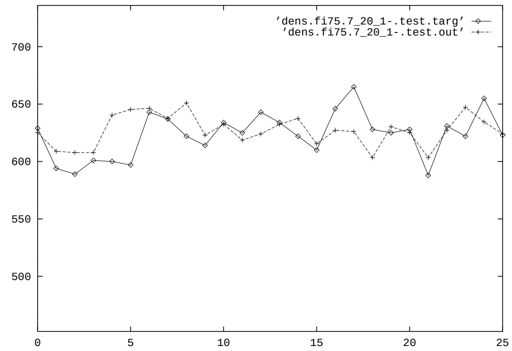


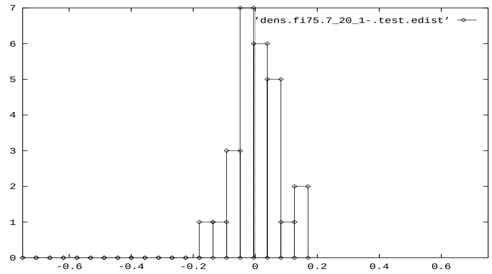


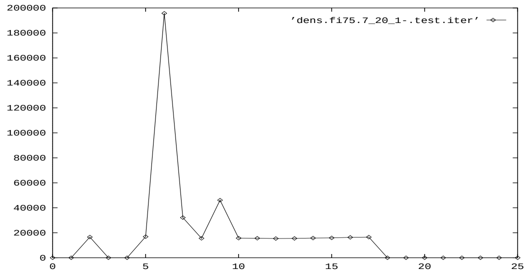


DENSITY

density			
arch	%sigma	%merr	maxerr
7_10_1	17.7	1.1	-57.6
7_10_10_1	17.1	7.1	71.2
7_30_1	14.5	4.8	52.2
7_1	14.4	7.0	50.3
7_20_1	**13.9**	**2.6**	**48.4**

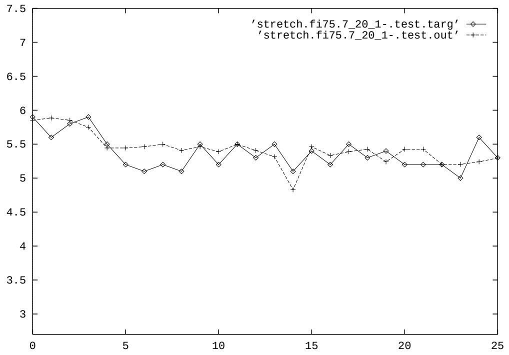




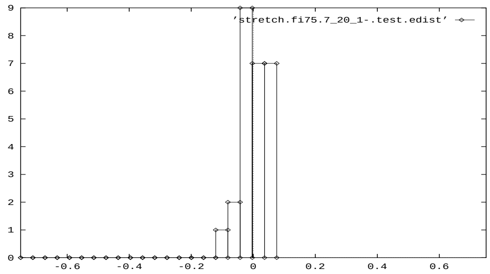


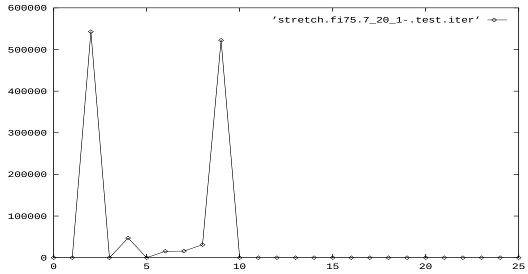
Upper Left: Performance for different architechtures. Upper Right: Best plot of target and prediction. Lower Left: Error distribution. Lower Right: Iterations to relearn.

ELONGATION

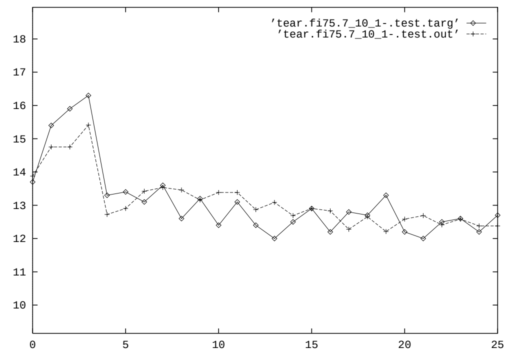


elongation			
arch	%sigma	%merr	maxerr
7_1	14.1	13.8	0.8
7_10_1	8.4	3.9	0.4
7_10_10_1	8.3	0.6	-0.3
7_30_1	8.2	5.9	0.4
7_20_1	**8.0**	**2.3**	**0.4**

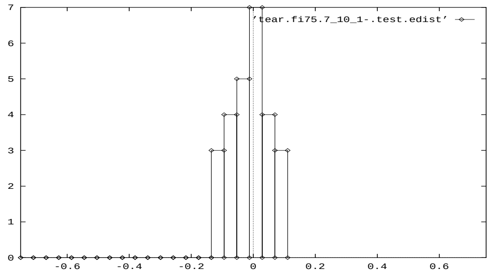


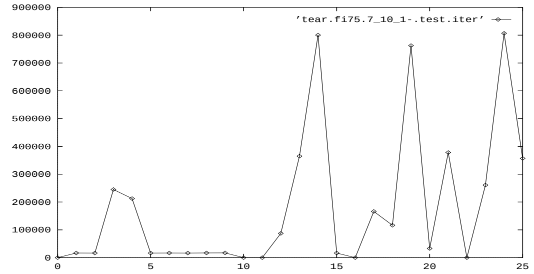


TEAR



tear			
arch	%sigma	%merr	maxerr
7_20_1	16.8	-1.7	-1.7
7_30_1	15.2	-0.7	-1.7
7_10_10_1	14.3	1.7	-1.5
7_10_1	**12.3**	**0.2**	**-1.1**
7_1	12.0	2.8	1.1

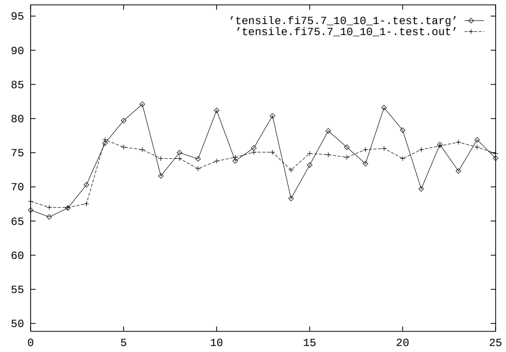


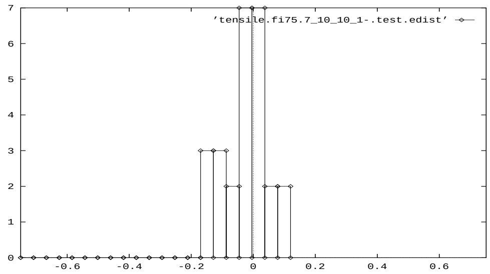


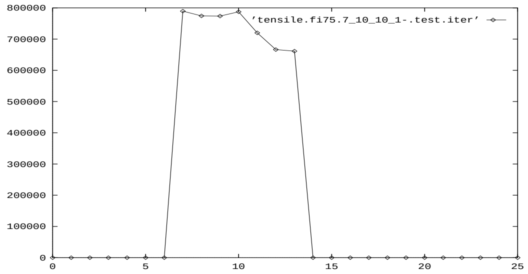
Upper Left: Performance for different architechtures. Upper Right: Best plot of target and prediction. Lower Left: Error distribution. Lower Right: Iterations to relearn.

TENSILE

tensile			
arch	%sigma	%merr	maxerr
7_1	26.1	-4.4	11.2
7_20_1	17.6	-3.6	12.8
7_30_1	15.9	-2.9	8.8
7_10_1	15.2	-4.0	8.1
7_10_10_1	**14.3**	**-3.3**	**-7.4**





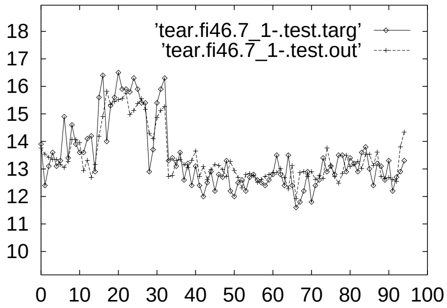


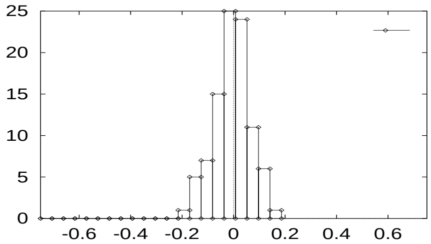
Upper Left: Performance for different architechtures. Upper Right: Best plot of target and prediction. Lower Left: Error distribution. Lower Right: Iterations to relearn.

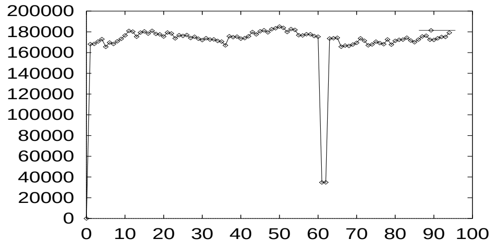
Appendix G : Continuous learning

TEAR 7_1

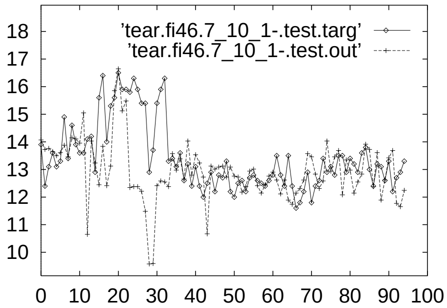
tear	$\%\sigma_{err}$	%merr	maxerr	$1-\frac{\sigma^2_{err}}{\sigma^2_{sig}}$
7_10_1	28.3	-9.9	-4.1	-0.31
7_30_1	19.1	-3.8	-3.7	0.40
7_10_10_1	18.3	-1.1	-2.1	0.45
7_20_1	18.2	-0.8	-3.8	0.46
7_1	14.1	0.2	-1.8	0.67

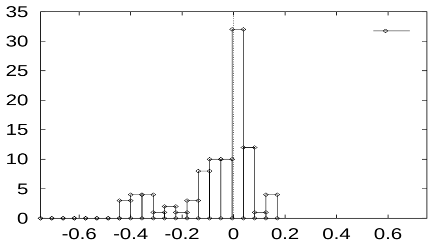


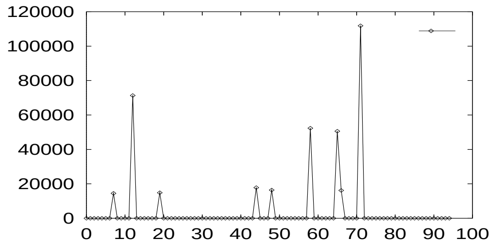




TEAR 7_10_1

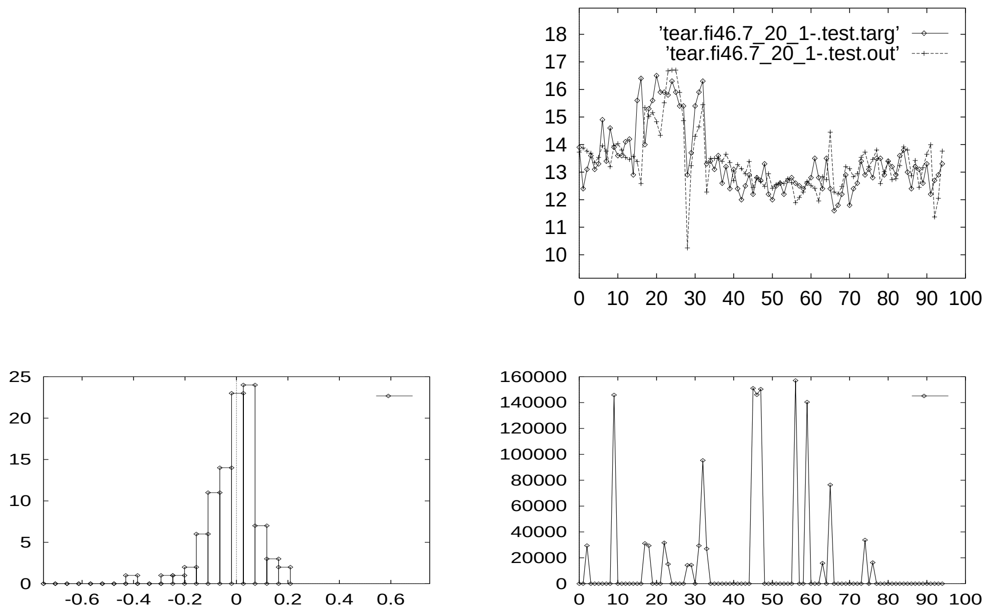




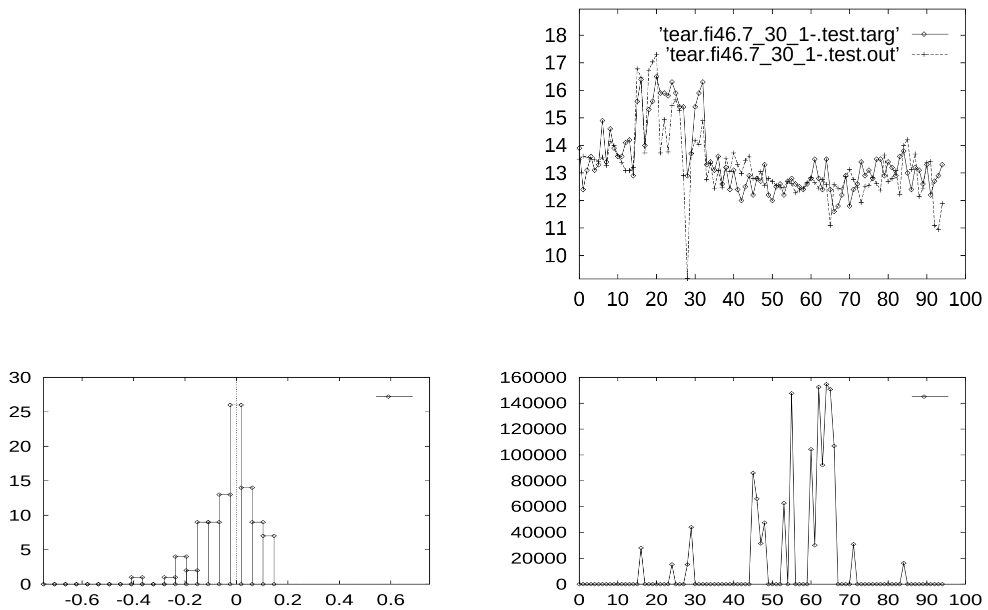


Upper Left: Performance table for *tear* on different architechtures. The table shows standard deviation for the error expressed as percentage of the range of *tear*. The same for the average error in percent. “maxerr” is the maximal error in engineering units. The measure “$1 - \frac{\sigma^2_{err}}{\sigma^2_{sig}}$” relates the distribution of the error with the distribution of the signal. Upper Right: Plot of target and prediction. Lower Left: Error distribution. Lower Right: Iterations to relearn.

TEAR 7_20_1

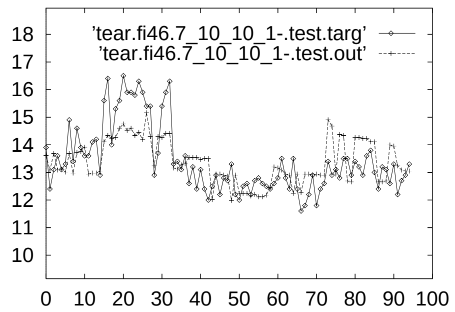


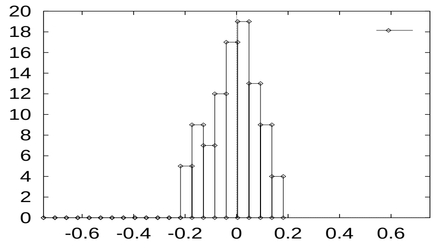
TEAR 7_30_1

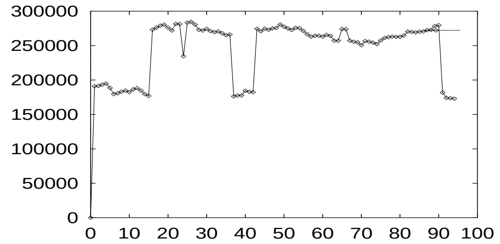


Upper Right: Plot of target and prediction. Lower Left: Error distribution. Lower Right: Iterations to relearn.

TEAR 7_10_10_1





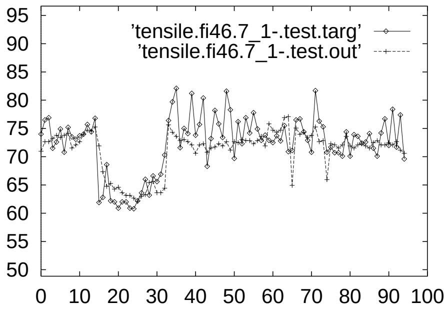


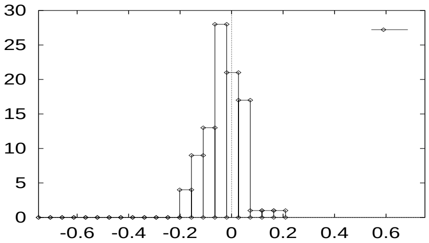
Upper Right: Plot of target and prediction. Lower Left: Error distribution. Lower Right: Iterations to relearn.

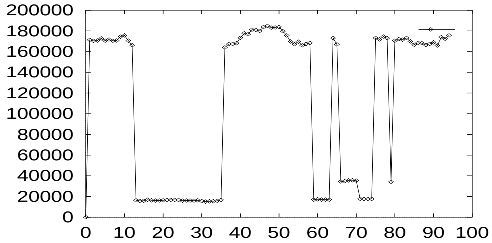
Appendix H : Continuous learning

TENSILE 7_1

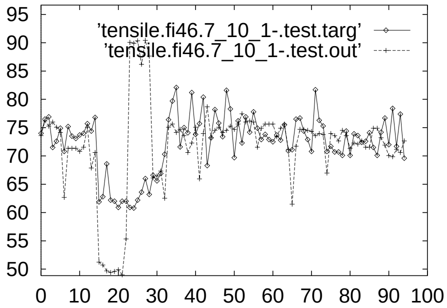
tensile	$\%\sigma_{err}$	%merr	maxerr	$1-\frac{\sigma^2_{err}}{\sigma^2_{sig}}$
7_10_1	35.6	-1.6	29.2	-1.77
7_20_1	31.4	5.6	32.4	-1.16
7_10_10_1	21.6	5.1	14.5	-0.02
7_30_1	19.8	-13.6	-18.0	0.145
7_1	14.2	-4.7	10.0	0.56

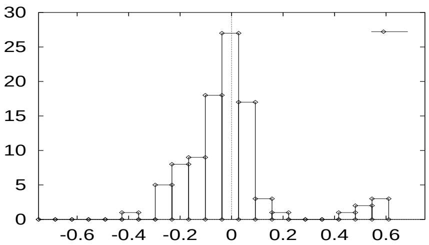


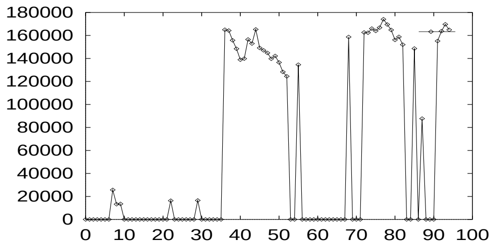




TENSILE 7_10_1

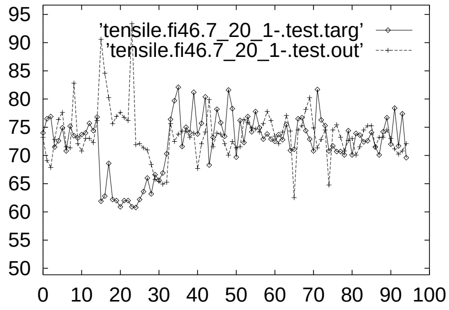


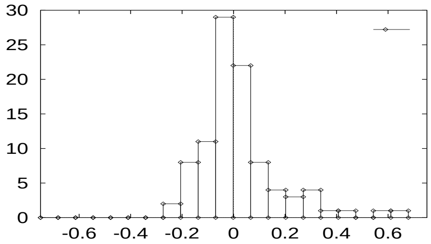


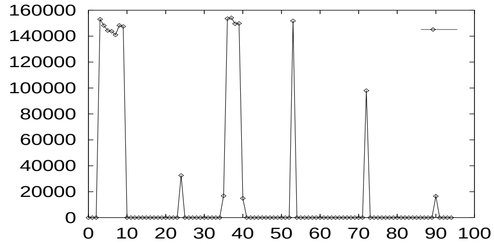


Upper Left: Performance table for *tensile* on different architechtures. The table shows standard deviation for the error expressed as percentage of the range of *tensile*. The same for the average error in percent. "maxerr" is the maximal error in engineering units. The measure "$1 - \frac{\sigma^2_{err}}{\sigma^2_{sig}}$" relates the distribution of the error with the distribution of the signal. Upper Right: Plot of target and prediction. Lower Left: Error distribution. Lower Right: Iterations to relearn.

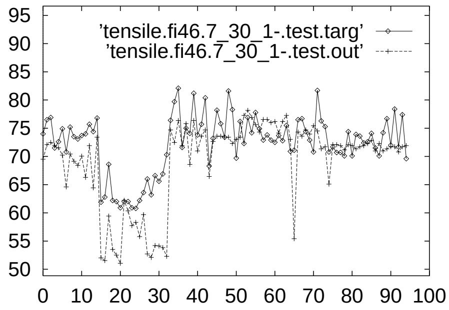
TENSILE 7_20_1

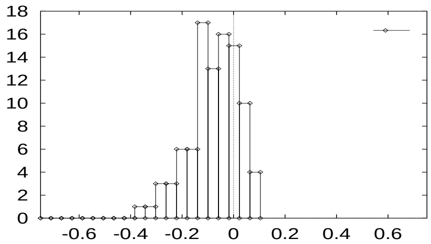


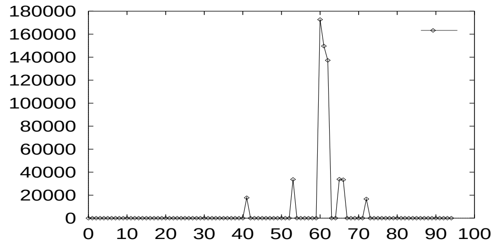




TENSILE 7_30_1

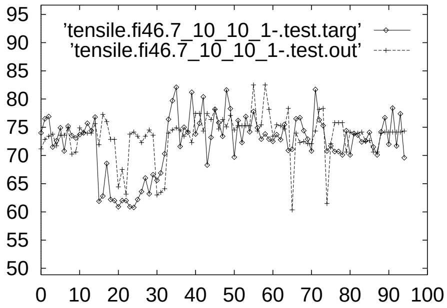


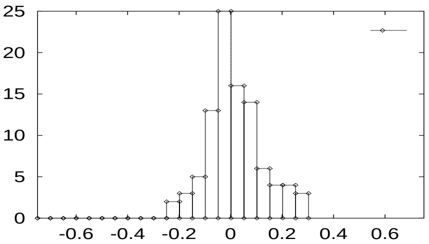


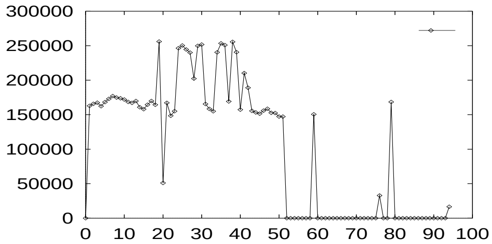


Upper Right: Plot of target and prediction. Lower Left: Error distribution. Lower Right: Iterations to relearn.

TENSILE 7_10_10_1





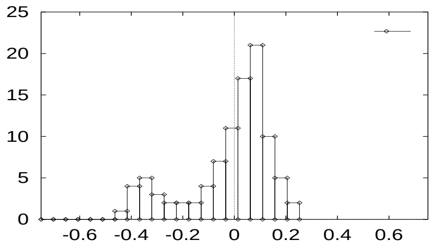


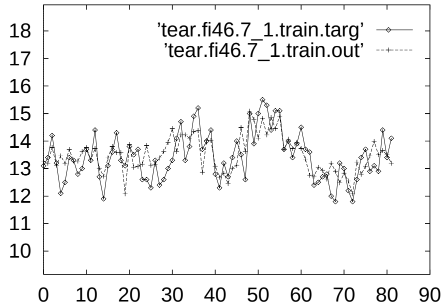
Upper Right: Plot of target and prediction. Lower Left: Error distribution. Lower Right: Iterations to relearn.

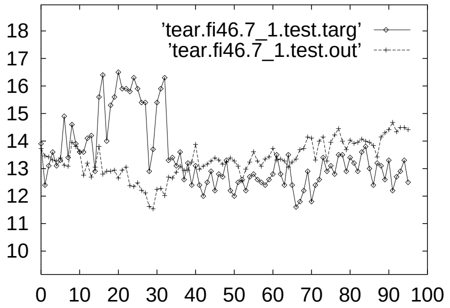
Appendix I : Fixed learning

tear.fi46	$\%\sigma_{err}$	%merr	maxerr	$1\text{-}\frac{\sigma^2_{err}}{\sigma^2_{sig}}$
7_10_10_1	39.7	-2.8	-4.9	-1.60
7_20_1	35.7	4.8	-4.2	-1.10
7_1	32.8	-1.6	-4.3	-0.77
7_10_1	31.5	1.6	-3.9	-0.64
7_40_1	29.6	8.3	3.5	-0.45
7_30_1	29.5	8.2	-3.6	-0.44

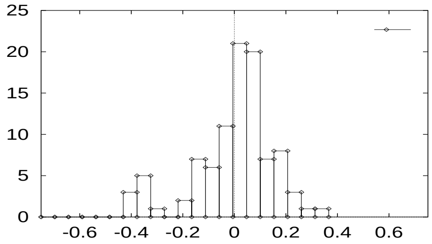
TEAR 7_1

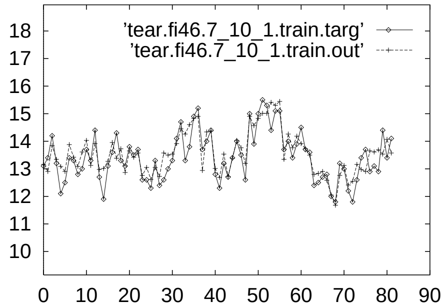


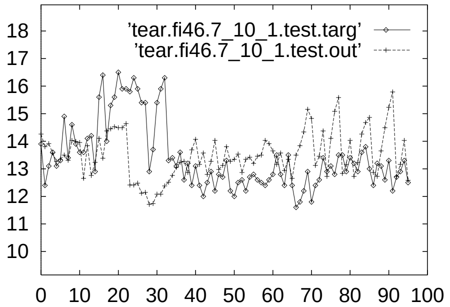




TEAR 7_10_1

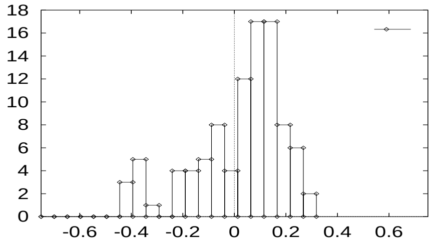


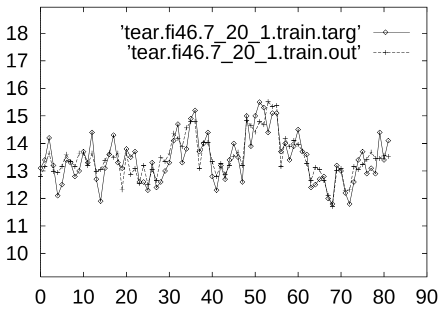


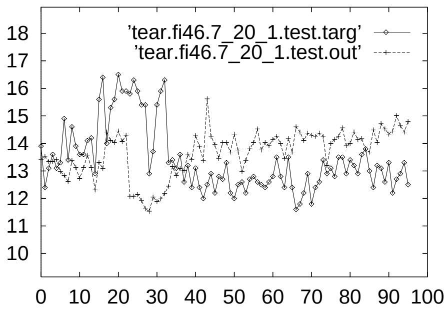


Upper Left: Performance table for *tear* on different architechtures. The table shows standard deviation for the error expressed as percentage of the range of *tear*. The same for the average error in percent. "maxerr" is the maximal error in engineering units. The measure "$1-\frac{\sigma^2_{err}}{\sigma^2_{sig}}$" relates the distribution of the error to the distribution of the signal. Upper Right: Error distribution for test set. Lower Left: Plot of train target and output. Lower Right: Plot of test target and output.

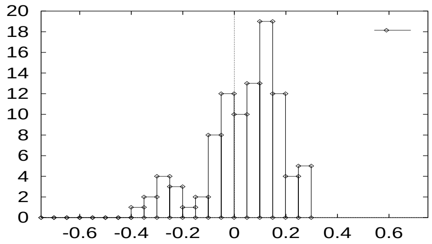
TEAR 7_20_1

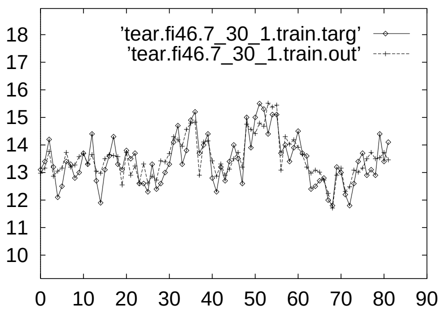


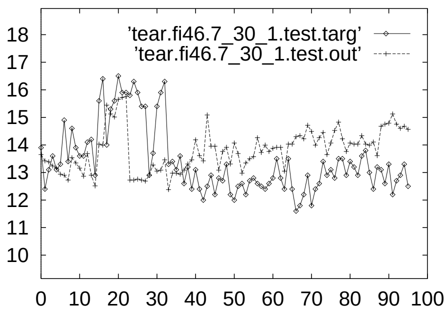




TEAR 7_30_1

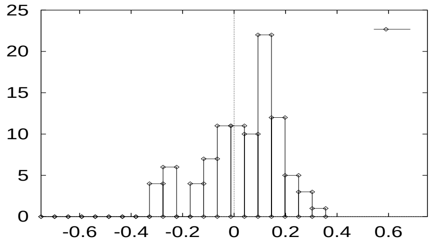


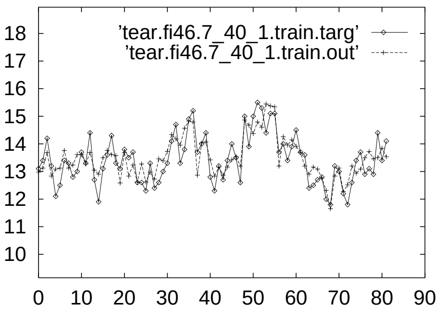


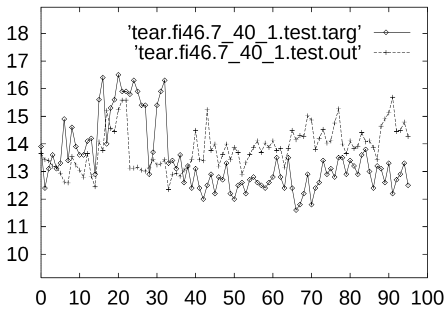


Upper Right: Error distribution for test set. Lower Left: Plot of train target and output. Lower Right: Plot of test target and output.

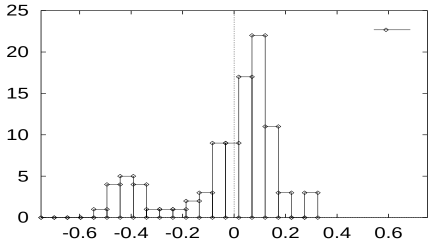
TEAR 7_40_1

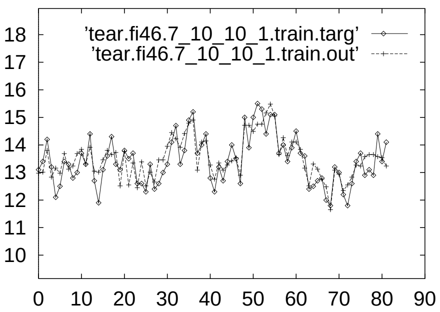


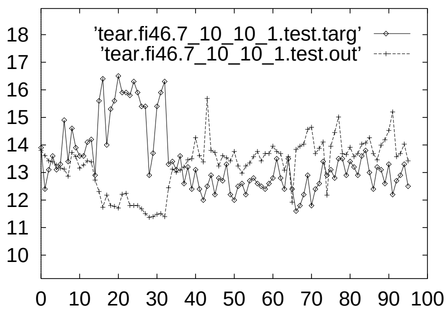




TEAR 7_10_10_1





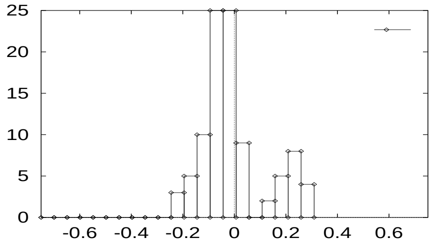


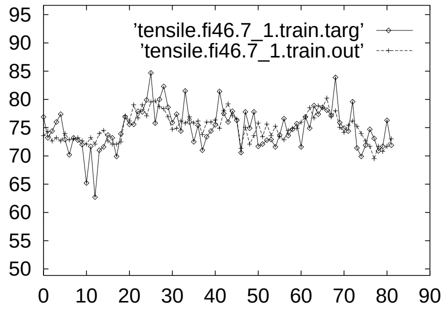
Upper Right: Error distribution for test set. Lower Left: Plot of train target and output. Lower Right: Plot of test target and output.

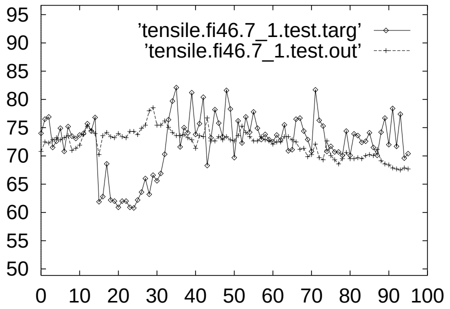
Appendix J : Fixed learning

tensile.fi46	$\%\sigma_{err}$	%merr	maxerr	$1-\frac{\sigma^2_{err}}{\sigma^2_{sig}}$
7_10_10_1	45.8	8.1	29.5	-3.63
7_20_1	29.2	2.5	26.0	-0.88
7_1	25.2	0.2	14.8	-0.41
7_10_1	24.6	-2.4	24.7	-0.33
7_30_1	24.4	-0.3	20.7	-0.32
7_40_1	21.8	-2.9	17.2	-0.05

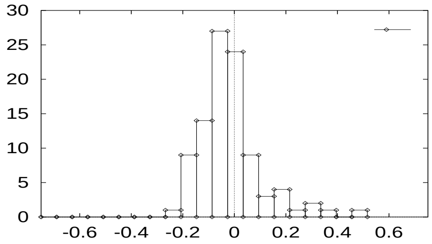
TENSILE 7_1

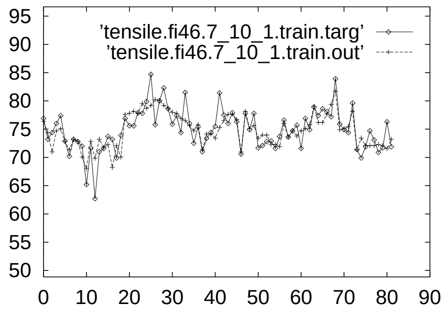


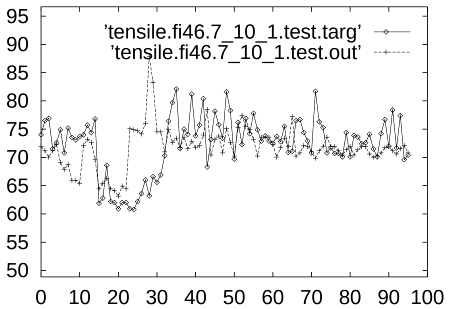




TENSILE 7_10_1

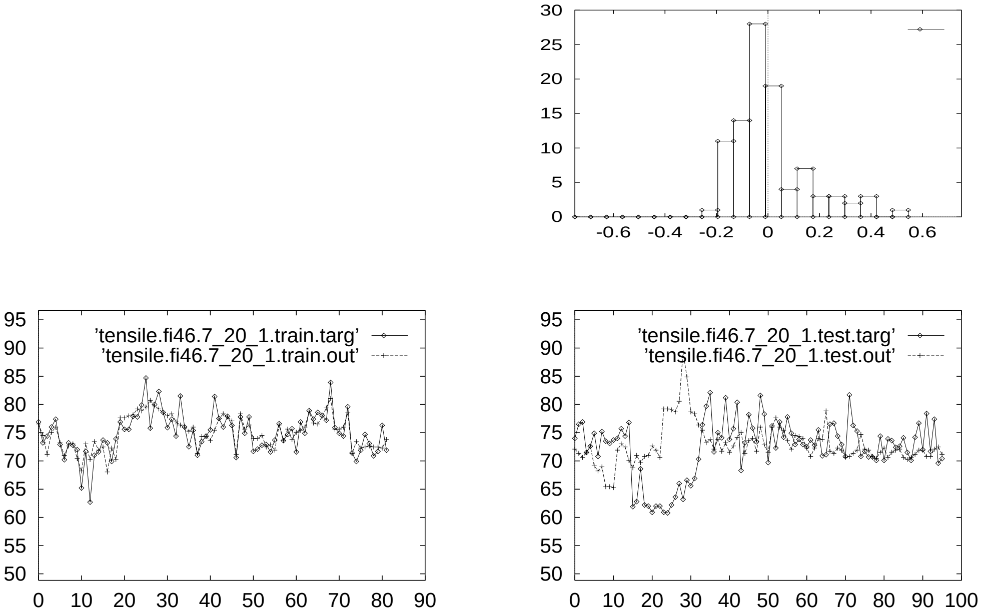




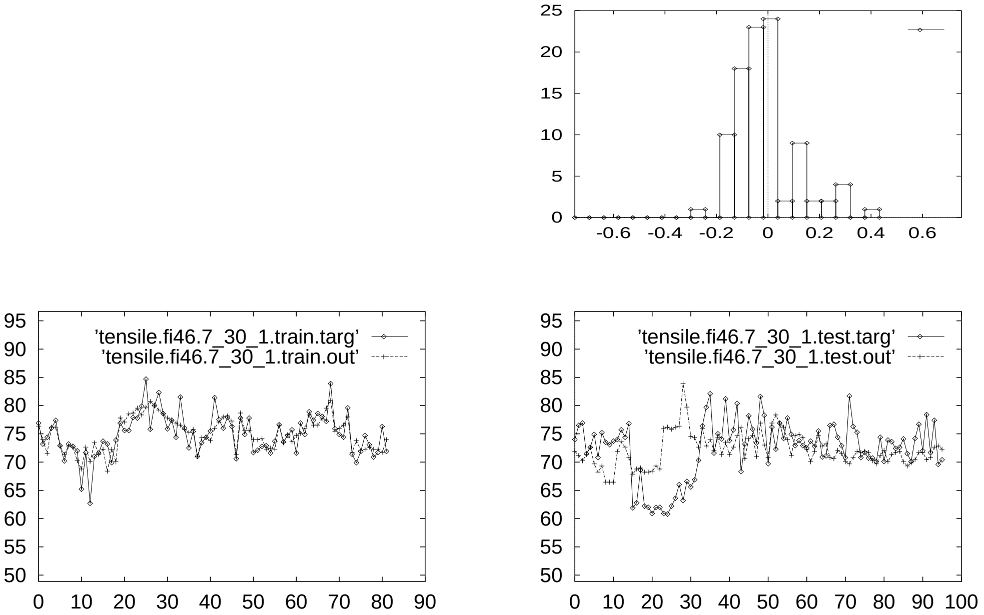


Upper Left: Performance table for *tensile* on different architechtures. The table shows standard deviation for the error expressed as percentage of the range of *tensile*. The same for the average error in percent. "maxerr" is the maximal error in engineering units. The measure "$1 - \frac{\sigma^2_{err}}{\sigma^2_{sig}}$" relates the distribution of the error to the distribution of the signal. Upper Right: Error distribution for test set. Lower Left: Plot of train target and output. Lower Right: Plot of test target and output.

TENSILE 7_20_1

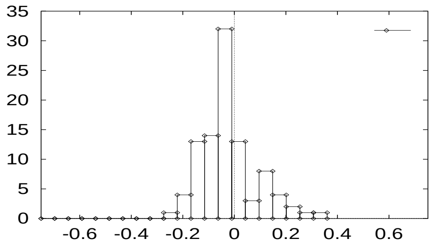


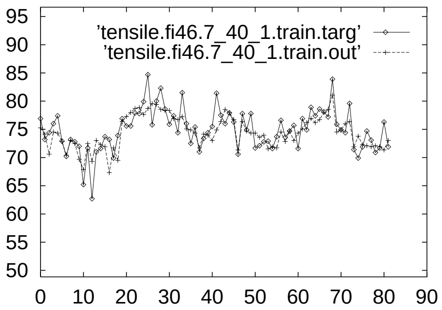
TENSILE 7_30_1

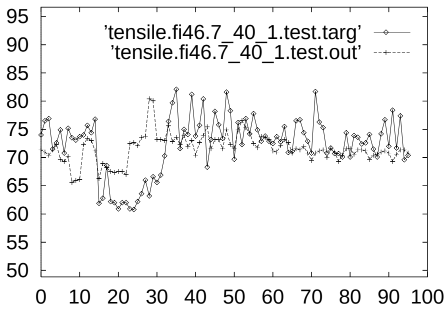


Upper Right: Error distribution for test set. Lower Left: Plot of train target and output. Lower Right: Plot of test target and output.

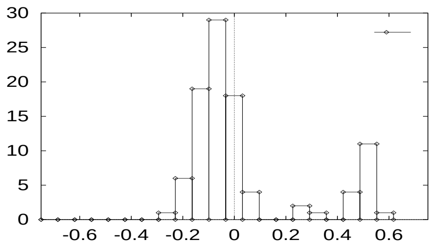
TENSILE 7_30_1

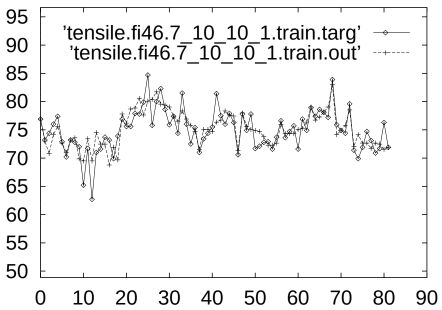


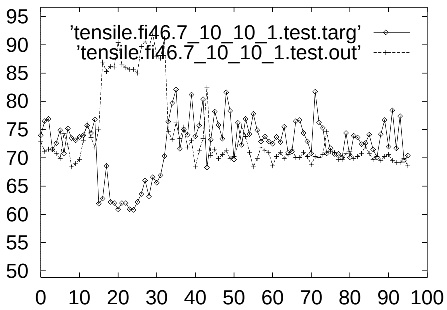




TENSILE 7_10_10_1







Upper Right: Error distribution for test set. Lower Left: Plot of train target and output. Lower Right: Plot of test target and output.

Appendix K : Input value statistics

	absrange	sigma%	sigma/mean%
tear	4.9	21.7	7.9
tensile	23.9	19.5	6.3
par1	6	31.4	76.2
par5	16.7	19.7	8.9
par7	30.9	39.7	57.0
par8	112	21.4	7.6
par9	24.0	19.3	7.9
par3	100	42.2	183.3
par4	100	47.6	72.4

This table shows some statistics for the training data. What is shown here is the rangesize *(absrange)*, i.e., the difference between min and max value of a signal; the standarddeviation in percent *(sigma%)* of the rangesize and the standarddeviation divided by the expectation value *(sigma/mean)*. The value sigma/mean is expressed in percent just to give nice numbers, *i.e.* , it is just multiplied with 100.