Article published in Journal of System Engineering (1996)6:128-136, Springer-Verlag London

address, email, tel and fax updated to more recent information.
old addr: SANS, Dept. of Numerical Analysis and Computing Science
old addr: Royal Institute of Technology, S–100 44 Stockholm, Sweden

Pulp Quality Modelling Using Bayesian Mixture Density Neural Networks

Roland Orre and Anders Lansner

NeuroLogic Sweden AB, AlbaNova University Center
SE-10691 Stockholm, Sweden
E–mail: orre@kth.se, Tel: +46–8–446 31 60
E–mail: roland.orre@neurologic.se, Tel,fax: +46–70–8269748, +46–8–55378214

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

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 feed-forward networks trained with the error backpropagation (BP) algorithm [Orre and Lansner, 1992]. 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 [Orre and Lansner, 1994].

Mixture density networks have been used for, *e.g.* classification of speech segments and satellite image pixels [Tråvén, 1993] and classification of globins in protein sequences [MacKay, 1994]. Function approximation has been done by, *e.g.* predicting the *a posteriori* density [Bishop, 1994] or using the EM-algorithm directly [Ghahramani, 1994]. 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.* [Holst and Lansner, 1993a] and [MacKay, 1994]. 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 [Tråvén, 1991] for the RBF-units and a BCPNN (Bayesian Confidence Propagation Neural Network), which have earlier been successfully used for pattern completion [Lansner and Örjan Ekeberg, 1989] and classification [Holst and Lansner, 1993b], to associate an explanatory conditioned density with a response density function.

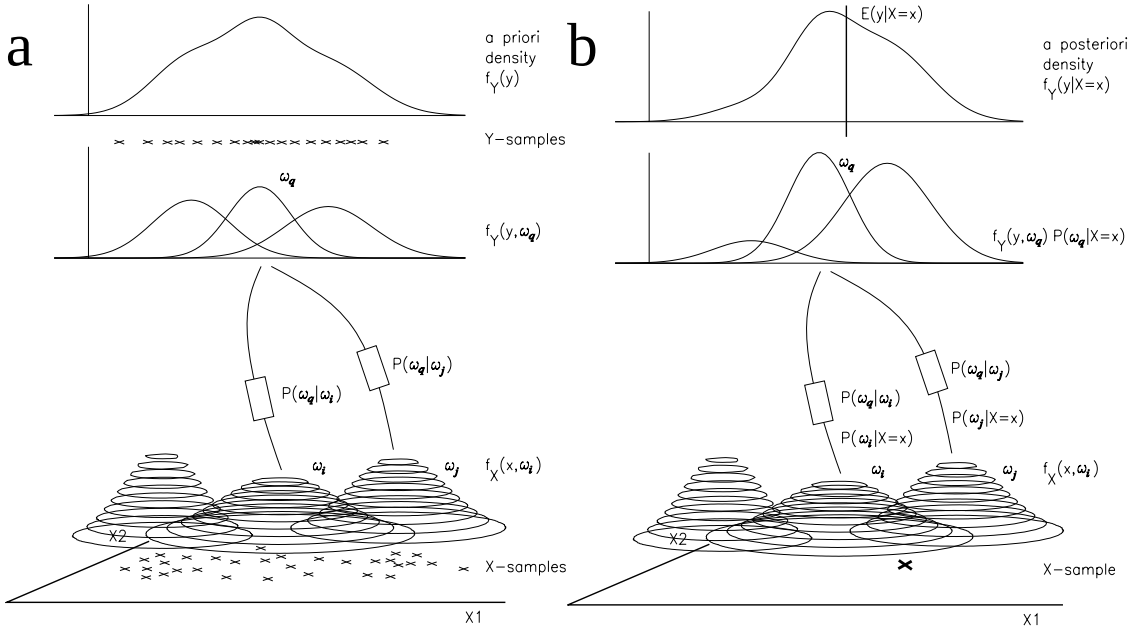

Figure 1: An overview of the density method. **a:** Training phase: We let a set of Gaussian density functions (ω_i) model the density of a set of data samples in explanatory (X) and response (Y) variable spaces. When an X-sample is drawn from the process ω_i in explanatory space there is a certain probability $P(\omega_q \mid \omega_i)$ 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 $X{=}x$.

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 15) 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.

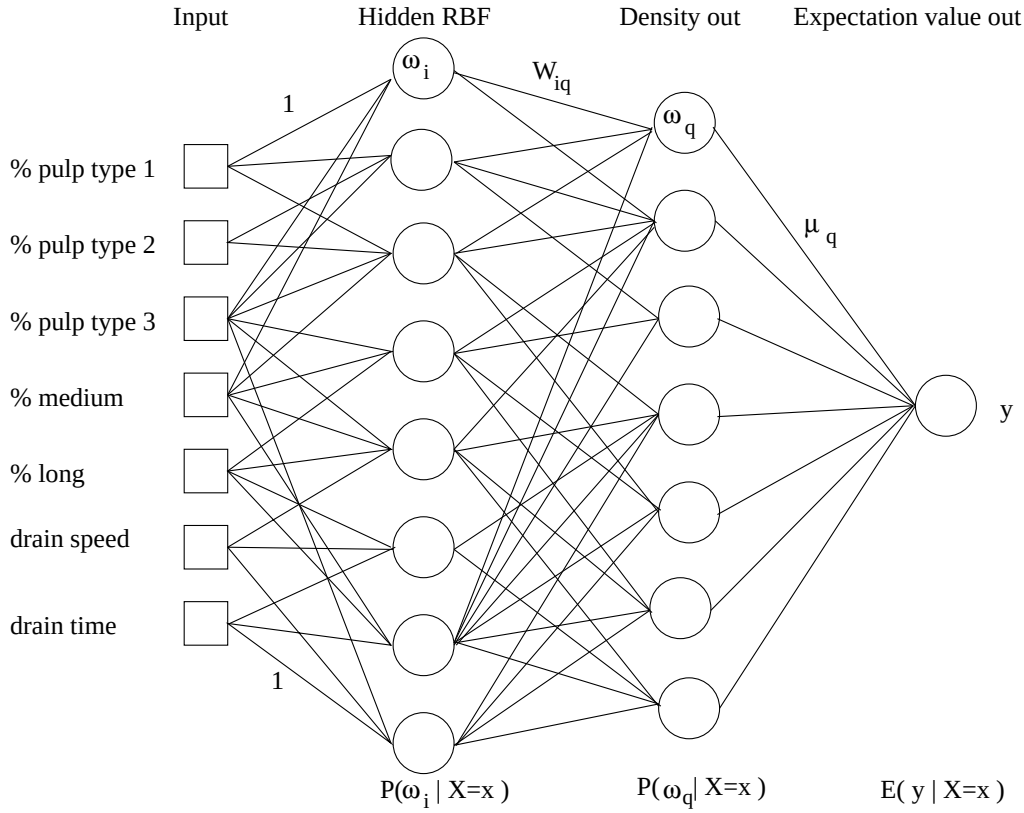

Figure 2: Setup of experiment. At left we input the process parameters, at right we output the response variable (e.g. y=tear or y=tensile) value. The input units just distribute the explanatory values to all hidden RBFs. Outputs from the hidden RBF units are $P(\omega_i|X = x)$, i.e. probability that 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	···	100
% pulp-type2	0	···	100
% pulp-type3	0	···	100
% middle	22.2	···	30.9
% long	51	···	75
drain-time	29.2	···	45.9
drain-speed	255	···	367
tear	11.6	···	16.5
tensile	60.8	···	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.

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).

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

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.

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

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)

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

A stochastic variant of the EM-algorithm [Tråvén, 1991], 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 (3)

$$\theta_N = \frac{\sum_{k=1}^{N} P(\omega|x_k)\theta(x_k)}{\sum_{k=1}^{N} P(\omega|x_k)}$$

$$\begin{aligned} \theta_{N+1} &= \frac{\sum_{k=1}^{N+1} P(\omega|x_k)\theta(x_k)}{\sum_{k=1}^{N+1} P(\omega|x_k)} \quad (3) \\ &= \frac{P(\omega|x_{N+1})\theta(x_{N+1}) + \sum_{k=1}^{N} P(\omega|x_k)\theta(x_k)}{\sum_{k=1}^{N+1} P(\omega|x_k)} \\ &= \frac{P(\omega|x_{N+1})\theta(x_{N+1}) + \sum_{k=1}^{N+1} P(\omega|x_k)\theta_N - P(\omega|x_{N+1})}{\sum_{k=1}^{N+1} P(\omega|x_k)} \end{aligned}$$

and simplified (4), 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.

$$\theta_{N+1} = \theta_N + \eta_{N+1}(\theta(x_{N+1}) - \theta_N) \quad , \quad \eta_{N+1} = \frac{P(\omega|x_{N+1})}{\sum_{k=1}^{N+1} P(\omega|x_k)} \, \delta \quad (4)$$

The denominator for η (4) 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 (5) and variance (7) we use an intermediate estimate of the next value. For the center value (5) this is just the next sample and for the variance (6) it is the squared distance over the number of dimensions.

$$\begin{aligned} \mu_{N+1} &= \mu_N + \eta_{N+1}(x_{N+1} - \mu_N) \quad (5) \\ \hat{\sigma}^2_{N+1} &= (x_{N+1} - \mu_N)^T (x_{N+1} - \mu_N)/d \quad (6) \\ \sigma^2_{N+1} &= \sigma^2_N + \eta_{N+1}(\hat{\sigma}^2_{N+1} - \sigma^2_N) \quad (7) \end{aligned}$$

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).

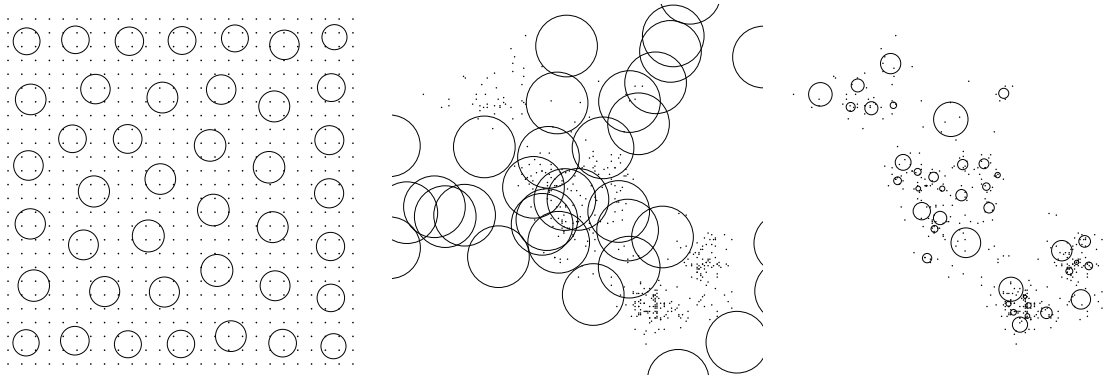

Figure 3: **Left:** 41 units are adapted to a square distribution. We see the RBF units with their σ plotted as circles and the data samples as dots. **Middle:** 40 units randomly initialized. **Right:** After about 200 iterations the units had become stable in this case.

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.

$$f_X(x) = \sum_{i=1}^{n} P(\omega_i) f(x|\omega_i), f_Y(y) = \sum_{q=1}^{m} P(\omega_q) f(y|\omega_q) \tag{8}$$

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 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 (9).

$$P(\omega_i|x) = \frac{P(\omega_i) f(x|\omega_i)}{f_X(x)} \propto P(\omega_i) f(x|\omega_i), \sum_{i=1}^{n} P(\omega_i|X{=}x) = 1 \tag{9}$$

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) (8). We start by rewriting the probability of a response variable component density (8) being conditioned by an explanatory variable value.

$$f_Y(y|X{=}x) = \sum_{q} f_Y(y|\omega_q) P(\omega_q|X{=}x) \tag{10}$$

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" (11) and thereafter Bayes rule (12). 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 (13).

$$P(\omega_q|X=x) = \sum_i P(\omega_q|\omega_i)P(\omega_i|X=x) \quad (11)$$

$$= \sum_i P(\omega_q)\frac{P(\omega_i|\omega_q)}{P(\omega_i)}P(\omega_i|X=x) \quad (12)$$

$$= P(\omega_q)\sum_i \frac{P(\omega_q \& \omega_i)}{P(\omega_q)P(\omega_i)}P(\omega_i|X=x) \quad (13)$$

We can now combine (10) and (13) to get the expression for $f_Y(y|X{=}x)$ (14). As was stated in (9) 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 $N(x, \mu_i, \sigma_i^2)$.

$$f_Y(y|X{=}x) = \sum_q f_{Y_q}(y|\omega_q)P(\omega_q)\sum_i \underbrace{\frac{P(\omega_q \& \omega_i)}{P(\omega_q)P(\omega_i)}}_{W_{iq}}P(\omega_i|X{=}x) \quad (14)$$

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 [Lansner and Örjan Ekeberg, 1989]. 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 (15) 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 (16). We also want some measure of the prediction *quality*. By integrating (17) the density function we may estimate a confidence interval (18) for the prediction.

$$E(y|X{=}x) = \int f_Y(y|X{=}x)\, y\, dy \quad (15) \qquad F_Y(\gamma) = \int_{-\infty}^{\gamma} f_Y(y|X{=}x)\, dy \quad (17)$$

$$E(y|X{=}x) = \sum_q P(\omega_q|X{=}x)\, \mu_q \quad (16)$$

$$\left.\begin{array}{rcl} 0.025 & < & F_Y(y_1) \\ 0.975 & > & F_Y(y_2) \end{array}\right\} \Rightarrow y_1 \leq Y_{95\%} \leq y_2 \quad (18)$$

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 $(0 \ldots 1)$. $\delta_i = y_{out} - y_{desired}$

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

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.

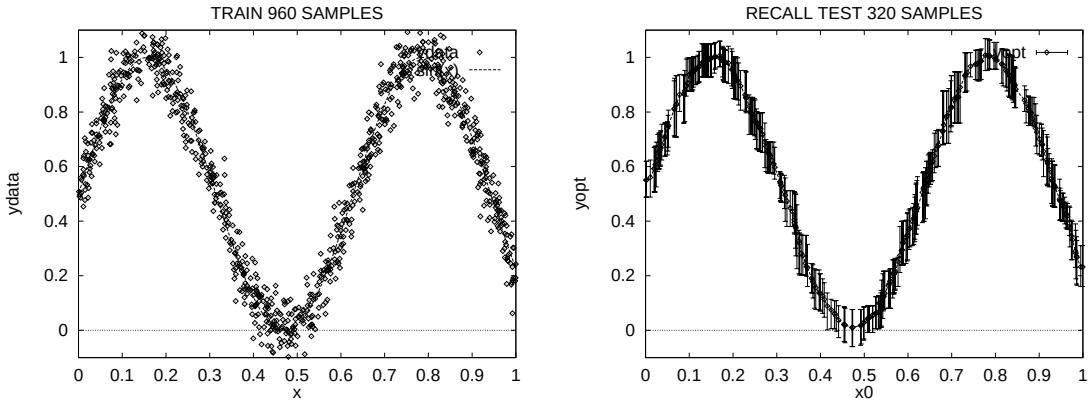

Figure 4: **Left:** Training data consisting of 960 samples with a normal distributed noise ($\sigma = 0.07$). **Right:** Recalled y-value from 320 samples with a network using 60 explanatory units and 40 response units. The error bars show one predicted standard deviation.

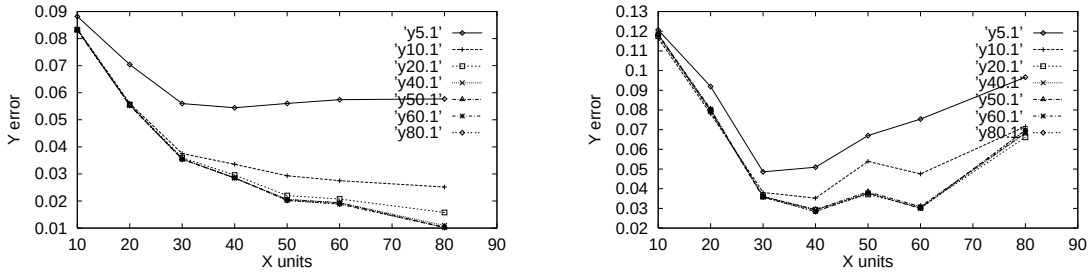

Figure 5: The performance for training set (left) and test set (right) when we vary the number of units in explanatory (X) and response (Y) layers. On the x-axis is the number of explanatory units. The Y units are shown by different curves in the same diagram.

This mixture density method is actually symmetrical for predictions $X \rightarrow Y$ and $Y \rightarrow X$, 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.

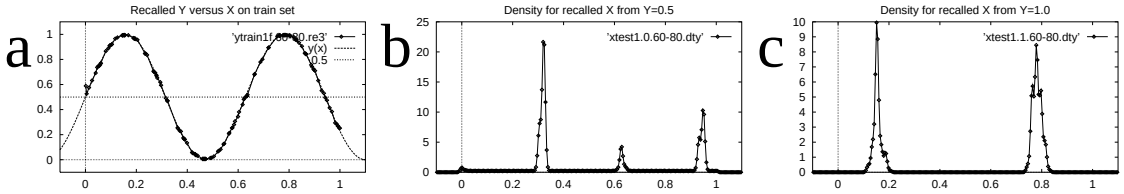

Figure 6: **a:** A recalled sine function as Y=f(X). **b:** Recalled density for X when input on Y is 0.5 (arcsin(Y)). **c:** Same as b when Y input is 1.

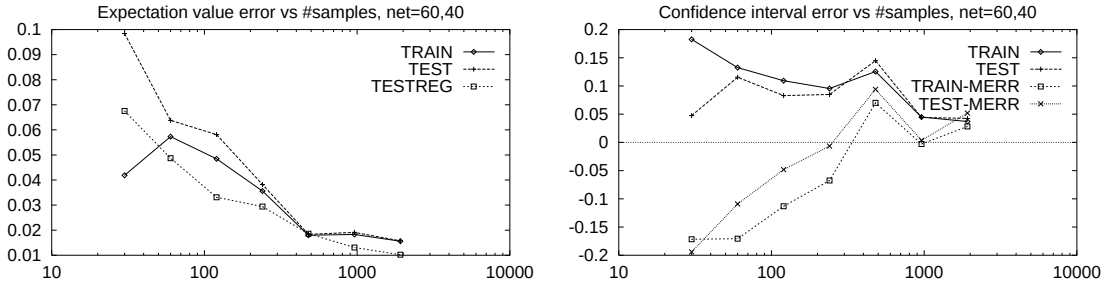

Figure 7: **Left:** The prediction performance of the expectation value improves for a specific network when the number of samples used for training increases. In this case we could perform better on the test set with regularization than on the training set without regularization most of the time. **Right:** 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.

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.

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 [Orre and Lansner, 1992]. 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 center, thus activating most units resulting in a predicted value close to the expectation value of the *a priori* density for the response variable.

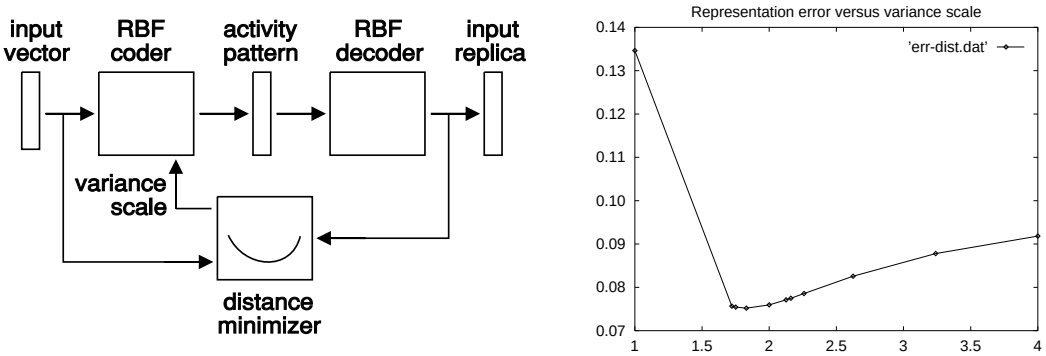

Figure 8: **Left:** Dynamic regularization by adapting the explanatory sample "fitness" by scaling the variances, thus improving the generalization capability. **Right:** Representation error vs. variance scaling (golden search used as distance minimizer).

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 [Orre and Lansner, 1992] (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.

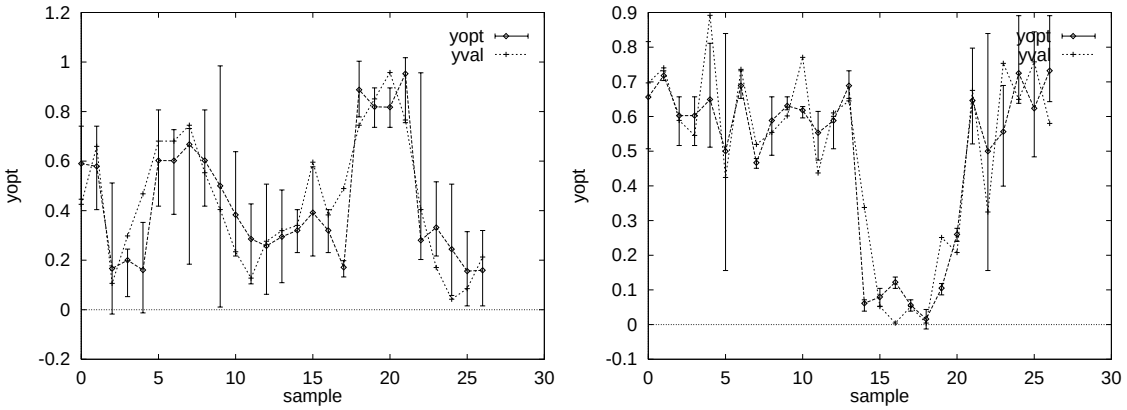

Figure 9: Two test sets with prediction outputs for *tear* and *tensile* with estimated 67% confidence intervals (*one standard deviation*) plotted as error bars. The predicted values are marked with ⋄:s (with bars) and the process output values are marked with +:s. In some cases like sample 9 for *tear* 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	
size	σ_{err}
80-80	8.01
80-10	8.43
80-20	8.51
80-40	8.94
40-40	9.33
40-20	9.70
40-10	9.91
40-80	10.21
20-20	10.45
20-10	11.56
20-80	11.75
20-40	11.83
10-20	15.15
10-10	15.83
10-40	15.99
10-80	16.86

tensile-TRAIN	
size	σ_{err}
80-20	7.77
80-80	8.09
40-20	8.22
80-40	8.56
40-80	9.13
80-10	9.20
40-40	9.28
40-10	9.53
20-10	9.73
20-80	10.38
20-40	10.47
20-20	10.51
10-20	11.69
10-80	12.50
10-40	12.56
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	
size	σ_{err}
40-10	11.86
40-20	13.81
20-10	15.29
40-80	15.34
20-40	15.63
10-80	16.56
40-40	18.02
20-80	18.06
10-10	18.10
80-20	18.45
80-10	18.47
10-40	18.55
20-20	18.80
80-80	18.88
10-20	20.31
80-40	20.98

tensile-TEST	
size	σ_{err}
20-80	9.89
20-40	9.90
20-20	10.09
20-10	12.02
10-10	12.94
40-80	13.09
10-80	13.59
40-40	14.78
10-40	14.82
40-20	15.47
80-40	15.74
40-10	15.88
10-20	16.98
80-20	17.81
80-10	21.75
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).

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.

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

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

[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.

[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.

[Lansner and Örjan Ekeberg, 1989] Lansner A. and Örjan Ekeberg. (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.

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

[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.

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