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 pulp quality is predicted from a set of process values. The values being predicted are results from time consuming laboratory experiments. In most *regression* models, like the error backpropagation network, the 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 a certain 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. Explanatory and response variable spaces are represented by Gaussian RBF:s *(Radial Basis Functions)* using the stochastic EM (*Expectation Maximization*) algorithm for calculation of positions and variances. These RBF:s or *(density functions)* model the *a priori* density for each variable space. Bayesian associative connections are used to generate the response variable *a posteriori* density when it is conditioned by an explanatory variable value. We found that this method for function approximation performs comparably well with the best backpropagation network we could find on the same pulp and paper data. It is also straight forward to use with just two design parameters, the number of units which code the explanatory and response variables respectively.

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 [OL92]. 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 [OL94].

Mixture density networks have been used for, *e.g.* classification of speech segments and satellite image pixels [Trå93] and classification of globins in protein sequences [Mac94]. Function approximation has been done by ,*e.g.* predicting the *a posteriori* density [Bis94] or using the EM-algorithm directly [Gha94]. 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.* [HL93a] and [Mac94]. 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 [Trå91] for the RBF-units and a BCPNN (Bayesian Confidence Propagation Neural Network), which have earlier been successfully used for pattern completion [LE89] and classification [HL93b], to associate an explanatory conditioned density with a response density function.

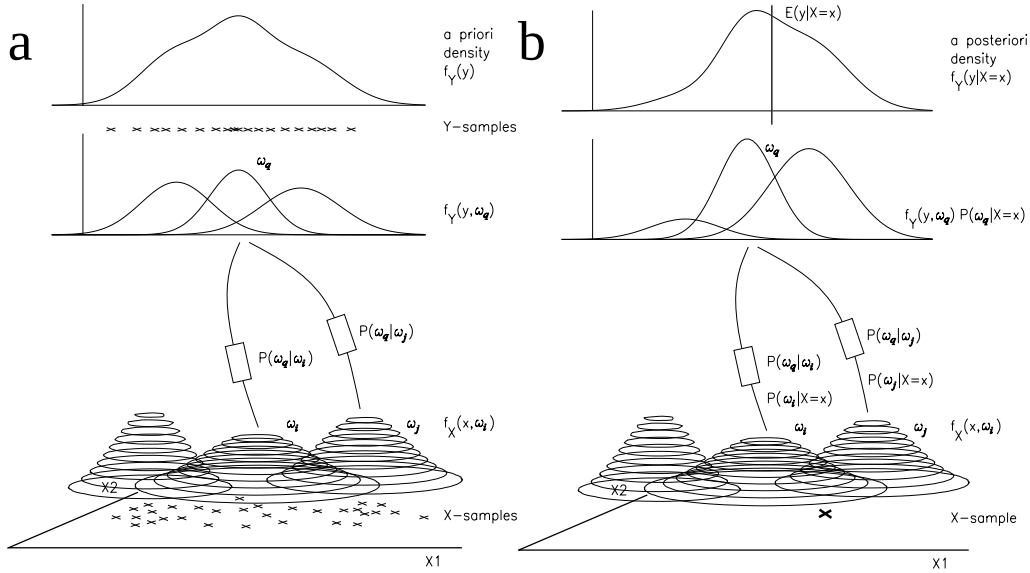

Figure 1: An overview of the density method. **a:** Training phase: We let a set of Gaussian density functions (ω_i) model the density of a set of data samples in explanatory (X) and response (Y) variable spaces. When an X-sample is drawn from the process ω_i in explanatory space there is a certain probability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

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).

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

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 probabilites α_i sum to 1, which can be solved by using the Lagragange multiplier method.

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

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 numerially. (C_i is the covariance matrix for component i and d is the number of dimensions for the variable)

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

A stochastic variant of the EM-algorithm [Trå91], 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 stepsize 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" parameterfree 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 update the variances when the positions have somewhat begun to stabilize. As an example we can, in figure 2, see how a set of 40 RBF:s have adapted to 676 data points forming a square. There is also an example with two variable 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 pdf:s is not necessarily unique, there may exist several solutions which maximize the likelihood (2).

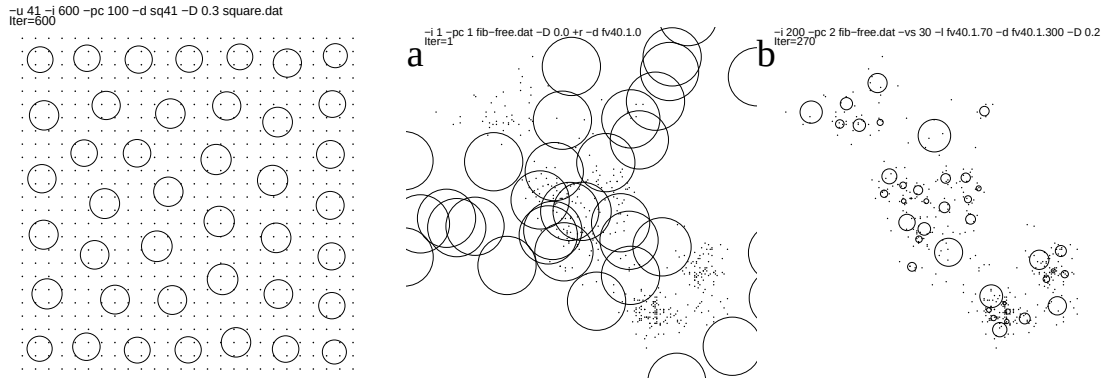

Figure 2: Left: 41 units are adapted to a square distribution. We see the RBF units with their σ plotted as circles and the data samples as dots. **a:** 40 units randomly initialized within. **b:** After about 200 iterations the units had become stable in this case.

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 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 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 to express the response variable density 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 $N(x, \mu_i, \sigma_i^2)$.

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

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 [LE89]. Finally we want

a predicted value as output. To get this we calculate the expectation value of the response value density function, which can be done by integrating the density function (16) to give "a center of mass". In the case with, for instance, Gaussian component densities we need not do an integration for this. It is enough to 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 estimate a confidence interval (19) for the prediction.

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

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

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

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

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 3, 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 4, where we used 150 samples for the training set and 50 samples for the test set, we see how the peformance on training and test set respectively on the same problem as above varies when the number of explanatory and response units are varied.

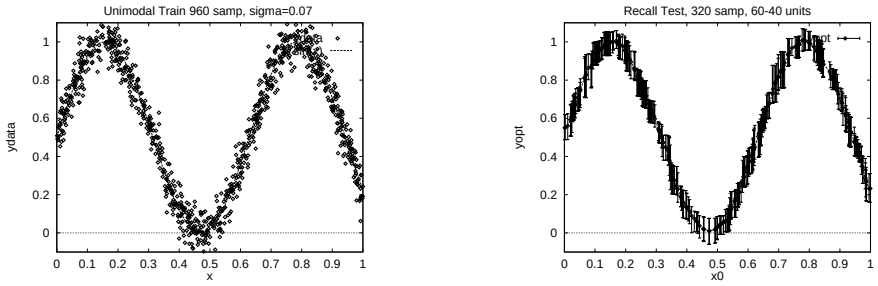

Figure 3: Left: Training data consisting of 960 samples with a normal distributed noise ($\sigma = 0.07$). Right: Recalled y-value from 320 samples with a network using 60 explanatory units and 40 response units. The error bars show one estimated standard deviation.

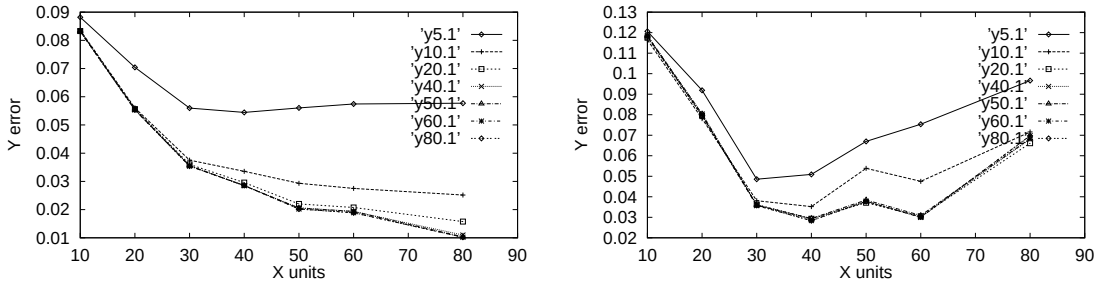

Figure 4: The performance for training set (left) and test set (right) when we vary the number of units in explanatory (X) and response (Y) layers. On the x-axis is the number of explanatory units. The Y units are shown by different curves in the same diagram.

Actually this mixture density method is symmetrical for predictions $X \rightarrow Y$ and $Y \rightarrow X$, which is illustrated in figure 5 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 6 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.

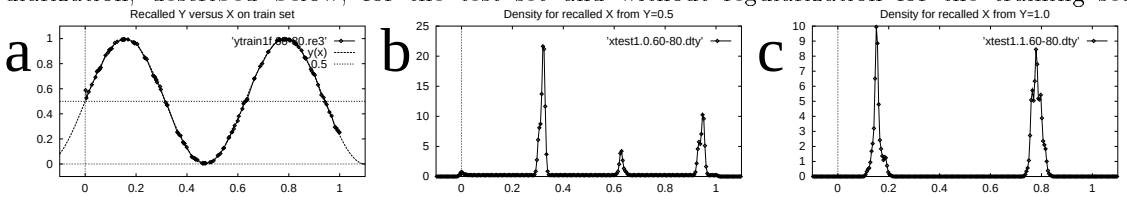

Figure 5: **a:** A recalled sine function as Y=f(X). **b:** Recalled density for X when input on Y is 0.5 (arcsin(Y)). **c:** Same as b when Y input is 1.

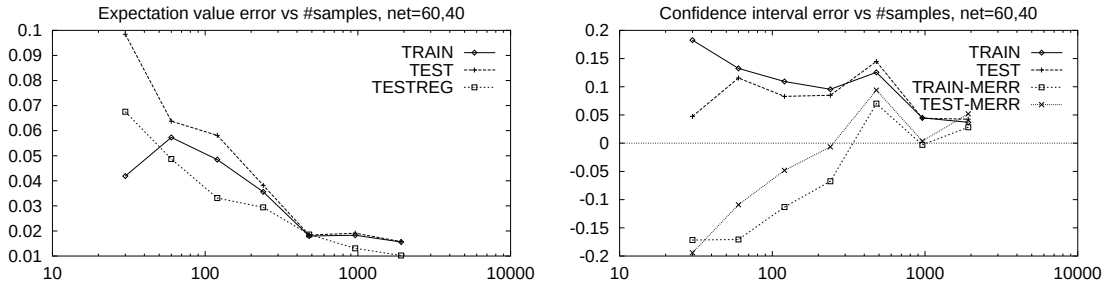

Figure 6: Left: The prediction performance of the expectation value improves for a specific network when the number of samples used for training increases. In this case we could perform better on the test set with regularization than on the training set without regularization most of the time. Right: 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

One way to improve the generalization performance when the RBF-model has been trained with too few examples is to increase the "fuzziness" in the system by scaling up the variances. The sample will be classified as possibly being generated from several close PDF:s 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 PDF:s which represent this value is minimized, figure 7.

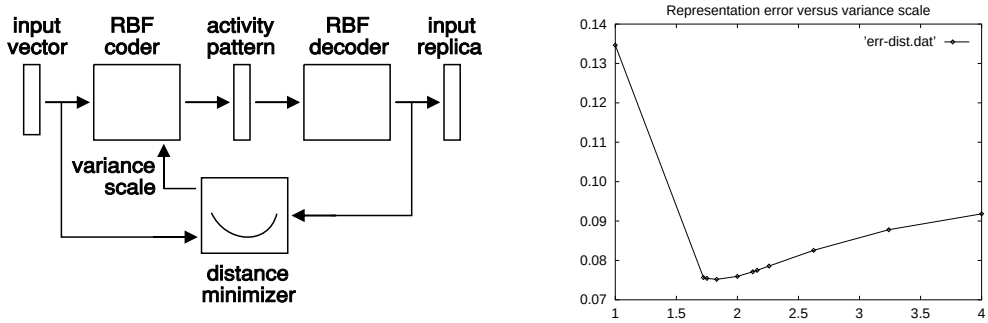

Figure 7: Left: Dynamic regularization by adapting the explanatory sample "fitness" by scaling the variances, thus improving the generalization capability. Right: Representation error vs. variance scaling (golden search used as distance minimizer).

In table 1 we see two laboratory measured pulp response variables *tear*, and *tensile* being predicted from six explanatory variables. We see how the prediction performance varies when the number of units in explanatory and response layer is increased from 10 to 80. The data set was randomly partitioned into 75 % training and 25 % test data. As a comparision the best BP results obtained on test set earlier [OL92] was 11.7 % for *tear* and 10.8 % for *tensile*. This indicates that the mixture density method performs equally well as BP considering the predicted value.

In figure 8 we see diagrams for the predictions of *tear* and *tensile* tensile on a test set. Here are also an estimation of a 67 % confidence interval for the prediction shown as error bars.

tear				*tensile*			
TRAIN		TEST		TRAIN		TEST	
size	σ_{err}	size	σ_{err}	size	σ_{err}	size	σ_{err}
80-80	8.01	40-10	11.86	80-20	7.77	20-80	9.89
80-10	8.43	40-20	13.81	80-80	8.09	20-40	9.90
80-20	8.51	20-10	15.29	40-20	8.22	20-20	10.09
80-40	8.94	40-80	15.34	80-40	8.56	20-10	12.02
40-40	9.33	20-40	15.63	40-80	9.13	10-10	12.94
40-20	9.70	10-80	16.56	80-10	9.20	40-80	13.09
40-10	9.91	40-40	18.02	40-40	9.28	10-80	13.59
40-80	10.21	20-80	18.06	40-10	9.53	40-40	14.78
20-20	10.45	10-10	18.10	20-10	9.73	10-40	14.82
20-10	11.56	80-20	18.45	20-80	10.38	40-20	15.47
20-80	11.75	80-10	18.47	20-40	10.47	80-40	15.74
20-40	11.83	10-40	18.55	20-20	10.51	40-10	15.88
10-20	15.15	20-20	18.80	10-20	11.69	10-20	16.98
10-10	15.83	80-80	18.88	10-80	12.50	80-20	17.81
10-40	15.99	10-20	20.31	10-40	12.56	80-10	21.75
10-80	16.86	80-40	20.98	10-10	13.46	80-80	23.00

Table 1: Prediction performance for *tear* and *tensile* for different sizes of explanatory and response layer.

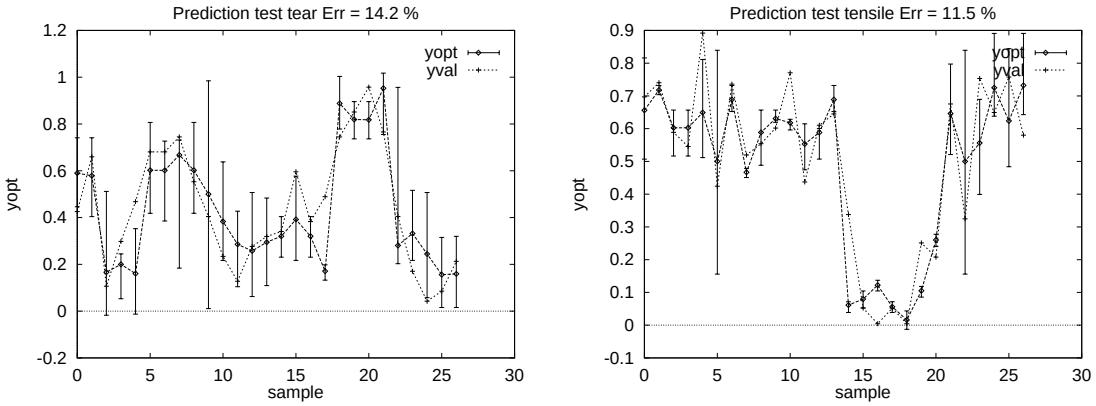

Figure 8: Two test sets with prediction outputs for *tear* and *tensile* with estimated 67% confidence intervals (*one standard deviation*) plotted as error bars. The predicted values are marked with ◇:s (with bars) and the process output values are marked with +:s. In some cases like sample 9 for *tear* and sample 5 for *tensile* we see a large confidence interval estimation due to the input vector being far from all the RBF units.

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 *unsupervised*, 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 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 relatively simple. 4) The supervised training part can be done as a *one shot* quick 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, which will show up as a multimodal density function. 6) A missing value in an input sample vector is still useable 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

[Bis94] Chris M. Bishop. Mixture density networks. Tech. Rep. NCRG/4288, Department of Computer Science, Aston University, 1994.

[Gha94] 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.

[HL93a] 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.

[HL93b] Anders Holst and Anders Lansner. A flexible and fault tolerant query-reply system based on a Bayesian neural network. *Int. J. Neural Systems*, 4(3):257–267, 1993.

[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. Also extended abstract in Proceedings from the Nordic symposium on Neural Computing, April 17–18, Hanasaari Culture Center, Espoo, Finland.

[Mac94] David J.C. MacKay. Bayesian neural networks and density networks. Proc. of Workshop on Neutron Scattering Scattering Data Analysis 1994, 1994.

[OL92] 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.

[OL94] 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.

[Trå91] 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.

[Trå93] Hans G.C. Tråvén. Invariance constraints for improving generalization in probabilistic neural networks. In *IEEE ICNN'93*, pages 1348–1353, 1993.