Study and Software Implementation of Variational Bayesian Approach to Mixed Deterministic/Stochastic Fuzzy Models

Sukhvir Singh
Department of Computer Science, Himachal Pradesh University,
Shimla -171005, India

Yogesh Mohan
Department of Computer Science, Himachal Pradesh University,
Shimla -171005, India

Kishori Lal Bansal
Department of Computer Science, Himachal Pradesh University,
Shimla -171005, India

ABSTRACT
The study explains a new emerging methodology Variational Bayesian Inference (VB) to structure optimization of Fuzzy System (Takagi-Sugeno fuzzy system). Recently, the study of (Kumar et al. 2010 a) introduced a mixed Takagi-Sugeno fuzzy filter whose antecedents are deterministic while the consequents are random variables. The parameters of fuzzy filters are inferred under VB framework. The objective of this study is to show how computational intelligence based model contribute to the methodology of constructing models of software processes and products. The study provides detailed software implementation of Variation Bayesian approach to mixed deterministic/stochastic fuzzy models and also helps in software developments of some computational optimization algorithms based on Variational Bayesian approach. The developed MATLAB software can be used in the field of image processing, signal processing, pattern recognition, machine learning.

General Terms
Variational Bayesian Inference, Algorithms, Fuzzy Modeling, MATLAB Tool.

Keywords
Fuzzy Modeling, Takagi-Sugeno Fuzzy Model, Variational Bayesian Inference, Stochastic Model, Matlab Software.

1. INTRODUCTION
A real world modeling problems [5], [6], [10] typically involves the uncertainties and fuzzy systems are considered suitable tools for dealing with uncertainties [2], [4]. Also many real-world physical processes are generally characterized by the presence of non-linearity, complexity and uncertainty. These processes cannot be represented by linear models used in conventional system identification [8]. Fuzzy logic, the logic underlying the modes of reasoning which are approximate rather than exact, can be used for the modeling of real-world uncertain complex processes. Fuzzy framework offers the following advantages in comparison to the classical techniques:

- Uncertainty can be handled by capturing the process behavior using fuzzy sets.
- The inference procedure becomes more robust and flexible with approximate reasoning methods.

The fuzzy rule-based systems are probably the most important application of fuzzy set theory of Zadeh [1]. Depending on the form of the fuzzy rules considered and types of inputs and outputs, two tyes of rule-based fuzzy systems are distinguished: 1) Mamdani, 2) Takagi-Sugeno. A fuzzy rule based model suitable for approximation of many systems and functions is the Takagi Sugeno fuzzy model [3]. The Takagi-Sugeno type fuzzy models are typically chosen in data driven modelling problems because Takagi-Sugeno fuzzy models facilitate the mathematical development and analysis of the systematic methods for generating fuzzy rules from a given input-output data set. In the last few years, a large research effort has been made to combine the learning capability of neural networks with the knowledge representation of fuzzy logic results in Neuro Fuzzy systems, which extract fuzzy rules from training data [7].

Now a day, the Variational Bayes (VB) method is not a new technique and has been widely studied by the researchers. The VB framework has the advantage over Bayesian framework which is a powerful technique for the statistical inference of model parameters. The reason being VB framework is less computationally intensive than other Bayesian methods. The VB method approximates the posterior distribution over a model in an analytical manner [11], and minimizes the Kullback-Leibler (KL) divergence of the approximate posterior from the true posterior density [12].

1.1 Variational Bayesian Inference
In a model selection task the role of a Bayesian is to calculate the posterior distribution over a set of models given some a priori knowledge and some new observation (data). The knowledge is represented in the form of a prior over model structures $p(m)$, and their parameters $p(w|m)$ which define the probabilistic between the variables in the model. By Baye's rule, the posterior over models m having seen data y is given by

$$p(m|y) = \frac{p(m)p(y|m)}{p(y)}$$

The most probable model or model structure is the one that maximises $p(m|y)$, where $p(y|m)$ is the marginal likelihood or evidence for a model m. For each model structure, the posterior distribution over the parameters can also be computed by:

$$p(w|y,m) = \frac{p(y|w,m)p(w|m)}{p(y|m)}$$

This implies estimation both of the parameters and uncertainties associated with their estimation. Thus Baye's

theorem provides the posterior probability of the parameters given the data and model. The analytical evaluation of posterior probability distribution is not possible in every case. The variational methods can be used to find approximate solutions in Bayesian inference problems. Variational Baye's approximates the full posterior as:

$$q(w) \approx p(w \mid y, m)$$

Where $q(w)$ is restricted to belong to a family of distributions of simpler form. This form is selected by minimizing the gap between q and true posterior. This difference is known as Kullback-Leibler divergence (KL) and is given by

$$KL(q \,\|\, p) = \int q(w) \log \frac{q(w)}{p(w \mid y, m)} dw.$$

Given a probabilistic model of the data, the log of 'evidence' or marginal likelihood can be written as:

$$\log p(y \mid m) = \log \int p(y, w \mid m) dw$$

$$= \log \int q(w) \frac{p(y, w \mid m)}{q(w)} dw$$

$$\geq \int q(w) \log \frac{p(y, w \mid m)}{q(w)} dw$$

$$\equiv \mathrm{F}\,(q(w), m)$$

Thus a lower bound on log evidence using Jensen's inequality is obtained, which is seek to maximize. Maximizing F is equivalent to minimizing the KL divergence between $q(w)$ and $p(w \mid y, m)$, so a tractable q can be an approximation to the intractable posterior.

2. PROBLEM

Consider a Takagi-Sugeno fuzzy model $(F_s : X \to Y)$ that maps n-dimensional real input space $(X = X_1 \times X_2 \times ..... \times X_n)$ to one dimensional real line. A Takagi-Sugeno type fuzzy filter in [9] represented as

$$y_f = G^T(x, \theta)\alpha, \quad c\theta \geq h \qquad (1)$$

2.1 *A Mixed Deterministic/Stochastic Fuzzy Model*

The filter, as seen from(1), is characterized by two types of parameters: antecedents (θ) and consequents (α). Expression (1) shows that the output of the fuzzy filter is linear in consequent i.e. in the elements of vector α while nonlinear in antecedents i.e. in the elements of vector θ. This type of model is used to study a filter with

- the nonlinear parameters θ being considered as deterministic, and
- the linear parameters α being considered as random variable.

2.2 *VB for Mixed Deterministic/Stochastic Fuzzy Models* [9]

The considered fuzzy filtering problem in VB framework is stated in Problem 1.

Problem 1: Given N pairs of inputs-output data pairs $\{x(j), y(j)\}_{j=1}^{N}$ and a structure m (i.e. number of membership functions and rules) of a Takagi-Sugeno filter of type (1), estimate θ and a probability distribution $q(\alpha)$ by maximizing the lower bound on the quantity: $\log p(y(1), ..., y(N) \mid x(1), ..., x(N), \theta, m)$.

Introduce the notations:

$$Y = [y(1).....y(N)]^T \in R^N, B(\theta) = \begin{bmatrix} G^T(x(1), \theta) \\ \vdots \\ G^T(x(N), \theta) \end{bmatrix} \in R^{N \times K},$$

$$v = [n_1.....n_N]^T \in R^N$$

Now,

$$Y = B(\theta)\alpha + v, \qquad (2)$$

and Problem 1 can be rewritten as

Problem 2: Given N pairs of inputs-output data pairs $\{x(j), y(j)\}_{j=1}^{N}$ and a structure m (i.e. number of membership functions and rules) of a Takagi-Sugeno filter of type (1), estimate θ and a probability distribution $q(\alpha)$ by maximizing the lower bound on the quantity: $\log p(Y \mid B(\theta), m)$.

The finite mixture models are widely used in stochastic modeling of data. It is assumed that there exist a finite set of random sources whose convex combination might have generated the observed data.

The concern is to model the observed data as a mixture of different Takagi-Sugeno fuzzy filters. Assume that there are S number of fuzzy filters (with their structures as $\{m^{s_i}\}_{s_i=1}^{S}$) such that for $s_i - th$ filter $(s_i = 1, ..., S)$, we have

$$Y = B(\theta^{s_i})\alpha^{s_i} + v, \qquad s_i = 1, ..., S \qquad (3)$$

Let $\pi = [\pi_1 ..... \pi_s]^T \in R^S$, with $0 \leq \pi_{s_i} \leq 1$ and $\sum_{s_i=1}^{S} \pi_{s_i} = 1$, be a vector of mixing proportions and s_i is a discrete indicator variable for the filter chosen to model data. Now, model the probability density function of the observed output data as a weighted average of the fuzzy filters' output density functions:

$$p(Y \mid \pi, \{B(\theta^{s_i})\}_{s_i=1}^{S}, \{\alpha^{s_i}\}_{s_i=1}^{S}, \phi, \{m^{s_i}\}_{s_i=1}^{S}) =$$

$$\sum_{s_i=1}^{S} p(s_i \mid \pi)\, p(Y \mid s_i, \{B(\theta^{s_i})\}_{s_i=1}^{S}, \{\alpha^{s_i}\}_{s_i=1}^{S}, \phi, \{m^{s_i}\}_{s_i=1}^{S})$$

(4)

Problem 3: Given N pairs of inputs-output data pairs $\{x(j), y(j)\}_{j=1}^{N}$ and S different structure $\{m^{s_i}\}_{s_i=1}^{S}$ of the Takagi-Sugeno filter of type (1) such that data satisfy (3) and (4), estimate $\{\theta^{s_i}\}_{s_i=1}^{S}$ and the probability distribution

$\{q(\alpha^{s_i}), q(\pi_{s_i}), q(s_i)\}_{s_{i=1}}^{S}$ by maximizing the lower bound on the quantity: $\log p(Y \mid \{B(\theta^{s_i})\}_{s_{i=1}}^{S}, \{m^{s_i}\}_{s_i=1}^{S})$.

3. THE MATHEMATICAL DERIVATION

Problem 3 (stated above) is concerned with the inference of S number of fuzzy filters (with their structures as $\left\{m^{s_i}\right\}_{s_i=1}^{S}$) such that for $s_i - th$ filter $(s_i = 1,...,S)$, $Y = B(\theta^{s_i})\alpha^{s_i} + v,\ \mathrm{s}_i = 1,...,S.$ Consider the case of additive Gaussian uncertainty v with mean 0 and a variance of $1/\phi$: $P(v) \sim N(0,\ \phi^{-1}I)$. Let $\pi = [\pi_1 ... \pi_S]^T \in R^S$, with $0 \le \pi_{s_i} \le 1$ and $\sum_{s_i=1}^{S} \pi_{s_i} = 1$, be a vector of mixing proportions and s_i is a discrete indicator variable for the filter chosen to model data. The probability density function of the observed output data is modeled as a weighted average of the fuzzy filters' output density functions:

$$p(Y \mid \pi, \{B(\theta^{s_i})\}_{s_{i=1}}^{S}, \{\alpha^{s_i}\}_{s_{i=1}}^{S}, \phi, \{m^{s_i}\}_{s_{i=1}}^{S}) = \sum_{s_i=1}^{S} p(s_i \mid \pi)\, p(Y \mid s_i, \{B(\theta^{s_i})\}_{s_{i=1}}^{S}, \{\alpha^{s_i}\}_{s_{i=1}}^{S}, \phi, \{m^{s_i}\}_{s_{i=1}}^{S}).$$

The mathematical derivation of the results has been directly taken from section III of [9].

4. THE ALGORITHMS [9]

Algorithm 1 VB inference of the fuzzy filter combination

Require: Data pairs $\{x(j), y(j)\}_{j=1,.....,N.}$

1: Choose a total of S fuzzy filters' structures $\{m^{s_i}\}_{s_{i=1}}^{S}$; hyper parameters $c_0, m_0, \Lambda_0, a_0, b_0$ which define the regularizing priors; parameters $\{c^{s_i}, h^{s_i}\}_{s_{i=1}}^{S}$ such that the interpretability constraints on the membership functions of the $s_i - th$ filter can be formulated as $c^{s_i}\theta^{s_i} \ge h^{s_i}$.

2: Define

$$\mathrm{F}(q(\pi), q(\alpha), q(\phi), \{q(s_i)\}_{s_{i=1}}^{S}, \{B(\theta^{s_i})\}_{s_{i=1}}^{S}, c_0, m_0, \Lambda_0, a_0, b_0, \{m^{s_i}\}_{s_{i=1}}^{S}) =$$
$$\int d\pi q(\pi) \log \frac{p(\pi \mid c_0 d_0)}{q(\pi)} + \int d\alpha q(\alpha) \log \frac{p(\alpha \mid m_0, \Lambda_0)}{q(\alpha)}$$
$$+ \int d\phi q(\phi) \log \frac{p(\phi \mid a_0, b_0)}{q(\phi)}$$
$$+ \sum_{s_{i=1}}^{S} q(s_i) \int d\alpha d\phi q(\alpha) q(\phi) \log p(Y \mid s_i, B(\theta^{s_i}), \alpha^{s_i}, \phi, m^{s_i})$$
$$+ \sum_{s_{i=1}}^{S} q(s_i) \int d\pi q(\pi) \log \frac{p(s_i \mid \pi)}{q(s_i)}$$

3: Derive analytically the correct expressions for distributions $(q(\pi), q(\alpha), q(\phi), q(s_i))$ by maximizing F over $(q(\pi), q(\alpha), q(\phi), q(s_i))$ (i.e. by setting the functional derivative of F with respect to each free distribution equal to zero). Let $(q^*(\pi), q^*(\alpha), q^*(\phi), q^*(s_i))$ denote the obtained expressions for the variational distributions.

4: Estimate the optimal values of antecedents of fuzzy filters, i.e. $(\theta^{1,*}, ..., \theta^{S,*})$, via maximizing F further over $(\theta^1, ..., \theta^S)$. That is, solve the following constrained nonlinear optimization problem:

$$(\theta^{1,*}, ..., \theta^{S,*}) = \arg \max_{(\theta^1, ..., \theta^S)} \left[\mathrm{F}_\theta(.); c^{s_i}\theta^{s_i} \ge h^{s_i}\right]$$

Where

$$\mathrm{F}_\theta(.) = \mathrm{F}(q^*(\pi), q^*(\alpha), q^*(\phi), \{q^*(s_i)\}_{s_{i=1}}^{S}, \{B(\theta^{s_i})\}_{s_{i=1}}^{S}, c_0, m_0, \Lambda_0, a_0, b_0, \{m^{s_i}\}_{s_{i=1}}^{S}).$$

5: return $(\theta^{1,*}, ..., \theta^{S,*})$, and $(q^*(\pi), q^*(\alpha), q^*(\phi), \{q^*(s_i)\}_{s_{i=1}}^{S})$

Algorithm 2 An algorithm for VB inference of the fuzzy filter combination

Require: Data pairs $\{x(j), y(j)\}_{j=1,.....,N.}$

1: Choose a total of S fuzzy filters' structures $\{m^{s_i}\}_{s_{i=1}}^{S}$; hyper parameters $c_0, m_0, \Lambda_0, a_0, b_0$ which define the regularizing priors; parameters $\{c^{s_i}, h^{s_i}\}_{s_{i=1}}^{S}$ such that the interpretability constraints on the membership functions of the $s_i - th$ filter can be formulated as $c^{s_i}\theta^{s_i} \ge h^{s_i}$.

2: Set iteration count t = 0 and choose a tolerance limit (say, equal to 0.01%).

3: **if** $\left(\max_{s_i} abs(q^*(s_i)|_{t+1} - q^*(s_i)|_t) < 0.0001\right)$ & $(t > 0)$ **then**

4: return $(\theta^{1,*}|_{t+1}, ..., \theta^{S,*}|_{t+1})$ and $(c|_{t+1}, \{d_{s_i}|_{t+1}\}_{s_{i=1}}^{S}, \{\Lambda^{s_i}|_{t+1}\}_{s_{i=1}}^{S}, \{m^{s_i}|_{t+1}\}_{s_{i=1}}^{S}, \{q^*(s_i)|_{t+1}\}_{s_{i=1}}^{S}, a|_{t+1}, b|_{t+1})$.

5: **else**

6: Update the parameters as follows:

$$\theta^{s_i,*}|_{t+1} = \arg \min_{\theta^{s_i}} \left[r(m^{s_i}|_t, \Lambda^{s_i}|_t, \theta^{s_i}); c^{s_i}\theta^{s_i} \ge h^{s_i}\right]$$
$$\Lambda^{s_i}|_{t+1} = \Lambda_0^{s_i} + q^*(s_i)|_t\, a|_t\, b|_t\, (B(\theta^{s_i,*}|_{t+1}))^T B(\theta^{s_i,*}|_{t+1})$$
$$m^{s_i}|_{t+1} = \left[\Lambda_0^{s_i} + q^*(s_i)|_t\, a|_t\, b|_t\, (B(\theta^{s_i,*}|_{t+1}))^T B(\theta^{s_i,*}|_{t+1})\right]^{-1}$$
$$\left[\Lambda_0^{s_i} m_0^{s_i} + q^*(s_i)|_t\, a|_t\, b|_t\, (B(\theta^{s_i,*}|_{t+1}))^T Y\right]$$

$$q^*(s_i)|_{t+1} = \frac{1}{Z} \exp\left(\Psi(c|_t\, d_{s_i}|_t) - \frac{a|_t\, b|_t}{2} r(m^{s_i}|_{t+1}, \Lambda^{s_i}|_{t+1}, \theta^{s_i,*}|_{t+1})\right)$$

where Z is such that $\sum_{s_{i=1}}^{S} q^*(s_i)|_{t+1} = 1.$

$$b\,|_{t+1} = b_0 + \frac{N}{2}$$

$$\frac{1}{a\,|_{t+1}} = \frac{1}{a_0} + \frac{1}{2}\sum_{s_{i=1}}^{S} q^*(s_i)\,|_{t+1}\; r(m^{s_i}\,|_{t+1}, \Lambda^{s_i}\,|_{t+1}, \theta^{s_i,*}\,|_{t+1})$$

$$c\,|_{t+1} = c_0 + 1$$

$$d_{s_i}\,|_{t+1} = \frac{1}{c_0+1}\left(\frac{c_0}{S} + q^*(s_i)\,|_{t+1}\right)$$

here

$$\left(\begin{array}{l}\{m^{s_i}\,|_0\}_{s_{i=1}}^{S}, \{\Lambda^{s_i}\,|_0\}_{s_{i=1}}^{S}, a\,|_0, b\,|_0, \{\theta^{s_i,*}\,|_0\}_{s_{i=1}}^{S}, \\ \{q^*(s_i)\,|_0\}_{s_{i=1}}^{S}, c\,|_0, \{d_{s_i}\,|_0\}_{s_{i=1}}^{S}\end{array}\right)$$

denote the initial guess.

7: **end if**

Remark 1: Algorithm 2 summarizes the method for inferring the parameters of the fuzzy filters combination via maximizing F..

Remark 2: The problem 2 is simpler and a particular case of the problem 3 where there is only one fuzzy filter whose parameters need to be inferred. Without loss of any generality, our approach of Algorithm 1 can be used to solve the problem 2. The update rules in this case are summarized in algorithm 3.

Algorithm 3 An algorithm for VB inference of the fuzzy filter parameters

Algorithm 3 represent an approach to inferred parameters of one fuzzy filter and can be started with an initial guess of $m\,|_0 = m_0,\ \Lambda|_0 = \Lambda_0,\ \mathrm{a}|_0 = a_0,\ \mathrm{b}|_0 = b_0$.

Require: Data pairs $\{x(j), y(j)\}_{j=1,\ldots\ldots,N.}$

1: Choose a fuzzy filter structure m ; hyper parameters $c_0, m_0, \Lambda_0, a_0, b_0$ which define the regularizing priors; parameters c, h such that the interpretability constraints on the membership functions can be formulated as $c\theta \geq h$.

2: Set iteration count t = 0 and choose a tolerance limit (say, equal to 0.01%).

3: **if** $\left(\mathrm{F}\,|_{t+1} - \mathrm{F}\,|_t < 0.0001\mathrm{F}\,|_t\right)$ & $(t > 0)$ **then**

4: **return** $(\theta^*\,|_{t+1}, \Lambda\,|_{t+1})$ and $(m\,|_{t+1}, a\,|_{t+1}, b\,|_{t+1})$.

5: **else**

6: Update the parameters as follows:

$$\theta^*\,|_{t+1} = \arg\min_{\theta}\ \left[r(m\,|_t, \Lambda\,|_t, \theta);\ c\,\theta \geq h\right]$$

$$\Lambda\,|_{t+1} = \Lambda_0 + a\,|_t\; b\,|_t\; (B(\theta^*\,|_{t+1}))^T B(\theta^*\,|_{t+1})$$

$$m\,|_{t+1} = \left[\Lambda_0 + a\,|_t\; b\,|_t\; (B(\theta^*\,|_{t+1}))^T B(\theta^*\,|_{t+1})\right]^{-1} \left[\Lambda_0 m_0 + a\,|_t\; b\,|_t\; (B(\theta^*\,|_{t+1}))^T Y\right]$$

$$b\,|_{t+1} = b_0 + \frac{N}{2}$$

$$\frac{1}{a\,|_{t+1}} = \frac{1}{a_0} + \frac{1}{2} r(m\,|_{t+1}, \Lambda\,|_{t+1}, \theta^*\,|_{t+1})$$ here, $(m\,|_0, \Lambda\,|_0, a\,|_0, b\,|_0, \theta^*\,|_0)$ denote the initial guess and $\mathrm{F}\,|_t$ is computed as

$$\begin{aligned}\mathrm{F}\,|_t &= \frac{N}{2}\left(\Psi(b\,|_t) + \log(a\,|_t)\right) - \frac{N}{2}\log(2\,pi) \\ &- \frac{a\,|_t\; b\,|_t}{2} r(m\,|_t, \Lambda\,|_t, \theta^*\,|_t) - \frac{1}{2}\log\left(\frac{|(\Lambda_0)^{-1}|}{|(\Lambda\,|_t)^{-1}|}\right) \\ &- \frac{1}{2}Tr\left(\Lambda_0(\Lambda\,|_t)^{-1}\right) - \frac{1}{2}(m\,|_t - m_0)^T \Lambda_0 (m\,|_t - m_0) \\ &+ \frac{K}{2} - \log(\Gamma(b_0)) - b_0\log(a_0) \\ &+ \log(\Gamma(b\,|_t)) + b_0\log(a\,|_t) \\ &- b\,|_t\;\Psi(b\,|_t) + b_0\Psi(b\,|_t) - b\,|_t\;\frac{a\,|_t}{a_0} + b\,|_t\;.\end{aligned}$$

7: **end if**

Remark 3**:** The algorithms 2 and 3 were implemented in MATLAB 6.5 [13] and involve a nonlinear constrained optimization problem. The parameter estimation, based on the nonlinear optimization problem, was performed by running a single iteration of the algorithm "fmincon" available in MATLAB Optimization Toolbox [14].

5. SOFTWARE IMPLEMENTATION

Algorithms explained have been implemented in MATLAB 6.5. This section presents some pieces of software code developed during the computational optimization process of algorithms.

5.1 *Code for the choice of antecedent matrix*

```
function[B]= AntecedentMatrix(inputs_data_matrix,FM)
[N,n] = size(inputs_data_matrix);
if FM.order == 0
K = length(FM.consequents.mean);
elseif FM.order == 1
   K_f = length(FM.consequents.mean);
   K = K_f/(n+1);
   B_f = zeros(N,K_f);
else
   disp('The order must be either 0 or 1!');
   return
end
B = zeros(N,K);
k1 = 1;
while (k1 <= N)
   if strcmp(FM.memberships.type,'GaussianClustering')
B(k1,:)=GaussianMembershipClustering((inputs_data_matrix
```

```
(k1,:))',FM.memberships.cluster_matrix,FM.memberships.
var_matrix);
   else
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      k2 = 1;
       while (k2 <= n)
ts.membership_values{k2}=MembershipValues(inputs_data_
matrix(k1,k2),FM.memberships.knots{k2},
FM.memberships.type);
number_memberships(k2)=length(ts.membership_values{k2});
         k2 = k2 +1;
      end
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      if n==1,
         B(k1,:) = ts.membership_values{1};
      end
      if n == 2
         j = 0;
         for i1 = 1:number_memberships(1),
            for i2 = 1:number_memberships(2),
B(k1,j+i2)=(ts.membership_values{1}(i1))*
                                  (ts.membership_values{2}(i2));
            end
            j = j+number_memberships(2);
         end
      end
      if n == 3
         j = 0;
         for i1 = 1:number_memberships(1),
            for i2 = 1:number_memberships(2),
               for i3 = 1:number_memberships(3),
B(k1,j+i3)=(ts.membership_values{1}(i1))*
      (ts.membership_values{2}(i2))*(ts.membership_values{3}(i3));
               end
               j = j+number_memberships(3);
            end
         end
      end
      if n == 4
         j = 0;
         for i1 = 1:number_memberships(1),
            for i2 = 1:number_memberships(2),
               for i3 = 1:number_memberships(3),
                  for i4 = 1:number_memberships(4),
B(k1,j+i4)=(ts.membership_values{1}(i1))*
                        (ts.membership_values{2}(i2))*
                              (ts.membership_values{3}(i3))*
                                    (ts.membership_values{4}(i4));
                  end
                  j = j+number_memberships(4);
               end
            end
         end
      end
   end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
B(k1,:) = (1/sum(B(k1,:)))*B(k1,:);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   if FM.order == 1
      for i = 1:K
         tmpvrb1 = n*(i-1) + i;
         tmpvrb2 = (n+1)*i;
    B_f(k1,tmpvrb1:tmpvrb2) =
                           (B(k1,i))*[inputs_data_matrix(k1,:) 1];
      end
   end
   k1 = k1+1;
end
if FM.order == 1
   B = B_f;
end
return
 function [mv] = MembershipValues(x,t,membership_type)
switch lower(membership_type)
   case 'trapezoidal'
      [mv] = TrapezoidalMembership(x,t);
   case 'gaussian'
      [mv] = GaussianMembership(x,t);
   case 'clustering'
      [mv] = ClusteringMembership(x,t);
   case 'triangular'
      [mv] = TriangularMembership(x,t);
end
return
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5.2 *Software code for Clustering Membership Value*

```
function [mv] = ClusteringMembership(x,t)
n1 = length(t);
if x <= t(1),
   mv(1) = 1;
elseif x <= t(2),
   mv(1) = ((x-t(2))^2)/((x-t(2))^2+(x-t(1))^2);
else
   mv(1) = 0;
end
if x >= t(n1-1) & x <= t(n1),
   mv(n1) = ((x-t(n1-1))^2)/((x-t(n1-1))^2+(x-t(n1))^2);
elseif x > t(n1),
   mv(n1) = 1;
else mv(n1) = 0;
end
for i = 2:(n1-1),
   if x >= t(i-1) & x <= t(i)
      mv(i) = ((x-t(i-1))^2)/((x-t(i-1))^2+(x-t(i))^2);
   elseif x >= t(i) & x <= t(i+1)
      mv(i) = ((x-t(i+1))^2)/((x-t(i+1))^2+(x-t(i))^2);
   else mv(i) = 0;
   end
end
return
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5.3 *Software implementation of study 1*

The first example taken from [15] as in [9], deals with the identification of a noisy time series.

$$x_j = 1.5x_{j-1}\exp\left(-\frac{x_{j-1}^2}{4}\right) + \grave{o}_j,\ \grave{o}_j \sim N(0,1)$$

$$y_j = x_j + v_j$$

Here v_j is generated by a gross error model defined as

$$F = (1-\grave{o})G + \grave{o}H$$

Where F is the noise distribution and G, H are the probability distributions that occur with probabilities $1-\grave{o}$ and $\grave{o}$, respectively. It is required to train a fuzzy model with noisy input-output pairs $\{y_{j-1}, y_j\}$ to approximate the true

function $f\left(x\right)=1.5x\exp\left(-\frac{x^{2}}{4}\right)$. The training data set consists of 200 input-output pairs generated from the true function. Algorithm 2 was used to infer the parameters of fuzzy filters combination.

```
function [] = study1()
N = 200;
x = zeros(200,1);
y = zeros(200,1);
for i = 1:200,
        if i > 1
                x(i) = 1.5*(x(i-1))*exp(-0.25*(x(i-1))^2)
                                                + randn(1,1);
        end
        G = (sqrt(0.05))*rand(1,1);
        H = (sqrt(3))*rand(1,1);
        tv = rand(1,1);
        if tv < 0.95,
                y(i) = x(i) + G;
        else
                y(i) = x(i) + H;
        end
end
inputs_data_matrix = y(1:199,1);
Y = y(2:200,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x_test = zeros(400,1);
x_test(1) = min(y);
y_test = zeros(400,1);
for i = 1:399,
        x_test(i+1) = x_test(1) + ((max(y)-min(y))/399)*i;
end
y_test = ProcessModel(x_test);
%%%%%%% models choices %%%%%%%%%%%%%
[FM{1},NFE(1),Param_a(1),Param_b(1),dl(1)]=
                        VBFuzzyFilter(inputs_data_matrix,Y,
    [min(y(1:199,1))max(y(1:199,1))],'clustering',[3],[0.1],1);
[FM{2},NFE(2),Param_a(2),Param_b(2),dl(2)]=
                        VBFuzzyFilter(inputs_data_matrix,Y,
    [min(y(1:199,1))max(y(1:199,1))],'clustering',[4],[0.1],1);
[FM{3},NFE(3),Param_a(3),Param_b(3),dl(3)]=
                        VBFuzzyFilter(inputs_data_matrix,Y,
    [min(y(1:199,1))max(y(1:199,1))],'clustering',[5],[0.1],1);
[FM{4},NFE(4),Param_a(4),Param_b(4),dl(4)]=
                        VBFuzzyFilter(inputs_data_matrix,Y,
    [min(y(1:199,1))max(y(1:199,1))],'clustering',[6],[0.1],1);
FE(1) = sqrt(mean((y_test-EvalFuzzyFilter(x_test,FM{1})).^2));
FE(2) = sqrt(mean((y_test-EvalFuzzyFilter(x_test,FM{2})).^2));
FE(3) = sqrt(mean((y_test-EvalFuzzyFilter(x_test,FM{3})).^2));
FE(4) = sqrt(mean((y_test-EvalFuzzyFilter(x_test,FM{4})).^2));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
S = length(FM);
%%%%%%% hyperparameters for priors%%%%%%%
Dirich_Param_c_o = S;
Dirich_Param_d_o = (1/S)*ones(1,S);
Gamma_Param_a_o = 1e+6;
Gamma_Param_b_o = 1e-6;
for i = 1:S,
    mean_o_struc{i} =  0*FM{i}.consequents.mean;
    Inv_Covar_o_struc{i}=eye(length(FM{i}.consequents.mean));
end
%%% Initialize the parameters of fuzzy filters mixture %%%
for i = 1:S,
   Inv_Covar_struc{i}=inv(FM{i}.consequents.Covar_matx);
        mean_struc{i} = FM{i}.consequents.mean;
    B_struc{i}=AntecedentMatrix(inputs_data_matrix,FM{i});
    r(i)=(Y-(B_struc{i})*(mean_struc{i}))'*(Y-(B_struc{i}) *
        (mean_struc{i})) + trace((inv(Inv_Covar_struc{i}))
                *(B_struc{i})'*(B_struc{i}));
   q(i) = exp(NFE(i)+max(abs(NFE)));
end
q = (1/sum(q))*q;
q
Gamma_Param_b = Gamma_Param_b_o + 0.5*N;
Gamma_Param_a = inv((1/Gamma_Param_a_o) + 0.5*q*r');
Dirich_Param_c = Dirich_Param_c_o + 1;
for i = 1:S,
        Dirich_Param_d(i)=(1/(Dirich_Param_c_o+1))*
                        ((Dirich_Param_c_o/S) + q(i));
end
close all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
F = [];
k = 1;
F(k)=NegativeFreeEnergy(Y,B_struc,Dirich_Param_c,
        Dirich_Param_d,Inv_Covar_struc,mean_struc,
Gamma_Param_a,Gamma_Param_b,q,Dirich_Param_c_o,
mean_o_struc,Inv_Covar_o_struc,Gamma_Param_a_o,
Gamma_Param_b_o);
del_q = 1000;
component_pmf(k,1:S) = q;
while (del_q > 0.01)
        qo = q;
        %------- update nonlinear parameters -------
         for i = 1:S,
            FM{i}=UpdateNonLinearParameters
                        (inputs_data_matrix,Y,FM{i},1);
            B_struc{i}=AntecedentMatrix(inputs_data_matrix,FM{i});
         end
        k = k+1;
        component_pmf(k,1:S) = component_pmf(k-1,1:S);
         F(k)=   NegativeFreeEnergy(Y,B_struc,
                 Dirich_Param_c,Dirich_Param_d,
                 Inv_Covar_struc,mean_struc,
                 Gamma_Param_a,Gamma_Param_b,q,
                 Dirich_Param_c_o,mean_o_struc,
                 Inv_Covar_o_struc,Gamma_Param_a_o,
                 Gamma_Param_b_o);
         if F(k) < F(k-1),
         disp('updating nonlinear parameters resulted in decrease');
        end
  %%update mean and covariance of linear parameters%%
         for i = 1:S,
           Inv_Covar_struc{i} = Inv_Covar_o_struc{i}+q(i)*
           Gamma_Param_a*Gamma_Param_b*(B_struc{i})'*
           B_struc{i};
           mean_struc{i}=(inv(Inv_Covar_struc{i}))*
           (Inv_Covar_o_struc{i}*mean_o_struc{i}+q(i)*
           Gamma_Param_a*Gamma_Param_b*
           (B_struc{i})'*Y);
          r(i)=   (Y-(B_struc{i})*(mean_struc{i}))'*
                  (Y-(B_struc{i})*
                  (mean_struc{i}))+
                  trace((inv(Inv_Covar_struc{i}))*
                  (B_struc{i})'* (B_struc{i}));
        FM{i}.consequents.mean = mean_struc{i};
        FM{i}.consequents.Covar_matx =
                        inv(Inv_Covar_struc{i});
        end
        k = k+1;
component_pmf(k,1:S) = component_pmf(k-1,1:S);
F(k)=NegativeFreeEnergy(Y,B_struc,Dirich_Param_c,
Dirich_Param_d,Inv_Covar_struc,mean_struc,Gamma_Param_a,
```

```
Gamma_Param_b,q,Dirich_Param_c_o,mean_o_struc,
Inv_Covar_o_struc,Gamma_Param_a_o,Gamma_Param_b_o);
        if F(k) < F(k-1),
            disp('updating mean and covariance resulted in
                                                decrease...');
        end
  %%%%---------- update q(si)----------%%%%
        for i = 1:S,
          q(i)=exp(psi(Dirich_Param_c*
                Dirich_Param_d(i)) - 0.5*

        Gamma_Param_a*Gamma_Param_b*r(i));
        end
        q = (1/sum(q))*q;
         k = k+1;
component_pmf(k,1:S) = q;
F(k)=NegativeFreeEnergy(Y,B_struc,Dirich_Param_c,
    Dirich_Param_d,Inv_Covar_struc,mean_struc,
    Gamma_Param_a,Gamma_Param_b,q,Dirich_Param_c_o,
    mean_o_struc,Inv_Covar_o_struc,Gamma_Param_a_o,
    Gamma_Param_b_o);
%%%------update gamma distribution parameters ------%%%
    Gamma_Param_b = Gamma_Param_b_o + 0.5*N;
    Gamma_Param_a = inv((1/Gamma_Param_a_o)+ 0.5*q*r');
    k = k+1;
component_pmf(k,1:S) = component_pmf(k-1,1:S);
F(k)=NegativeFreeEnergy(Y,B_struc,Dirich_Param_c,
     Dirich_Param_d,Inv_Covar_struc,mean_struc,
     Gamma_Param_a,Gamma_Param_b,q,Dirich_Param_c_o,
     mean_o_struc,Inv_Covar_o_struc,Gamma_Param_a_o,
     Gamma_Param_b_o);
        if F(k) < F(k-1),
        disp('updating a and b resulted in decrease...');
        end
        %---- update dirichlet distribution parameters---%
        Dirich_Param_c = Dirich_Param_c_o + 1;
        for i = 1:S,
          Dirich_Param_d(i) = (1/(Dirich_Param_c_o + 1))*
                            ((Dirich_Param_c_o/S) + q(i));
        end
        k = k+1;
component_pmf(k,1:S) = component_pmf(k-1,1:S);
F(k)=NegativeFreeEnergy(Y,B_struc,Dirich_Param_c,
     Dirich_Param_d,Inv_Covar_struc,mean_struc,
     Gamma_Param_a,Gamma_Param_b,q,Dirich_Param_c_o,
     mean_o_struc,Inv_Covar_o_struc,Gamma_Param_a_o,
     Gamma_Param_b_o);
        if F(k) < F(k-1),
            disp('updating c and d resulted in decrease...');
        end
        %-------------------------------------------------------%
subplot(211), plot([1:k],F(1:k)), ylabel('negative free energy'),
xlabel('iteration count');
subplot(212), semilogy(component_pmf,':','LineWidth',2),
ylabel('components probability'), xlabel('iteration count');
        pause(0.0001);
        del_q = 100*(max(abs(q-qo)));
end
q
optimal_model_index = find(q==max(q));
optimal_model_index
FEC=sqrt(mean((y_test-EvalFuzzyFilter(x_test,FM
                {optimal_model_index})).^2));
no_fuzzy_rules = optimal_model_index + 2;
no_parameters = no_fuzzy_rules*(3);
NFE
FE
sprintf('%f %f %f %f',no_fuzzy_rules,no_parameters,FEC,(k-1)/5)
figure;
plot(x_test,y_test,'color','red'), hold on,
plot(x_test,EvalFuzzyFilter(x_test,FM{optimal_model_index
})), plot(inputs_data_matrix,Y,'x'), hold off;
DrawMembershipFunction(FM{optimal_model_index});
return

function [y] = ProcessModel(x)
m = length(x);
y = zeros(m,1);
i = 1;
while (i <= m),
        y(i) = 1.5*(x(i))*exp(-0.25*(x(i))^2) ;
        i = i +1;
end
return
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5.4 *Software implementation of study 2*

The second example taken from [16] as in [9], where the aim is to approximate the following nonlinear function:

$$f(x)=\left[a_1+a_2\frac{x}{b}+a_3\left(\frac{2x^2}{b^2}-1\right)\right]\exp\left(-\frac{x^2}{2b^2}\right)$$

Where $a_1=1,\ a_2=2,\ a_3=-2,\ and\ b=1.8$. The training data set consists of 100 data pairs $\{x(j),y(j)\}_{j=1}^{100}$. Here, $x(j)$ is a random number generated from a uniform distribution on $[-10,10]$ and $y(j)=f(x(j))+n_j$, where n_j is a uniform random number on $[-2.5,2.5]$. The approximation quality was measured in term of root mean squared error (RMSE) on uniform grid of 201 points on $[-10,10]$.

```
function [] = study2()
clear all;
warning off all;
N = 100;
inputs_data_matrix = 10*(1-2*rand(N,1));
Yo = ProcessModel(inputs_data_matrix);
n = 2.5*(1-2*rand(N,1));
Y = Yo + n;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x_test = [-10:0.1:10]';
y_test = ProcessModel(x_test);
%%%% models choices %%%%%%%%%%%%
[FM{1},NFE(1),Param_a(1),Param_b(1),dl(1)]=
VBFuzzyFilter(inputs_data_matrix,Y,[-10  10],
'GaussianClustering',[4],[0.1],1);
[FM{2},NFE(2),Param_a(2),Param_b(2),dl(2)]=
VBFuzzyFilter(inputs_data_matrix,Y,[-10 10],
'GaussianClustering',[5],[0.1],1);
[FM{3},NFE(3),Param_a(3),Param_b(3),dl(3)]=
VBFuzzyFilter(inputs_data_matrix,Y,[-10 10],
'GaussianClustering',[6],[0.1],1);
[FM{4},NFE(4),Param_a(4),Param_b(4),dl(4)]=
VBFuzzyFilter(inputs_data_matrix,Y,[-10 10],
'GaussianClustering',[7],[0.1],1);
FE(1) = sqrt(mean((y_testEvalFuzzyFilter(x_test,FM{1})).^2));
FE(2) = sqrt(mean((y_test-EvalFuzzyFilter(x_test,FM{2})).^2));
FE(3) = sqrt(mean((y_test-EvalFuzzyFilter(x_test,FM{3})).^2));
FE(4) = sqrt(mean((y_test-EvalFuzzyFilter(x_test,FM{4})).^2));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
S = length(FM);
%%%%%%%%%% hyperparameters for priors %%%%%%
Dirich_Param_c_o = S;
Dirich_Param_d_o = (1/S)*ones(1,S);
Gamma_Param_a_o = 1e+6;
Gamma_Param_b_o = 1e-6;
for i = 1:S,
    mean_o_struc{i} = FM{i}.consequents.mean;
 Inv_Covar_o_struc{i}=inv(FM{i}.consequents.Covar_matx);
end
%%% Initialize the parameters of fuzzy filters mixture %%
for i = 1:S,

Inv_Covar_struc{i}=inv(FM{i}.consequents.Covar_matx);
mean_struc{i} = FM{i}.consequents.mean;
B_struc{i}=AntecedentMatrix(inputs_data_matrix,FM{i});
r(i)=(Y-(B_struc{i})*(mean_struc{i}))'*
(YB_struc{i})*(mean_struc{i}))+trace((inv(Inv_Covar_struc{i}))*
(B_struc{i})'*(B_struc{i}));
q(i) = exp(NFE(i)+max(abs(NFE)));
end
q = (1/sum(q))*q;
q
Gamma_Param_b = Gamma_Param_b_o + 0.5*N;
Gamma_Param_a = inv((1/Gamma_Param_a_o) + 0.5*q*r');
Dirich_Param_c = Dirich_Param_c_o + 1;
for i = 1:S,
Dirich_Param_d(i)=(1/(Dirich_Param_c_o+1))*
((Dirich_Param_c_o/S) + q(i));
end
close all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
F = [];
k = 1;
F(k)=NegativeFreeEnergy(Y,B_struc,Dirich_Param_c,
Dirich_Param_d,Inv_Covar_struc,mean_struc,Gamma_Param_a,
Gamma_Param_b,q,Dirich_Param_c_o,mean_o_struc,
Inv_Covar_o_struc,Gamma_Param_a_o,Gamma_Param_b_o);
del_q = 1000;
component_pmf(k,1:S) = q;
while (del_q > 0.01)
    qo = q;
    %----------- update nonlinear parameters -------%
    for i = 1:S,
    FM{i}=
    UpdateNonLinearParameters(inputs_data_matrix,Y,FM{i},1);
    B_struc{i}=AntecedentMatrix(inputs_data_matrix,FM{i});
    end
    k = k+1;
    component_pmf(k,1:S) = component_pmf(k-1,1:S);
    F(k)=NegativeFreeEnergy(Y,B_struc,Dirich_Param_c,
Dirich_Param_d,Inv_Covar_struc,mean_struc,Gamma_Param_a,
Gamma_Param_b,q,Dirich_Param_c_o,mean_o_struc,
Inv_Covar_o_struc,Gamma_Param_a_o,Gamma_Param_b_o);
        if F(k) < F(k-1),
            disp('updating nonlinear parameters
            resulted in decrease...');
        end
  %- update mean and covariance of linear parameters -%
        for i = 1:S,
            Inv_Covar_struc{i}=Inv_Covar_o_struc{i}+
        q(i)*Gamma_Param_a*Gamma_Param_b*
        (B_struc{i})'*B_struc{i};
        mean_struc{i}=(inv(Inv_Covar_struc{i}))*
        (Inv_Covar_o_struc{i}*mean_o_struc{i}+q(i)*
        Gamma_Param_a*Gamma_Param_b*(B_struc{i})'*Y);
        r(i) = (Y-
        (B_struc{i})*(mean_struc{i}))'*(Y(B_struc{i})*
        (mean_struc{i})) +trace((inv(Inv_Covar_struc{i}))*
        (B_struc{i})'*(B_struc{i}));
        FM{i}.consequents.mean = mean_struc{i};
        FM{i}.consequents.Covar_matx
                    =inv(Inv_Covar_struc{i});
        end
        k = k+1;
        component_pmf(k,1:S) = component_pmf(k-1,1:S);
        F(k)=NegativeFreeEnergy(Y,B_struc,Dirich_Param_c,
        Dirich_Param_d,Inv_Covar_struc,mean_struc,
        Gamma_Param_a,Gamma_Param_b,q,
        Dirich_Param_c_o,mean_o_struc,Inv_Covar_o_struc,
        Gamma_Param_a_o,Gamma_Param_b_o);
        if F(k) < F(k-1),
            disp('updating mean and covariance
            resulted in decrease...');
        end
        %---------- update q(si)---------------%
        for i = 1:S,
        q(i) = exp(psi(Dirich_Param_c*Dirich_Param_d(i)) - 0.5*
        Gamma_Param_a*Gamma_Param_b*r(i));
        end
        q = (1/sum(q))*q;
        k = k+1;
        component_pmf(k,1:S) = q;
        F(k)=NegativeFreeEnergy(Y,B_struc,Dirich_Param_c,
        Dirich_Param_d,Inv_Covar_struc,mean_struc,
        Gamma_Param_a,Gamma_Param_b,q,
        Dirich_Param_c_o,mean_o_struc,Inv_Covar_o_struc,
        Gamma_Param_a_o,Gamma_Param_b_o);
        %---- update gamma distribution parameters ---%
        Gamma_Param_b = Gamma_Param_b_o + 0.5*N;
        Gamma_Param_a = inv((1/Gamma_Param_a_o) +
                    0.5*q*r');
        k = k+1;
        component_pmf(k,1:S) = component_pmf(k-1,1:S);
        F(k)=NegativeFreeEnergy(Y,B_struc,Dirich_Param_c,
        Dirich_Param_d,Inv_Covar_struc,mean_struc,Gamma_Param_a,
        Gamma_Param_b,q,Dirich_Param_c_o,mean_o_struc,
        Inv_Covar_o_struc,Gamma_Param_a_o,Gamma_Param_b_o);
        if F(k) < F(k-1),
        disp('updating a and b resulted in decrease...');
        end
        %--------------------------------------------------------
        %---- update dirichlet distribution parameters --
        Dirich_Param_c = Dirich_Param_c_o + 1;
        for i = 1:S,
        Dirich_Param_d(i) = (1/(Dirich_Param_c_o + 1))*
        ((Dirich_Param_c_o/S) + q(i));
        end
        k = k+1;
        component_pmf(k,1:S) = component_pmf(k-1,1:S);
        F(k)=NegativeFreeEnergy(Y,B_struc,Dirich_Param_c,
        Dirich_Param_d,Inv_Covar_struc,mean_struc,
        Gamma_Param_a,Gamma_Param_b,q,Dirich_Param_c_o,
        mean_o_struc,Inv_Covar_o_struc,Gamma_Param_a_o,
        Gamma_Param_b_o);
        if F(k) < F(k-1),
        disp('updating c and d resulted in decrease...');
        end
  %--------------------------------------------------------%
subplot(211), plot([1:k],F(1:k)), ylabel('negative free
eneergy'), xlabel('iteration count');
subplot(212), semilogy(component_pmf,':','LineWidth',2),
ylabel('components probability'), xlabel('iteration count');
pause(0.0001);
del_q = 100*(max(abs(q-qo)));
```

```
end
q
optimal_model_index = find(q==max(q));
optimal_model_index
FEC =
sqrt(mean((y_testEvalFuzzyFilter(x_test,FM{optimal_model_index})).^2));
no_fuzzy_rules = optimal_model_index + 3;
no_parameters = no_fuzzy_rules*(3+1);
NFE
FE
sprintf('%f %f %f %f',no_fuzzy_rules,no_parameters,FEC,(k-1)/5)
figure;
plot(x_test,y_test,'color','red'), hold on,
plot(x_test,EvalFuzzyFilter(x_test,FM{optimal_model_index})),
plot(inputs_data_matrix,Y,'x'), hold off;
DrawMembershipFunction(FM{optimal_model_index});
return
function [y] = ProcessModel(x)
m = length(x);
y = zeros(m,1);
i = 1;
while (i <= m),
        y(i) = (1 + (2/1.8)*x(i) - 2*((2*(x(i))^2)/((1.8)^2)) +
        2)* exp(-(0.5*(x(i))^2)/((1.8)^2));
         i = i +1;
end
return
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5.5 *Software implementation of study 3*

A real-world example of modeling an ECG signal as in [9] was taken from [16]. The ECG signal data belong to the MITBIH database. As in [16], the fuzzy model was trained with 450 data pairs and tested on another 5000 pairs. Algorithm 3 was used to infer the parameters of a fuzzy filter that define three different clusters on the input data.

```
function [] = study3()
clear all;
warning off all;
[trndata,testdata] = study3_data;
d = size(trndata,2)-1;
inputs_data_matrix = trndata(:,1:d);
Y = trndata(:,d+1);
inputs_range_matrix=[(min(trndata(:,1:d)))'(max(trndata(:,1:d)))'];
[FM,NFE,Gamma_Param_a,Gamma_Param_b,dl]=
VBFuzzyFilter(trndata(:,1:d),trndata(:,d+1),
[(min(trndata(:,1:d)))'(max(trndata(:,1:d)))'],
'GaussianClustering',[3 ones(1,d-1)],0.1*ones(1,d),1);
no_fuzzy_rules = 3;
no_parameters = no_fuzzy_rules*(3*d+1);
FE =
sqrt(mean((testdata(:,d+1)EvalFuzzyFilter(testdata(:,1:d),FM)).^2));
sprintf('The number of fuzzy rules = %f',no_fuzzy_rules)
sprintf('The number of model pareameters = %f',no_parameters)
sprintf('RMSE = %f',FE)
figure;
plot(testdata(:,d+1)-EvalFuzzyFilter(testdata(:,1:d),FM));
DrawMembershipFunction(FM)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

5.6 *Implementation of Variational Bayesian fuzzy filter.*

```
function [FM,F_end,Gamma_Param_a,Gamma_Param_b,dl]
=VBFuzzyFilter(inputs_data_matrix,Y,inputs_range_matrix,
membership_type,number_memberships_for_each_input,min
_gap_knots_for_each_input,model_order)
N = length(Y);
if strcmp(membership_type,'GaussianClustering')
        [FM]=InitializeFuzzyFilterClustering(inputs_data_
matrix,inputs_range_matrix,prod(number_memberships_for_e
ach_input),model_order);
else
        [FM]=InitializeFuzzyFilter(inputs_range_matrix,me
mbership_type,number_memberships_for_each_input,min_ga
p_knots_for_each_input,model_order);
end
%%%%% hyperparameters for priors%%%%%%%%%%
Gamma_Param_a_o = 1e+6;
Gamma_Param_b_o = 1e-6;
mean_o_struc = FM.consequents.mean;
Inv_Covar_o_struc = eye(length(FM.consequents.mean));
%%%% Initialize the parameters of fuzzy filters mixture %%
Inv_Covar_struc = Inv_Covar_o_struc;
mean_struc =  mean_o_struc;
FM.consequents.mean = mean_struc;
FM.consequents.Covar_matx = inv(Inv_Covar_struc);
B_struc = AntecedentMatrix(inputs_data_matrix,FM);
Gamma_Param_a = Gamma_Param_a_o;
Gamma_Param_b = Gamma_Param_b_o;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
k = 1;
F(k)=
NegativeFreeEnergySingle(Y,B_struc,Inv_Covar_struc,
mean_struc,Gamma_Param_a,Gamma_Param_b,mean_o_struc,
Inv_Covar_o_struc,Gamma_Param_a_o,Gamma_Param_b_o);
del_F = 1000;
while (del_F > 0.01)
        Fo = F(k);
         %------- update nonlinear parameters -------%
        FM=
        UpdateNonLinearParameters(inputs_data_matrix,Y,FM,1);
        B_struc =
        AntecedentMatrix(inputs_data_matrix,FM);
         k = k+1;
        F(k)=
        NegativeFreeEnergySingle(Y,B_struc,Inv_Covar_struc,
        mean_struc,Gamma_Param_a,Gamma_Param_b,
        mean_o_struc,Inv_Covar_o_struc,Gamma_Param_a_o,
        Gamma_Param_b_o);
        if F(k) < F(k-1),
                disp('updating nonlinear parameters
        resulted in decrease...');
        end
%---- update mean and covariance of linear parameters ----%
        Inv_Covar_struc = Inv_Covar_o_struc +
        Gamma_Param_a*Gamma_Param_b*(B_struc)'*
        B_struc;
        mean_struc=(inv(Inv_Covar_struc))*
        (Inv_Covar_o_struc*mean_o_struc+Gamma_Param_a*
        Gamma_Param_b*B_struc'*Y);
        FM.consequents.mean = mean_struc;
       FM.consequents.Covar_matx=inv(Inv_Covar_struc);
         k=k+1;   F(k)=
        NegativeFreeEnergySingle(Y,B_struc,Inv_Covar_struc,
        mean_struc,Gamma_Param_a,Gamma_Param_b,
        mean_o_struc,Inv_Covar_o_struc,Gamma_Param_a_o,
        Gamma_Param_b_o);
        if F(k) < F(k-1),
                disp('updating mean and covariance
                resulted in decrease...');
         end
        %-- update gamma distribution parameters ---%
         Gamma_Param_b = Gamma_Param_b_o + 0.5*N;
         r=(YB_struc*mean_struc)'*(YB_struc*mean_struc) +
        trace((inv(Inv_Covar_struc))*B_struc'*B_struc);
```

```
        Gamma_Param_a = inv((1/Gamma_Param_a_o) +
        0.5*r);
        k = k+1;  F(k)=
        NegativeFreeEnergySingle(Y,B_struc,Inv_Covar_struc,
        mean_struc,Gamma_Param_a,Gamma_Param_b,
        mean_o_struc,Inv_Covar_o_struc,Gamma_Param_a_o,
        Gamma_Param_b_o);
        if F(k) < F(k-1),
                disp('updating a and b resulted in
                decrease...');
         end
        %-------------------------------------------------------%
semilogy(1:k,F(1:k)), ylabel('negative free eneergy'),
xlabel('iteration count');
        pause(0.0001);
        del_F = 100*(abs(F(k)-Fo)/abs(Fo));
end
F_end = F(end);
dl=DataLikelihoodSingle(Y,B_struc,Inv_Covar_struc,mean_s
truc, Gamma_Param_a,Gamma_Param_b);
sprintf('The number of iterations = %f',(k-1)/3)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
return
```

6. CONCLUDING REMARKS

The paper present software implementation of Variation Bayesian approach to mixed deterministic/stochastic fuzzy models. A potential application of developed software is in the analysis of biomedical signals, where a stochastic fuzzy model is inferred from the biomedical data. The details have been provided as in [10].

7. REFERENCES

[1] L. A. Zadeh, Fuzzy Sets, Information and Control, 1965.

[2] L. A. Zadeh, "Outline of a New Approach to the Analysis of Complex Syestems and Decision Processes," *IEEE Trans. Syst., Man, Cybern*., vol. SMC-3, no. 1, pp. 28-44 Jan. 1973.

[3] T. TAKAGI and M. SUGENO, "Fuzzy identification of system and its application to modelling and control," *IEEE Trans. Syst., Man, Cybern*., vol. 15, no. 1, pp. 116-132, 1985.

[4] L. A. Zadeh, "The role of fuzzy logic in the management of uncertainty in expert systems," *Fuzzy Sets Syst.*, vol. 11, pp. 199-227, 1983.

[5] M. Kumar, D. Arndt, S. Kreuzfeld, K. Thurow, N. Stoll, and R. Stoll, "Fuzzy techniques for subjective workload score modelling under uncertainties," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 6, pp. 1449-1464, Dec. 2008.

[6] M. Kumar, M. Weippert, R. Vilbrandt, S. Kreuzfeld, and R. Stoll, "Fuzzy evaluation of heart rate signals for mental stress assessment," *IEEE Transaction on Fuzzy Systems*, vol. 15, no.5, pp. 791-808, 2007

[7] J.-S.R. jang, "ANFIS: Adaptive-network-based fuzzy inference systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665-685, May 1993.

[8] L. Ljung, System Identification, Theory for the User. New Jersey: Prentice-Hall, 1987.

[9] M.Kumar, N. Stoll, and R. Stoll, "Variational Bayes for a Mixed Stochastic/Determinstic Fuzzy Filter," *IEEE Transactions on Fuzzy Systems*, vol. 18, no.4, pp. 787-801, 2010.

[10] M. Kumar, M. Weippert, N. Stoll, and R. Stoll, "A mixture of fuzzy filters applied to the analysis of heartbeat intervals," *Fuzzy Optim. Decis. Making*, vol. 9, no. 4, pp. 383-412, 2010.

[11] H. Attias, "A Variational Bayesian framework for graphical models," In Advances In Neural Information Processing Systems 12. Cambridge, MA: MIT Press, 2000, pp. 209-215.

[12] H. Lappalainen and J. W. Miskin, "Ensemble learning," in Advances in independent component Analysis, M. Girolami, Ed. New York: Springer-Verlag, 2000.

[13] MATLAB-The Language of Technical Computing [Online]. Available:http://www.mathworks.com/products/matlab/

[14] Optimization Toolbox—MATLAB [Online]. Available: http://www.mathworks.com/products/optimization/

[15] C. C. Chuang, S. F. Su, and S. S. Chen, "Robust TSK fuzzy modeling for function approximation with outliers," *IEEE Trans. Fuzzy Syst*., vol. 9, no. 6, pp. 810-821, Dec. 2001.

[16] J. M. Leski, "TSK-fuzzy modeling based on ò-intensive learning," *IEEE Trans. Fuzzy Syst*., vol. 13, no. 2, pp. 181-193, Apr. 2005.