Development and Maintenance of Fuzzy Models in Financial Applications

Piero P. Bonissone

General Electric Global Research Center Schenectady, NY 12309, USA
bonissone@research.ge.com

Our goal is to illustrate the typical life cycle of a fuzzy knowledge-based model, starting from its development, testing, optimization, and deployment, and ending with the maintenance of its knowledge base. We illustrate this process within the context of an underwriting insurance application. First we define some key concepts of soft computing models and discuss some design tradeoffs that must be addressed. Then we focus on the design and implementation of a fuzzy rule-based classifier (FRC). We establish a standard reference dataset (SRD), consisting of 3,000 insurance applications with their corresponding decisions. The SRD exemplifies the results achieved by an ideal, optimal classifier, and represents the target for our design. We apply evolutionary algorithms to perform an off-line optimization of the design parameters of the classifier, modifying its behavior to approximate this target. The SRD is also used as a reference for testing and performing a five-fold cross-validation of the classifiers. Finally, we focus on the monitoring and maintenance of the FRC. We describe a fusion architecture that supports an off-line quality assurance process of the on-line FRC. The fusion module takes the outputs of multiple classifiers, determines their degree of consensus, and compares their overall agreement with the decision made by the FRC. From this analysis, we can identify the most suitable cases to update the SRD, to audit, or to be reviewed by senior underwriters.

1 Introduction

1.1 Soft Computing

The literature of Soft Computing (SC) is expanding at a rapid pace, as evidenced from the numerous congresses, books, and journals devoted to this issue. Its original definition provided by Zadeh [23]) denotes systems that "... exploit the tolerance for imprecision, uncertainty, and partial truth to achieve

tractability, robustness, low solution cost, and better rapport with reality." As discussed in previous articles (Bonissone [3]; Bonissone *et al.* [4]), we view soft computing as the synergistic association of computing methodologies that includes as its principal members *fuzzy logic*, *neuro-computing*, *evolutionary computing* and *probabilistic computing*. We also stress the synergy derived from hybrid SC systems that are based on a loose or tight integration of their constituent technologies. This integration provides complementary reasoning and search methods that allow us to combine domain knowledge and empirical data to develop flexible computing tools and solve complex problems.

1.2 Characteristics of Real World Applications

Integration of Knowledge and Data

When addressing real-world problems, we realize that we are dealing with systems that are typically ill defined, difficult to model, and possess large solution spaces. In these cases, precise models are usually impractical, too expensive, or non-existent. Therefore, we need to generate approximate solutions by leveraging the two types of resources that are generally available: *problem domain knowledge* of the process (or product) and *field data* that characterize the system's behavior. The relevant available domain knowledge is typically a combination of first principles and empirical knowledge. This knowledge is often incomplete and sometimes erroneous. The available data are typically a collection of input-output measurements, representing instances of the system's behavior, and are generally incomplete and noisy. Soft computing is a flexible framework in which we can find a broad spectrum of design choices to perform the integration of knowledge and data in the construction of approximate models.

Need to Support Model Maintenance

In real-world applications before we can use a model in a production environment we must address the model's entire life cycle, from its design and implementation, to its validation, tuning, production testing, use, monitoring and maintenance. By maintenance we mean all the steps required to keep the model vital (e.g., non-obsolete) and able to adapt to changes. Two reasons justify our focus on maintenance. Over the life cycle of the model, maintenance costs are the most expensive component (as software maintenance is the most expensive life cycle component of software). Secondly, when dealing with mission-critical software we need to guarantee continuous operations or at least fast recovery from system failures or model obsolescence to avoid lost revenues and other business costs.

1.3 Structure of paper

In the next section, we describe a variety of soft computing models and some typical search methods that could be used to generate those models. Then we focus on the insurance underwriting process, the problem domain used to illustrate the life cycle of a fuzzy knowledge-based model. Within this context, we further focus on models that assess risk using a fixed resolution (rate classes). Therefore, in this case study, the model is a discrete classifier. However, many of the concepts illustrated in this paper can be easily extended to situations in which the model is used as a predictor of continuous values, rather than a classifier. In Section 4, we provide a brief description of the design of a fuzzy rule-based classifier. In Sections 5 through 7, we describe the steps required to ensure the stable operation of the classifiers: the establishment of a standard reference dataset (SRD) to provide an ideal reference; the optimization of the classifiers' design parameters to ensure their performance; their cross-validation to ensure their robustness; the run-time monitoring of their decisions to track stable operations; and an off-line quality assurance process, based on the fusion of multiple classifiers, to enhance the SRD and identify questionable cases for manual audit.

2 Models and Fuzzy Models

2.1 Model Generation

In general terms, we can consider a model to be characterized by its *representation* (structural and parametric information) and its associated *reasoning mechanism*, which is usually related to the representation. The generation (and updating) of an optimal model requires a *search* method to define the model's representation and to characterize its reasoning mechanism. We will illustrate this concept with examples taken from conventional and soft computing models.

Differential Equations

Classical engineering models based on linear differential equations have a representation that can be decomposed into a *structure* (the order of the differential equation that determines the number of required coefficients), and a set of *parameters*, (the value of those coefficients). The *reasoning mechanism* is based on the (exact or approximated) solution of the differential equations. The *search* method used to generate the model could be based on energy-balance approaches (e.g., Bond Graphs), least mean squared error minimization, etc.

Bayesian Belief Networks and Neural Networks

For graphical models we have a similar situation. In the case of Bayesian Belief Networks (BBN's) their *structure* (graph topology) and *parameters* (prior probabilities on the node and conditional probabilities on the links) define the BBN's representation. Their underlying *reasoning mechanism* is based on conditioning, and consists of calculating and propagating posterior probabilities along the network topology, according to Bayes' rule. The *search* method used to generate the BBN's ranges from manual construction, to EM algorithms, evolutionary algorithms (EA's) (Larrañaga and Lozano [16]), et cetera. Other graphical models, such as Neural Networks (NN's), have a similar representation: their *structure* is the network topology, while their *parameters* are the biases on the nodes and weights on the links. *Reasoning* with NN's is a similar process, consisting in evaluating and propagating values along the network topology according to the operators defined in each neuron. The *search* method used to generate the NN structure could be manual, forward or backward pruning, evolutionary algorithms (Vonk *et al.* [21], Yao [11]), while local search methods such as backpropagation, conjugate gradient, etc., or global search approaches, such as evolutionary algorithms could be used to generate the NN parameters.

Fuzzy Systems and Radial Basis Functions

Many knowledge-based models, which on the surface might appear quite different from graphical models, can be easily transformed and represented graphically. For instance, the domain knowledge in fuzzy systems is usually represented by a set of *if-then rules* that approximate a mapping from a state space $\overrightarrow{X}$ to an output space $\overrightarrow{Y}$. In a Mamdani-type fuzzy system (Mamdani and Assilian [17]) the KB is completely defined by a set of scaling factors, determining the ranges of values for the state and output variables; a term set, defining the membership function of the values taken by each state and output variable and a rule set, characterizing a syntactic mapping of symbols from $\overrightarrow{X}$ to $\overrightarrow{Y}$. The *structure* of the underlying model is the rule set, while the *parameters* are the scaling factors and term sets. The reasoning mechanism is based on *generalized modus ponens*, and consists in interpolating among the outputs of all relevant rules. A Takagi-Sugeno-Kang (TSK) type of fuzzy system (Takagi and Sugeno [19]) increases its representational power by allowing the use of a first-order polynomial (defined on the state space), to be the output of each rule in the rule set. This enhanced representational power, at the expense of local legibility (Babuska *et al.* [1]) results in a model that is equivalent to Radial Basis Functions (RBF's) (Bersini *et al.* [2]). RBF's have a topology similar to classical NN's with the sigmoid node replaced by one containing a localized distribution. The same model can be translated into a structured network, such as the adaptive neural fuzzy inference systems (ANFIS) (Jang [15]). In ANFIS, the rule set determines the topology of the net

(model structure), while dedicated nodes in the corresponding layers of the net (model parameters) define the term sets and the polynomial coefficients. In all these cases, the *search* method for the structure is either manual or based on evolutionary algorithms, while the determination of the parameters is usually based on local (backpropagation) or global (EA's) search.

Instance-based Models

Non-graphical models, such as instance-based models, follow a similar pattern. Their representation contains *structural* information (such as the dimensionality and definition of their attribute space), and *parametric* information (such as the weights assigned to each attribute, etc.). Their *reasoning* mechanism is based on a retrieval of instances, a similarity-driven ordering of the instances, the evaluation and potential modification of their outcomes, and an aggregation of the (modified) outputs. This mechanism might require the value of some parameters, such as the relevance of each attribute in computing similarity, the extent of the retrieval, the type of aggregation, etc. For simplicity, we will include all these parameters in the model representation.

2.2 Search

Regardless of the underlying representation of the model, Evolutionary Algorithms can provide a robust, global search method to define the models' structure and parameters. Given a fixed model structure, using a wrapper approach, EA's can be used to determine the inputs and pre-processors (attribute selection, weighing, and construction) as well as the parameter values required by the models. The appealing aspect of this approach is the ease of incorporating knowledge in the EA to control the search, as described by Bonissone *et al.* [7]. Furthermore, the same methodology can be applied after deployment to retune and or reconfigure the model, hence extending its vitality. It is also possible to intertwine global search (more robust but less efficient) with local search, such as greedy induction, gradient-based techniques, etc, in the quest to design and maintain better models. In the following sections, we will illustrate the application of EA's in the development and maintenance of optimal fuzzy classifiers.

3 Problem description

3.1 The Classification Problem

To illustrate the life cycle of a classifier, we will use a representative, challenging classification problem: the process of underwriting insurance applications. Insurance underwriting is a complex decision-making task that is traditionally

performed by trained individuals. An underwriter must evaluate each insurance application in terms of its potential risk for generating a claim, such as mortality in the case of term life insurance. An application is compared against standards adopted by the insurance company, which are derived from actuarial principles related to mortality. Based on this comparison, the application is classified into one of the risk categories available for the type of insurance requested by the applicant. The accept/reject decision is also part of this risk classification, since risks above a certain tolerance level will typically be rejected. The estimated risk, in conjunction with other factors such as gender, age, and policy face value, will determine the appropriate price (premium) for the insurance policy. When all other factors are the same, to retain the fair value of expected return higher risk entails higher premium.

We represent an insurance application as an input vector $\overrightarrow{X}$ that contains a combination of discrete, continuous, and attribute variables. These variables represent the applicant's medical and demographic information that has been identified by actuarial studies to be pertinent to the estimation of the applicant's claim risk. Similarly, we represent the output space $\overrightarrow{Y}$, e.g. the underwriting decision space, as an ordered list of rate classes. Due to the intrinsic difficulty of representing risk as a real number on a scale, e.g., 97% of nominal mortality, the output space $\overrightarrow{Y}$ is subdivided into bins (rate classes) containing similar risks. For example 96%-104% nominal mortality could be labeled the *Standard* rate class. Therefore, we consider the underwriting (UW) process as a discrete classifier mapping an input vector $\overrightarrow{X}$ into a discrete decision space $\overrightarrow{Y}$, where $|\overrightarrow{X}| = n$ and $|\overrightarrow{Y}| = T$.

This problem is complicated by several requirements: *a)* the *nonlinearity* of the UW mapping; *b)* the need for inputs *interpretations*; *c)* the *flexibility* required to balance *risk-tolerance*, to preserve price competitiveness, and *risk-avoidance*, to prevent overexposure to risk; *d)* legal and compliance regulations that require the models to be *transparent* and *interpretable*. To address these requirements we decided to develop a hybrid soft computing system, based on fuzzy logic and evolutionary algorithms. With this system we were able to provide flexibility and consistency, while maintaining interpretability and accuracy as part of an underwriting and a risk management platform.

4 Design and Implementation

4.1 Design Tradeoffs

In the development of a classifier, we usually face design trade-offs, such as *accuracy-versus-coverage* and *accuracy-versus-interpretability* (Guillaume [14]; Casillas *et al.* [10]). The first trade-off is similar to the *precision-versus-recall* compromise found in the design of information retrieval systems. We could tune a classifier to maximize the number of correct decisions it produces, declining to make any commitment if we are not confident about the

conclusion. This behavior would increase accuracy at the expense of coverage. Alternatively, we could tune the same classifier to always issue a decision for each case, thus increasing coverage at the expense of accuracy. The second trade-off, typically dictated by legal or compliance regulations, constrains the underlying technologies used to implement the classifier. When we use SC techniques, the equation **"model = structure + parameters (& search)"**, takes on a different connotation, as we have a much richer repertoire to represent the structure, to tune the parameters, and to iterate this process (Bonissone *et al.* [4]). This repertoire enables us to choose among different trade-offs between the model's interpretability and its accuracy. For instance, one approach aimed at maintaining the model's transparency usually starts with *knowledge-derived linguistic models*, in which domain knowledge is translated into an initial structure and parameter values. The model's accuracy is further improved by using global or local data-driven search methods to tune the structure and/or parameters. An alternative approach, aimed at building more accurate models might start with *data-driven search methods*. Then we can embed domain knowledge into the search operators to control or limit the search space, or to maintain the model's interpretability. Post-processing approaches can also be used to extract explicit structural information from the models. The commonalities among these models are the tight integration of knowledge and data, leveraged in their construction, and the loose integration of their outputs, exploited in their off-line use.

4.2 Fuzzy Rule-Based Classifier (FRC)

The fuzzy rule-based and case-based classifiers have been briefly described in (Bonissone *et al.* [5]). The Fuzzy Rule-based Classifier (FRC) uses rule sets to encode underwriting standards. Each rule set represents a set of fuzzy constraints defining the boundaries between rate classes. These constraints were first determined from the underwriting guidelines. They were then refined using knowledge engineering sessions with expert underwriters to identify factors such as blood pressure levels and cholesterol levels, which are critical in defining the applicant's risk and corresponding premium. The goal of the classifier is to assign an applicant to the most competitive rate class, providing that the applicant's vital data meet all of the constraints of that particular rate class to a minimum degree of satisfaction. The constraints for each rate class r are represented by n fuzzy sets: $A_i^r(x_i)$, $i = 1, \ldots, n$. Each constraint $A_i^r(x_i)$ can be interpreted as the *degree of preference* induced by value x_i, for satisfying constraint A_i^r. After evaluating all constraints, we compute two measures for each rate class r. The first one is the degree of intersection of all the constraints and measures the *weakest* constraint satisfaction[1]:

[1]This expression implies that each criterion has equal weight. If we want to attach a weight $\omega_i \in [0, 1]$ to each criterion A_i we could use the weighted minimum operator $\Gamma(r) = \cap_{i=1}^{n} \omega_i A_i^r(x_i) = \min_{i=1}^{n} \left[\max\left((1 - \omega_i), A_i^r(x_i)\right)\right]$.

$$I(r) = \bigcap_{i=1}^{n} A_i^r(x_i) = \min_{i=1}^{n} A_i^r(x_i).$$

The second one is a cumulative measure of missing points (the complement of the average satisfaction of all constraints), and measures the *overall tolerance* allowed to each applicant i.e.:

$$MP(r) = \sum_{i=1}^{n} \left(1 - A_i^r(x_i)\right) = n\left(1 - \frac{1}{n}\sum_{i=1}^{n} A_i^r(x_i)\right) = n(1 - \overline{A^r}).$$

The final classification is obtained by comparing the two measures, $I(r)$ and $MP(r)$ against two lower bounds defined by thresholds τ_1 and τ_2. The parametric definition of each fuzzy constraint $A_i^r(x_i)$ and the values of τ_1 and τ_2 are design parameters that were initialized with knowledge engineering sessions.

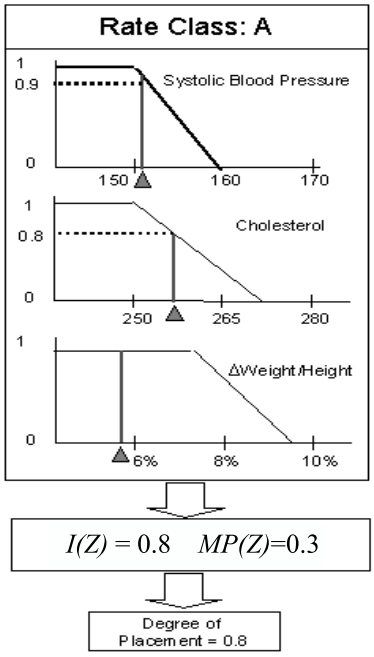

Fig. 1. Example of three fuzzy constraints for rate class Z

Figure 1 illustrates an example of three constraints (trapezoidal membership functions) associated with rate class Z, the input data corresponding to an application, and the evaluation of the first measure, indicating the weakest degree of satisfaction of all constraints.

5 Optimization of Design Parameters of the FRC Classifier

The FRC design parameters must be tuned, monitored, and maintained to assure the classifier's optimal performance. To this end, we have chosen to use EA's. Our EA is composed of a population of individuals or *chromosomes*. Each chromosome contains a vector of elements that represent distinct tunable parameters to configure the FRC classifier, i.e., the parametric definition of the fuzzy constraints $A_i^r(x_i)$ and thresholds τ_1 and τ_2.

A chromosome, the genotypic representation of an individual, defines a complete parametric configuration of the classifier. Thus, an instance of such classifier can be initialized for each chromosome, as shown in Figure 2. Each chromosome c_i, of the population $P(t)$ (left-hand side of Figure 2), goes through a decoding process to allow them to initialize the classifier on the right. Each classifier is then tested on all the cases in the case base, assigning a rate class to each case. We can determine the quality of the configuration encoded by the chromosome (the "fitness" of the chromosome) by analyzing the results of the test. Our EA uses mutation (randomly varying parameters of a single chromosome) to produce new individuals in the population. The more fit chromosomes in generation t will be more likely to be selected for this and pass their genetic material to the next generation $t + 1$. Similarly, the less fit solutions will be culled from the population. At the conclusion of the EA's execution the *best* chromosome of the *last* generation determines the classifier's configuration.

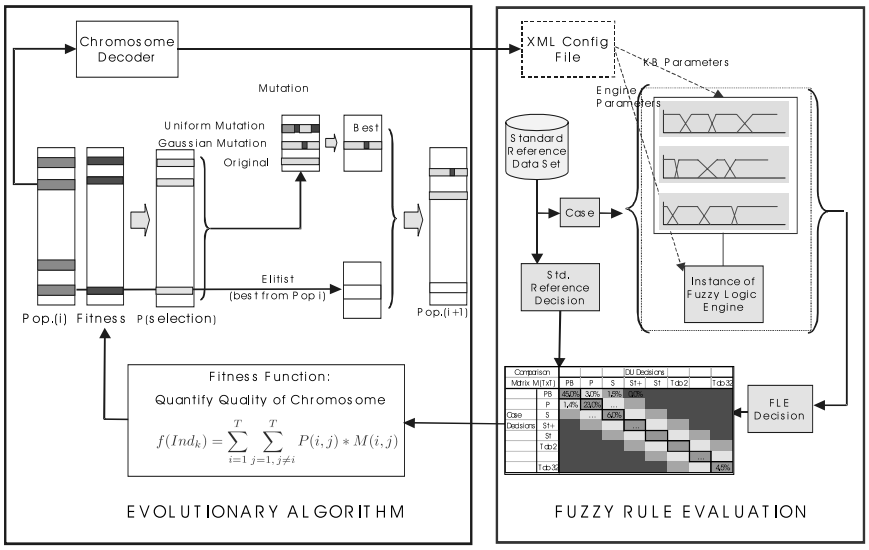

Fig. 2. FRC Optimization Using EA

5.1 Standard Reference Dataset (SRD)

To test and tune the classifiers, we need to establish a benchmark. Therefore, we generated a *standard reference dataset* (SRD) of approximately 3,000 cases taken from a stratified random sample of the historical case population. Each of these cases received a rate category decision when it was originally underwritten. However, to reduce variability in these decisions a team of experienced underwriters performed a *blind* review of selected cases to determine the *standard reference* decisions. These cases were then used to create and optimize the FRC model.

5.2 Fitness Function

In discrete classification problems we can use two matrices to construct the fitness function that we want to optimize. The first matrix is a $T \times T$ *confusion matrix* M that contains frequencies of correct and incorrect classifications for all possible combinations of the Standard Reference Decisions (SRD)[2] and classifier decisions. The first $(T-1)$ columns represent the rate classes available to the classifier. Column T represents the classifier's choice of not assigning any rate class, sending the case to a human underwriter. The same ordering is used to sort the rows for the SRD. The second matrix is a $T \times T$ *penalty matrix* P that contains the cost of misclassification. The fitness function f combines the values of M, resulted from a test run of the classifier configured with chromosome c_i, with the penalty matrix P to produce a single value:

$$f(c_i) = \sum_{j=i}^{T} \sum_{k=i}^{T} M(j,k) * P(j,k).$$

. Function f represents the overall misclassification cost.

6 Testing and Validation of FRB

After defining measures of coverage, relative, and global accuracy[3], we performed a comparison against the SRD. The results, partially reported in reference (Bonissone *et al.* [5]), show a remarkable improvement in all measures. Specifically we obtained the following results:

[2]Standard Reference Decisions represent ground truth rate class decisions as reached by consensus among senior expert underwriters for a set of insurance applications.

[3]*Coverage*: Percentage of cases as a fraction of the total number of input cases; *Relative Accuracy*: Percentage of correct decisions on those cases that were not referred to the human underwriter; *Global Accuracy*: Percentage of correct decisions, including making correct rate class decisions and making a correct decision to refer cases to human underwriters as a fraction of total input cases

Table 1. Typical performance of the un-tuned and tuned rule-based decision system (FRC)

METRIC	Initial parameters based on written guidelines	Best knowledge engineered parameters	Optimized parameters
Coverage	94.01%	90.38%	91.71%
Relative Accuracy	75.92%	92.99%	95.52%
Global Accuracy	74.75%	90.07%	93.63%

Using the initial parameters (first column of Table 1) we can observe a large moderate *Coverage* ($\sim$ 94%) associated with a low *Relative Accuracy* ($\sim$ 76%) and a lower *Global Accuracy* ($\sim$ 75%). These performance values are the result of applying a strict interpretation of the UW guidelines, without allowing for any tolerance. Had we implemented such crisp rules with a traditional rule-based system, we would have obtained these same evaluations. This strictness would prevent the insurer from being price competitive, and would not represent the typical *modus operandi* of human underwriters. However, by allowing each underwriter to use his/her own interpretation of such guidelines, we could introduce a large underwriters' variability. One of our main goals of this project was to provide a *uniform* interpretation, while still *allowing for some tolerance*. This goal is addressed in the second column of Table 1, which shows the results of performing *knowledge engineering* and encoding the desired tradeoff between risk and price competitiveness as fuzzy constraints with *preference* semantics. This intermediate stage shows a different tradeoff since both *Global* and *Relative Accuracy* have improved. *Coverage* slightly decreases ($\sim$ 90%) for a considerable gain in *Relative Accuracy* ($\sim$ 93%). Although we obtained this initial parameter set by interviewing the experts, we had no guarantee that such parameters were optimal. Therefore, we used an evolutionary algorithm to tune them. We allowed the parameters to move within a predefine range centered around their initial values and, using the SRD and the fitness function described above, we obtained an optimized parameter set, whose results are described in the third column of Table 1. The results of the optimization show the point corresponding to the final parameter set dominates the second set point (in a Pareto sense), since both *Coverage* and *Relative Accuracy* were improved. Finally, we can observe that the final metric, Global Accuracy (last row in Table 1), improves monotonically as we move from using the strict interpretation of the guidelines ($\sim$ 75%), through the knowledge-engineered parameters ($\sim$ 90%), to the optimized parameters ($\sim$ 94%). While the reported performance of the optimized parameters (in Table 1) is typical of the performance we achieved through the optimization, a five-fold cross-validation on the optimization was also performed to identify stable parameters in the design space and stable metrics in the performance space. Specifically:

Table 2. Average FRC performance over 5 tuning case sets compared to 5 disjoint test sets

METRIC	Average performance on tuning sets	Average performance on disjoint test sets
Coverage	91.81%	91.80%
Relative Accuracy	94.52%	93.60%
Global Accuracy	92.74%	91.60%

7 Monitoring and Maintenance of the FRB

A serious challenge to the successful deployment of intelligent systems is their ability to remain valid and accurate over time, while compensating for drifts and accounting for contextual changes that might otherwise render their knowledge-base stale or obsolete. This issue has been a constant concern in deploying AI expert systems and continues to be a critical issue in deploying knowledge-based classifiers. The maintenance of a classifier is essential to its long-term usefulness since, over time, the configuration of the FRC may become sub-optimal. Thus, we want to be able to modify the SRD to reflect these contextual changes. We developed specialized editors to achieve this objective. By modifying the SRD to incorporate the desired changes (by altering some previous standard reference decisions), we can create a new *target* for the classifier. Then, we use the same evolutionary optimization tools to find a new configuration (parameter values) for the classifier to *approximate the new target*.

It is also vital to monitor the classifier performance over time to identify those new, highly reliable cases that could be used to update the SRD. To address this objective, we have implemented an off-line quality assurance (QA) process, based on a fusion module, to test and monitor the production FRC that performs on-line rate classification. At periodic intervals, e.g., every week, the fusion module and its components will review the decisions made by the FRC during the previous week. The purpose of this fusion is to assess the quality of the FRC performance over that week. In addition, this fusion will identify the *best* cases, which could be used to tune the production engine, as well as *controversial* or *unusual* cases that could be audited or reviewed by human underwriters.

7.1 Fusion Architecture

We developed four independent classifiers specially tuned for accuracy:

1) *Neural Networks* (NN),
2) *Multivariate Regression Splines* (MARS),
3) *Support Vector Machine* (SVM), and
4) *Random Forest* (RF).

NN: We developed multiple binary neural networks to perform the multi-class classification. Each binary network had the structure of 12-5-1, i.e., 12 input nodes, 5 hidden neurons, and 1 output node. We used logistic sigmoidal functions as the activation functions for both hidden and output neurons. The range of target values was scaled to the range $[0.1, 0.9]$ to prevent saturation during training process. The Levenberg-Marquardt numerical optimization technique was used as the back-propagation learning algorithm to achieve second-order training speed. Each binary network represented an individual rate class and was trained with the targets of one-vs-other. ***MARS:*** We created a collection of MARS models, each of which solves a two-class problem, and we collated their outputs in a manner similar to the one used for the NN classifiers. MARS is an adaptive nonparametric regression technique, able to capture main and interaction effects in a hierarchical manner (Friedman [13]). ***SVM:*** We used this model to find the optimal hyper-plane that correctly classifies data points as much as possible, while separating the points of two classes as far as possible via structural risk minimization (Vapnik [20], Cristianini and Taylor [11]). ***RF:*** We built an ensemble of 1,000 trees such that each tree depended on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The RF's overall prediction was obtained by combining the predictions of the trees (Breiman [9]). These classifiers, which don't need to meet the same interpretability requirements as the FLE, are part of the fusion process described in Figure 3.

This fusion process can be decomposed into four steps:

1. *Collection, discounting and post-processing of modules' outputs.* Each classifier's output is a possibility (or probability, depending on the classifier) distribution representing the degree to which a given rate class is selected. The set of all possible rate classes represents the universe of possible answers that can be considered by the classifiers. A classifier's ignorance can be modeled by a weight assignment to this universe.
2. *Determination of combined decision via associative fusion of modules' outputs.* We combine the outputs of each classifiers by using an outer-product with the T-norm operator that better represents the possible correlation among the classifiers. The final output is a rate class distribution and a measure of conflict among all the classifiers.
3.) *Determination of degree of confidence.* By normalizing the final output we identify the strongest selection of the fusion, and qualify it with a degree of confidence that is the complement of the measure of conflict.
4. *Identification of candidate cases for test set, auditing, and SRD process.* We use the confidence measure and the agreement/disagreement of the fused modules' decision with the production engine's decision to assess the quality of the production engine. We label the cases in terms of their decision confidence, e.g. *'low'*, *'medium'*, *'high'*, or *'unknown'*. When the fused decision exhibits a low degree of confidence for a given application, the case is selected for *auditing*. When the fused decision exhibits a *high*

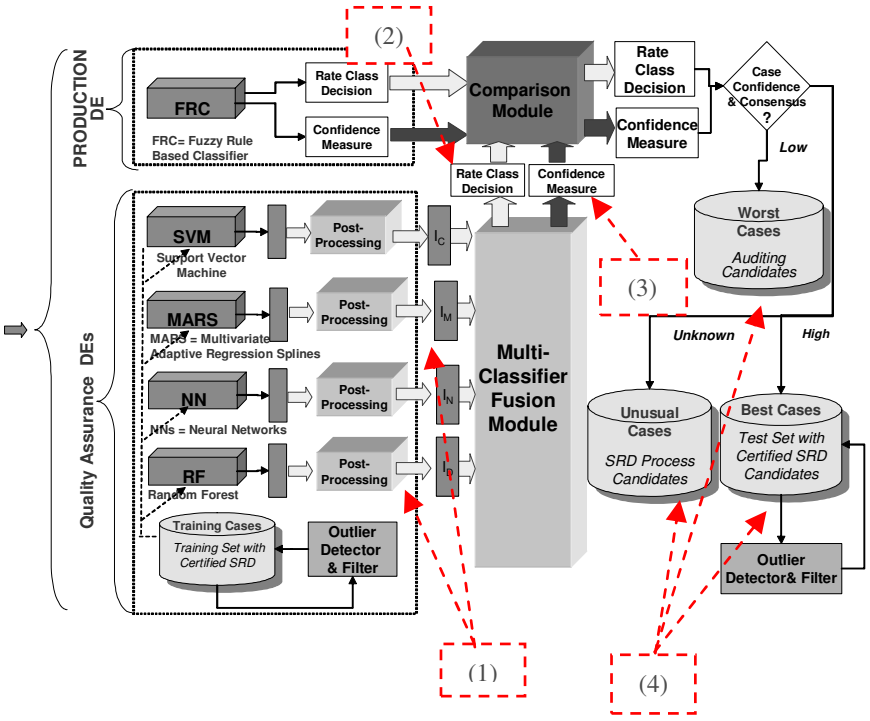

Fig. 3. Top-level Fusion Architecture

degree of confidence and agrees with the decision of the production engine, the case becomes a candidate for *augmenting the SRD*. When the fused decision is non-committal or *unknown* (i.e., it does not show a strong commitment for any class), the case is a candidate for a *review by senior underwriters* who will generate a standard reference decision. These cases will be later used to retrain the offline classifiers. When the fused decision exhibits a *medium* degree of confidence, we will not use the case, but we will monitor the frequency of its occurrence.

A more detailed description of the fusion mechanism can be found in (Bonissone [6], [8]).

8 Conclusions

We have developed a design methodology for a fuzzy knowledge-based classifier that automatically determines risk categories for insurance applications. The classifier design was based on the integration of domain knowledge with field data. The knowledge was leveraged in selecting the attributes that define

the classifier's structure, in designing the chromosome for the EA's, and in defining a fitness function to enforce a specific trade-off between classification coverage and accuracy. The field data was refined, corrected, and used to establish a repository of cases representing ground-truth decisions, i.e. the SRD.

The model's parameters and decision thresholds were first initialized by elicitation from expert underwriters, and then tuned using a multi-stage mutation-based evolutionary algorithm, wrapped around the classifier, to achieve a specific trade-off between accuracy and coverage. The fitness function selectively penalized different degrees of misclassification, and served as a forcing function to drive correct classifications. The structure of the model was determined by knowledge engineering sessions, given that legal constraints defined the only variables that could be used. In other less constrained applications, we could have derived the structure of the model by evolutionary algorithms following the same *wrapper* approach (Freitas [12]).

More importantly, we have established a reliable method to *design* and *maintain* the fuzzy knowledge-based classifier. Our design methodology explicitly addresses classifiers' obsolescence to facilitate their maintenance. Maintenance of the classification accuracy over time is an important requirement considering that decision guidelines may evolve, and so can the set of certified cases. In our approach we have designed the classifiers around a set of *standard reference dataset* (SRD), which embodies the results of the ideal behavior that we want to *reach* during development and that we want to *track* during production use.

We have proposed the use of a fusion module to update the SRD with new cases, without resorting to manual screening. Furthermore, during the life of the classifier we might need to change the underwriting rules. These modifications could be driven by new government regulations, changes among data suppliers, new medical findings, etc. These rule changes are used to update the SRD via maintenance tools. We identify the subset of SRD cases whose decisions are affected by the changes and must be altered. The updated SRD represents the new *target* that we want our classifier to approximate. At this point, we can use the same EA-based optimization tools, employed during the initial tuning, to find a parametric configuration that structures a classifier whose behavior better approximates the new SRD. A more detailed description of the salient steps in the life cycle of a fuzzy knowledge-based classifier can be found in (Patterson *et al.* [18]; Bonissone [6]).

References

1. Babuska, R., Jager, R. and Verbruggen, H.B. (1994). Interpolation Issues in Sugeno-Takagi Reasoning. In: *Proc. Third IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE'94)*, Orlando, pp. 859–863.

2. Bersini, H., Bontempi, G. and Decaestecker, C. (1995). Comparing RBF and fuzzy inference systems on theoretical and practical basis. In: *Proc. of Int. Conf. on Artificial Neural Networks (ICANN '95), Vol. 1*, Paris, pp. 169–174.
3. Bonissone, P. (1997). Soft Computing: the Convergence of Emerging Reasoning Technologies. *Soft Computing - A Fusion of Foundations, Methodologies and Applications* **1**(1), 6–18.
4. Bonissone, P., Chen, Y.-T., Goebel, K. and Khedkar, P. (1999). Hybrid Soft Computing Systems: Industrial and Commercial Applications. *Proceedings of the IEEE* **87**(9), 1641–1667.
5. Bonissone, P., Subbu, R. and Aggour, K. (2002). Evolutionary Optimization of Fuzzy Decision Systems for Automated Insurance Underwriting. In: *Proceedings of the IEEE Intern. Conference on Fuzzy Systems (FUZZ-IEEE'02), Vol. 2*, Honolulu, Hawaii, pp. 1003–1008.
6. Bonissone, P. (2003). The Life Cycle of a Fuzzy Knowledge-based Classifier. In: *Proc. of 2003 North American Fuzzy Information Processing Society*, Chicago, pp. 488–494.
7. Bonissone, P., Subbu, R., Eklund, N. and Kiehl, T. (2004). Evolutionary Algorithms + Domain Knowledge = Real-World Evolutionary Computation. (Submitted).
8. Bonissone, P. (2004). Automating the Quality Assurance of an On-line Knowledge-Based Classifier By Fusing Multiple Off-line Classifiers. In: *Proc. Information Processing and Management of Uncertainty 2004*, Perugia.
9. Breiman, L. (1999). Random Forests. Technical Report, Statistics Department, University of California Berkeley, CA. (http://www.stat.berkeley.edu).
10. Casillas, J., Cordón, O., Herrera, F. and Magdalena, L. (Editors) (2003). *Interpretability Issues in Fuzzy Modeling, and Accuracy Improvements in Linguistic Fuzzy Modeling*. Studies in Fuzziness and Soft Computing, Vols. 128-129, Springer-Verlag, Heidelberg.
11. Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines*. Cambridge University Press.
12. Freitas, A. (2002). *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag, Berlin.
13. Friedman, J.H. (1991). Multivariate Adaptive Regression Splines. *Annals of Statistics* **19**, 1–141.
14. Guillaume, S. (2001). Designing fuzzy inference systems from data: An interpretability-oriented review. *IEEE Trans. on Fuzzy Systems* **9**(3), 426–443.
15. Jang, J.S.R. (1993). ANFIS: Adaptive-network-based-fuzzy-inference-system. *IEEE Trans. on Systems, Man, and Cybernetics* **233**, 665–685.
16. Larrañaga, P. and Lozano, J. (Guest Editors) (2002). Synergies between evolutionary computation and probabilistic graphical models. Special Issue of the *International Journal of Approximate Reasoning* **31**(3), Nov. 2002.
17. Mamdani, E.H. and Assilian, S. (1975). An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *Int. J. Man Machine Studies* **7**(1), 1–13.
18. Patterson, A., Bonissone, P. and Pavese, M. (2004). Six Sigma Quality Applied Throughout the Lifecycle of and Automated Decision System. *International Journal of Quality and Reliability*, (to appear).
19. Takagi, T. and Sugeno, M. (1985). Fuzzy Identification of Systems and Its Applications to Modeling and Control. *IEEE Trans. on Systems, Man, and Cybernetics* **15**, 116–132.

20. Vapnik, V.N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.
21. Vonk, E., Jain, L.C. and Johnson, R.P. (1997). *Automatic Generation of Neural Network Architecture Using Evolutionary Computation*. World Scientific Publ. Co., Singapore.
22. Yao, X. (1999). Evolving Artificial Neural Networks. *Proceedings of the IEEE* **87**(9), 1423–1447.
23. Zadeh, L.A. (1994). Fuzzy Logic and Soft Computing: Issues, Contentions and Perspectives. In: *Proc. of Third Int. Conf. on Fuzzy Logic, Neural Nets and Soft Computing (IIZUKA'94)*, Iizuka, pp. 1–2.