Chapter 1

A REVIEW OF COMPUTATIONAL INTELLIGENCE ALGORITHMS IN INSURANCE APPLICATIONS

S. Salcedo-Sanz[*], L. Cuadra[†], A. Portilla-Figueras[‡], S. Jiménez-Fernández[§] and E. Alexandre[¶]
Department of Signal Theory and Communications
Universidad de Alcalá, Madrid, Spain.

Abstract

Insurance sector, one of the cornerstones the financial system is based on, is currently facing major changes because of the urgent necessity of being adapted to the new context of global economic crisis. Within this frame, the financial system of any modern country needs to develop novel procedures aiming to make it more competitive and efficient, along with the compliance with commitment agreed with the policyholders. Part of this important responsibility relies on the inclusion of more effective and accurate computing techniques able to provide better solutions for crucial problems in different key components of the system, including, of course, the insurance sector. In this respect, this chapter presents a review of those Soft-Computing applied in the field of insurance companies and related problems, which have been very active and successful over the past 10 years. The aims of this chapter are thus: 1) to illustrate in a comprehensive way those features of Soft-Computing algorithms that make them suitable to tackle insurance problems; 2) to provide a good and up-dated review of their main applications in insurance-related problems; and 3) to show to what extent Soft-Computing algorithms work accurately in a real application. Just in this regard, this chapter ends up with details about the application of several of these techniques in predicting vehicle accidents using drivers and vehicles' data, which is useful for vehicle insurance companies.

Key Words: Soft-Computing techniques, Insurance companies, Review of methods and applications.

[*]E-mail address: sancho.salcedo@uah.es
[†]E-mail address: lucas.cuadra@uah.es
[‡]E-mail address: antonio.portilla@uah.es
[§]E-mail address: silvia.jimenez@uah.es
[¶]E-mail address: enrique.alexandre@uah.es

1. Introduction

The insurance sector plays a growing and crucial role among the different components of the financial system in modern economies. The insurance industry is fundamental for modern countries, both from the economic and social points of view. Because of this key role in the financial system of countries, it has long been recognized that there is a need for some form of *prudential* supervision of insurance entities, to attempt to minimize failure risk. In recent years, Solvency I and II projects have led the reform of the existing solvency rules in European Union. The main concern of the regulatory authority, through Solvency II, is the protection of the insured person, in the sense that the insurance company have to be able to guarantee the payment of financial obligations that have contracted with the insured person. That is, it is required for insurance companies to maintain an adequate economic level: additionally to the "technical" capital reserves, it is compulsory to keep the so-called Solvency Capital Requirements (SCR). The objective is thus to ensure the stability of the company *against adverse situations*, as currently is occurring. Regarding this, Solvency II suggests that part of the Solvency Capital Requirements, which have to be secured, is related to the *non-zero possibility of having underestimated the risks assumed with a novel client*. Soft-Computing (SC) or Computational Intelligence (CI) methods, as will be shown throughout this Chapter, can play a key role in assisting a company in accurately predicting such a risk.

However, despite their importance, the insurance sector has received less attention from researches that that given to the banking sector. In fact, as a consequence of its business peculiarities, it is not possible to translate the conclusions from banking sector analysis to the insurance one, and therefore a specific analysis is needed. Consequently, the development of novel methods and algorithms to tackle insurance problems is a key point, and opens interesting new lines of research all around the world.

Many important problems in the insurance and reinsurance industry are related with the processing of a *huge amount of uncertain, imprecise and incomplete information*. The insurance companies face classical problems such as the calculation of fair premiums and reserves, estimating the risks assumed with a novel client, prediction of future uncertain events (car accidents, dates of death, insolvencies, catastrophes, etc.), design of optimal strategies, application of different risk measures, etc. And all of these problems have in common the big difficulty of forecasting the future from present and past information about the characteristics of the insured risks. The classical methodologies for studying these problems are based on well known statistical techniques. These methodologies, however, cannot be applied to many modern problems in insurance, because of their aforementioned properties of involving huge, uncertain, imprecise and incomplete information. In these cases, the application of Soft-Computing techniques has been massive in the last years.

Soft-Computing techniques refer to a set of modern computational methodologies, focused on the design of intelligent systems, which are able to process huge amount of uncertain, imprecise and incomplete data, and to find approximate solutions for problems that otherwise would be intractable. Soft-Computing is a key part of Artificial Intelligence (AI), and many of its methods also belong to the area of knowledge called "Natural Computing" (NC). The wider term Natural Computing refers to algorithms inspired by the way Nature solves extremely complex problems. It draws inspiration from Evolution (leading to

Evolutionary Computation), Physics (Simulated Annealing), social living being networks (social insects, what leads to Ant Colony Optimization (ACO) Algorithm, and Swarm Intelligence), Neural Networks (human brain metaphor inspires Artificial Neural Networks), Immune Systems (leading to Imnunocomputing), and so on. This classification is flexible in the sense that some algorithms can belong simultaneously to different groups: for instance a Genetic Algorithm (GA) is a population-based metaheuristic (like Ant Colony Optimization), and is also an evolutionary algorithm. Another example: A fuzzy neural network or neuro-fuzzy system is a learning machine that finds the parameters of a fuzzy system (i.e., fuzzy sets, fuzzy rules) by exploiting approximation techniques from neural networks. It thus belongs to the intersection between Neural Computation and Computation based on Fuzzy concepts.

With this in mind, Figure 1 will assist us in showing the different fields of SC, and in motivating the structure of this Chapter. The meaning of the acronyms in Figure 1 have been listed in Table 1 for the sake of clarity, since they will be used throughout this Chapter. As motivated before, we have represented basically five sets of Soft-Computing techniques, some of them exhibiting non-null intersection. These sets are: Evolutionary Computation, Neural Computation, Computation based on Fuzzy concepts, Physics-Inspired Computation, and Computation inspired by Social living-being with collective behavior. We have adopted this classification for convenience and for illustrative purposes, although other authors could carried out others since, as mentioned, some algorithms may belong to different soft-computing approaches.

Although not represented in Figure 1 for clarity, SC include hundred of algorithms and numerical methods which together form the wide family of soft-computing approaches. SC techniques have been successfully applied to a wide variety of problems in Science and Engineering, including general finance applications [32] and problems in the insurance sector [78]. The applications of Soft-Computing techniques −see [4, 50, 87] for a general overview− are huge, and cover a variety of different fields in Science and Technology, such as Electrical Engineering [14], Biometrics [62], Green and Renewable Energy Systems [34], Ecology [49], Biology [3], Bio-medical applications [28], Humanities and Social Sciences [77], and, of course, Economy and Finance [8, 24, 32, 78]. In the last few years, as will be shown in Section 7. in a more detailed way, the applications of Soft-Computing in problems related to Insurance have been important, and currently, the majority of economical institutions, from private banks and insurance companies to government-depending analysis services, count with artificial intelligence departments that make use of Soft-Computing techniques, as a base for the their predictions and analysis. This is because Soft-Computing exhibits the capacity to tackle problems that, otherwise, would be intractable. Currently, the guiding principle of SC is that, in general, better results can be achieved through the use of methodologies of SC in combination, rather than in a standalone mode. That is, many complex problems are successfully tackled by hybridizing several SC strategies.

With all the aforementioned ideas in mind, and making use of Figure 1 as a guide, the structure of the rest of the Chapter is as follows:

- Section 2. focuses on basic concept of FL computation that could be useful for insurance-related problems

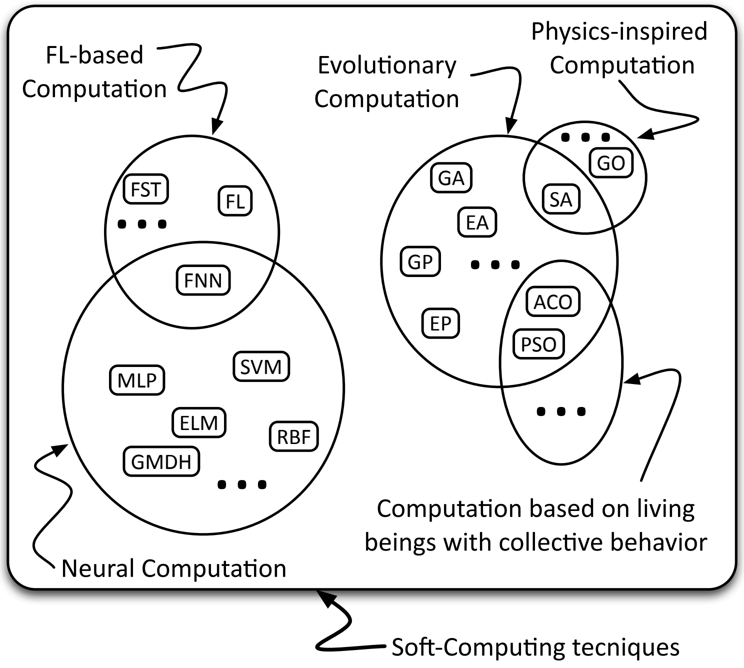

Figure 1. Outline of the structure of Soft-Computing methods. We have represented basically five sets of Soft-Computing techniques (some of them exhibiting non-null intersection): Evolutionary Computation, Neural Computation, Computation based on Fuzzy concepts, Physics-Inspired Computation, and Computation inspired by Social living-being networks. The meaning of the acronyms have been listed in Table 1 for the sake of clarity. See the main text for further details.

- Section 3. describes the main Neural Nomputation-based approaches
- Section 4. centers on the most common Evolutionary-based algorithms
- Section 5. focuses on the SA method, an instance of Physics-inspired algorithm that has been used in insurance-related problems
- Section 6. describes some algorithms inspired by the collective behavior of social insects and birds
- Section 7. reviews the most important applications of these techniques to insurance problems, published during the last years in important international journals
- Section 8. centers on the application of some of the revised techniques to a real problem in automobile insurance: the prediction of vehicles accidents on the basis of drivers and cars' data
- Finally, Section 9. completes this Chapter by giving some final concluding remarks.

Table 1. List of acronyms used throughout this Chapter.

Acronyms	Meaning
ASR	Average Squared Residual
ACO	Ant Colony Optimization
AI	Artificial Intelligence
CEP	Classical Evolutionary Programming algorithm
CI	Computational Intelligence
DA	Differential Evolution
DEA	Data Envelopment Analysis
EC	Evolutionary Computation
ELM	Extreme Learning Machine
FEP	Fast Evolutionary Programming
FNN	Fuzzy Neural Network
FSP	Feature Selection Problem
GA	Genetic Algorithm
GMDH	Group Method of Data Handling algorithm
GO	Gravity Optimization
GP	Genetic Programming
IFEP	Improved Fast Evolutionary Programming
KS	Kolmogorov-Smirnov
MLP	Multi-Layer Perceptron
NC	Natural Computing
NN	Neural Network
PSO	Particle Swarm Optimization
RS	Rough Set
SA	Simulated Annealing
SC	Soft-Computing
SCR	Solvency Capital Requirements
SLFN	Single-hidden layer feed-forward network
SVM	Support Vector Machine
SVMC	Support Vector Machine for classification
SVMr	Support Vector Machine for regression

2. Fuzzy logic-based computation

Fuzzy logic-based computation, the first subset of Soft-Computing methods illustrated in Figure 1, are inspired by the fact that humans exhibit the extraordinary capability to reason and make decisions in an environment of uncertainty, incomplete information, and partiality of class membership. The main goal of Fuzzy Logic (FL) is the formalization/mechanization of this capability [115], as briefly shown in the next subsection.

2.1. Fuzzy logic

The original concept of FL was first proposed by Zadeh [103], and is based on the concept of "fuzzy set", which plays a central role in fuzzy logic. Classical set theory has a crisp concept of membership: an element either belongs to a set *or* it does not. However, fuzzy set (FS) theory differs from the traditional one in the fact that *partial membership* is allowed (that is, an element can belong to a set with a certain degree). This *degree of membership* is commonly referred to as the *membership value* and is usually represented by using a real value in [0,1], where 0 and 1 correspond to *full non-membership* and *membership*, respectively. Usually, triangular or trapezoidal functions are used as *membership functions* because of their simplicity, although, however, more smooth or complex shapes can be used if necessary [115]. Based on these ideas, predicates in fuzzy logic can have *partial degrees of truth*, in the same way as elements can have partial membership in fuzzy set theory. The grade of truth of a predicate is represented using a real number in [0,1]. These ideas will assist us in introducing two basic concepts in FL. These are the concepts of "graduation" and "granulation" [111], which form the very core of FL, and are the mayor distinguishing properties of fuzzy logic [115] when comparing to the classical one. In FL everything is or is allowed to be graduated, that is, be a matter of degree or, equivalently, fuzzy. Furthermore, in FL everything is or is allowed to be granulated: for example, the concept "size" is granulated when its values are described as "small", "medium" and "big". Thus, the principal contributions of fuzzy logic consist of the concept of a *linguistic variable* (idea of using words instead of numbers) [104–106, 115], the machinery of fuzzy "if-then" rules, and the capability to compute with information described in natural language. See the illustrative review [115] for further details.

Fuzzy logic makes it possible to construct better models of reality [110–113, 115] in human-centric science such as economics, medicine, psychology and linguistics [107–109]. In particular, FL-based computation plays a relevant role in insurance-related problems. We will review this topic in detail in Section 7..

Another theory that also makes use of the fact that some objects described by the same data or knowledge can be classified into different classes is the Rough Set (RS) theory.

2.2. Rough set

Rough Set theory is an Artificial Intelligence (AI) tool which is considered as a high-performance classifier method. Unlike other AI methods, one of RS characteristics is its explicative character, i.e. the obtained model is easily understandable and explained the knowledge hidden in the dataset. Briefly, RS theory was firstly developed by Pawlak [68] as a mathematical method to deal with the uncertainty or vagueness inherent in a decision making process. But, unlike other methods that deal with uncertainty (such as statistical probability or fuzzy set theory), RS theory deals with the uncertainty produced when some objects described by the same data or knowledge (so, they are indiscernible) can be classified into different classes. This fact prevents their precise assignment to a set. Therefore, the classes in which the objects are to be classified are imprecise, but they can be approximated with precise sets [61, 64].

RS approach works by discovering dependencies between attributes in an information table, and reducing the set of attributes by removing those that are not essential to characterize knowledge. A *reduct* is defined as the minimal subset of attributes which provides the same quality of classification as the set of all attributes. A reduced information table may provide decision rules of the form “if conditions then decisions”. These rules specify what decisions (actions) should be undertaken when some conditions are satisfied, and can be used to assign new objects to a decision class by matching the condition part of one of the decision rule to the description of the object. RS is one of the most used approach in classification problems arising in insurance applications. We will review this topic in detail in Section 7..

3. Neural computation-based and related algorithms

Neural computation, the second subset illustrated in Figure 1, is inspired by the way human brain works. In this respect, an Artificial Neural Network (ANN) mimics human brain in the sense that an ANN is a massively parallel and distributed information processing system. As will be shown in detail in this Section 3., examples of neural computation approaches are the Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), Extreme Learning Machine (ELM), and Group Method of Data Handling algorithm (GMDH). These neuro-computing approaches have successfully been applied in modeling a large variety of nonlinear problems.

3.1. Multilayer perceptrons

A MLP is a massively parallel and distributed information processing system, successfully applied in modeling a large amount of nonlinear problems [36], [6]. The MLP learns from given sample examples, by constructing an input-output mapping to perform predictions of future samples. MLPs are known to be universal approximators of a large class of functions, with a high degree of accuracy. Single hidden layer perceptrons are the most widely used model in forecasting problems [36], and its more common structure is given in Figure 2.

It consists of an input layer, a hidden layer and an output layer. The number of neurons in the hidden layer is a parameter to optimize when using this type of neural networks. The relationship between the output and the input signals of a given neuron is the following:

$$y = \varphi\left(\sum_{j=1}^{n} w_i x_j - \theta\right), \tag{1}$$

where y is the output signal, x_j, for $j = 1, 2, \ldots, n$ are the input signals, w_j is the weight associated with the j-th input and θ is a threshold. The transfer function φ is usually considered as the logistic function:

$$\varphi(x) = \frac{1}{1+e^{-x}}. \tag{2}$$

Usually, the well-known Levenberg-Marquardt algorithm is used to train the MLP [35]. The Levenberg-Marquardt algorithm was designed to approach second-order training speed, without having to compute the Hessian matrix. This matrix is estimated using the Jacobian matrix instead, which can be computed through a standard back-propagation

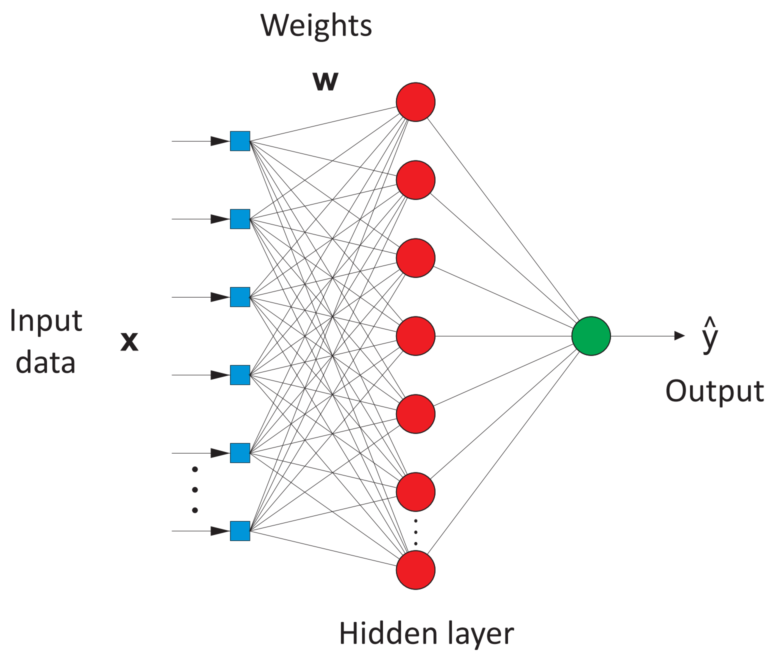

Figure 2. Simplified structure of a single hidden layer perceptron.

technique, much less complex than computing the Hessian matrix [35]. The Levenberg-Marquardt algorithm works by using the following Newton-like update:

$$\mathbf{x}_{k+1} = \mathbf{x_k} - \left(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}\right)^{-1}\mathbf{J}^T\mathbf{e} \tag{3}$$

where $\mathbf{J}$ is the Jacobian matrix, $\mathbf{e}$ is a vector of network errors and μ is a parameter which controls the process: when $\mu = 0$ we have the Newton's method, when μ is large, it becomes a gradient descent method with small step size.

3.2. Extreme learning machines

The Extreme Learning Machine (ELM) is a novel and fast learning method recently proposed in [40]. Put it simple, an ELM is a generalized single-hidden layer feedforward network (SLFN), its essence being that the hidden layer of the SLFN does *not* need to be tuned. Compared to other traditional neurocomputing-based approaches such as neural networks and support vector machines, ELM can provide better generalization performance *at a much faster learning speed* [40–44] . This is the reason why it has been applied to a large number of classification and regression problems. Its beauty is its simplicity along with its surprising results, comparable or even superior to that of other classification and regression techniques such as multi-layer perceptrons or support vector machines, but with the added bonus of learning at much faster speed.

The ELM algorithm can be described as follows: given a training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i | \mathbf{x}_i \in \mathbb{R}^n, \mathbf{t}_i \in \mathbb{R}^m, i = 1, \cdots, N\}$, an activation function $g(x)$ and and number of hidden nodes $(\tilde{N})$,

1. Randomly assign input weights $\mathbf{w}_i$ and biases b_i, $i = 1, \cdots, \tilde{N}$.

2. Calculate the hidden layer output matrix $\mathbf{H}$, defined as

$$\mathbf{H} = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x_1} + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x_1} + b_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x_N} + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x_N} + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}} \tag{4}$$

3. Calculate the output weight vector β as

$$\beta = \mathbf{H}^{\dagger}\mathbf{T}, \tag{5}$$

where $\mathbf{H}^{\dagger}$ stands for the Moore-Penrose inverse of matrix $\mathbf{H}$ [40], and $\mathbf{T}$ is the training output vector, $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]^T$.

Note that the number of hidden nodes ($\tilde{N}$) is a free parameter of the ELM training, and must be estimated for obtaining good results. The solution to this problem is usually to evaluate a different number of values for $\tilde{N}$.

3.3. Support Vector Machines for regression problems

One of the most important statistical models for prediction are the Support Vector Regression algorithms (SVMr) [83]. The SVMrs are appealing algorithms for a large variety of regression problems, since they do not only take into account the error approximation to the data, but also the generalization of the model, i.e., their capability to improve the prediction of the model when a new dataset is evaluated by it. Although there are several versions of SVMr, in this case we are going to describe the classic model presented in [83].

The ε-SVMr method for regression consists of training a model of the form $y(\mathbf{x}) = f(\mathbf{x}) + b = \mathbf{w}^T \phi(\mathbf{x}) + b$, given a set of training vectors $C = \{(\mathbf{x_i}, y_i), i = 1, \ldots, l\}$, to minimize a general risk function of the form

$$R[f] = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l} L(y_i, f(\mathbf{x})) \tag{6}$$

where $\mathbf{w}$ controls the smoothness of the model, $\phi(\mathbf{x})$ is a function of projection of the input space to the feature space, b is a parameter of bias, $\mathbf{x_i}$ is a feature vector of the input space with dimension N, y_i is the output value to be estimated and $L(y_i, f(\mathbf{x}))$ is the loss function selected. In this paper, we use the L1-SVMr (L1 support vector regression), characterized by an ε-insensitive loss function [84]

$$L(y_i, f(\mathbf{x})) = |y_i - f(\mathbf{x_i})|_{\varepsilon} \tag{7}$$

In order to train this model, it is necessary to solve the following optimization problem [84]:

$$\min\left(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)\right) \tag{8}$$

subject to the restrictions:

$$y_i - \mathbf{w}^T\phi(\mathbf{x_i}) - b \leq \varepsilon + \xi_i, \quad i = 1,\ldots,l \tag{9}$$

$$-y_i + \mathbf{w}^T\phi(\mathbf{x_i}) + b \leq \varepsilon + \xi_i^*, \quad i = 1,\ldots,l \tag{10}$$

$$\xi_i, \xi_i^* \geq 0, \quad i = 1,\ldots,l \tag{11}$$

The dual form of this optimization problem is usually obtained through the minimization of the Lagterm function, constructed from the objective function and the problem constraints. In this case, the dual form of the optimization problem is the following:

$$\max\left(-\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(\mathbf{x_i},\mathbf{x_j}) - \right.$$

$$\left. -\varepsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*)\right) \tag{12}$$

subject to

$$\sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \tag{13}$$

$$\alpha_i, \alpha_i^* \in [0, C] \tag{14}$$

In addition to these constraints, the Karush-Kuhn-Tucker conditions must be fulfilled, and also the bias variable, b, have to be obtained. We do not detail this process for simplicity, the interested reader can consult [84] for reference. In the dual formulation of the problem the function $K(\mathbf{x_i},\mathbf{x_j})$ is the kernel matrix, which is formed by the evaluation of a kernel function, equivalent to the dot product $\langle\phi(\mathbf{x_i}),\phi(\mathbf{x_j})\rangle$. An usual selection for this kernel function is a Gaussian function, as follows:

$$K(\mathbf{x_i},\mathbf{x_j}) = exp(-\gamma\cdot\|\mathbf{x_i} - \mathbf{x_j}\|^2). \tag{15}$$

The final form of the function $f(\mathbf{x})$ depends on the Lagterm multipliers α_i, α_i^*, as follows:

$$f(\mathbf{x}) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)K(\mathbf{x_i},\mathbf{x}) \tag{16}$$

In this way it is possible to obtain a SVMr model by means of the training of a quadratic problem for given hyper-parameters C, ε and γ. However, obtaining these parameters is not a simple procedure, being necessary to implement search algorithms to obtain the optimal ones or the estimation of them [65].

3.4. Support Vector Machines for classification problems

The support vector machines for classification (SVMC) consists of, given a training set $C = (\mathbf{x_i}, y_i, i = 1, \ldots, l)$, where $\mathbf{x_i} \in R^n$ are the input vectors and $y_i \in \{-1, 1\}$ is the label associated to each input vector, obtaining a classification model by solving the following optimization problem:

$$\min_{\mathbf{w},\xi_\mathbf{i}} \left(\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{l} \xi_i \right) \tag{17}$$

constrained to

$$y_i(\mathbf{w^T}\phi(\mathbf{x_i}) + b) \geq 1 - \xi_i \quad i = 1 \ldots l \tag{18}$$

where $\mathbf{w}$ is the generalization term which is inversely related to the margin of the model, ξ_i is the slack variable for each input vector $\mathbf{x_i}$ and C is a parameter which controls the trade-off between an approximation to misclassification errors, given by the variables ξ_i and the ability of generalization of the model when future input vectors are introduced to the model. Note that the slack variables ξ_i are included in the model because we consider non-separable classification problems.

In order to solve this optimization problem, the Wolfe-dual formulation is usually used, obtaining a new representation which does not depend directly on the mapping function $\phi(\mathbf{x_i})$, but on the dot product in a mapped space $k(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i})^T \phi(\mathbf{x_j})$, i.e., the kernel function.

$$\min_{\alpha_i} \left(\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j k(\mathbf{x_i}, \mathbf{x_j}) - \sum_{i=1}^{l} \alpha_i \right) \tag{19}$$

subject to

$$\sum_{j=1}^{l} \alpha_j y_j = 0 \tag{20}$$

$$0 \leq \alpha_i \leq C \quad i = 1 \ldots l \tag{21}$$

In this optimization problem the solution is given by the Lagrange multipliers α_i, and must fulfil the necessary and sufficient Karush-Kuhn-Tucker conditions to be optimal. In this case we only show the most important KKT conditions for our study:

$$\xi_i(C - \alpha_i) = 0 \quad i = 1, \ldots, l, \tag{22}$$

$$\alpha_i(y_i(\mathbf{w^T}\phi(\mathbf{x_i}) + b) - 1 + \xi_i) = 0 \quad i = 1, \ldots, l. \tag{23}$$

Regarding to the kernel function, we focus our study on the most classical function used in the literature [75], the Gaussian kernel function. We only consider the case in which one parameter (γ) controls the width of the gaussian function in any direction.

$$k(\mathbf{x_i}, \mathbf{x_j}) = e^{-\gamma \cdot \|\mathbf{x_i} - \mathbf{x_j}\|^2} \tag{24}$$

Taking into account the optimal solution of the dual optimization problem and the kernel function selected, the final classification model is given by the following expression:

$$f(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\phi(\mathbf{x_i}) + b = \sum_{i=1}^{L} \alpha_i y_i k(\mathbf{x_i}, \mathbf{x}) + b. \tag{25}$$

3.5. The Group Method of Data Handling algorithm

It is well known that the relationship between any sets of input-output variables can be approximated by Volterra functional series, the discrete analogue of which is the Kolmogorov-Gabor polynomial [1]:

$$p = a_0 + \sum_{i=1}^{m} a_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} a_{ijk} x_i x_j x_k + \dots \tag{26}$$

where, $x = (x_1, x_2, \dots, x_m)$ are the inputs and $A = (a_0, a_1, a_2, \dots, a_m)$ are the coefficients (weights). The Kolmogorov-Gabor polynomial is a universal format for non-linear function modeling as they can approximate any continuous function on a compact data set to an arbitrary precision, in an Average Squared Residual sense (ASR), if there are enough terms [30].

$$ASR = \frac{1}{N}\sum_{i=1}^{N}(y_i - p(x_i))^2 \tag{27}$$

where y_i is the output of and $p(x)$ is the Kolmogorov-Gabor polynomial.

The Kolmogorov-Gabor polynomial has an important drawback, because it is necessary to have a large numbers of samples in order to calculate all the coefficients a_i [47]. In order to overcome this drawback, Ivakhnenko proposed a new algorithm which approximates the Kolmogorov-Gabor polynomial by using low order polynomials in an iterative method very similar to the multi-layer perceptron neural network. This method does not need so many samples and the time consuming is much lower. In fact, Ivakhnenko proved that using second order polynomials, the complete Kolmogorov-Gabor Polynomial can be reconstructed [47]. This is the key idea behind the GMDH neural network.

There are several GMDH types [1], in this article we work with the Multi-layer GMDH algorithm [47]. This algorithm constructs hierarchical cascade of bivariate second order polynomials in the nodes and variables in the leaves. The next steps explain how the multi-layer GMDH algorithm works:

1. Obtain the data set of the problem $D = (x_i, y_i)_{i=1}^{N}, x_i = (x_{i1}, x_{i2}, \dots, x_{iK})$, where N is the number of samples and K the input dimension.

2. Split the data set in two subsets, one is used to calculate the polynomial coefficients of the nodes. The second subset checks the goodness of every polynomial created before.

3. Make all combination of variables in pairs (x_i, x_j) in order to generate all the possible bivariate polynomials $L = \frac{K(K-1)}{2}$.

4. Calculate the coefficients of every polynomial with an ordinary Least Square method.

5. Apply an external criteria to choose the best nodes of the current layer. This election is based on new information (samples are not involved in the process of calculation of coefficients) in order to avoid over fitting. There are several criteria, the most popular in GMDH is called regularity criterion which consists of splitting the training set in two subsets A and B, one is used to calculate the coefficients and the other to apply the ASR by using the second order polynomial (27).

6. The outputs of the selected nodes are the inputs for the new layer.

This process is repeated until a stop criteria is achieved. The stop criteria is usually based on the ASR: when the ASR does not decrease from one layer to the next the algorithm is halted, because it means that the generated structure is complex enough to make good estimations.

3.5.1. The least squares approach to obtain the polynomial coefficients in GMDH

In order to calculate the polynomial coefficients in the GMDH, we have N different equations

$$A\beta = Y \tag{28}$$

where β is the vector of unknown coefficients of the polynomial given by

$$\beta = \left[\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5\right] \tag{29}$$

where A is the polynomial matrix given by

$$A = \begin{bmatrix} 1 & x_{11} & x_{21} & x_{11}^2 & x_{21}^2 & x_{11}x_{21} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{1N} & x_{2N} & x_{1N}^2 & x_{2N}^2 & x_{1N}x_{2N} \end{bmatrix} \tag{30}$$

and Y is the vector of the output values given by

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \tag{31}$$

In order to solve this system of equations usually least squares technique is applied which has the following expression:

$$\beta = (A^T A)^{-1} A^T Y \tag{32}$$

This equation determines the vector of coefficients which minimizes the mean square error.

4. Evolutionary computation-based algorithms

While Neural Computation is inspired by the human brain metaphor, the third subset of Soft-Computing techniques illustrated in Figure 1, Evolutionary Computation is inspired by the principles of Genetics and Natural Selection. The most representative strategy in this subset is the concept of Genetic Algorithm (GA). It is an optimization and search technique which exhibits useful properties for solving problems that, otherwise, would be intractable. Among these advantages, it is worth mentioning that GAs deal with a large number of variables, provide a global solution for multi-local extrema problems, optimize functions with continuous or discrete variables, optimize variables with extremely complex cost surfaces, and do not require derivative information.

Although still not clear at this point, these benefits are due to the fact that GAs are inspired by the way nature finds solutions to extremely complex problems such as the survival-of-the-fittest individual in a changing ecosystem. Put it simple, a GA is based on three key facts: a) Encoding the candidate solution; b) Generate an initial population of candidate solutions, and c) Applying genetic mechanisms. In nature, the random creation of novel genetic information may lead to the ability to survive. The better an individual is suited to an environment, the higher its probability of survival. This is the so-called “survival of the fittest” and the longer the individual’s life is, the higher its chances of having descendants. In this procreation process, the parent chromosomes are combined (“recombination”) to provide a novel chromosome. Sporadically, and because of unavoidable errors in copying genetic information or external factors (for instance, radiation), mutations (random variations) occur. The consequence is the creation of a generation of living beings with some novel characteristic which is slightly different from those of their progenitors. If the new attribute makes the offspring better suited to the varying environment, the probabilities of survival and of having descendants also increase. Part of the offspring could inherit the modified genes and the corresponding external characteristic. In this way, the population of individuals evolves and for a number of generations the described process results in the creation of individuals better adapted to the environment and in the extinction of those worst suited.

Evolutionary algorithms (EAs) [27, 29, 33], are robust problem’s solving techniques based on natural evolution processes. They are population-based techniques which codify a set of possible solutions to the problem, and evolve it through the application of the so called *evolutionary operators* [27, 33]. There are different types of EAs, depending on the type of encoding (binary, integer, real, etc.), evolution mechanism (genetic algorithms, differential evolution, evolutionary programming), or paradigm simulated in the algorithm (particle swarm). Note that, with this approach, particle swarm optimization (PSO), belongs to the intersection between Evolutionary- and Social Network-based methods.

4.1. Genetic and evolutionary algorithms

Genetic algorithms (GAs) [27,33] are a class of robust problem-solving techniques, usually applied to discrete optimization problems, based on a population of solutions (individuals), where each individual represents a specific solution to a the problem. Starting from a random generated population, the population of individuals evolves through successive generations by means of the application of three genetic operators: selection, crossover and mutation [27], which emulate the random changes occurring in nature. After a number of generations, highly fit individuals will emerge corresponding to good solutions to the given optimization problem. GAs are useful to perform search in huge search spaces, where other methods (local or gradient searches, etc) do not provide good results. The pseudo-code of a standard GA is the following:

```
Genetic Algorithm
Initialize GA population at random
while(max. number of generations not reached) do
    evaluate($f(\mathbf{x})$);
    selection
    crossover
    mutation
end while
```

4.1.1. Selection operator

The Selection operator is responsible for choosing which individuals will survive for the next generation of the EA. Among the different types of selection procedures existing [27], the *roulette wheel* is one of the most used. In roulette will selection the probability of an individual to be selected for the next generation ($P(i)$) depends on its current fitness value:

$$P(i) = \frac{f_i}{f_T} \tag{33}$$

where f_i is the fitness value associated to individual i and f_T is the total fitness of the population, which is defined as $f_T = \sum_{i=1}^{\xi} f_i$. Thus, it is probable that the fittest individuals receive a larger number of samples in the next generation than individuals with less associated fitness values.

Alternative selection mechanisms are Tournament selection, considered nowadays as the most effective crossover procedure. It consists of making direct comparisons (tournaments) between pairs of individuals, ranking them afterwards depending on the number of winning tournaments. In this selection mechanism the survival probability depends on the final position obtained in this ranking of the stronger (most winning) individuals. A specific implementation of the Tournament selection is outlined later in this Chapter, in the description of the Evolutionary Programming algorithm.

4.1.2. Crossover operator

The crossover operator has been described as the key to the EA's power [27], as it promotes structured yet randomized information exchange between individuals. However, if the crossover operator is applied to every individual in the population, there will be a discontinuity from the previous to the present populations, as none of the individuals of the population from the previous generation will be retained in the new one. In order to avoid this, a *crossover probability* P_c is defined. It has been suggested therefore that $\alpha_x < 1$, and the range of values usually considered lies between $P_c = 0.5 - 0.6$ [33]. The pseudo-code of the general Crossover operator in a genetic algorithm is as follows:

```
Pseudo-code of Crossover Operator.
Couple all individuals, at random.
  for(each couple)
    if(random_variable(0,1) $\leq P_c$)
     Perform_Crossover($\mathbf{x_i}, \mathbf{x_j}$);
    end(if)
  end(for)
```

4.1.3. Mutation operator

After the crossover operation described above, every single individual in the population may undergo a further random change with a very small probability P_m. This change may consists of choosing two points in the string of numbers representing an individual and swapping the values in them, or in changing several points in the individual, replacing the current value by a different one.

4.2. Differential evolution

The Differential Evolution algorithm [70, 86] is quite similar to a genetic algorithm [33], whose main process has mutation, crossover and selection. The main difference between differential evolution and a genetic algorithm is the mutation operator. In a genetic algorithm, mutation consists of small changes in genes of each individual, whereas in a differential evolution algorithm, mutation is carried out by the arithmetical combinations of selected individuals. A differential evolution algorithm maintains a population of N possible solutions to the problem, $\mathbf{x}_i, i \in \{1, \cdots, N\}$. The algorithm works as follows:

First, the population is initialized, generating a population of N random vectors $\mathbf{x}_i$. Each individual $\mathbf{x}_i$ produces a mutant individual:

$$\mathbf{x}_i^m = \mathbf{x}_p + F(\mathbf{x}_q - \mathbf{x}_r) \tag{34}$$

where p, q and r are randomly chosen integers, mutually different, and also different from index i. Parameter F is a real factor kept constant during the entire optimization process, usually taken from the interval $(0, 2]$.

The mutated vector $\mathbf{x}_i^m$ crosses with $\mathbf{x}_i$, generating an offspring individual $\mathbf{u}_i$. If this individual is better than $\mathbf{x}_i$, $\mathbf{u}_i$ substitutes it in the population, and it is discarded in other case. Note that with this implementation of the differential evolution, all the individuals in the next generation are as good or better than their counterparts in the current generation. The algorithm is usually stopped when a maximum of generations is reached.

4.3. Evolutionary programming algorithm

The Classical Evolutionary Programming algorithm (CEP) was first described in the work by Bäck and Schwefel in [2], and analyzed later by Yao et al. in [94] and [92]. It is used to optimize a given function $f(\mathbf{x})$ (the validation error in our case), i.e. obtaining $\mathbf{x}_o$ such that $f(\mathbf{x}_o) < f(\mathbf{x})$, with $\mathbf{x} \in [lim\_inf, lim\_sup]$ (note that in this case, the values of $[lim\_inf, lim\_sup]$ are set by the initial coarse grain). Then, the CEP algorithm performs as follows:

1. Generate an initial population of μ individuals (solutions). Let t be a counter for the number of generations, set it to $t = 1$. Each individual is taken as a pair of real-valued vectors $(\mathbf{x}_i, \boldsymbol{\sigma}_i)$, $\forall i \in \{1, \cdots, \mu\}$, where $\mathbf{x}_i$'s are objective variables (C, ε and γ), and $\boldsymbol{\sigma}_i$'s are standard deviations for Gaussian mutations.

2. Evaluate the fitness value for each individual ($\mathbf{x}_i$, $\boldsymbol{\sigma}_i$) (using the problem's objective function, the error obtained with the SVM using the parameters $\mathbf{x_i}$, in a validation set).

3. Each parent ($\mathbf{x}_i$, $\boldsymbol{\sigma}_i$), $\{i = 1, \cdots, \mu\}$ then creates a single offspring ($\mathbf{x}'_i$, $\boldsymbol{\sigma}'_i$) as follows:

$$\mathbf{x}'_i = \mathbf{x}_i + \boldsymbol{\sigma}_i \cdot \mathbf{N_1}(\mathbf{0}, \mathbf{1}) \tag{35}$$

$$\boldsymbol{\sigma}'_i = \boldsymbol{\sigma}_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot \mathbf{N}(\mathbf{0}, \mathbf{1})) \tag{36}$$

   where $N(0,1)$ denotes a normally distributed one-dimensional random number with mean zero and standard deviation one, and $\mathbf{N}(\mathbf{0},\mathbf{1})$ and $\mathbf{N_1}(\mathbf{0},\mathbf{1})$ are vectors containing random numbers of mean zero and standard deviation one, generated anew for each value of i. The parameters τ and τ' are commonly set to $(\sqrt{2\sqrt{n}})^{-1}$ and $(\sqrt{2n})^{-1}$, respectively [94], where n is the length of the individuals.

4. If $x_i(j) > lim\_sup$ then $x_i(j) = lim\_sup$ and if $x_i(j) < lim\_inf$ then $x_i(j) = lim\_inf$.

5. Calculate the fitness values associated with each offspring ($\mathbf{x}'_i$,$\boldsymbol{\sigma}'_i$), $\forall i \in \{1, \cdots, \mu\}$.

6. Conduct pairwise comparison over the union of parents and offspring: for each individual, p opponents are chosen uniformly at random from all the parents and offspring. For each comparison, if the individual's fitness is better than the opponent's, it receives a "win".

7. Select the μ individuals out of the union of parents and offspring that have the most "wins" to be parents of the next generation.

8. Stop if the halting criterion is satisfied, and if not, set $t = t + 1$ and go to Step 3.

A second version of the algorithm is the so called Fast Evolutionary Programming (FEP). The FEP was described and compared with the CEP in [94]. The FEP is similar to the CEP algorithm, but it performs a mutation following a Cauchy probability density function, instead of a Gaussian based mutation. The one-dimensional Cauchy density function centered at the origin is defined by

$$f_t(x) = \frac{1}{\pi} \frac{t}{t^2 + x^2} \tag{37}$$

where $t > 0$ is a scale parameter. See [94] for further information about this topic. Using this probability density function, we obtain the FEP algorithm by substituting step 3 of the CEP, by the following equation:

$$\mathbf{x}'_i = \mathbf{x}_i + \boldsymbol{\sigma}_i \cdot \boldsymbol{\delta} \tag{38}$$

where $\boldsymbol{\delta}$ is a Cauchy random variable vector with the scale parameter set to $t = 1$.

Finally, in [94] the *improved FEP* (IFEP) is also proposed, where the best result obtained between the Gaussian mutation and the Cauchy mutation is selected to complete the process.

5. Physics-inspired methods

Within the framework of complex problems in insurance companies, the Physics-inspired algorithm more commonly use, hybridized with other SC ones, is the Simulated Annealing (SA) algorithm.

5.1. Simulated Annealing

Simulated Annealing is an optimization procedure that has been widely applied to solve combinatorial optimization problems [51, 52, 93]. It is inspired by the physical process of heating a substance and then cooling it slowly, until a strong crystalline structure is obtained. This process is simulated by lowering an initial temperature by slow stages until the system reaches to an equilibrium point, and no more changes occur. Each stage of the process consists of changing the configuration several times, until a thermal equilibrium is reached, and then a new stage starts, with a lower temperature. In this process, each configuration of the system stands for a given solution to the optimization problem. Thus, the final solution of the problem is the configuration obtained in the last stage. In the standard SA, the changes in the configuration are performed in the following way: A new configuration is built by a random displacement of the current one. If the new configuration is better, then it replaces the current one, and if not, it may replace the current one probabilistically. This probability of replacement is high in the first stages of the algorithm, and decreases in every stage. Just like in evolutionary-based algorithms, the solution found by SA can be considered a “good enough” solution, but it is not guaranteed to be the best.

Also similarly to evolutionary algorithms, the most important parts in a SA algorithm are: the objective function to be minimized during the process, the chosen representation for solutions and the mutation or configuration change operator.

SA *algorithm:*

```
$k = 0$;
$T = T_0$;
 Obtain an initial solution at random, $\mathbf{X}$;
 evaluate($\mathbf{X}, f(\mathbf{X})$);
repeat
  for $j = 0$ to $\xi$
    $\mathbf{X}_{mut}$ = mutate($\mathbf{X}$);
 evaluate($\mathbf{X}_{mut}, f(\mathbf{X_{mut}})$);
    if(($f(\mathbf{X_{mut}}) < f(\mathbf{X})$) OR ($random(0,1) < e^{(\frac{-\alpha}{T})}$)) then
    $\mathbf{X} = \mathbf{X}_{mut}$;
    endif
  endfor
$T = f_T(T_0, k)$;
$k = k + 1$;
until($T < T_{min}$);
```

In this SA pseudo-code k counts the number of iterations performed; T keeps the current temperature; T_0 is the initial temperature; T_{min} is the minimum temperature to be reached; $\mathbf{X}$ stands for the current configuration and $\mathbf{X}_{mut}$ for the new configuration after the mutation operator is applied; f represents the cost function considered; ξ is the number of changes performed for a given temperature T; f_T is the freezer function; and α is a constant. Parameter α and the initial temperature T_0 are chosen to have an initial acceptance probability about 0.8, a value usually used. The freezer function is defined as

$$f_T = \frac{T_0}{1+k}. \tag{39}$$

The minimum temperature T_{min} is calculated on the basis of the desired number of iterations (*numIt*) as:

$$T_{min} = f_T(T_0, numIt). \tag{40}$$

6. Computation inspired by the collective behavior of social living beings

Ant colony optimization (ACO) and Particle swarm optimization (PSO) are two of the most important SC algoritms more commonly used in insurance. Both are population-based metaheuristics that can be used to find approximate solutions to complex optimization problems in many other fields. In ACO [25], a set of software agents called *artificial ants* search for good solutions to a given optimization problem. To apply ACO, the optimization problem is transformed into the problem of finding the best path on a *weighted graph*. The artificial ants (or simply, ants) incrementally build solutions by moving on the graph. The process of solution construction is stochastic, and is biased by a *pheromone model*, that is, a set of parameters associated with graph components (either nodes or edges) whose values are modified at runtime by the ants. Since Particle swarm optimization (PSO) has been used in some insurance problems, we summarize it in the following sub-section.

6.1. Particle swarm optimization algorithm

Particle swarm optimization (PSO) is a population-based stochastic optimization technique developed by Eberhart and Kennedy [26], inspired by social behavior of bird flocking and fish schooling. A PSO system is initialized with a population of random solutions, and searches for the optimal one by updating the population over several generations. PSO has no evolution operators, such as crossover and mutation as genetic algorithms do, but potential solutions instead, called *particles*, which fly through the problem search space to look for promising regions according to its own experiences and experiences of the whole group. Thus, social information is shared, and also individuals profit from the discoveries and previous experiences of other particles in the search. The PSO is considered a global search algorithm.

Mathematically, given a swarm of N particles, each particle $i \in \{1,2,\cdots,N\}$ is associated with a position vector $\mathbf{x}_i = (x_1^i, x_2^i, \cdots, x_K^i)$, with K the number of parameters to be optimized in the problem (note that in this case $K = 3$, and $x_1^i = C_i$ –parameter C of the i-th particle–, $x_2^i = \varepsilon_i$ –parameter ε of the i-th particle– and finally $x_3^i = \gamma_i$ –parameter γ of the i-th particle–). Let $\mathbf{p}_i$ be the best previous position that particle i has ever found, i.e. $\mathbf{p}_i = (p_1^i, p_2^i, \cdots, p_K^i)$, and $\mathbf{g}$ be the group's best position ever found by the algorithm, i.e $\mathbf{g} = (g_1, g_2, \cdots, g_K)$. At each iteration step $k+1$, the position vector of the ith particle is updated by adding an increment vector $\Delta\mathbf{x}_i(k+1)$, called velocity $\mathbf{v}_i(k+1)$, as follows:

$$v_d^i(k+1) = v_d^i(k) + c_1 r_1 (p_d^i - x_d^i(k)) + c_2 r_2 (g_d - x_d^i(k)), \tag{41}$$

$$v_d^i(k+1) = \frac{v_d^i(k+1) \cdot V_d^{max}}{|v_d^i(k+1)|}, if |v_d^i(k+1)| > v_d^{max}, \tag{42}$$

$$x_d^i(k+1) = x_d^i(k) + v_d^i(k+1), \tag{43}$$

where c_1 and c_2 are two positive constants, r_1 and r_2 are two random parameters which are found uniformly within the interval $[0,1]$, and v_d^{max} is a parameter that limits the velocity of the particle in the dth coordinate direction. This iterative process will continue until a stop criterion is satisfied, and this forms the basic iterative process of a standard PSO algorithm [26].

Once we have in mind the essential aspects of the main algorithms, we can now focus on their potential applications in the field of insurance. This is just the goal of Section 7..

7. Review: Computational intelligence algorithms in insurance problems

The use of Soft-Computing techniques has been a trending topic in insurance applications since the last decade of the last century. Many different techniques have been applied to a huge number of specific problems, and the interest in these approaches has been growing with time, being today the most important family of algorithms in solving insurance-related problems. A first review of computational intelligence techniques in insurance-related problems was carried out in [78], where the different evaluated techniques were structured by their possible application in several major problems arising in the insurance sector. That paper covers quite well the techniques and problems until year 2001. However, since then, there is not a review of the main and novel problems and algorithms developed in this field. There is however a good review of neural networks, genetic algorithms and fuzzy logic techniques in [79]. This section aims to carry out a comprehensive review of techniques and their application to insurance problems, following the same structure that was proposed in [78], i.e., structured on the basis of the different problems arising in the insurance sector.

7.1. Claim fraud detection

Claim fraud detection is an important research area in which Soft-Computing techniques have been successfully applied.

In [19] it was reported on a SC-based fraud detection system for managed healthcare. The essence of the system was that it detected "anomalous" behavior by comparing an individual medical provider to a peer group. The method involved three steps: identify the proper peer population, identify behavior patterns, and analyze behavior pattern properties. The peer population was assumed to be as a three-dimensional space: organization type, geographic region, and organization size. The behavior patterns were developed using the experience of a fraud-detection department, an unsupervised NN that learnt the relationships inherent in the claim data, and a supervised approach that produce a fuzzy model from a knowledge of the decision variables. The behavior pattern properties were analyzed using the statistical measures mean, variance, standard deviation, mean absolute deviation, Kolmogorov-Smirnov (KS) test, skewness, and kurtosis [80].

Other approaches, such as, [85] and [67], have also used fuzzy concepts more recently. In [85], a fuzzy expert system is proposed to evaluate potential fraud cases in insurances. A fuzzy logic expert algorithm is used in [67] to identify claims that carry an element of fraud from amongst all settled claims. Already in 2012, [81] focuses on a scoring model to detect abuse billings patterns in health insurance claims is presented. The scoring is including in a decision tree in order to obtain the corresponding final decision. This scoring improves the performance of decision trees in detecting anomalous billing patterns in health insurance claims.

In [5], the author used an evolutionary-fuzzy approach to investigate suspicious home insurance claims. To that end, genetic programming was employed to evolve FL rules that classified claims into "suspicious" and "non- suspicious" classes. It used clustering to develop membership functions and committee decisions to identify the best-evolved rules. Regarding the former, the features of the claims were clustered into low, medium, and high groups, and the minimum and maximum value in each cluster was used to define the domains of the membership functions. The committee decisions were based on different versions of the system that were run in parallel on the same data set and weighted for intelligibility, which was defined as inversely proportional to the number of rules, and accuracy. It was reported that the results of his model when applied to actual data agreed with the results of previous analysis [80] .

Within the research brach of *neuro-computing techniques*, in [37], a multilayer perceptron was used to classify the practice profiles of a sample of medical general practitioners into four different classes, ranging from having “normal” to having “abnormal” profiles. Then, a Kohonen’s self organizing map was utilized to analyze the classification, and it was found that it was more appropriate to use only two classes instead of four. In [12], a Kohonen self-organizing feature map (a kind of auto-organizing neural network) was applied to uncover automobile bodily injury claims fraud in the insurance industry, including also a neural network with back-propagation to validate the feature map approach. Bayesian learning neural networks have been recently applied to the discovering of personal injury protection (PIP) automobile insurance claim fraud [90]. The paper presents the neural network method and applies it to real data PIP claims from accidents that occurred in Massachusetts, USA, during 1993, and that were previously investigated for suspicion of fraud by domain experts.

Most recently, *evolutionary algorithms* have been used aiming at predicting insurance fraud. In [59], two different evolutionary techniques are applied to predict insurance fraud: a genetic algorithm, and a particle swarm optimization algorithm. It has been found that genetic algorithms and a particle swarm optimization algorithms reach better results than those achieved by using a k-means algorithm.

7.2. Bankruptcy and insolvency prediction

Another important area of research that is gaining popularity in the last few years (and especially during the current crisis) is the *bankruptcy and insolvency predicition* (and related problems) in insurance companies, in which work has been massive. Most of them have been use of combinations of neural-based and evolutionary-based computation methods.

First articles focused on the application of Soft-Computing techniques applied to predict bankruptcy within the insurance sector include, for instance, [66], [10], [46], [11], [55] and [53]. In [66], a multi-layer perceptron designed by means of a GA was used to predict bankruptcy of insurance companies. Soon later, in [10], a three-layer perceptron with a back-propagation algorithm was used to develop an early warning system (2 years prior to insolvency) for US property-liability insurers. In [46], a neural network and a GA were applied aiming at forecasting financial distress in life insurers. In [11] a multi-layer perceptron was applied to predict insurer propensity into insolvency. The proposed neural model was applied to the prediction of insolvency in Texas (USA) companies, with available data between 1987 and 1990. In [55] a model for the evaluation of non-life insurance companies was proposed. The model includes expert systems and neural computation in order to perform an evaluation of the state of companies in such a way that the system output gives the priority (low, medium or high) of the examination of the company state for a human expert. The work presents different results on Dutch non-life insurance companies in 1992 and 1993. In [53] neural networks are proposed to predict automobile insurance losses. The results obtained allow to make the accept/reject decision but not to perform the premium setting function.

More recently, in [73] a Genetic Programming algorithm has been used to predict the insolvency of non-life insurance companies. Different test in Spanish non-life insurance companies with data ranging from 1983 to 1994 have elucidated the good performance of this approach, better than those achieved by other algorithms such as SVMs or rough set. In [31] an interesting model for analyzing and predicting insurance companies rating has been presented. The model is based on a combination of logistic regression and Support Vector Machines to model linear and non-linear relations. A variety of different experiments, with several types of insurance companies, have been carried out to check the validity of the proposal. In [9] a Data Envelopment Analysis approach is used to evaluate the effect of solvency in the efficiency of insurance companies (claims paying ability of the company). In [72], Support Vector Machines for classification with feature selection processes have been applied to the prediction of insolvency in non-life insurance companies in Spain. This model has been extended in [76] to include genetic algorithms and simulated annealing in similar data, also in Spanish insurance companies. In [74], a Rough Set approach has been used in a similar problem of insurance companies insolvency. In [91] a support vector machine has been put into practice to predict bankruptcy, a GA being used estimate the optimal parameters of the SVM.

7.3. Underwriting and establishment of premiums/compensation rates

An important class of problems in insurance is related to underwriting and the adequate establishment of premiums/compensation rates.

In this regard, the early work [18] focused on applying neural networks to replicate the decisions made by mortgage insurance-underwriters, leading to the conclusion that when the results of the neural network differ from the underwriter's, the system's classifications are more consistent with the guidelines than the underwriter's judgement. A different approach was adopted in [56], which describes the computation of fuzzy premiums for an endowment policy. In this paper the underwriting focus was on the definition of a preferred policyholder in life insurance. In [100] fuzzy sets were used to model the selection process in group health insurance.

However, in [63], genetic algorithms and classification trees have been employed to build a knowledge base of rules for being used by an expert system aiming at determining when an insurance policy should be terminated.

In [38], fuzzy logic has been used to study medical underwriting of life insurance. In [101], a fuzzy logic system was used to make pricing decisions in group health insurance and in [102] a fuzzy logic system was applied to adjust workers compensation insurance rates.

Using again the neural network approach, in [89] a multilayer perceptron has been applied to classify life insurance applicants standard and high-risk patterns. In [96] neural networks are used to model the effects of premium changes on motor insurance customer retention rates. In [7] a fuzzy decision system optimized by means of a evolutionary algorithm is used to decide the risk categories of insurance applications. In [97] a model of the premium price sensitivity of automobile insurance customers is developed using a neural network. More recently, in [58] a multi-layer perceptron with back-propagation training is used to improve fire-insurance premiums rates, depending on the corresponding risk level of each policyholder. This method is shown to improve the results obtained with a classical linear interpolation method in a real application in Taiwan. In [54] analyzes the impact of neural networks models as tools in underwriting of private passenger automobile insurance policies. In [22] some computational methods based on statistical mechanic are applied to pricing insurance premiums. In [95] a neural network is used for the problem of automated insurance underwriting. In [39], a neural network with back propagation training is applied to the problem of selecting the best investment-linked insurance, a life insurance type that combines investment and protection, in such a way that premiums provide not only a life insurance cover, but also part of the premiums are invested in specific investment funds to generate benefits. In this paper the neural network is applied to predict the policyholder purchase decision depending on the different characteristics of products. The results of the neural network is compared to that of a logistic regression, obtaining good results. A related problem is the Dynamic Proportion Portfolio insurance problem, solved by using genetic programming in [15].

7.4. Prediction of payments problems

Other interesting applications are related to payments prediction and related problems in insurance companies. For example, in [60] the problem of forecasting when checks issued to retirees and beneficiaries will be presented for payment, in order to forecast daily cash flow requirements was investigated. In [98] a comparison between soft computing methods and traditional approaches for the claim cost prediction in the automobile insurance industry is presented. In [17] a hybrid approach based on a combination of kernel logistic regression and a support vector regression algorithm is applied to obtain tariffs in motor vehicle insurance companies. In [71], a study is made regarding the use of support vector machines to estimate the expected claim rates in the context of catastrophes in China. In [20] fuzzy systems and neural networks are used to estimate total claim amount payments of an insurance company. In fact, in that paper, the neural network is used to estimate the parameters of a fuzzy inference system. The proposed approach is validated in a real problem of total claim amount payments prediction in an insurance company in Turkey, where the proposed hybrid neural system improves the results provided by a multiple linear regression system. A problem of loss risk reduction model suitable for application in construction projects of many different risk profiles is described in [16]. An evolutionary Support Vector Machine is applied in this case to solve the problem.

8. Some experiments and results: prediction of vehicles accidents from drivers and cars' data using Soft-Computing techniques

In the effort of illustrating the suitability of these methods, we give in this section an example of the application of different Soft-Computing approaches to a prediction problem in the insurance field. Specifically, the problem consists in the prediction of having a car accident from a given set of drivers and car's data. As will be shown in detail later on, the database has been provided by a Spanish car insurance company. There are some key aspects that have motivated this application: 1) In Spain, the insurance companies and pension plans management companies control approximately one quarter of a billion euros. 2)

The insurance industry is a strategic sector for our economy from the standpoint of net job creation. In 2009, about 135000 people were working in this sector, 49.25% of them being women. 3) In the Spanish insurance sector, the *non-life* industry is higher than that of life insurance sector. 4) Furthermore, within the non-life industry, the auto insurance industry is the non-life sector that exhibits *the largest premium volume*, representing 36% of the total.

As mentioned in the Introduction, Solvency II compels insurance companies to keep the so-called Solvency Capital Requirements (SCR), aiming to ensure the stability of the company against adverse situations. In order to estimate SCR, four generic types of risk have been defined: market risk (related to financial assets, both fixed income and equity), credit risk (losses due to defaults by debtors of the company), operational risk (because of human and/or technical failures, both internal and external to the insurance company), and subscription risk. Regarding this, Solvency II suggests that part of the Solvency Capital Requirements, which have to be secured, is related to the non-zero possibility of having underestimated the risks assumed with a novel client. This is one of the reasons why we propose a Soft-Computing method −based on data related to different types of clients and cars− to predict the probability for a given client to have an accident. This is of the key importance for the insurance car company in order to estimate how much money would be necessary to guarantee the payment of financial obligations that have contracted with such insured person: medical expenses, liability, fire, support costs, legal costs, repair costs, and so on.

As a consequence, the problem is of importance, of course, for the majority of insurance companies covering vehicles liabilities. Moreover, insurance companies aim to classify the insured policies into homogeneous tariff classes, assigning the same premium to all the policies belonging to the same class. The classification of the policies into the classes is based on the selection of the so-called risk factors, which are characteristics or features of the policies that help the companies to predict their claim amounts in a given period of time (usually one year). As mentioned before, in automobile insurance, these are observable variables concerning the driver, vehicle and traffic, like age, driving license date, kind of vehicle, circulation zone, etc., that are correlated with the claim rates, and therefore can be useful in order to predict the future claims.

Using a modern modeling, the prediction of having a traffic accident can be stated as a *binary classification problem*, in the sense that if an insured person has an accident, he/she will belong to the first class ("accident-prone class"), while otherwise, it will belong to the second class (that grouping "non accident-prone" ones). The problem will thus consist in designing a soft-computing-based intelligent machine, ξ, capable of predicting whether or not a candidate client will be prone to car accident.

In the effort of an accurate design, it is necessary for the intelligent machine ξ to be designed (“learning process”) by using a sufficiently long number of data (belonging to a “design set”) different from those used to test (“test set”) to what extent the machine is able to properly predict whether or not *other* person (not belonging to the training set) will have an accident. The set formed by the union of both training and test sets is usually called “database”. Or in other words, the database is divided into a training set and a test set. Sometimes, when designing classifiers such as MLPs, the design set is in turn divided into a training set and a validation set.

In a more detailed way, the structure of the design set S_{design} plays a key role. It contains L samples that represent M clients, each sample being formed by N “features” (or variables presumably involved in the success), and a “label” ($y = 1$ means accident-prone, while $y = -1$ represents non accident-prone) associated to each client. The test set S_{test} exhibits the same structure than S_{design} but with data corresponding to persons different from those belonging to the training set. Usually, the number of samples in S_{design} is longer than that in S_{test}. Once the database has been properly designed, the problem consists now in training a classification machine ξ with the data of S_{design} in such a way that it provides the minimum possible classification error when comparing the result achieved by the classifier (under design using data belonging to S_{design}) to the corresponding label. The classification machine ξ will be thus the one that minimizes the error *over the test set* S_{test} (that is, using data of clients different from those used in the training process). This will ensure the generalization of the results given by the classification machine ξ, i.e., given a new candidate client represented by sample s, the probability of misclassify him/her in terms of his/her prone to accident is as minimum as possible.

The number of features in this prediction problem is reduced, since they described data from drivers and cars (limited values of variables). Thus, it is not necessary a feature selection systems. This means that the different algorithms described in Section 3. may be directly applied to the problem. Note that for insurance companies it is very important to select an adequate set of risk factors in order to predict the future claim rates correctly and to charge fair premiums to the drivers. There are several works dealing with the subject of the risk classification of policyholders [21, 23, 57], though not many of them apply Soft-Computing techniques.

Specifically, in our problem of car accident prediction from drivers and cars’ data, we have made use of 13 prediction variables. These variables can be considered as risk factors. They are qualitative and quantitative variables. Table 2 shows the considered variables, including a brief description of each one.

Regarding to the database used in the experimental work, we have considerer a total of 5500 Spanish automobile and drivers’ data, all belong to year 2005. The 13 prediction variables listed in Table 2 have been completed with the corresponding labels ($y = 1$, for accident or $y = -1$, for not accident). As mentioned before, the database have been divided into a design set (80%) and a test set (20%).

We have explored the use of a number of classifiers, such as the well known K-Nearest Neighbors (KNN) approach, the Extreme Learning Machine (ELM), a Multi-Layer Perceptron trained with the Levenberg-Marquardt algorithm [35], the two decision trees algorithms (the PART and the C4.5) and the Support Vector Machine for classification (SVMC). Table 3 list the main parameters we have considered in the experimental work we have carried out in this particular problem.

Fig. 3 will assist us in clearly illustrating the results achieved by any of the explored classifiers. It represents the accuracy classification, ACC (%), as a function of the method used for the prediction of accidents from driver and cars’ data. All the results have been computed *over the test set*. Note that the best approach is the Support Vector Machine for classification (ACC=84.63%), which outperforms those reached by other techniques: KNN (ACC=78.72%), ELM (ACC=79.32%), MLP (ACC=80.03%), PART (ACC=80.15%) AND C4.5 (ACC=78.17%). In this case, all the algorithms considered are about 80% of accuracy in the problem (test set), which is a pretty good result. The SVMC, however, is closed to

Table 2. Prediction variables (and their definition) involved in the problem of prediction of vehicles accidents from drivers and cars' data. These variables can be considered as risk factors, and may be qualitative and quantitative features.

Variable	Definition
Kind of vehicle	Six types of vehicles are considered.
Use	Use to which the vehicle is devoted. It takes 10 values.
CV	Vehicle power.
Private	Private or public vehicle.
Tare	Tare (weight).
Plazas	Number of seats of the vehicle.
Ambit	Circulation area of the vehicle.
Years of the vehicle	The age of the vehicle.
Policyholder age	The age of the policyholder.
Driving license	Years of validity of the driving license.
Gender	Male or female.
Region	Autonomous region.
Diesel	Diesel or gasoline.

Table 3. Summary of the different classifying methods used for comparative purposes and the main parameters utilized in the experiments we have carried out.

Method	Values for the main parameters
ELM	from 5 to 40 neurons, in steps of 5
KNN	from 1 to 10 neighbors, in steps of 1
MLP	from 7 to 20 neurons, in steps of 1
SVMC	γ: from 2^{15} to 2^{3}
	C from 2^{5} to 2^{15}, using a logarithmic scale
PART	Standard values of classical decision tree algorithms
C4.5	Standard values of classical decision tree algorithms.

the 85% of accuracy computed over the test set.

9. Conclusions

In this Chapter we have carried out a review of several Computational Intelligence algorithms centered on applications in insurance problems. The insurance sector plays currently a growing and crucial role among the different components of the financial system. And just because of this importance, it is long recognized that there is a need for some form of *prudential* supervision of insurance entities aiming to minimize failure risk. The main concern of the regulatory authority, through Solvency II, is the protection of the insured

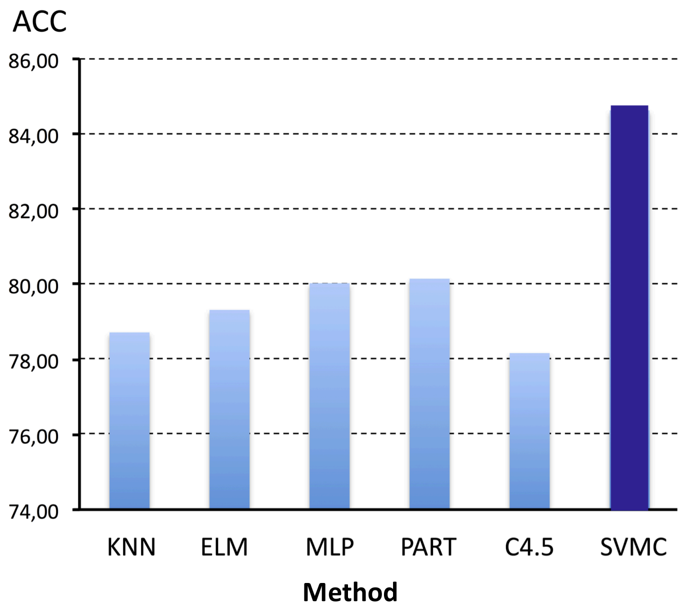

Figure 3. Representation of the accuracy classification ACC (%) as a function of the method used for the prediction of accidents from driver and car data.

person, in the sense that the insurance company have to be able to guarantee the payment of financial obligations that have contracted with the insured person. Thus, it is compulsory to keep the Solvency Capital Requirements, whose objective is to ensure the stability of the company *against adverse situations*, as currently is occurring. Regarding this, Solvency II suggests that part of the Solvency Capital Requirements is related to the *non-zero possibility of having underestimated the risks assumed with a novel client*. Soft-Computing or Computational Intelligence methods play a key role in assisting a company in accurately predicting such a risk, and in facing many other important problems related with the processing of a *huge amount of uncertain, imprecise and incomplete information*. In these cases, the application of Soft-Computing techniques has been massive in the last years.

He have shown how Soft-Computing techniques refer to a set of modern computational methodologies, focused on the design of intelligent systems, which are able to process huge amount of uncertain, imprecise and incomplete data, and to find approximate solutions for problems that otherwise would be intractable. As motivated, we have classified the hugh amount of algorithms basically into five sets of Soft-Computing techniques (some of them exhibiting non-null intersection): Evolutionary Computation, Neural Computation, Computation based on Fuzzy concepts, Physics-Inspired Computation, and Computation inspired by Social living-being networks. We have adopted this classification for convenience and illustrative purposes, although other authors could carried out others since, as mentioned, some algorithms may belong to different soft-computing approaches.

Because of their importance, we have specifically reviewed the most important that have been applied in the insurance sector: Rough Sets, Multi-layer Perceptrons, Extreme Learning Machines, Support Vector Machines (classification and regression), Group Method of

Data Handling, Genetic Algorithms, Differential Evolution, Evolutionary Programming, Simulated Annealing, and Particle Swarm Optimization. We have also discussed some of their most important applications in insurance problems, which have been published in top international journals. Finally, in the effort of illustrating the suitability of these methods, we have focused on the real problem of predicting vehicle accidents on the basis of drivers and vehicles data, reaching very good results.

We hope this review chapter can be used as introductory material for those researchers interested in tackle different insurance problems with modern computation techniques.

Acknowledgement

This work has been partially supported by Spanish Ministry of Science and Innovation under project number ECO2010-22065-C03-02, and by the Universidad de Alcalá, under project number UAH2011/EXP-026.

References

[1] L. Anastasakis and N. Mort, "The development of self-organization techniques in modelling: A review of the group method of data handling (GMDH)," *Research Report*, Department of Automatic Control and Systems Engineering, The University of Sheffield, (813), 2001.

[2] T. Bäck, G. Rudolph and H. P. Schwefel, "Evolutionary programming and evolution strategies: similarities and differences," In *Proc. of the Second Annual Conference on Evolutionary Programming*, pp. 11-22, 1993.

[3] S. Bandyopadhyay, "Analysis of Biological Data: A Soft Computing Approach, " In *Science, Engineering, and Biology Informatics*, vol. 3, World Scientific, 2007.

[4] V. E. Balas, J. Fodor and A. Varkonyi-Koczy, "New Concepts and Applications in Soft Computing, " In *Studies in Computational Intelligence*, vol. 417, Springer, 2012.

[5] P. J. Bentley, "Evolutionary, my dear Watson, investigating committee-based evolution of fuzzy rules for the detection of suspicious insurance claims, " In *Proceedings of the Second Genetic and Evolutionary Computation Conference (GECCO)*, vol. 1, pp. 8-10.

[6] C. M. Bishop, *Neural networks for pattern recognition*, Oxford University Press, 1995.

[7] P. P. Bonisone, R. Subbu and K.S. Aggour, "Evolutionary optimization of fuzzy decision systems for automated insurance underwriting", In *Proceedings of the 2002 IEEE International Conference on Fuzzy Systems*, vol. 2, pp. 1003-1008, 2002.

[8] A. Brabazon, M. O'Neill and D. Maringer, "Natural Computing in Computational Finance", In *Studies in Computational Intelligence*, vol. 4, Springer, 2011.

[9] P. L. Brockett, W. W. Cooper, L. L. Golden, J. J. Rousseau and Y. Wang, "Evaluating solvency versus efficiency performance and different forms of organization and marketing in US property–liability insurance companies," *European Journal of Operational Research*, vol. 154, pp. 492-514, 2004.

[10] P. L. Brockett, W. W. Cooper, L. L. Golden and U. Pitaktong, "A neural network method for obtaining an early warning of insurer insolvency," *Journal of Risk and Insurance*, vol. 61, no. 3, pp. 402-424, 1994.

[11] P. L. Brockett, W. W. Cooper, L. L. Golden and X. Xia, "A case study in applying neural networks to predicting insolvency for property and casualty insurers," *Journal of the Operational Research Society*, vol. 48, vol. 12, 1153-1162, 1997.

[12] P. L. Brockett, X. Xia and R. A. Derrig, R.A., "Using Kohonen's self-organizing feature map to uncover automobile bodily injury claims fraud," *Journal of Risk and Insurance* vol. 65, no. 2, pp. v245-274, 1998.

[13] C. C. Chang and C. J. Lin, "Libsvm: a library for support vector machines," 2001, available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm. pp. 989-993, 1994.

[14] D.K. Chaturvedi, “Soft Computing: Techniques and Its Applications in Electrical Engineering,” *Studies in Computational Intelligence*, vol. 103, Springer, 2010.

[15] J. S Chen, C. L. Chang, J. L. Hou and Y. T. Lin, “Dynamic proportion portfolio insurance using genetic programming with principal component analysis,” *Expert Systems with Applications*, vol. 35, pp. 273-278, 2008.

[16] M. Y. Cheng, H. S. Peng, Y. W. Wua and Y. H. Liao, “Decision making for contractor insurance deductible using the evolutionary support vector machines inference model,” *Expert Systems with Applications*, vol. 38, pp. 6547-6555, 2011.

[17] A. Christmann, “On a strategy to develop robust and simple tariffs from motor vehicle insurance data,” *Acta Mathematicae Applicatae Sinica, English Series*, vol. 21, no. 2, pp. 193-208, 2005.

[18] E. Collins, S. Ghosh and C. Scofield, “An application of a multiple neural network learning system to emulation of mortgage underwriting judgements,” In *IEEE International Conference on Neural Networks*, pp. 459-466, 1988.

[19] E. Cox, “Fuzzy system for detecting anomalous behaviors in healthcare provider claims.,” In *Intelligent Systems for Finance and Business*, pp. 111-135, Wiley, 1995.

[20] T. E. Dalkilic, F. Tank and K. S. Kula, “Neural network approach for determining total claim amounts in insurance,” *Insurance: Mathematics and Economics*, vol. 45, pp. 236-241, 2009.

[21] S. D’Arcy, “Predictive modeling in automobile insurance: a preliminary analysis,” In *Proc. of the World Risk and Insurance Economics Congress*, August, Salt Lake City, No. 302, November, 2005.

[22] A. H. Darooneha ”Non-life insurance pricing: multi-agent model,” *European Physical Journal B*, vol. 42, pp. 119-122, 2004.

[23] M. Denuit, X. Maréchal, S. Pitrebois, and J. F. Walhin, “Index in Actuarial Modelling of Claim Counts: Risk Classification,” *Credibility and Bonus-Malus Systems*, John Wiley & Sons, Ltd, Chichester, UK, 2007.

[24] L. Dymowa, ”Soft Computing in Economics and Finance,” In *Intelligent Systems Reference Library*, vol. 6, Springer, 2011.

[25] M. Dorigo and C. Blum, “Ant colony optimization theory: A survey,” *Theoretical Computer Science*, vol. 344, pp. 243-278, 2005.

[26] R. Eberhart and Y. Shi, “Particle swarm optimization: developments, applications and resources”, In *Proc. IEEE Congress on Evolutionary Computation*, 2001.

[27] A. E. Eiben and J. E. Smith, “Introduction to evolutionary computing,” Springer-Verlag, 2003.

[28] J. Enderle and J. Bronzino, “Introduction to Biomedical Engineering,” In *Academic Press series in biomedical engineering*, Academic Press, 2011.

[29] D. B. Fogel, “An introduction to simulated evolutionary optimization,” *IEEE Trans. Neural Networks*, vol. 5, no. 1 pp. 3-14, 1994.

[30] D. Gabor, W. Wildes and R. Woodcock,“A universal non-linear filter predictor and simulator which optimices itself by a learning process,” *Proceedings of the IEEE*, vol. 108B, pp. 422-438, 1961.

[31] T. V. Gestel, D. Marten, B. Baesens and D. Feremans, J. Huysmans and J. Vanthienen, “Forecasting and analyzing insurance companies’ ratings,” *International Journal of Forecasting*, vol. 23, pp. 513-529, 2007.

[32] M. Gilli and E. Schumann, “Heuristic optimisation in financial modelling,” *Annals of Operation Research*, vol. 193, pp. 129-158, 2012.

[33] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, Reading, MA: Addison-Wesley, 1989.

[34] K. Gopalakrishnan, S. K. Khaitan and S. Kalogirou, *Soft Computing in Green and Renewable Energy Systems*, In *Studies in Fuzziness and Soft Computing*, vol. 269, Springer, 2011.

[35] M. T. Hagan and M. B. Menhaj, “Training feed forward network with the Marquardt algorithm,” *IEEE Transactions on Neural Networks*, vol. 5, no. 6, 1994.

[36] S. Haykin, *Neural networks: a comprenhensive foundation*, Prentice Hall, 1998.

[37] H. He, J. Wang and W. Graco, “Application of neural networks to detection of medical fraud,” *Expert systems with applications*, vol. 13, no. 4, pp. 329-336, 1997.

[38] P. Horgby, R. Lohse and N. Sittaro, “Fuzzy underwriting: an application of fuzzy logic to medical underwriting,” *Journal of Actuarial Practice*, vol. 5, no. 1, pp. 79-104, 1997.

[39] K. T. Hsu, “Using a back propagation network combined with grey clustering to forecast policyholder decision to purchase investment-inked insurance,” *Expert Systems with Applications*, vol. 38, pp. 6736-6747, 2011.

[40] G. B. Huang, Q. Y. Zhu and C. K. Siew, “Extreme Learning Machine: Theory and Applications,” *Neurocomputing*, Vol. 70, N. 1–3, pp. 489–501, 2006.

[41] G. B. Huang, L. Chen and C. K. Siew, “Universal approximation using incremental constructive feedforward networks with random hidden nodes,” *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879-892, 2006.

[42] G. B. Huang and L. Chen, “Convex incremental extreme learning machine,” *Neurocomputing*, vol. 70, pp. 3056-3062, 2007.

[43] G. B. Huang and L. Chen, “Enhanced random search based incremental extreme learning machine,” *Neurocomputing*, vol. 71, pp. 3460-3468, 2008.

[44] G. B. Huang, D. H. Wang and Y. Lan, “Extreme learning machines: a survey,” *International Journal of Machine Learnign and Cybernetics*, vol. 2, no. 2, pp. 107-122, 2011.

[45] G. B. Huang, H. Zhou, X. Ding and R. Zhang, “Extreme learning machine for regression and multi-class classification,” *IEEE Transactions on Systems, Man and Cybernetics, Part B*, in press, 2012.

[46] C. Huang, R. E. Dorsey and M. A. Boose, “Life insurer financial distress prediction: a neural network model,” *Journal of Insurance Regulation*, vol. 13, no. 2, pp. 131-167, 1994.

[47] A.G. Ivakhnenko. “The group of method of data handling, a rival of the method of stochastic approximation,” *Soviet Automatic Control c/c of Automatica*, vol. 1, no. 3, pp. 43-55, 1968.

[48] A. G. Ivakhenko and G. A. Ivakhenko, “The review of problems solvable by algorithms of the group method of data handling (GMDH),” *Pattern Recognition and Image Analysis*, vol. 4, no. 2, pp. 185-196, 2003.

[49] F. Jopp, “Modelling Complex Ecological Dynamics: An Introduction Into Ecological Modeling for Students, Teachers and Scientists,” Springer, 2011.

[50] J. Kacprzyk, “New Learning Paradigms in Soft Computing,” In *Studies in Fuzziness and Soft Computing*, vol. 84, Physica-Verlag, 2011.

[51] S. Kirpatrick, C. D. Gerlatt and M. P. Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, pp. 671-680, 1983.

[52] S. Kirpatrick, “Optimization by simulated annealing–Quantitative studies,” *J. Stat. Phys.*, vol. 34, pp. 975-986, 1984.

[53] F.L. Kitchens, J.D. Johnson and J.N.D. Gupta, “Predicting automobile insurance losses using artificial neural networks,” In *Neural networks in business: techniques and applications*, K.A. Smith, J.N.D. Gupta Eds., pp. 167-187, IRM Press, 2002.

[54] F.L. Kitchens, “Financial implications of artificial Neural Networks in automobile insurance underwriting,” International Journal of Electronic Finance, vol. 3, no. 3, 2009.

[55] B. Kramer, “N.E.W.S.: A model for the evaluation of non-life insurance companies,” *European Journal of Operational Research*, vol. 98, pp. 419-430, 1997.

[56] J. Lemaire, “Fuzzy insurance,” *ASTIN Bulletin*, vol. 20, no. 1,pp. 33-55, 1990.

[57] J. Lemaire, *Bonus-Malus Systems in Automobile Insurance*, Boston Kluwer Academic Publisher, 1995.

[58] C. Lin, “Using neural networks as a support tool in the decision making for insurance industry,” *Expert Systems with Applications*, vol. 36, pp. 6914-6917, 2009.

[59] J.L. Liu and C.L. Chen, “Application of evolutionary data mining algorithms to insurance fraud prediction,” In *Proc. of the IACSIT Hong Kong conferences*, IPCSIT vol. 25, 2012.

[60] L. Lokmic and K. A. Smith, “Cash flow forecasting using supervised and unsupervised neural networks,” In *Proc. of the of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, vol. 6, pp. 343-347, 2000.

[61] T. McKee, “Developing a bankruptcy prediction model via Rough Sets theory,” *International Journal of Intelligent Systems in Accounting, Finance and Management*, vol. 9, pp. 159-173, 2000.

[62] P. Melin, “Soft Computing for Recognition Based on Biometrics,” In *Studies in Computational Intelligence*, vol. 312, Springer, 2010.

[63] C. Nikolopoulos and S. Duvendack, “A hybrid machine learning system and its application to insurance underwriting,” In *Proc. of the First IEEE Conference on Evolutionary Computation*, pp. 692-695, 1994.

[64] H. Nurmi, J. Kacprzyk and M. Fedrizzi, “Probabilistic, fuzzy and rough concepts in social choice,” *European Journal of Operational Research*, vol. 95, pp. 264-277, 1996.

[65] E. Ortiz-García, S. Salcedo-Sanz, A. Pérez-Bellido and J. A. Portilla-Figueras, “Improving the training time of support vector regression algorithms through novel hyper-parameters search space reductions,” *Neurocomputing*, vol. 72, pp. 3683-3691 , 2009.

[66] J. Park, *Bankruptcy prediction of banks and insurance companies: an approach using inductive methods*, Ph.D. Dissertation, University of Texas, Austin, 1993.

[67] J. Pathak, N. Vidyarthi and S.L. Summers, “A fuzzy-based algorithm for auditors to detect elements of fraud in settled insurance claims,” *Managerial Auditing Journal*, vol. 20 no. 6, pp. 632 - 644, 2005.

[68] Z. Pawlak, *Rough sets. Theoretical aspects of reasoning about data*, London: Kluwer Academic Publishers, 1991.

[69] C.A. Peña-Reyes and M. Sipper, “A fuzzy-genetic approach to breast cancer diagnosis,” *Artif. Intell. Med.*, vol. 17, pp. 131 - 155, 1999.

[70] K. Price, R. Storn and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Springer Verlag, 2005.

[71] Z. Ren and M. Chen, “The Study on Measurement of Non-Life Insurance Output and Added Value in the Context of Catastrophes in China”, In *Proc. of the second international symposium on intelligent information technology (IITA’08)*, vol. 2, pp. 928-934, 2008

[72] S. Salcedo-Sanz, M. DePrado-Cumplido, M. J. Segovia-Vargas, F. Pérez-Cruz and C. Bousoño-Calzón, “Feature selection methods involving SVMs for the prediction of insolvency in non-life insurance companies,” *Intelligent Systems in Accounting Finance and Management*, vol. 12, pp. 261-281, 2004.

[73] S. Salcedo-Sanz, J. L. Fernández-Villacañas, M. J. Segovia-Vargas and Carlos Bousoño-Calzón, "Genetic programming for the prediction of insolvency in non-life insurance companies," *Computers & Operations Research*, vol. 32, pp. 749-765, 2005.

[74] A. Sanchis, M. J. Segovia, J. A. Gil, A. Heras, J. L. Vilar, "Rough Sets and the role of the monetary policy in financial stability (macroeconomic problem) and the prediction of insolvency in insurance sector (microeconomic problem)," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1554-1573, 2007.

[75] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18-28, 2008.

[76] M. J. Segovia-Vargas, S. Salcedo-Sanz and C. Bousoño-Calzón, "Prediction of insolvency in non-life insurance companies using support vector machines, genetic algorithms and simulated annealing," *Fuzzy Economic Review*, vol. 9, pp. 79-97, 2004.

[77] R. Seising and V. Sanz, "Soft Computing in Humanities and Social Sciences," In *Studies in Fuzziness and Soft Computing*, vol. 273, Springer, 2011.

[78] A. Shapiro, "Soft-Computing applications in actuarial science," *ARCH*, pp. 1-14, 2001.

[79] A. Shapiro, "The merging of neural networks, fuzzy logic, and genetic algorithms," *Insurance: Mathematics and Economics*, vol. 31, pp. 115-131, 2002.

[80] A. F. Shapiro, "Fuzzy logic in insurance," *Insurance: Mathematics and Economics*, vol. 35, pp. 399-424, 2004.

[81] H. Shin, H. Park, J. Lee and W. C. Jhee, "A scoring model to detect abusive billing patterns in health insurance claims," *Expert Systems with Applications*, vol. 39, pp. 7441-7450, 2012.

[82] J. Y. Shyng, F. K. Wang, G. H. Tzeng and K. S. Wu, "Rough Set Theory in analyzing the attributes of combination values for the insurance market," *Expert Systems with Applications*, vol. 32, pp. 56-64, 2007.

[83] A. J. Smola, N. Murata, B. Scholkopf and K. Muller, "Asymptotically optimal choice of ε-loss for support vector machines," In *Proc. of the 8th International Conference on Artificial Neural Networks, Perspectives in Neural Computing*, 1998.

[84] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression", *Statistics and Computing*, vol. 14, pp. 199-222, 2004.

[85] B. Stefano and F. Gisella, "Insurance fraud evaluation: a fuzzy expert system," In *Proc. of the 10th IEEE International Conference on Fuzzy systems*, vol. 3, pp. 1491-1494, 2001.

[86] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341-359, 1997.

[87] A. Tettamanzi, M. Tomassini and J. Janβen, "Soft Computing: Integrating Evolutionary, Neural, and Fuzzy Systems", Springer, 2010.

[88] V. N. Vapnik, "Statistical Learning Theory", In Adaptive and Learning Systems for Signal Processing, Communications and Control, S.Haykin, Ed. John Wiley and Sons, 1998.

[89] M. L. Vaughn, E. Ong and S. J. Cavill, "Interpretation and knowledge discovery from a multilayer perceptron network that performs whole life assurance risk assessment," *Neural Computing and Applications*, vol. 6, pp. 201-213, 1997.

[90] S. Viaenea, G. Dedeneb and R. A. Derrig, "Auto claim fraud detection using Bayesian learning neural networks," *Expert Systems with Applications*, vol. 29, pp. 653-666, 2005.

[91] C.H. Wu, G.H. Tzeng, Y.J. Goo and W.C. Fang, "A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy," *Expert systems with applications*, vol. 32, no. 2, pp. 397-408, 2007.

[92] C. Y. Lee and X. Yao, "Evolutionary programming using mutations based on the Levy Probability distribution," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 1-13, 2004.

[93] X. Yao, "A new simulated annealing algorithm," *International Journal of Computer Mathematics*, vol. 56, pp. 161-168, 1995.

[94] X. Yao, Y. Liu and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82-102, 1999.

[95] W. Yan and P.P. Bonissone, "Designing a Neural Network Decision System for Automated Insurance Underwriting," In *International joint conference on neural networks*, pp. 2106-2113, 2006.

[96] A. Yeo, K. Smith, R. Willis and M. Brooks, "Modeling the Effect of Premium Changes on Motor Insurance Customer Retention Rates Using Neural Networks, " In Lecture notes in computer science, vol. 2074, pp. 390-399, Springer Berlin / Heidelberg, 2001.

[97] A.C. Yeo, K.A. Smith, R.J. Willis and M. Brooks, "Using neural networks to model premium price sensitivity of automobile insurance customers," In *Neural networks in business: techniques and applications*, K.A. Smith, J.N.D. Gupta Eds., pp. 41-54, IRM Press, 2002.

[98] A.C. Yeo, K.A. Smith, R.J. Willis and M. Brooks, "A comparison of soft computing and traditional approaches for risk classification and claim cost prediction in the automobile insurance industry," In *Soft computing in measurement and information acquisition*, L. Reznik and V. Kreinovich Eds., Springer-Verlag, 2003.

[99] J. H. Yoo, B. H. Kang and J.U. Choi, , "A hybrid approach to auto-insurance claim processing system," In *IEEE Int. Conf. Syst. Man Cybern.*, vol. 1, pp. 537-542, 1994.

[100] V. R. Young, “The application of fuzzy sets to group health underwriting,” *Transactions of the Society of Actuaries*, vol. 45, pp. 551-590, 1993.

[101] V. R. Young, “Insurance rate changing: a fuzzy logic approach,” *Journal of Risk and Insurance*, vol. 63, pp. 461-483, 1996.

[102] V. R. Young, “Adjusting indicated insurance rates: fuzzy rules that consider both experience and auxiliary data,” *Proceedings of the Casualty Actuarial Society*, vol. 84, pp. 734-759, 1997.

[103] L. A. Zadeh, “Fuzzy sets,” *Information and Control*, vol. 8, pp. 338-353, 1965.

[104] L. A. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning. Part I,” *Information Sciences*, vol. 8, pp. 199-249, 1975.

[105] L. A. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning. Part II,” *Information Sciences*, vol. 8, pp. 301-357, 1975.

[106] L. A. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning. Part III,” *Information Sciences*, vol. 9, pp. 43-80, 1975.

[107] L. A. Zadeh, “A new direction in AI - toward a computational theory of perceptions,” *AI Magazine*, vol. 22, pp. 73-84, 2001.

[108] L. A. Zadeh, “Toward a perception-based theory of probabilistic reasoning with imprecise probabilities,” *Journal of Statistical Planning and Inference*, vol. 105, pp. 233-264, 2002.

[109] L. A. Zadeh, “Precisiated natural language (PNL),” *AI Magazine*, vol. 3, pp. 74-91, 2004.

[110] L. A. Zadeh, “Toward a generalized theory of uncertainty (GTU) - an outline,” *Information Sciences,* , vol. 172, pp. 1-40, 2005.

[111] L. A. Zadeh, “From imprecise to granular probabilities,” *Fuzzy Sets and System*, vol. 154, pp. 370-374, 2005.

[112] L. A. Zadeh, “From search engines to question answering systems - the problems of world knowledge relevance deduction and precisiation,” (Chapter 9) in: Elie Sanchez (Ed.),” *Fuzzy Logic and the Semantic Web* , vol. 1, pp. 163-210, 2006.

[113] L. A. Zadeh, “Generalized theory of uncertainty (GTU) - principal concepts and ideas,” *Computational Statistics and Data Analysis*, vol. 51, pp. 15-46, 2006.

[114] L. A. Zadeh, “Understanding Fuzzy Logic: An Interview with Lotfi Zadeh,” *IEEE Signal Processing Magazine*, vol. 5, pp. 101-105, 2007.

[115] L. A. Zadeh, “Is there a need for fuzzy logic?,” *Information Sciences*, vol. 178, pp. 2751-2779, 2008.