© International Journal of Combinatorial Optimization Problems and Informatics, Vol. 10, No. 1, Jan-April 2019, pp. 40-50. ISSN: 2007-1558.

Fuzzy GA-SVR for Mexican Stock Exchange's Financial Time Series Forecast with Online Parameter Tuning

Javier Alberto Rangel González [1], Juan Frausto Solís[1], Héctor Joaquín Fraire Huacuja[1], Juan Javier González Barbosa[1], Rodolfo Abraham Pazos Rangel[1]

[1]Tecnológico Nacional de México - Instituto Tecnológico de Ciudad Madero
E-mail: javieralberto64@hotmail.com, juan.frausto@itcm.edu.mx, automatas2002@yahoo.com.mx, jjgonzalezbarbosa@hotmail.com, r_pazos_r@yahoo.com.mx

Abstract. Time series play a major role in a large number of economic problems such as predicting the volatility of price indices and market rates on the stock exchanges. Volatility reflects the behavior of fluctuations in the prices of assets, allowing to measure the risk of portfolios. However, volatility is not directly observable, is not constant throughout the period, is dependent on the past and does not follow Gaussian normality. In the literature, there have been many studies using classic methods for the Mexican Stock Exchange. New heuristic methods of supervised learning for forecasting financial time series are presented in this paper. These methods are structured in such a way that, knowing only the time series, the values of the dependent variable several periods ahead are estimated with low errors. This work presents a hybrid method based on support vector machines, fuzzy controllers, evolutionary algorithms and classic methods applied to the asset portfolios optimization of the Mexican Stock Exchange. New heuristic methods of supervised learning for forecasting financial time series are presented in this paper. Besides, a fuzzy logic approach with a Genetic Support Vector Regression (Fuzzy GA-SVR) algorithm for the prediction of time series with a broad spectrum of application is presented. This algorithm uses an online tuning process to produce superior efficacy than the classic forecasting methods. Experimentation presented in the paper shows that Fuzzy GA-SVR outperforms both, the classic ARIMA methods and the individual use of SVR.

Keywords. SVR, Fuzzy Logic, Genetic Algorithm, Mexican Stock Exchange

1 Introduction

Time series classic methods have limited application in problems of predicting the volatility of investment instruments. New Support Vector Regression (SVR) methods have proven to be successful in such problems; however, they have only been applied to certain types of series, usually distinguished by parameters in the area [1].

Thousands of purchase and sale transactions of various stock exchange instruments are carried out continuously. In economic problems, one of the most important variables is volatility. In essence, this variable reflects the behavior of fluctuations in asset prices, allowing the portfolio risk to be measured [2].

However, volatility determination has three main issues:

1) Volatility is not a direct observable variable.

2) Volatility's data dispersion metrics are unknown.

3) Considering a Gaussian distribution, a volatility measure considers a variance along the time series [3]. This measure seeks resolution of the above problems and is implemented in volatility forecasting classic methods. However, volatility is value-dependent throughout the time series and does not follow a Gaussian distribution. Support Vector Regression (SVR) and fuzzy logic algorithms are not affected by these problems. Nevertheless, one of the main problems of SVR applications is to tune correctly the kernel parameters.

Received May 24, 2016 / Accepted Sep 11, 2017

In this paper a fuzzy controller with automatic rules generation by a genetic algorithm for tuning the kernel of an SVR model is proposed; this allows forecasting the value of the time series several periods ahead. The proposed method has high-quality prediction, and low time-consuming solutions that compete with classical forecasting time-series methods. This novel approach has shown significantly better results than the classic methods.

2 Problem Description

A time series is a sequence of observations (usually over time) of a changing phenomenon. The underlying idea of time series forecasting is that the patterns associated with past values in a data series can be used to project future values [4]. If the sequence of observations of a time series is expressed by $x(t)\ t = 1,2, \ldots$ then the expression $\hat{x}(t+h)$ is the prediction horizon for period h at time t, where $h = 1$ means a step forward and $h > 1$ means multiple steps in the prediction ahead. Forecasting time series methods can be classified into 3 types: critical, univariate and multivariate [5].

The critical forecast is based on the interpretation of any relevant information. The prediction in univariate methods depends only on actual or past values of a single time series. The forecast for a variable in multivariate methods depends, at least partially, on the values of the variables of other time series. Univariate and multivariate methods are organized into two subclasses [6]: traditional statistical methods (exponential smoothing, ARIMA, GARCH, etc.) and data mining methods such as neural networks and SVRs.

3 Forecasting Methods for Financial Series

This section briefly reviews support vector regression, fuzzy controller, genetic algorithm, and ARIMA. The main characteristics by which these methods were chosen for the proposed algorithm in this paper are:

- The SVR ability to learn from a training set and to be able to predict future values in a time series.
- The capacity of genetic algorithms to generate different random populations and to take them as a set of fuzzy rules.
- The possibility of setting the parameters of a fuzzy controller through fuzzy rule sets on a fuzzy controller.
- SVR forecasting methods are not affected by the distribution of the time series data.

The above suggests that SVR forecasting methods are theoretically superior to the classic methods. This is verified in this paper with the method proposed, and the details are presented in the following sections.

3.1 Support vector regression

Vapnik proposes in 1994 the Vector Support Machine (SVM) [7] which is a supervised learning machine used in both classification and regression problems. The purpose of SVM is to find the hyperplane capable of describing the behavior of a time series by fitting this function to the sample data while reducing the error between the hyperplane and the data. However, there is a large number of hyperplanes that could be adjusted. Therefore, SVM restricts this number using an ε-size *tube*, or also known as ε−tube, which is the dimension of the environment where the hyperplane is. This ε−tube, also called margin, reduces the space of possible hyperplanes. Moreover, it allows explaining the analyzed process using a subset of data called support vectors. The structure of the hyperplane is defined by the points lying on the margin, while all those contained within the ε−tube are irrelevant. SVM is formulated as an optimization problem that minimizes the training error and maximizes the margin.

When the data come from a non-linear function or are contaminated with noise, the SVM add in their formulation functions known as kernels, which have been used for classification and regression problems by different methods [8]. A kernel function allows to project data from its original space (input dimension) to another greater dimension space called space of characteristics. Using a kernel, we can map a nonlinear regression problem in the input space to a linear regression in the feature space.

While traditional neural networks implement the principle of empirical risk minimization (ERM), SVMs consider the principle of structural risk minimization (SRM) that seeks to minimize an upper limit in the dimension of Vapnik-Chervonenkis (VC), unlike ERM that minimizes the error in the data within the sample to be evaluated.

Based on the SRM principle, the SVM achieves a balance between error training and generalization error, which leads to better forecast performance than traditional neural networks. The selection of the best SVM model is equivalent to solving a quadratic programming problem, which gives another merit to the SVM of a single global solution. SVM was originally developed for classification problems (SVC) and then extended to regression problems (SVR).

The SVR is executed through nonlinear mapping of the input space in a high-dimensional characterized space and then runs the linear regression in the output space. Therefore, the linear regression in the output space corresponds to the non-linear regression in the low dimensional input space. As the name implies, the design of the SVR depends on the extraction of a training data subset that serves as support vectors and therefore represents a stable feature of the data.

For the financial time series, the objective is to find a function with the greatest ε deviation of the values obtained for all training data, and at the same time be as flat as possible. That means that errors do not take a great importance when they are smaller than the ε value; in other words, the forecasted values are within the hyperplane. Nevertheless, larger errors will not be accepted.

3.2 Fuzzy Controller

Fuzzy logic is an extension of traditional (Boolean) logic that uses concepts of group membership more similar to the way of human thinking. This is because humans think with varying degrees of truth that are diffuse. For example: *much, little, young, adult* [9]. A fuzzy set, compared to a *classical set*, has no discrete and clearly defined boundaries. In fuzzy logic, the truth of each statement becomes a degree of membership. Each state is assigned a degree of membership between 0 and 1.

The fuzzy inference system is a very popular computational structure based on the concepts of fuzzy logic theory, if-then rules, and methods of fuzzy inference. Fuzzy inference systems are applied to a wide variety of areas such as automatic control, data classification, decision analysis, expert systems, time series prediction, robotics, and pattern recognition. Because of their multidisciplinary nature, fuzzy inference systems are also known as expert systems, fuzzy models, fuzzy logic controllers or simply as fuzzy systems. The following are the specifications of each segment of type-1 fuzzy logic system diagrams (T1FLS) [10]:

- Fuzzifier: Crisp inputs are mapped into fuzzy sets that trigger the inference engine.
- Rule set: T1FLS rules consist of antecedents and consequents that are represented by a type-1 fuzzy set.
- Inference: Inference segment assigns fuzzy inputs to fuzzy outputs using the rule set and operators such as union and intersection.
- Defuzzification: The outputs of the type reduction segment are given to the Defuzzification segment. The reduced type sets are determined by their points at the left and right ends, the defuzzified value is calculated by the average of these points.

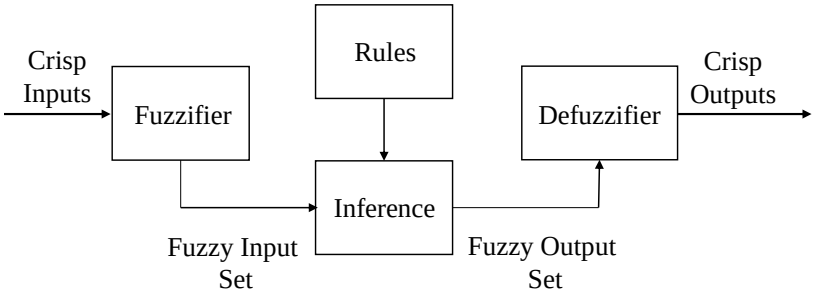


Figure 1. Fuzzy Inference

3.3 Genetic Algorithm

The original motivation for the GA approach was a biological analogy [11]. In the selective crossbreeding of plants or animals, for example, desired traits were sought in offspring, characteristics that are determined at a genetic level by the way the parent chromosomes are combined. A population of arrays is used, and these arrays are commonly referred to in the literature as chromosomes. The recombination of the arrays is generated using simple analogies of the cross and genetic mutation, and the search is guided by the results of evaluating the objective function f for each array in the population. In the case of genetic algorithms, a population based on this evaluation, the arrangements that have a greater aptitude (e.g., represented by better solutions) can be identified, and are given a greater opportunity of crossing.

Crossbreeding consists in replacing some of the genes in one parent with the corresponding genes in the other. An example of a one point crossbreeding is shown in Table 1. Given the parents $P1$ and $P2$, with the crossbreeding in the 4th gen, the offspring will be $O1$ and $O2$:

Table 1. Crossbreeding in genetic algorithm

Parents	Parent Genes	Offspring	Offspring Genes
$P1$	1 0 1 **0** 0 1 0	$O1$	1 0 1 **1** 0 0 1
$P2$	0 1 1 **1** 0 0 1	$O2$	0 1 1 **0** 0 1 0

The other common operator is mutation, in which a gene (or subset of genes) is chosen randomly, and the allele value of the chosen genes is changed. In the case of binary arrays, this means complementing the chosen bits. For example, the array $O1$, with the genes 3 and 5 mutated, would become 1 0 0 1 1 0 1, this is shown in Table 2.

Table 2. Mutation in genetic algorithm

Original Offspring	Original Gene	Mutated Offspring	Mutated Gene
$O1$	1 0 **1** 1 **0** 0 1	Mutated $O1$	1 0 **0** 1 **1** 0 1

A pseudocode of a genetic algorithm is shown in Algorithm 1.

Algorithm 1. Evolutive

```
1.  Function Evolutive()
2.    Population = generate_population()
3.    while (Condition_Stop ! = TRUE) do
4.      NextGenPopulation = Crossover()
5.      NextGenPopulation = Mutation()
6.      Population = Elite_Selection()
7.    end while
8.  end Function
```

3.4 ARIMA

One of the most popular and important forecasting models is called ARMA, which is the combination of the Auto Regressive model (AR) and Moving Average model (MA) [12].

The AR model is represented as follows:

$$Y_t = \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \cdots + \varphi_p Y_{t-p} + \varepsilon_t \quad \textbf{(1)}$$

where Y_t is an estimated variable in the period t in terms of the first data p in the time series; ε_t is the error of the model against the real data in period t, and all φ are determined by a simple regression model.

In the MA model, Y_t is estimated near the mean μ of the time series; this is done through the weighting of the ε_t errors in several q periods prior to period t:

$$Y_t = \mu_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \ldots - \theta_q \varepsilon_{t-q} \quad \textbf{(2)}$$

The ARMA model combines the equations above as follows:

$$Y_t = \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \cdots + \varphi_p Y_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \ldots - \theta_q \varepsilon_{t-q} \quad \textbf{(3)}$$

The ARMA model is the combination of the AR (Auto-Regressive) and MA (Moving Average) model, while the ARIMA model includes the stationary component to the ARMA model. When a time series is not stationary, it can be integrated by defining a new variable as follows:

$$Z_t = y_t - y_{t-1} \quad \textbf{(4)}$$

where $t = 2,3, \ldots, n$.
The second differences are determined by:

$$z_t = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) \quad \textbf{(5)}$$

for $t = 3,4, \ldots, n$.

The resulting model, integrating seasonality, is known as ARIMA (Auto-Regressive Integrated Moving Average) which is an ARMA model with a non-seasonal component. In order to integrate the seasonal component, ARMA is adapted and the SARMA model is generated. By combining the ARIMA and SARMA model, the SARIMA (Seasonal Auto-Regressive Integrated Moving Average) model is generated.

Processes delays and random disturbances can be represented by a periodic form in each seasonal pattern. Seasonal delays occur due to dependence on similar periods of successive years. Another model is able to represent the correct trend, seasonality and non-seasonal components of a time series. This model is known as the $ARIMA(p,d,q) \times SARIMA(P,D,Q)$, where:

- p: AR model order.
- d: differencing order of the regular and non-regular parts of a stationary series.
- q: MA model order.
- P: SAR model order.
- D: differencing order of the seasonal parts of the series.
- Q: SMA order.

4 Fuzzy GA-SVR

The SVR model developed in this paper receives a tuning of the parameter ε of the linear kernel through the fuzzy controller; also, the fuzzy controller receives a fuzzy rule set generated by an evolutionary algorithm to be able to tune the SVR.

The genetic algorithm (Algorithm 2) begins with the random generation of the population; each chromosome will have the size of the rule set followed by the rules (coded by numbers) and MAPE as we can see in Table 3. In this table (Ant1, Cons1), (Ant2, Cons2), …, (Antn,Consn) come from the fuzzy logic theory, meaning antecedent and consequent [9].

Table 3. Population

Size	Ant1	Cons1	Ant2	Cons2	Ant3	Cons3	⋯	MAPE
2	4	1	5	1	NA	NA	⋯	0.668
15	3	2	4	1	1	5	⋯	0.117
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋯	⋮

The crossover method uses a binary tournament which takes 4 individuals randomly, choosing the 2 best parents out of these 4 individuals, which generates 2 children through a cross point. For adding diversity, a 5% of chance of accepting a bad parent was applied. The cross point has to be $n - size(fuzzy\_set)$ from the lower size parent to maintain the diversity of sizes. This method

is shown in Table 4 and Table 5, and is executed $n/2$ times to obtain a new population of children with the size of the initial population.

Table 4. Parents binary tournament

	Size	Ant1	Cons1	Ant2	Cons2	Ant3	Cons3	⋯	MAPE
P1	2	4	1	5	1	NA	NA	⋯	0.668
P2	15	3	2	4	1	1	5	⋯	0.117
P3	12	4	4	5	1	4	1	⋯	0.479
P4	9	3	4	2	3	3	3	⋯	0.318

Table 5. Crossover

	Size	Ant1	Cons1	Ant2	Cons2	Ant3	Cons3	⋯	MAPE
S1	15	3	2	2	3	3	3	⋯	
S2	9	3	4	4	1	1	5	⋯	

The mutation method has a perturbation per bit with a 5% probability, in which each individual can increase or decrease the size of the rule set, and each rule can be disturbed. An example of this mutation can be seen between Table 3 and Table 6. Once the initial population and the population of children are evaluated, an elite selection is made, where 50% of the best individuals in each population are taken to generate the new population. This process will continue until there is no change in the best individual after n generations.

Table 6. Mutation

Size	Ant1	Cons1	Ant2	Cons2	Ant3	Cons3	⋯	MAPE
12	4	3	5	1	2	5	⋯	0.395
15	1	2	4	1	1	4	⋯	0.892
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋯	⋮

The fuzzy controller (Algorithm 3) implemented in this work consists of an input variable, the mean absolute percentage error (MAPE), and an output variable ε that affects the linear kernel of the SVM. Each variable consists of five Gaussian membership functions (*low*, *low-mid*, *mid*, *mid-high*, *high*) slightly overlapping each other. Rule sets can have all possible rules (from 1 to 25 rules). The rules are coded as follows: 1 for *low*, 2 for *low-mid*, 3 for *mid*, 4 for *mid-high* and 5 for *high*. The fuzzy controller evaluates the input variable with the rule set taken from each individual of the population generating a defuzzification for the output variable and tune the parameter ε of the SVR kernel.

Algorithm 2. GA-FUZZY-SVR

```
Function GA-FUZZY-SVR()
  generate_population()
  SVR_Fuzzy_eval(Population)
  Best_set = get_best_set (Population)
  while (stop_condition ! = TRUE) do
    Genetic_operators(Offspring)
     Offspring = SVR_Fuzzy_eval(Offspring)
    Population = Elite_Selection()
  end while
end Function
```

Algorithm 3. SVR_Fuzzy_eval

```
Function SVR_Fuzzy_eval(Population)
  for i = 1 to Population_size
    SD=Generate_fuzzy_system()
    for j = 1to stop_condition
      Evaluate_SVR()
      fuzzy_inference()
      defuzzify()
    end for
    Predict_SVR()
    Best_MAPE(Population[i])
  end for
end Function
```

5 GA-SVR

An alternative algorithm was generated trying to expand the possibilities of configuration of the Fuzzy GA-SVR (Algorithm 4), in order to improve the performance of said algorithm and to reduce the computational time. To this end, the genetic algorithm was redesigned, which, unlike the previous algorithm, does not configure the fuzzy system rulesets. This genetic algorithm directly configures all parameters of the SVR without using the fuzzy controller.

Each individual in the population of the GA-SVR algorithm has the following configurable parameters of the SVR of the library *e1071* from the R project:

- Kernel: Select the kernel with which the SVR will work.
  Linear: $u'v$
  Polinomial: $(\gamma u'v + coef0)^{degree}$
  Radial: $e^{(-\gamma|u-v|^2)}$
  Sigmoid: $\tan h(\gamma u'v + coef0)$
- Kernel type: Kernel types used for regression. The kernels used were eps-regression and nu-regression.
- Degree: Parameter needed for the polynomial kernel.
- γ: Parameter needed for all kernels except the linear one.
- Coef 0: Parameter needed for polynomial and sigmoid kernels.
- Cost: Value of the infringement of the restriction, it is the constant C of the regularization term of the Lagrange function.
- ν: Parameter needed for the nu-regression.
- ε: Value in the insensitive loss function.
- MAPE: Mean Absolute Percentage Error, it is the measure for the evaluation.

The crossover and mutation method were calculated similarly to those of the Fuzzy GA-SVR with the only exception when generations without improvements are $n-2$, the probability of 5% will rise to 25% to force the algorithm to do a larger exploration.

Algorithm 4. GA-SVR

```
1.  Function GA-SVR()
2.    Population = generate_population()
3.    Population = SVR_eval(Population)
4.    while (stop_condition != TRUE) do
5.      Genetic_Operators(Offspring)
6.      Offspring = SVR_eval(Offspring)
7.      Population = Elite_Selection()
8.    end while
9.  Prediction(Best_Set)
10. end Function
```

6 Implementation and Results

The implementation in this paper is based on the generation of a fuzzy controller with automatic generation of rule sets for tuning an SVR to predict time series of the Mexican Stock Exchange. This implementation was developed in R language using the libraries: *e1071*, *sets* and *tseries*.

The time series used in these experiments were taken from contests M1, M2, M3 and M5, and two time series of the Mexican Stock Exchange: the CARSO group and CEMEX. Four experiments were performed for the SVR algorithm to measure its performance, and three classic methods were used: AR, ARMA and ARIMA.

The ARIMA method was chosen because it is one of the classic methods with greater use for the prediction of financial time series; likewise, the AR and ARMA methods were evaluated to demonstrate that SVRs have the potential to outperform classic methods. Table 7 shows the Mean Absolute Percentage Error (MAPE) generated by each algorithm for the seven time series:

Table 7. MAPE results

Time Series	AR	ARMA	ARIMA	Standard SVR	Tuned SVR	Fuzzy GA-SVR	GA-SVR
Carso	3.61	4.81	3.61	6.84	6.11	3.56	3.76
Cemex	74.73	84.08	28.17	50.04	31.55	16.43	13.96
M1	3.55	Inf.	0.83	23.65	17.01	13.82	0.86
M2_PANTER	31.62	32.70	25.92	77.52	75.43	74.67	14.15
M2_REALGNP	4.94	Inf.	1.23	4.81	4.21	3.63	0.56
M3	34.36	30.17	29.03	24.60	24.98	34.72	27.81
M5	99.29	98.88	98.88	82.53	88.42	125.17	98.66

Table 8. Time results

Time Series	AR	ARMA	ARIMA	Standard SVR	Tuned SVR	Fuzzy GA-SVR	GA-SVR
Carso	0.27 sec	41 sec	1 min	3 sec	25 min	2 hr	36 min
Cemex	0.27 sec	41 sec	1 min	3 sec	12 min	4 hr	1 hr
M1	0.27 sec	41 sec	1 min	3 sec	2 min	23 min	5 min
M2_PANTER	0.27 sec	41 sec	1 min	3 sec	2 min	11 min	23 sec
M2_REALGNP	0.27 sec	41 sec	1 min	3 sec	2 min	15 min	41 sec
M3	0.27 sec	41 sec	1 min	3 sec	2 min	8 min	3 min
M5	0.27 sec	41 sec	1 min	3 sec	25 min	9 min	4 min

As we can see in Table 7, GA-SVR surpasses Fuzzy GA-SVR in 3 of 7 time series. We can also observe that the difference in performance between GA-SVR and Fuzzy GA-SVR is very minimal for the CARSO series with a difference of 0.2, and for the series M1 with a difference of 0.03 between GA-SVR and ARIMA. The standard SVR only obtained the best performance for the M3 and M5 time series, and even in this case the MAPE is very high because the M3 and M5 are full of Poisson jumps.

The GA-SVR obtained better performance than Fuzzy GA-SVR mainly because the Fuzzy GA-SVR only changes the ε parameter and only use the linear kernel unlike the GA-SVR, which changes every parameter of the SVR and also uses 4 different kernels.

It would be interesting to determine which of the two algorithms, GA-SVR and Fuzzy GA-SVR, has the best efficiency from a complexity point of view. In the case of genetic algorithms, a probabilistic analysis and the execution of the programs is required [13]. However, in the case of GA-SVR and Fuzzy GA-SVR, a mathematical analysis of the complexity is not possible, because the execution of both algorithms should be made until the equilibrium be reached, which is non-deterministic. For this reason, we measure the efficiency of these algorithms through an experimental process. To this end, the same instances of the problem were taken, but with series that goes from 50 to 1000 values; these instances actually form part of the same time series used in this paper.

Figures 2, 3 and 4 show the performance of the GA-SVR and Fuzzy GA-SVR algorithms for the best case, average case, and worst case, respectively. The sizes of the instances are shown on the x-axis while the execution times are shown on the y-axis; in addition, we add the trend of each algorithm. The performance of the GA-SVR algorithm for the three cases corresponds to the values that are located in the lowest part of these graphs, while the performance of Fuzzy GA-SVR is at the top of each figure. We can observe that in all the cases GA-SVR has the best efficiency.

Figure 2. Best performance

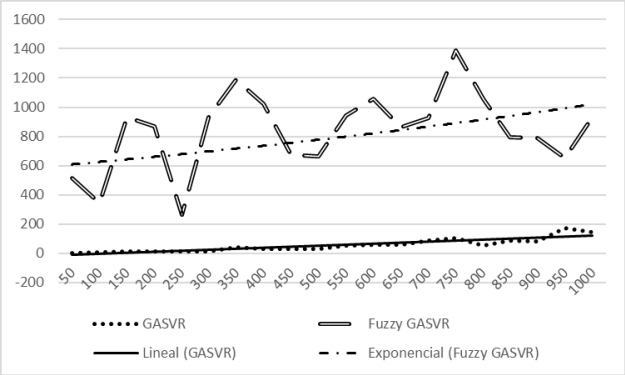


Figure 3. Mean performance

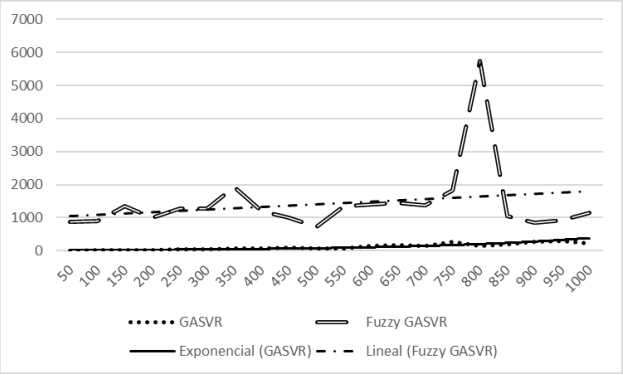
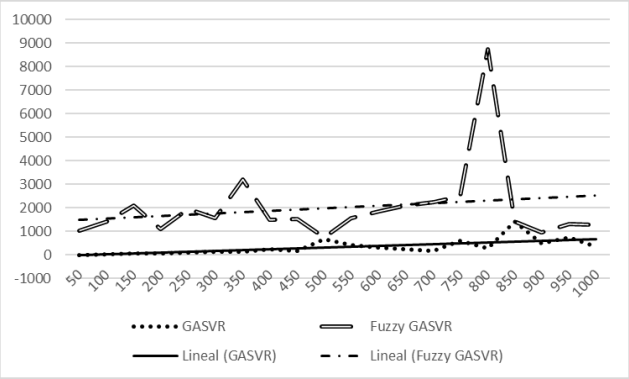


Figure 4. Worst performance



7 Conclusions

A fuzzy controller with an evolutionary generator of automatic rules to tune an SVR and thus predict financial series of the Mexican Stock Exchange was presented in this paper. Besides, an evolutionary algorithm was developed with an automatic configuration of all the parameters of the SVR.

The results generated by the two proposed forecasting algorithms show that Fuzzy GA-SVR and GA-SVR surpass the individual use of SVR, the tuned SVR method and the classic AR, ARMA and ARIMA methods. Finally, according with the experimentation, GA-SVR obtains even better results than Fuzzy GA-SVR.

We emphasize that Fuzzy GA-SVR worked using only the linear kernel which has only one parameter to be adjusted. In future research different kernels should be tested; in addition other heuristics instead of GA can be applied with a configuration similar to that of GA-SVR.

Acknowledgements

The authors would like to acknowledge the Consejo Nacional de Ciencia y Tecnología (CONACYT). Besides, they acknowledge the Laboratorio Nacional de Tecnologías de la Información (LaNTI) of the Instituto Tecnológico de Ciudad Madero for the access to the cluster. Also, Javier Alberto Rangel González thanks the scholarship 429340 received from CONACYT in his Ph.D.

8 References

[1] G. Santamaría-Bonfil, J. Frausto-Solís, and I. Vázquez-Rodarte, "Volatility forecasting using support vector regression and a hybrid genetic algorithm," *Comput. Econ.*, vol. 45, no. 1, pp. 111–133, 2015.

[2] G. Cornuejols and R. Tütüncü, *Optimization Methods in Finance*. Cambridge University Press, 2006.

[3] T. Bollerslev, "Generalized autoregressive conditional heteroskedasticity," *J. Econom.*, vol. 31, no. 3, pp. 307–327, 1986.

[4] I. Z. Batyrshin and L. B. Sheremetov, "Perception-based approach to time series data mining," *Appl. Soft Comput.*, vol. 8, no. 3, pp. 1211–1221, 2008.
[5] C. Chatfield, *Time-Series Forecasting*. Boca Raton, Florida: Chapman & Hall/CRC Press, 2002.
[6] X. Wang, K. Smith-Miles, and R. Hyndman, "Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series," *Neurocomputing*, vol. 72, no. 10–12, pp. 2581–2594, 2009.
[7] V. Vapnik, *Statistical Learning Theory*. New York: Wiley Publishing Inc, 1998.
[8] A. Scholkopf, B. and Smola, *Learning with Kernels*. Cambridge: MIT Press, 2002.
[9] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338–353, 1965.
[10] E. Kayacan, "Interval type-2 fuzzy logic systems: Theory and design," Bogazi{ç}i University, 2011.
[11] J. H. Holland, "Genetic algorithms," *Sci. Am.*, vol. 267, no. 1, pp. 66–72, 1992.
[12] J. Frausto-Solis, E. Pita, and J. Lagunas, "Short-term streamflow forecasting: ARIMA vs Neural Networks," in *American Conference on Applied Mathematics (MATH'08), Harvard, Massachusetts, USA*, 2008, pp. 402–407.
[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 3rd ed. MIT press Cambridge, 2009.