Jafar ABUKHAIT, Ayman M MANSOUR, Mohammad OBEIDAT

Tafila Technical University, Jordan

doi:10.15199/48.2018.05.03

Classification based on Gaussian-kernel Support Vector Machine with Adaptive Fuzzy Inference System

***Abstract**. In this paper, we propose a new classification approach which combines the advantages of both Gaussian-kernel Support Vector Machine and Adaptive Fuzzy Inference System. Instead of generating a large number of candidate rules as in fuzzy classification, the proposed method adopts the decision trees to generate rules directly from training data. Decision trees provide architecture to generate fuzzy IF–THEN rules from the training data where the fuzzy parameters of the rules would be optimized using Genetic Algorithm. The Gaussian-kernel SVM will be used in the classification phase using the parameters obtained from Particle Swarm Optimization. Experimental results of the proposed approach has proved significantly better accuracy than other state-of-the-art classification methods by testing it on benchmark UCI datasets.*

Streszczenie.** Zaproponowano nową metodę klasyfikacji łączącą zalety metod: Gaussian-Kernel Support Vector Machine i Adaptive Fuzzy Interference System. Wykorzystano drzewo decyzyjne do tworzenia zasad klasyfikacji bezpośrednio z danych treningowych. Parametry logiki rozmytej określano wykorxzystując algorytm genetyczny. A parametry SVM wykorzystując lagorytm mrówkowy. **Metoda klasyfikacji bazująca na SVM I adaptacyjnej logice rozmytej

Keywords: Support Vector Machines, Membership Functions, Fuzzy Inference System, Decision Trees, and Genetic Algorithm.
Słowa kluczowe: klasyfikacja danych, metoda SVM, logika rozmytas.

Introduction

Support Vector Machine (SVM) has been widely used as a technique for solving pattern classification and prediction problems. It can be viewed as an approximate implementation of what Vapnik has defined as Structure Risk Minimization (SRM), an inductive principle that aims to minimize the upper bound on the generalization error of a model, rather than minimizing the mean-square-error over the training data set.

In the last two decades, Fuzzy methodology has been successfully applied in a variety of areas including control and system identification, signal and image processing, pattern classification, and information retrieval [1, 2, 3, 4]. In general, building a fuzzy system consists of three basic steps: structure identification (variable selection, partitioning input and output spaces, specifying the number of fuzzy rules, and choosing a parametric/nonparametric form of membership functions); parameter estimation (obtaining unknown parameters in fuzzy rules via optimizing a given criterion); and model validation (performance evaluation and model simplification). The design methodology for the construction of fuzzy models involves both rule extraction and parameter learning aspects.

In spite of the popularity of both SVM and Fuzzy systems, there had been almost no work in the literature that relates both these methods. In this paper, we will focus on the rule extraction methods which have been formulated using neural networks, genetic algorithms, and a variety of clustering-based techniques in an effort to select only those rules that contribute to the inference consequence. We will investigate the connection between fuzzy rule-base systems and kernel machines. We then relate kernel function to fuzzy basis function and develop a new fuzzy rule-based inference system that fuses these two concepts thus; preserving the advantages of both these systems. The overall fuzzy inference system can be represented as series expansion of fuzzy basis function.

The key contribution of the proposed work is the development of a novel classification approach that uses Decision trees to extract features and then combines the advantages of both Gaussian-kernel SVM and fuzzy system in the classification process. Parameters of both fuzzy and SVM have been optimized using Genetic algorithm and Particle Swarm Optimization respectively. The performance comparison with state-of-the-art methods proposed in the literature is presented in the experimental part.

Mathematical Background

Decision Tree

Classification process involves four phases: data acquisition/gathering, data pre-processing, training/learning, and testing. The goal of data gathering phase is to obtain the training and test sets. The second phase aims to perform data cleaning, sampling, creating new records (attributes) and records selection of the experimental data. The relatively less correlated and redundant records in a given data set will be removed in this phase.

A decision tree partitions the input space of a data set into mutually exclusive regions by giving each region a label. The decision tree consists of a root and internal nodes and grows from a root node by determining the best split that partition the region at internal nodes into disjoint smaller subset and proceed down to the leaf node (terminal nodes) labelled as – present or not present. In order to perform the split, an error function is used that quantifies the performance of a node t in separating data from different classes. The used error function is named as impurity function. The best known impurity functions for splitting are entropy function and Gini index.

Using the impurity function φ, the impurity measure of a given node t as in equation 1.

$$(1) \quad E(t) = \varphi(p_1, p_2, \dots p_j) = -\sum_{j=1}^{J} p_j log_2 p_j$$

where p_j = P(j|t), the probability of class j at node t.

The entropy measures the homogeneity of a node. The maximum entropy value (log nc) results when all records belong to one class, implying most information while the minimum value of entropy(0) results when records are evenly distributed among all classes implying least information.

Similarly, the impurity measure of a tree T can be expressed as

$$(2) \quad E(T) = \sum_{t \in \Psi} \frac{n_t}{n} E(t)$$

where Ψ is the set of terminal nodes in the tree T, n_t equals number of records at child t, and n equals number of records at the terminal node.

The information gain is calculated as:

(3) $$GAINSplit = E(P) - E(T) = \boldsymbol{E(P)} - \sum_{i=1}^{k} \frac{\boldsymbol{n_i}}{\boldsymbol{n}} \boldsymbol{E(i)}$$

where Parent Node P (non leaf node, node with partition) is split into k partitions (children), ni is number of records in partition i, n = number of records at the terminal node.

Fuzzy Inference System

Fuzzy logic is a well-established methodology that is effective for systematic handling of deterministic uncertainty and subjective information [35]. Using Fuzzy rule based approach will enhance the classification performance. Fuzzy Inference System is experience-based as experience plays a key role in the design of it and it contains a set of fuzzy IF-THEN rules and a set of membership functions of fuzzy sets.

Firing strength ωi is generated with product method.

The last step in the fuzzy inference process is defuzzification. The decision task is performed by the Inference Engine that evaluates all the rules in the rule base and combines the weighted consequents of all relevant rules into a single output fuzzy set. That set is then defuzzified to produce a crisp similarity value. Fuzziness helps us to evaluate the rules, but the final output of a fuzzy system has to be a crisp number. There are several defuzzification methods, but probably the most popular one is the centroid technique. It finds the point where a vertical line would slice the aggregate set into two equal masses. Mathematically this center of gravity (COG) can be expressed as:

(4) $$COG = \frac{\int_a^b (\mu_\varsigma(x).x)dx}{\int_a^b \mu_\varsigma(x)dx}$$

Support vector machine (SVM)

Support vector machine (SVM) is a linear classifier that deploys statistical learning theory and kernel function for classification. SVM with sigmoid kernel function is similar to two-layer feed forward neural network response but SVM is more efficient than neural network in solving complex problems with large data set and high dimensionality and avoiding overfitting [5,6,7]. The concept of SVM stands on suggesting the optimal hyperplane that maximize the margin between the hyperplane itself and the closest vectors belonging to both classes. Fig. 1 shows an optimal hyperplane that separates the classes with maximum margin [7,8,9].

Linear Support Vector Machine

In a case where input space is:

$$\{x_1, x_2, x_3, \ldots\ldots, x_n\}$$

and output space is:

$$y \in \{-1,1\}$$

The hyperplane separating the two classes can be represented by:

(5) $$\vec{w}.\vec{x} + b = 0$$

where w (weight) is the orthogonal vector to the hyperplane determining its orientation and b (bias) is the distance from the origin to the hyperplane.

Any training sample should satisfy:

(6) $$\vec{w}.\vec{x} + b \geq 1 \text{ for } y = +1$$

(7) $$\vec{w}.\vec{x} + b \leq -1 \text{ for } y = -1$$

These two inequalities can be combined to get:

(8) $$y(\vec{w}.\vec{x} + b) - 1 \geq 0$$

The formulation of this problem would be:

$$\text{maximize} \quad \frac{2}{\|w\|} \quad \text{or minimize} \ \frac{1}{2}\|w\|^2$$

Such that

$$y(\vec{w}.\vec{x} + b) - 1 \geq 0$$

To solve this optimization problem, a Lagrange multiplier is suggested and the problem becomes:

$$minimize\ L_p \equiv \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{l} \alpha_i y_i (\vec{w}.\vec{x_i} + b) + \sum_{i=1}^{l} \alpha_i$$

Such that $\alpha_i \geq 0$ Where α is the Lagrange multiplier.

The result of solving SVM with Lagrange is a decision function in terms of Lagrange multipliers α and bias (b) for test input xt:

(9) $$y_t = \sum_{i=1}^{l} \alpha_i y_i < \vec{x_i}.\vec{x_t} > +b$$

If data is not linearly separable; Slack variables ξi can be added to allow mis-classification of difficult or noisy data points and the formulation would be:

$$minimize\ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi i$$

where C is a cost function.

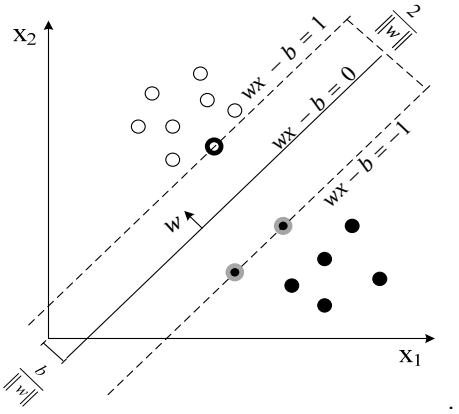

Fig.1. SVM optimal hyper-plane example

Nonlinear Support Vector Machine

In the case where the two classes are not linearly separable, a mapping function ϕ(x) can be used to map the input space into a higher dimension space where the classes can be separated linearly. This method is called kernel trick. The feature space is defined as the inner product of the mapping function:

$$k(x, \acute{x}) = \emptyset(x)^T.\emptyset(x)$$

and the decision function becomes as:

(10) $$y_t = \sum_{i=1}^{l} \alpha_i y_i k(x_i, x_t) + b$$

Kernel function can have different forms such as [5, 10]:

Polynomial: $k(x_i, x_j) =< x_i, x_j >^d$

Gaussian: $k(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$

Sigmoid: $k(x_i, x_j) = \tanh(\beta_0 x_i^T x_j + \beta_1)$

Particle Swarm Optimization (PSO)

It is a swarm Intelligence algorithm used to optimize a problem by exploring the best parameter within a search space that maximize or minimize a particular objective function. PSO is modelled by a population of particles that share information with each other and interact with their environment. Although there is no centralized control that governs how these particles interact, the local, and somehow random, interaction between these particles leads to global solution. PSO is most suited for problems which have a solution that can be represented by a point on a surface or n-dimensional space and has the advantage that it does not stuck at local minima or maxima [10, 11].

Particle swarm optimization can be used in pattern recognition problems to increase the classification rate by selecting the best classifier parameters. PSO starts by suggesting a population (swarm) of candidate solutions (particles) in the search space of the objective function. Each particle has a position, which consists of the candidate solution and its corresponding fitness function, and velocity. These particles move in the search space according to their own best known positions (best local positions) and the entire swarm best position (global best position) [12, 13]

Particles movement in the solution space is evaluated according to a fitness function in each iteration. These particles are then accelerated towards the particle with the best value for the fitness function. The advantage of using this approach over the other algorithms is that the large number of particles moving in the solution space prevents the algorithm from being trapped in local minima or maxima.

PSO consists of three steps which are repeated until a termination condition is met [11]:

1. Evaluate fitness function of each particle in the swarm.
2. Determine local and global best fitness functions.
3. Update the position and velocity of each particle.

The update of velocity and position of each particle is governed by the following two equations:

$$v_{k+1}^i = w v_k^i + c_1 r_1 \left(p_k^i - x_k^i\right) + c_2 r_2 (p_k^g - x_k^i) \quad (11)$$

$$x_{k+1}^i = x_k^i + v_{k+1}^i$$

where:

x_k^i : Particle position.

v_k^i : Particle velocity.

p_k^i : Best remembered individual particle position.

p_k^g : Best remembered swarm position.

c_1, c_2 : Cognitive and social parameters.

r_1, r_2 : Random numbers between 0 and 1.

w : The inertia weight.

Genetic Algorithm (GA)

Genetic Algorithm (GA) is one of the most popular derivative free optimization techniques which is based on the principles of evolution and natural genetics [13]. The GA starts by encoding each point in the parameter space into concatenated binary strings in which each concatenated value composed of a set binary bits using binary coding techniques. Different binary encoding techniques are available such that Excess-3 code, BCB and gray code. The resulted binary bits is called chromosome. A set of chromosomes in the solution space is called population. Chromosome consists of a set genes that contains information about the key parameters in a candidate solution. Genetic algorithms make multiple way search by creating a population of candidate solutions instead of just test one single solution. A starts by constructing a new population using genetic operation such as crossover and mutation through an iterative process until some convergence criteria are met.

The resulted new population will be decoded back to its original format. The process is: 1- Evaluation: Sort the population based on chromosomes scores (fitness).2- Selection: Choose the best chromosomes to generate the next population (natural selection). 75% of the sorted population will be kept in the new population. 3- Crossover: Merge the chromosomes by mixing their genes. Repeat the crossover operation until the new population is fully generated. 4- Mutation: change some chromosomes arbitrarily. Usually, around 1% of the crossover chromosomes will go through the mutation process. Mutation process prevents any single bit from converging to a value throughout the entire population

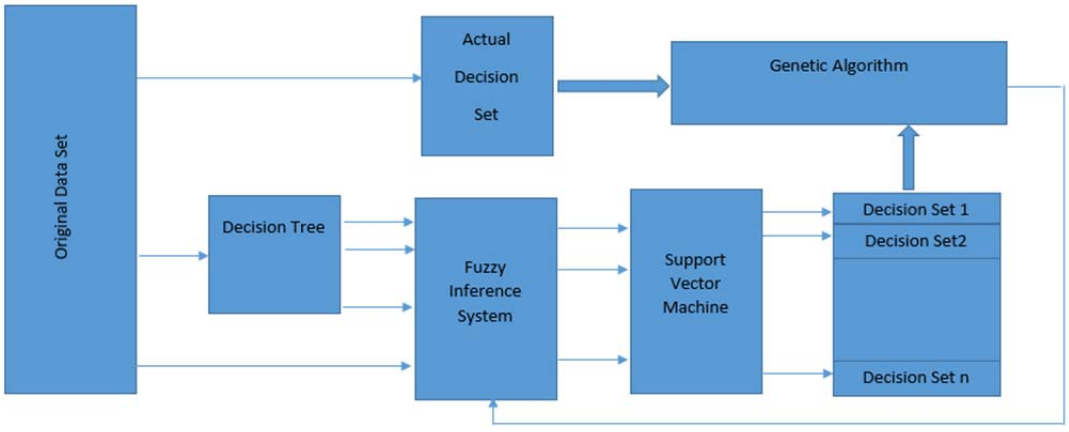

Fig.2. The developed system methodology (Training stage)

A new generation is created by repeating the selection, recombination and mutation processes until all chromosomes in the new population replace the initial population.

The Proposed system Methodology

In the learning phase, the target is to build a model. Testing Phase is used to determine the accuracy of the model. Usually, the given data set is divided into training

and test sets, with training set used to build the model and test set used to validate it.

Training Phase

A new novel system utilized the Fuzzy Inference System and the SVM is being proposed and developed. This preserves the advantages of both systems. Decisions tree will extract the rules that will be used for FIS and no need to get the rules from domain. This combination will help us in many problems specially the problem that is logistically described. The proposed system, as shown in Fig.2, can learn to make human-like decisions and uses fuzzy membership functions for the soft constraints (input variables).

Rule Development using Decision Tree

The use of fuzzy membership functions is convenient, because they express their decisions in terms of linguistic descriptions for the input constraints. Some decisions are highly correlated and virtually non-deterministic. The proposed adaptive fuzzy inference system uses a given input/output data set, and constructs the rules of fuzzy inference system (FIS) by Decision Tree. The algorithm for DT induction is shown in Fig.4.

Input: A given dataset described by m attributes and n examples, and the predefined fuzzy data base;

1 for $a = 1$ to m do
2 for $b = 1$ to n do
3 if Attribute Att_a is continuous then
4 for $x = 1$ to the total number of linguistic values defining Att_a do
5 *calculate $A_{Att_{a_x}}$, as the membership degree of the input value of attribute Att_a, example b, in the fuzzy set defining the x^{th} linguistic value of attribute Att_a;*
6 *Replace the continuous value of attribute Att_a, example b, with the linguistic value with highest membership degree with it;*

Fig.3. Fuzzification algorithm [14]

Initialization
Partition the training set space into {Iris-Setosa Iris-Versicolor, Iris-Virginica}
Calculate the total Entropy of the system
Calculate the $Gain_{Split}$ for each one of the input attributes
Choose the attribute with the highest information gain as the root node
Expand the Tree starting from the previously selected node
Repeat until all records are classified
Eliminate the previous selected node from the list of attributes
Partition attribute values of the remaining attributes in the space into {Iris-Setosa Iris-Versicolor, Iris-Virginica}
Calculate the Entropy of the attributes
Calculate the $Gain_{Split}$ of the new list of attributes
Choose the attribute with the highest information gain as the next node
End by return Tree

Fig.4. The Decision Tree induction algorithm

The decision tree is used as indirect method to extract fuzzy inference system rules. Classifying input records based on decision tree is by using a collection of "if x then y" rules where x is a conjunctions of attributes and y is the class label. The continuous attributes of the training set will be replaced by the linguistic terms of the fuzzy sets. Fuzzyfication of the continuous values of the training set and the number of fuzzy sets defining each attributes is estimated using Algorithm proposed in [14]. Fig.3 shows fuzzification Algorithm.

Feature Extraction using Fuzzy Inference System

Fuzzy inference is normally adapted in systems whose rule structure is essentially predetermined by the user's interpretation of the characteristics of the variables in the model. However, in some modelling situations, it cannot be distinguish what the membership functions should look like simply from looking at data. Rather than choosing the parameters associated with a given membership function arbitrarily, these parameters could be chosen so as to tailor the membership functions to the input/output data in order to account for these types of variations in the data values.

Fig.6 shows the architecture of the AFIS, comprising by input, fuzzification, inference and defuzzification layers. For simplicity, it is assumed that the fuzzy inference system under consideration has two inputs x and y and one output z as shown in Figure (1).

AFIS architecture consists of five layers; output of each layer is the following.

Layer 1: the input layer which is the fuzzification layer.

The input layer consists of n input. Each input is fazzified by three membership function. The output of this layer is denoted by

(12) $y_{1,i}^{j} = \mu_{i}^{j}(x_i)\,; i = 1,2,3$ and j=1 to n

For example as shown in Fig. 6:

$$\mu_{i}^{1}(x_i) = \mu_{Ai}(x); i = 1,2,3$$

$$\mu_{i}^{n}(x_i) = \mu_{Bi}(x); i = 1,2,3$$

and j is the total number of membership functions in the input layer,

$$j = 3 * n$$

the membership function is the generalized bell function.

(13) $$\mu_{A,i}(x) = \frac{1}{1 + \left|\frac{a_i - x}{b_i}\right|^{2c_i}}\,; c_i > 0$$

Where ai, bi, ci are premise parameters. The type of membership functions (MF) of the inputs are generalized bell functions, each MF has 3 nonlinear parameters. If the consequent MF is trapezoidal membership function, then each MF has 4 nonlinear parameters to be adjusted. In this paper the used membership function is the generalized bell function.

Layer 2: inference layer or rule layer

$y_{2,i} = \mu_i^R(x); i = 1,2,..,m$; where m is the number of rules

For example as shown in Fig.5.

(14) $$\mu_i^R(x) = \prod_{j=1}^{n} \mu_i^j(x)\,; i = 1,2,3$$

(15) $$\mu_1^R(x) = \mu_1^1(x) * \mu_1^1(x) = \mu_{A1}(x) * \mu_{B1}(x)$$

Layer 3: implication layer

(16) $y_{3,i} = y_{2,i} o C_i; i = 1,2,\dots,q$

Where q is the number of membership function μ of output

Implication operator is product.

Layer 4: aggregation layer

(17) $y_4 = \sum_{i=1}^{q} y_{2,i} o C_i$

Aggregate operator is sum. The consequent parameters are determined by Ci. If the consequent MF is trapezoidal membership function, each MF has 4 nonlinear parameters to be adjusted.

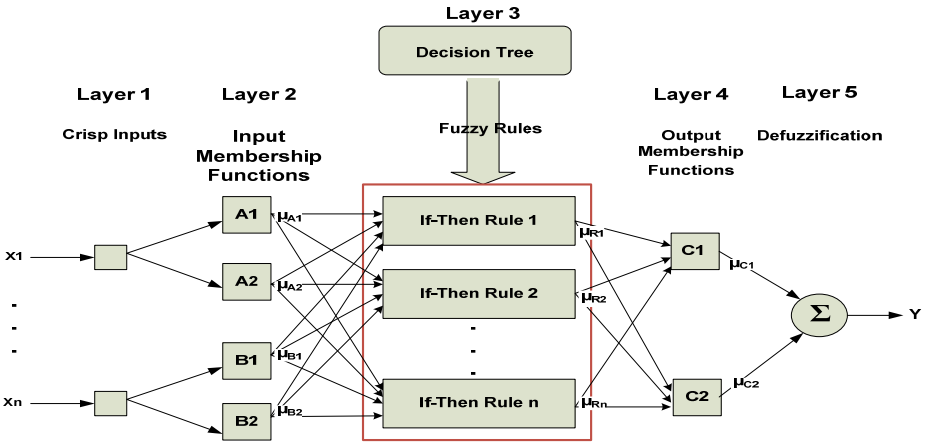

Fig.5. Decision Tree -based fuzzy basis function inference system architecture

Layer 5: defuzzification layer

(18) $$y_5 = \frac{\int \mu_{y5}(x) x \, dx}{\int \mu_{y5}(x) \, dx}$$

The crisp output f is achieved with the defuzzification method, center of gravity (COG).

The resulted membership function parameters are tuned (adjusted) using SVM and Genetic Algorithm. The system will learn the fuzzy rules that will be used in proposed system for classification purpose. With this combination, the system will have the benefits from all systems.

Classification Model Construction using SVM

This module constructs the classification model based on the features extracted from the fuzzy inference system. The output of this module which is the predicted class labels would be used along with the actual class labels to optimize the fuzzy weights by the genetic algorithm in the next module. Gaussian-kernel SVM is deployed on one-versus-all fashion to build up a multi-class classification model. The Gaussian function is defined as:

(19) $$k(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$$

Parameter values of the Gaussian-kernel SVM (σ, C) were selected using Particle swarm optimization (PSO) where each particle is characterized in the parameter space by a 2-dimensional vector x= [σ, C]. Each particle is initialized randomly with C and σ; then the fitness function which is the classification rate is calculated by training SVM after partitioning the training as shown in Fig.6. The fitness function of the best particle would be used to update particles' positions and velocities and the operation will repeat until the termination condition is satisfied [15,16].

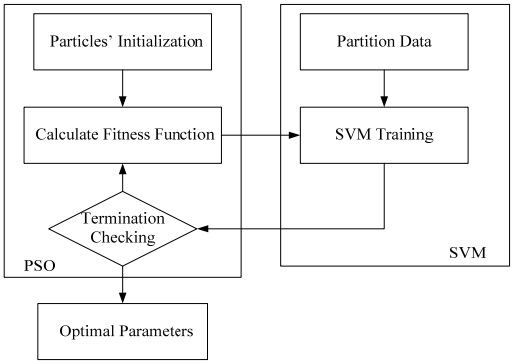

Fig.6. PSO-SVM model

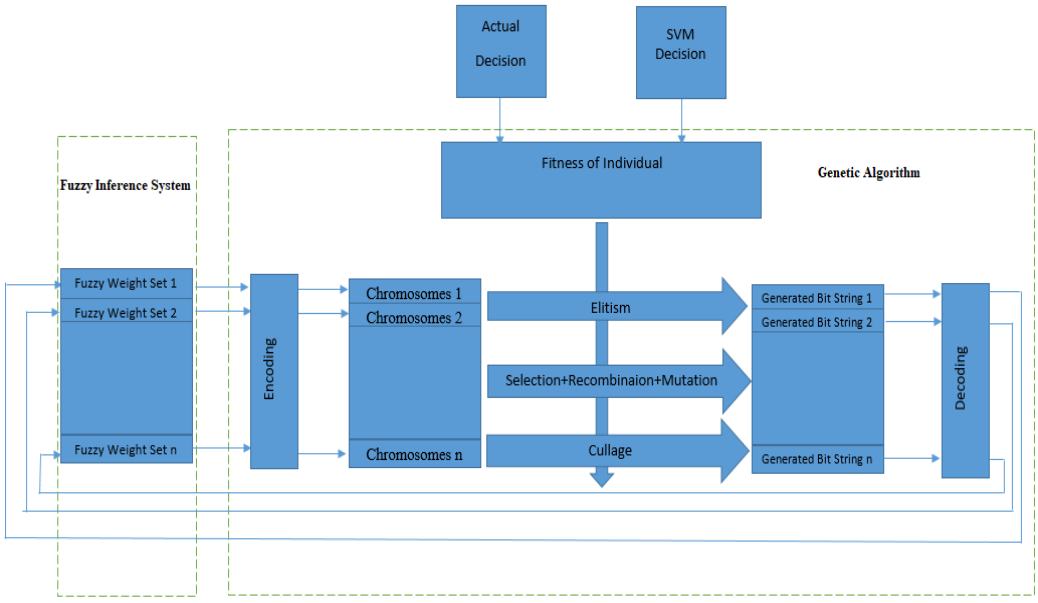

Fig.7. Genetics Algorithm model

Fuzzy Weights Optimization using Genetic Algorithm

The main step in GA is to calculate the fitness value of each member in the population as in Fig.7. The fitness value fi of the i[th] weight parameter is the objective function evaluated at this weight set. The fitness function is chosen to be the root mean squared differences between the correct decision specified by physician T and the decision given by the decision tree $\hat{T}$. The Root Mean Squared Error (RMSE) is given by equation (20). Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the actual class labelled by physician,

$$(20) \quad Objective\ Function = RMSE = \sqrt{\frac{\sum_{i=1}^{n}(T-\hat{T})^2}{n}}$$

where n is number of records of training data.

By this definition, then, the lower the fitness, the better the developed model and a fitness of zero means that the model achieves the desired behaviour for all inputs [17, 18]. As long as the fitness measure ranks the individuals accurately based on their performance, the exact form of the fitness is irrelevant to the working of the algorithm.

Fig.8 shows the algorithm used to determine the optimal solution.

```
Initialization
  Set Parameters and choose encode method
  Create Initial Population
  Repeat until iteration number= max number of iterations and Fitness=Max Fitness
    For each chromosome, Evaluate Fitness
      Update Fuzzy Inference System with new weights
      Create Decision Tree
      Train Decision Tree
      Test Decision Tree
      Fitness = Classification Accuracy
    End
  Parent Select
  Crossover
  Mutation
  Elitism
  Decode individual with maximum fitness
  End by return best solution: Output Fittest Chromosome
```

Fig.8. GA-based decision tree Optimization algorithm.

Testing Phase

The testing phase consists of two stages which are feature extraction using fuzzy inference system (FIS) and classification using SVM. These stages are shown in Fig.9.

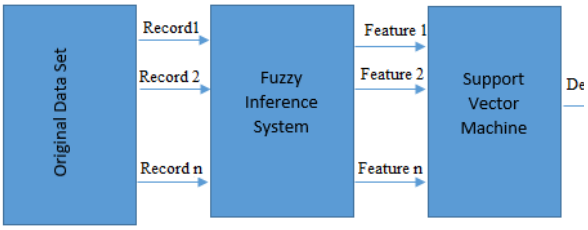

Fig.9. The developed system methodology (Testing stage)

Feature Extraction using FIS

FBF provide the mathematical formula that defines the mapping function for a FIS. Their expansion for the FIS mapping function is

$$(21) \quad y = f(x) = \frac{\sum_{j=1}^{M} y'_j \prod_{i=1}^{n} \mu_i^j(x_i)}{\sum_{j=1}^{M} \prod_{i=1}^{n} \mu_i^j(x_i)}$$

Where M denotes the number of rules, y'_j represents the center of gravity of the output fuzzy set that is associated with the rule R_j, n is the dimension of vector x, and $\mu_i^j(x_i)$ represents input membership functions. This expansion is valid only when we choose singleton fuzzification functions, product inference, maximum-product composition, and height defuzzification [8]. Equation (22) can be rewritten as

$$(22) \quad y = f(x) = \sum_{j=1}^{M} y'_j \Phi_j(x)$$

where the $\Phi_i(x)$ are called FBFs and are given by

$$(23) \quad \Phi_j(x) = \frac{\prod_{i=1}^{n} \mu_i^j(x_i)}{\sum_{j=1}^{M} \prod_{i=1}^{n} \mu_i^j(x_i)}$$

Equation (23) is valid for singleton fuzzification. The representation in equation is referred to as the fuzzy basis function expansion. It can be seen that the FBF expansion is essentially a sum over M rules, each of which generates an FBF.

Classification using SVM

This module is responsible on categorizing input instances to their classes based on the model constructed in the training phase. The input instances are represented by the extracted features from the FIS using the fuzzy weights obtained from module 4 in the training phase. A Gaussian-kernel function is used to map input features to higher space and the decision function becomes:

$$(24) \quad y(x_t) = sgn\left(\sum_{i=1}^{N_s} \alpha_i y_i \emptyset(s_i).\emptyset(x_t) + b\right) = sgn(\sum_{i=1}^{N_s} \alpha_i y_i k(s_i, x_t) + b)$$

where si are the support vectors, Ns is the number of support vectors, αi are the Lagrangian multipliers, b is the bias, k is the kernel function, yi is the class label, and xt is the test data.

Experimental Results

In this section, we present some experimental facts regarding the proposed SVM-AFIS classification methodology and compare it with other state-of-the-art classification algorithms in terms of accuracy. The proposed SVM-AFIS classification methodology has been deployed on eight datasets from UCI ML repository [19]. The eight datasets and their description with number of instances, attributes and classes are shown below in the Table 1. These data sets include a wide range of domains and a variety of data characteristics such as number of classes, instances, and attributes. The prediction accuracy has been measured by applying a 10-fold cross-validation where each dataset is randomly partitioned into 10 approximately equally sized subsets (or folds or tests). The induction algorithm is executed 10 times; in each time it is trained on the data that is outside one of the subsets and the generated classifier is tested on that subset. The

estimated accuracy for each cross-validation fold is a random variable that depends on the random partitioning of the data. So, for each dataset, we repeated 10- fold cross-validation 10 times. The estimated accuracy is the average over the ten 10- fold cross-validations.

Table 1. Comparison of C4.5,Naïve Bayes, Neural ,CBA, CMAR and SVM-AFIS on accuracy

Data Set	Instances	Attrib	Classes
heart	270	14	2
horse	368	22	2
Tic-tac	958	10	2
glass	214	10	7
zoo	101	18	7
german	1000	20	2
pima	768	8	2
iris	150	5	3

The proposed SVM-AFIS classification methodology has many stages as described earlier in section 3. The results of each stage would be addressed separately in the following sub-sections:

Rule Development Results

The generated decision tree of IRIS Data set as an example is shown in Fig.10.

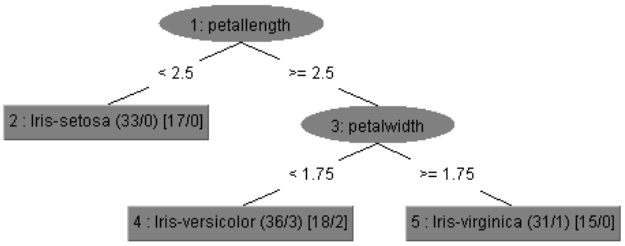

Fig.10. IRIS dataset decision tree

Feature Extraction Results

The extracted Rule from decision tree of IRIS data set is the following:

If Petal Length is Low Iris-Setosa is High

If Petal Length is High and Petal Width is Low Iris-Versicolor is High

If Petal Width is High and Petal Width is High Iris-Virginica is High

Fig.11 shows the Surface Analysis between Petal Width and Peta Length for Iris Setosa.

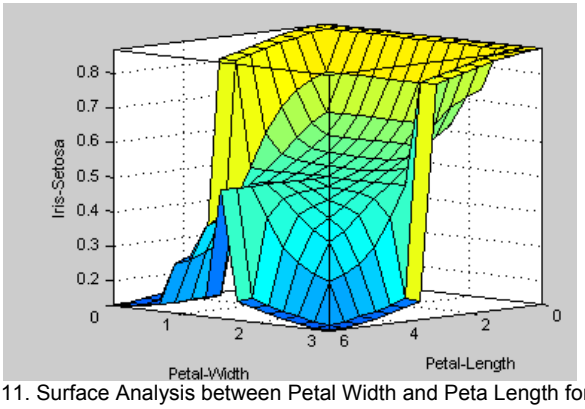

Fig.11. Surface Analysis between Petal Width and Peta Length for Iris Setosa.

Both Petal Length and Petal Width are fuzzy variables and their fuzzy values (Low, Medium, and High) are represented by triangular membership functions. Fig.12 and Fig.13 show the fuzzy sets for time Petal Length and Petal Width.

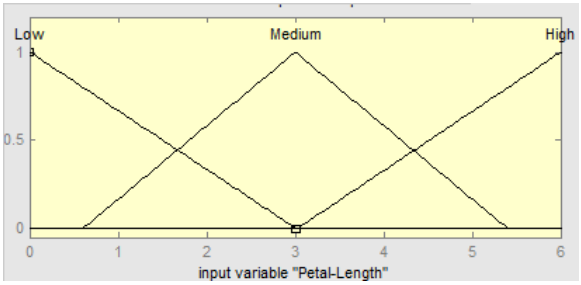

Fig.12. Membership functions for Petal Length

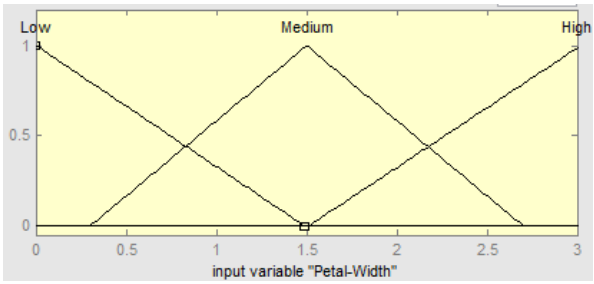

Fig.13. Membership functions for Petal Width

The corresponding membership functions of the output variables are shown in Fig. 14, Fig.15, and Fig.16 respectively. For each linguistic variable a combination of membership functions are defined. Sufficient overlapping degree between the membership functions is ensured to obtain a smooth output relationship.

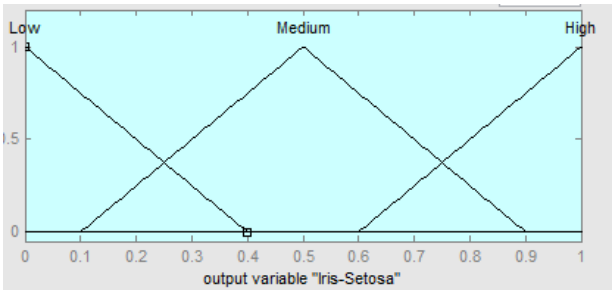

Fig.14. Membership functions for Iris Setosa

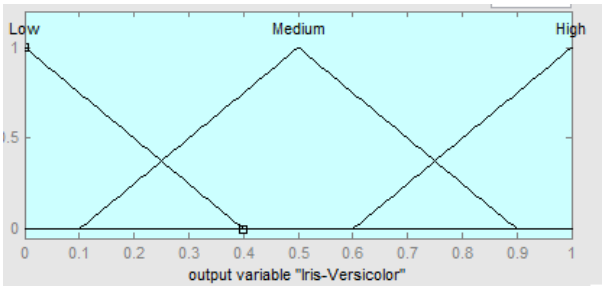

Fig.15. Membership functions for Iris Versicolor

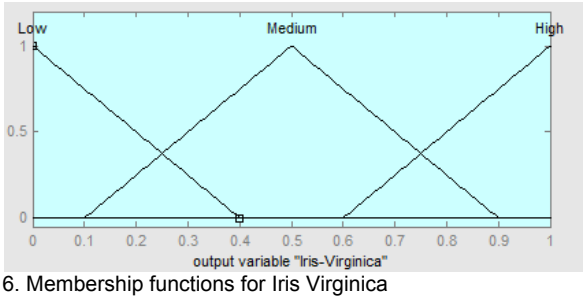

Fig.16. Membership functions for Iris Virginica

Classification Model Construction Results

The classification model has been constructed separately for each data set using. Five different Gaussian-kernels SVM classifiers have been deployed in a one-versus-one fashion to classify the data sets that have two classes which are: Heart, Horse, Tic-tac, German, and Pima data sets. In addition, three different Gaussian-kernels SVM classifiers have been deployed in a one-versus-all fashion to classify the data sets that have more than two classes which are: Glass, Zoo, and Iris data sets.

Parameter values of Gaussian-kernel SVM (σ, C) were selected using PSO where each particle is characterized in the parameter space by a 2-dimensional vector x= [σ, C]. Each particle is initialized randomly with C and σ; then the fitness function which is the classification rate is calculated by training SVM after partitioning the training data to 10 partitions. The fitness function of the best particle would be used to update particles' positions and velocities and the operation will repeat until the termination condition is satisfied. Table 2 shows the values of both σ and C for each classifier using PSO.

Table 2. SVM parameter values using PSO

Data Set	SVM Parameters	
	σ	C
Heart	4.01	9.83
Horse	.34	15.38
Tic-tac	2.45	28.23
Glass	1.2	19.14
Zoo	.65	21.66
German	.28	6.58
Pima	2.64	12.92
Iris	.1	1

Fuzzy Weights Optimization Results

In all the experiments the GA operates with the configuration shown in Table 3.

Table 3. GA setting

Parameter	Value
Population Size	20
Variable Range	[1,J], j is number of attributes
Maximum Generation	200
Crossover Points	2 points
Crossover Probability	0.75
Mutation Probability	0.005
Elitism	yes
Selection Method	Uniform selection

If the best-fitting instances are selected with genetic algorithm, this might lead to over-fitting. So, in order to make sure that there is no over fitting, 1/4 of the data was held (37 cases out of 150 cases) and will be used in testing the proposed algorithm

Table 4 shows the Classification model evaluation results after optimization. The proposed DT model shows high accuracy on IRIS data set (up to 98.17%) on the test data. To further validate the results, K-fold cross validation was used. In K-fold the training set will be randomly splitted into K that have approximately the same size. Then the Decision Tree will be trained using (K-2) subsets. One of the two remaining subsets will be used for validation and the last for testing. This process will be repeated K times, while a different subset is used for testing and validation.

Table 4. Overall performance results (training and validation set)

Correctly Classified Instances	98.5714 %
Incorrectly Classified Instances	1.4286 %
Kappa statistic	0.9703
Mean absolute error	0.0273
Root mean squared error	0.1199
Relative absolute error	5.6443%
Root relative squared error	24.3688 %

Using the held 37 cases not previously used in the training or cross validation. The achieved results are shown in Table 5.

Table 5. Overall performance results (testing set)

Correctly Classified Instances	68 %
Incorrectly Classified Instances	2.8571 %
Kappa statistic	0.9405
Mean absolute error	0.0359
Root mean squared error	0.1572
Relative absolute error	7.4114 %
Root relative squared error	31.9186 %

The final performance of decision tree has been improved compared with initial performance of decision tree shown in Table 6.

Table 6. Initial decision tree performance results

Correctly Classified Instances	59.0476 %
Incorrectly Classified Instances	40.9523 %
Kappa statistic	0
Mean absolute error	0.4135
Root mean squared error	0.643
Relative absolute error	90.5263%
Root relative squared error	123.9239 %

Table 7. Comparison of C4.5,Naïve Bayes, Neural ,CBA, CMAR and SVM-AFIS on accuracy

Data Set	SVM-AFIS	C4.5	Naïve Bayes	Neural	CBA	CMAR	SVM
heart	82.1	80.8	84.07	82.7	81.9	82.2	83.7
horse	83.8	82.6	78.8	65.3	82.1	82.6	82.61
Tic-tac	99.7	99.4	70.04	92.6	99.6	99.2	98.33
glass	73.2	68.7	48.59	62.1	73.9	70.1	59.81
zoo	97.2	92.2	93.2	93.07	96.8	97.1	96.04
german	74.92	72.3	74.7	73.4	73.4	74.9	74.7
pima	79.3	75.5	76.04	75.8	72.9	75.1	76.95
iris	98.5	95.3	95.33	92.9	94.7	94.0	96.0

Classification Accuracy Results

The average accuracy of the proposed SVM-AFIS approach has been compared with Naïve bayes, Neural, CBA, CMAR and C4.5 on eight different data sets. The proposed SVM-AFIS has shown excellent accuracy results as shown in Table 7.

Conclusions

This paper investigates the relationship between SVM and fuzzy rule-based systems establishing a link between kernels and fuzzy basis functions. The Decision Tree provides architecture to extract support vectors for generating fuzzy IF–THEN rules from the training data, and a method to describe the fuzzy system in terms of indirect

rule based. The main advantage of the proposed framework is that the model does not have to determine the number of rules in advance, and the overall fuzzy inference system can be represented as series expansion of fuzzy basis functions. Fuzzy rules are extracted directly from the given training data. The performance of the proposed algorithm is demonstrated using classification problems. Classification rates achieved in the experiments confirm that the developed systems perform better than the state-of-the-art techniques proposed in the literature.

***Authors**: Dr. Jafar Abukhait, Department of Communication, Electronics and Computer Engineering, Faculty of Engineering, Tafila Technical University, Tafila 66110, Jordan, E-mail: jafar@ttu.edu.jo; Dr. Ayman M. Mansour, Department of Communication, Electronics and Computer Engineering, Faculty of Engineering, Tafila Technical University, Tafila 66110, Jordan, E-mail: mansour@ttu.edu.jo; Dr. Mohammad A Obeidat, Department of Electrical Power and Mecatronics Engineering, Faculty of Engineering, Tafila Technical University, Tafila 66110, Jordan, E-mail: maobaidat76@ttu.edu.jo.*

REFERENCES

[1] Zadeh L. A., Fuzzy Sets, Information and Control, 8 (1965), No. 3, 338-353.

[2] Liu G., Chen J., Zhong J., An integrated SVM and fuzzy AHP approach for selecting third party logistics providers, Przegląd Elektrotechniczny , 88 (2012), No. 18, 5-8.

[3] Yang L., Mingzi X., Jin Z., A Novel Adaptive Fuzzy Controller Approach of Brushless DC Motors without Hall and Position Sensors, Przegląd Elektrotechniczny, 88 (2012), No. 12a, 290-294.

[4] Cao H., Wang Y., Jia L., Adaptive Neuro-Fuzzy Inference System-Based Pulverizing Capability Model for Running Time Assessment of Ball Mill Pulverizing System, Przegląd Elektrotechniczny , 89 (2013), No. 5, 122-127.

[5] Scholkopf B., Smola A. J., Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. Cambridge, MA, USA: MIT Press, 2001.

[6] Karatzoglou A., Meyer D., Hornik K., Support Vector Machines in R, Journal of Statistical Software, 15(2006), No. 9, 1-28.

[7] Burges C. J. C., A Tutorial on Support Vector Machines for Pattern Recognition, Data Min. Knowl. Discov., 2 (1998), No. 2, 121-167.

[8] Christianini N, Shawe-Taylor J., An introduction to support Vector Machines: and other kernel-based learning methods, Newyork 1999.

[9] Vapnik V. N., The Nature of Statistical Learning Theory, USA: Springer-Verlag, 1995.

[10] N. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines: And Other Kernel-Based Learning Methods. New York, NY, USA: Cambridge University Press, 2000.

[11] Kennedy J., Eberhart R., Particle swarm optimization, IEEE International Conference on Neural Networks, 4 (1995), 1942-1948.

[12] Sheng Ding and Shunxin Li, PSO parameters optimization based support vector machines for hyperspectral classification, 2009 1st International Conference on Information Science and Engineering (ICISE), Nanjing, 2009, 4066-4069.

[13] Eberhart R. C., Shi Y., Kennedy J., Swarm Intelligence (the Morgan Kaufmann Series in Evolutionary Computation). Morgan Kaufmann, 2001.

[14] Cintra, M. E., Monard, M. C., Camargo, H. A., A Fuzzy decision tree algorithm based on C4.5, Mathware & Soft Computing Magazine. 20 (2013), No. 1, 56-62.

[15] Ardjani F., Sadouni K., Benyettou M., Optimization of SVM MultiClass by particle swarm (PSO-SVM), 2010 2nd International Workshop on Database Technology and Applications (DBTA), Wuhan, 2010, 1-4.

[16] De Souza B. F., De Carvalho A. C. P. L. F., Calvo R., Ishii R. P., Multiclass SVM model selection using particle swarm optimization, *2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06)*, Rio de Janeiro, Brazil, 2006, 31-36.

[17] JAJCZYK J., Optimisation using a parallelised genetic algorithm on a personal computer, Przegląd Elektrotechniczny , 91 (2015), No. 7, 36-38.

[18] Hekim M., ANN-based classification of EEG signals using the average power based on rectangle approximation window, Przegląd Elektrotechniczny , 88 (2012), No.18, 210-215.

[19] Blake C., Merz C., UCI repository of machine learning databases.