



Constraint learning using adaptive neural-fuzzy inference system

Constraint
learning using
ANFIS

Hadi Sadoghi Yazdi and Reza Pourreza

*Department of Computer Engineering, Ferdowsi University of Mashhad,
Mashhad, Iran, and*

Mehri Sadoghi Yazdi

*Department of Electrical and Computer Engineering,
Shahid Beheshti University of Tehran, Tehran, Iran*

257

Received 2 November 2008
Revised 21 March 2009
Accepted 17 April 2009

Abstract

Purpose – The purpose of this paper is to present a new method for solving parametric programming problems; a new scheme of constraints fuzzification. In the proposed approach, constraints are learned based on deductive learning.

Design/methodology/approach – Adaptive neural-fuzzy inference system (ANFIS) is used for constraint learning by generating input and output membership functions and suitable fuzzy rules.

Findings – The experimental results show the ability of the proposed approach to model the set of constraints and solve parametric programming. Some notes in the proposed method are clustering of similar constraints, constraints generalization and converting crisp set of constraints to a trained system with fuzzy output. Finally, this idea for modeling of constraint in the support vector machine (SVM) classifier is used and shows that this approach can obtain a soft margin in the SVM.

Originality/value – Properties of the new scheme such as global view of constraints, constraints generalization, clustering of similar constraints, creation of real fuzzy constraints, study of constraint strength and increasing the degree of importance to constraints are different aspects of the proposed method.

Keywords Constraint handling, Learning, Fuzzy logic, Parametric measures, Programming

Paper type Research paper

1. Introduction

In this paper, we present a new method for solving parametric programming problems; a new scheme is proposed for fuzzification of constraints. In the proposed approach, constraints are learned based on deductive learning, adaptive neural-fuzzy inference system (ANFIS) is used for constraint learning by generating input and output membership functions (MFs) and suitable fuzzy rules. Finally, we use this idea for modeling of constraint in the support vector machine (SVM) classifier. In the following sub-sections, we discuss the SVM history and the fuzzy SVM, fuzzy programming (FP) is also introduced so the reader gets familiar with this concept, and finally a brief introduction to our approach is brought.

1.1 Support vector machines

Support vector machines (SVMs) (Vapnik, 1995) are very popular and powerful in learning systems because of the utilization of kernel machine in linearization, providing good generalization properties, their ability to classify input patterns with



minimized structural misclassification risk and finding the optimal separating hyperplane between two classes in the feature space.

Lin and Wang (2002, 2004) proposed fuzzy support vector machine (FSVM) by considering the noise in the training samples. They used the MF to express the membership value of a sample to positive or negative class. Importance degree of training data are modeled in the FSVM by insertion of membership value μ_i in the penalty term of cost function in the form of:

$$\frac{1}{2} \|W\|^2 + C \left(\sum_{i=1}^l \mu_i \xi_i \right).$$

It is noted that slack variable term ξ_i is scaled by μ_i . The fuzzy membership values are used to weight the soft penalty term in the cost function of SVM. The weighted soft penalty term reflects the relative fidelity of the training samples during training. Important sample with large membership value will have more emphasis in the FSVM training procedure and more effect over determination of hyperplanes.

Liu and Chen (2007) presented total margin-based adaptive fuzzy support vector machines (TAF-SVM). TAF-SVM is a type of FSVM, which have been presented in Lin and Wang (2002, 2004). Authors in Liu and Chen (2007) correct the skew of the optimal separating hyperplane due to the very imbalanced data sets by using different cost algorithm. This is performed by dividing the training data into two categories with different importance and result in dual problem has different boundary for Lagrange multipliers.

In Lin and Wang (2002), linear and quadratic functions are presented for μ_i in the FSVM and two main targets are followed, increasing margin and decreasing misclassification error. In Chu and Wu (2004), authors present two new methods for calculation of MF of μ_i based on geometry distribution of the training samples. Those samples, which are near to optimal hyperplane, have similar geometry property. The main idea of FSVM (Lin and Wang, 2002) is that if the input is detected as an outlier or noisy sample, MF decreases so that the total error term decreases. In Wang *et al.* (2005), a new method for calculation of μ_i of FSVM is presented which follows the same idea that one input is assigned a low membership of the class if it is detected as an outlier. However, the method of Wang *et al.* (2005) treat each input as an input of the opposite class with higher membership and it makes full use of the data and achieves better generalization ability. Also, in two different works Jayadeva and Khemchandani (2005) and Wang and Chiang (2007), authors try to determine MF in multi-category data classification but aforementioned works discuss over noisy or vague samples and reducing their effects. Each vague sample in SVM is converted to one fuzzy constraint so we must consider the FP over crisp and fuzzy numbers with emphasis over constraints. In the following section, we study FP in details.

1.2 Fuzzy programming

Herrera and Verdegay (1995) presented methods to solve fuzzy integer linear programming (FILP) problems with different forms of fuzzy constraints, fuzzy numbers in the objective function and fuzzy numbers defining the set of constraints. They defined constraints with fuzzy inequalities as presented in equation (1):

$$\begin{aligned}
 &\text{Maximize } z = \sum_{j=1}^N c_j x_j \\
 &\text{subject to: } \sum_{j=1}^N a_{ij} x_j < b_i, \quad i = 1, \dots, M \\
 & \quad \quad \quad x_j \geq 0, \quad j = 1, \dots, N \quad x_j \text{ is integer, } j = 1, \dots, N
 \end{aligned} \tag{1}$$

where M is the number of constraints and N is the number of variables to be optimized. They considered fuzzy constraints defined by fuzzy MFs:

$$\mu_i : R^n \rightarrow (0, 1], \quad i = 1, \dots, M \tag{2}$$

and i -th constraint corresponds to a MF is shown in equation (3):

$$\mu_i(x) = \begin{cases} 1, & \text{if } \sum_{j=1}^N a_{ij} x_j \leq b_i \\ \frac{[(b_i+d_i)-a_i x]}{d_i}, & \text{if } b_i \leq \sum_{j=1}^N a_{ij} x_j \leq b_i + d_i \\ 0, & \text{if } \sum_{j=1}^N a_{ij} x_j \geq b_i + d_i \end{cases} \tag{3}$$

where $d_i > 0$, $i = 1, \dots, M$. Then, the problem is converted to a parametric programming with the parameter α -cut of the constraint ($\mu_X(x) \geq \alpha$), where $X(\alpha) = \{x \in R^n | \mu_X(x) \geq \alpha\}$, $\forall x \in R^n$, $\mu_X(x) = \inf\{\mu_i(x), i \in M\}$. Therefore, each constraint in equation (1) can be written as:

$$\begin{aligned}
 &\text{Maximize } z = \sum_{j=1}^N c_j x_j \\
 &\text{Subject to: } a_i x_j \leq b_i + d_i(1 - \alpha), \quad i = 1, \dots, M \\
 & \quad \quad \quad x_j \geq 0, \quad j = 1, \dots, N \quad x_j \text{ is integer, } j = 1, \dots, N, \quad \alpha \in (0, 1]
 \end{aligned} \tag{4}$$

Also, Herrera and Verdegay (1995) studied FILP problems with imprecise coefficients in the objective function that is, with coefficients defined by fuzzy numbers in the form of equation (5):

$$\begin{aligned}
 &\text{Maximize } z = \sum_{j=1}^N \underline{c}_j x_j \\
 &\text{Subject to: } \sum_{j=1}^N a_{ij} x_j \leq b_i, \quad i = 1, \dots, M \\
 & \quad \quad \quad x_j \geq 0, \quad j = 1, \dots, N \quad x_j \text{ is integer, } j = 1, \dots, N
 \end{aligned} \tag{5}$$

Moreover, they studied a linear programming problem with fuzzy coefficients and fuzzy right hand-side numbers as:

$$\begin{aligned} \text{Maximize } z &= \sum_{j=1}^N c_j x_j \\ \text{Subject to : } & \sum_{j=1}^N a_{ij} x_j < b_i, \quad i \in M \quad x_j \geq 0, \\ & j = 1, \dots, N \quad x_j \text{ is integer, } \quad j = 1, \dots, N \end{aligned} \quad (6)$$

Rommelfanger (1996) used aggregation operator instead of addition in constraints with better-obtained results. Jimenez and his team with companionship of Verdegay (Jimenez *et al.*, 2003) focused on solving fuzzy nonlinear programming problems in the form of equation (7):

$$\begin{aligned} \text{Minimize } & f(x) \\ \text{subject to : } & g_i(x) < b_i, \quad i = 1, \dots, m \\ & x_i \in [l_i, u_i], \quad i = 1, \dots, n, \quad l_i \geq 0 \end{aligned} \quad (7)$$

where $x = (x_1, \dots, x_n) \in R^n$ is a n-dimensional real-valued parameter vector. They solve it using evolutionary algorithm because of non-convexity. Moreover, an evolutionary algorithm was used as search algorithm to find optimum solution for nonlinear parametric programming problem.

Leon and Vercher (2004) presented a class of fuzzy linear programming problems in which coefficients in the constraints were modeled as LR-fuzzy numbers. Maiti and Maiti (2007) used fuzzy constraints for finding optimal production with the objective function of minimum cost in the context of a multi-item dynamic production inventory control system. In their model, constraints included fuzzy members. Mula *et al.* (2006) tried to use fuzzy mathematical programming model for production planning under uncertainty in an industrial environment. This model considered fuzzy constraints related to the total cost, the market demand and the available capacity of the productive resources and fuzzy coefficients for costs due to the backlog of demand and for the required capacity.

1.3 Our approach

Some resultant notes from previous work are:

- if each defuzzified constraint for each selected parameter is not satisfied, the answer is not accepted;
- selected range of parameters in obtained parametric programming problem is questionable; and
- some forgotten constraints affect in the solution.

We attempt to present a new method for solving parametric programming and add interesting properties to the previous works. Therefore, a learning-based scheme is selected to learn the constraints, for this purpose intelligent computing tools such as artificial neural network and fuzzy logic approaches can be used. Recently, there has been

a growing interest in combining both approaches, and as a result, neuro-fuzzy computing techniques have been evolved (Jang, 1993). ANFIS model combines the neural network adaptive capabilities and the fuzzy logic qualitative nature. ANFIS was first presented by Jang (1993). It has attained its popularity due to a broad range of useful applications in such diverse areas in recent years as optimization of fishing predictions (Nuno *et al.*, 2005), vehicular navigation (Noureldin *et al.*, 2007), identify the turbine speed dynamics (Kishor *et al.*, 2007), radio frequency power amplifier linearization (Lee and Gardner, 2006), microwave application (Ubeyli and Guler, 2006), image de-noising (Qin and Yang, 2007; Daoming and Jie, 2006), prediction in cleaning with high-pressure water (Mula *et al.*, 2006), sensor calibration (Depari *et al.*, 2007), fetal electrocardiogram extraction from ECG signal captured from mother (Assaleh, 2007), identification of normal and glaucomatous eyes from the quantitative assessment of summary data reports of the Stratus optical coherence tomography in Taiwan-Chinese population (Huang *et al.*, 2007) and noise reduction in images (Çivicioglu, 2007). These applications show that ANFIS is a good universal approximator, predictor, interpolator and estimator. They demonstrate that any nonlinear function of many inputs and outputs can be easily constructed with ANFIS.

This study aims to using ANFIS for constraints learning in a convex or non-convex optimization problem[1]. Learning of constraints leads to substitution of crisp constraints with a fuzzy system tuned with neural networks (NNs). Converting crisp constraints to fuzzy rules and MFs with uncertainty property, results in better finding optimal solution. Many problems exist in constraints satisfaction problems, which are resolved due to using ANFIS model. In conventional optimization methods, unsatisfying of any constraint causes searching procedure hit a change in search trace, whereas in the proposed model, unsatisfying of one constraint increases satisfactory membership, which is a type of penalty. In our method, all constraints are substituted with fuzzy inference system (FIS), which is tuned using neural network. In this system, if one constraint is not satisfied, just MF of corresponding set decreases toward zero. For the first time, we use ANFIS for expressing constraints in a form similar to the human thinking. Constraints are converted to rules with uncertainty and are mixed just the same as occur in a fuzzy system. Also, we apply this scheme in the SVM, a famous tool of classification task.

The paper is organized as following: the survey of adaptive neuro-fuzzy inference system is explained in Section 2. Section 3 is devoted to solving constraint parametric programming using ANFIS algorithm. Section 4 appropriates to SVM constraint modeling using ANFIS. Experimental results are discussed in Section 5 and in Section 6 conclusions are presented.

2. Adaptive neuro-fuzzy inference system (ANFIS)

Recently, there has been a growing interest in combining neural network and FIS. As a result, neuro-fuzzy computing techniques have been evolved. Neuro-fuzzy systems are fuzzy systems, which use NNs theory in order to determine their properties (fuzzy sets and fuzzy rules) by processing data samples. Neuro-fuzzy integrates to synthesize the merits of both NNs and fuzzy systems in a complementary way to overcome their disadvantages (Abraham, 2005; Lin and Lee, 1996).

ANFIS has been proved to have significant results in modeling nonlinear functions (Jang, 1993). In an ANFIS, the MFs are extracted from a data set that describes the system behavior. The ANFIS learns features in the data set and adjusts the system

parameters according to given error criterion. In the ANFIS architecture, NN learning algorithms are used to determine the parameters of FIS. Below, we have summarized the advantages of the ANFIS technique:

- Real-time processing of instantaneous system input and output data. This property helps using this technique for many operational researches problems.
- Offline adaptation instead of online system-error minimization, thus easier to manage and no iterative algorithms are involved.
- System performance is not limited by the order of the function since it is not represented in polynomial format.
- Fast learning time.
- System performance tuning is flexible as the number of MFs and training epochs can be altered easily.
- The simple if-then rules declaration and the ANFIS structure are easy to understand and implement.

2.1 Adaptive neuro-fuzzy inference system (ANFIS) architecture

A typical architecture of ANFIS is shown in Figure 1 for modeling of function $f(x, y)$, in which a circle indicates a fixed node, and a square indicates an adaptive node. For simplicity, we consider two inputs x, y and one output z in the FIS. The ANFIS used in this paper implements a first-order Sugeno fuzzy model. Among many FIS, the Sugeno fuzzy model is the most widely used due to its high interpretability and computational efficiency, and built-in optimal and adaptive techniques. For example, for a first-order Sugeno fuzzy model, a common rule set with two fuzzy if-then rules can be expressed as equations (8) and (9):

$$\text{Rule 1 : If } x \text{ is } A_1 \text{ and } y \text{ is } B_1, \text{ then } z_1 = p_1x + q_1y + r_1 \quad (8)$$

$$\text{Rule 2 : If } x \text{ is } A_2 \text{ and } y \text{ is } B_2, \text{ then } z_2 = p_2x + q_2y + r_2 \quad (9)$$

where A_i, B_i ($i = 1, 2$) are fuzzy sets in the antecedent, and p_i, q_i, r_i ($i = 1, 2$) are the design parameters that are determined during the training process. As in Figure 1, the ANFIS consists of five Layers.

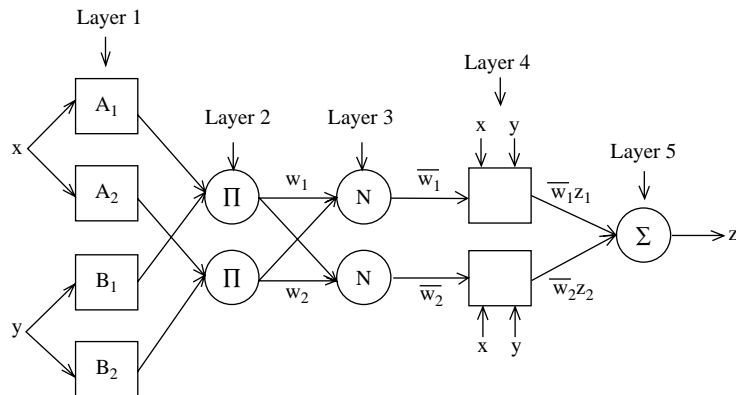


Figure 1. ANFIS architecture

Layer 1, every node i in this Layer is an adaptive node with a node function like equation (10):

$$O_i^1 = \mu_{A_i}(x), \quad i = 1, 2 \quad O_i^1 = \mu_{B_i}(y), \quad i = 3, 4 \quad (10)$$

where x, y are the input of node i , and $\mu_{A_i}(x)$ and $\mu_{B_i}(y)$ can adopt any fuzzy MF. In this paper, Gaussian MFs, which are defined by equation (11) are used:

$$gaussian(x, c, \sigma) = e^{-(1/2)((x-c)/\sigma)^2} \quad (11)$$

where c is center of Gaussian MF and σ is standard deviation of this cluster.

Layer 2, every node in the second Layer represents the ring strength of a rule by multiplying the incoming signals and forwarding the product as equation (12):

$$O_i^2 = \omega_i = \mu_{A_i}(x)\mu_{B_i}(y), \quad i = 1 \quad (12)$$

Layer 3, the i -th node in this Layer calculates the ratio of the i -th rule's ring strength to the sum of all rules' rings strengths:

$$O_i^3 = \varpi_i = \frac{\omega_i}{\omega_1 + \omega_2}, \quad i = 1, 2 \quad (13)$$

where ϖ_i is referred to as the normalized ring strengths.

Layer 4, the node function in this Layer is represented by equation (14):

$$O_i^4 = \varpi_i z_i = \varpi_i (p_i x + q_i y + r_i), \quad i = 1, 2 \quad (14)$$

where ϖ_i is the output of Layer 3, and $\{p_i, q_i, r_i\}$ is the parameter set. Parameters in this Layer are referred to as the consequent parameters.

Layer 5, the single node in this Layer computes the overall output as the summation of all incoming signals like equation (15):

$$O_1^5 = \sum_{i=1}^2 \varpi_i z_i = \frac{\omega_1 z_1 + \omega_2 z_2}{\omega_1 + \omega_2} \quad (15)$$

It is seen from the ANFIS architecture that when the values of the premise parameters are fixed, the overall output can be expressed as a linear combination of the consequent parameters:

$$z = (\varpi_1 x)p_1 + (\varpi_1 y)q_1 + (\varpi_1)r_1 + (\varpi_2 x)p_2 + (\varpi_2 y)q_2 + (\varpi_2)r_2 \quad (16)$$

The hybrid learning algorithm (Jang, 1993; Jang *et al.*, 1997) combining the least square method and the back propagation (BP) algorithm can be used to solve this problem. This algorithm converges much faster since it reduces the dimension of the search space of the BP algorithm. During the learning process, the premise parameters in Layer 1 and the consequent parameters in Layer 4 are tuned until the desired response of the FIS is achieved. The hybrid learning algorithm has a two-step process. First, while holding the premise parameters fixed, the functional signals are propagated forward to Layer 4, where the consequent parameters are identified by the least square method. Second, the consequent parameters are held fixed while the error signals, the derivative of the error measure with respect to each node output, are propagated from the output end to the input end, and the premise parameters are updated by the standard BP algorithm.

3. The proposed method

Let us consider the following parametric programming problem:

$$\begin{aligned} & \text{Minimize } f(x, \theta) \\ & \text{Subject to : } h_i(x, \theta) \leq 0, \quad i = 1, \dots, m \quad \theta \in [\alpha, \beta] \end{aligned} \quad (17)$$

α, β are constants and $x \in R^n$.

Here, $f(x, \theta)$ and $h_i(x, \theta)$, $i = 1, \dots, m$ are functions with respect to the first argument and $\theta \in [\alpha, \beta]$. Constraints $h_i(x, \theta)$ may be non-differentiable, discontinuous and nonlinear.

In conventional method for solving parametric programming problems or FP problems, which is converted to a parametric programming problem, if each constraint is not satisfied x must be deleted from answer set. But in the proposed method, value of unsatisfactorily of each constraints decrease membership of certainty for set of constraints or increase penalty value to final cost function as shown in Figure 2. The value of penalty and its relation can be defined with user or by separate procedure. Also increasing importance to each constraint is performed to simple form.

In the proposed approach, first set of $(x, \theta) \in R^n$ are generated and $h_i(x, \theta)$ ($i = 1, \dots, m$) are observed which is performed using Monte-Carlo simulation[2]. In this procedure, we try to generate all possible states for constraint. For example, for constraint $2x + 4y < 3$, $(0, 0)$ satisfies it and, $(0, 1)$ generate a penalty value. Obtained values for each constraint $h_i(x, \theta)$, $i = 1, \dots, m$ are points in m-dimensional space. Penalty values may be defined in linear or nonlinear form whereas $h_i(x, \theta)$, $i = 1, \dots, m$ depend on user decision. Fitting procedure is done in the second step, which is performed using ANFIS. In the following sub-sections, a complete explanation is presented.

3.1 Constraint learning by ANFIS

In this work, the ANFIS is used to learn constrains for solving a parametric programming problem. We define a feature vector as input to ANFIS which each feature includes penalty value of related constraint. Therefore, the length of the feature vector is m . The feature vector for kth sample is as shown in equation (18):

$$\vec{F}_k = \{\hat{h}_1(x, \theta), \hat{h}_2(x, \theta), \dots, \hat{h}_m(x, \theta)\}_k, \quad k = 1, \dots, N \quad (18)$$

where:

$$\hat{h}_i(x, \theta) = \begin{cases} h_i(x, \theta) & h_i(x, \theta) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (19)$$

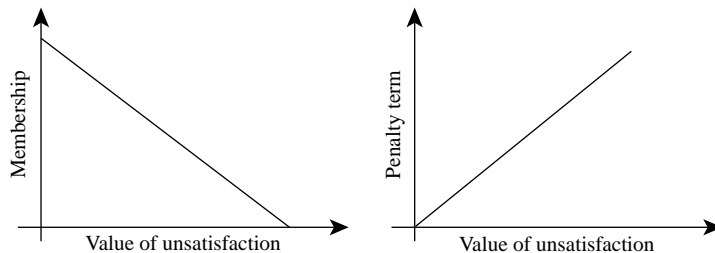


Figure 2.
Membership of certainty for set of constraints (left figure) and penalty value of satisfaction in linear form (right figure)

and \vec{F}_k 's are feature vectors for N training/generated samples. $\hat{h}_i(x, \theta)$ is penalty value and in this paper is selected in the form of equation (19). It can be defined to nonlinear form. Monte-Carlo simulation is used for generating N samples of feature vectors. Also for kth sample, desired output in linear form is defined in equation (20). Of course, in the general case, desired output is given by user:

$$\vec{d}_k = \frac{\sum_{i=1}^m \alpha_i \hat{h}_i(x, \theta)}{\max_i \{\alpha_i \hat{h}_i(x, \theta)\}}, \quad k = 1, \dots, N, \quad i = 1, \dots, M \quad (20)$$

Here, α_i is the weight of each constraint. Denominator of equation (20) is a normalization factor and its numerator is linear combiner of constraints so the desired output is a linear function of constraints, albeit constraints can be nonlinear functions of x and θ . Of course, α_i determine the slop of interpolated hyperplane. If a constraint $h_i(x, \theta)$ is not satisfied, \vec{d}_k increases proportional to $|h_i(x, \theta)|$, as a penalty value. This penalty affects the output with selected slop of α_i . If each constraint is satisfied, then the related penalty is zero. A FIS is generated with \vec{F}_k and \vec{d}_k , and tuned using neural network which is mentioned in Section 2 and more details in Jang (1993) and Jang *et al.* (1997). The obtained system has properties of ANFIS system and is like equation (21):

$$g(x, \theta) = N(\vec{F}_k, \vec{d}_k) \quad (21)$$

$g(x, \theta)$ is continuously differentiable and linear whereas constraints do not have these properties. $N(\vec{F}_k, \vec{d}_k)$ is the constructed FIS. Now problem (17) is converted to a conventional optimization model (22):

$$\text{minimize}_x \quad p(x, \theta) = f(x, \theta) + \eta g(x, \theta) \quad \theta \in [\alpha, \beta] \quad (22)$$

where η is a positive number. We can solve this problem with different methods, like optimization or neural network-based methods (Effati and Jafarzadeh, 2006), or intelligence-based approach (Sadoghi Yazdi and Effati, 2007).

3.2 Properties of new scheme

In this sub-section, some properties of the proposed approach are explained.

3.2.1 Global view of constraints. As we mentioned, a trained system is created instead of constraints using a suitable algorithm like ANFIS. From view point of application in the operation research (OR), noisy sample and missing data are inseparable in the OR problems. Some algorithms are used for finding optimum stochastic parameters from missing data or loss samples also, we encounter in the OR applications in data collection, and entering with missing samples. In the SVM, each training sample is one constraint in the classification task; so lack of each constraint is caused imperfective solution or incorrect solution. But in the new scheme missing constraints are estimated and learning procedure can compensate those if architecture of system is suitable and training algorithm is robust. For this purpose, we selected ANFIS with proved architecture and hybrid learning procedure.

3.2.2 Constraints generality. ANFIS construct a strength interpolator and weak extrapolator if enough learning samples are entered. So, this property can give generality to obtained system. With utilization of ANFIS or each other similar module in the proposed scheme, generality obtain easily.

3.2.3 Clustering of similar constraints. One problem in programming problems is increasing of constraints and so increasing in computational complexity. Clustering of constraints is standard solution for this purpose, which in the ANFIS, clustering is main procedure for reducing number of rules, computation and functions.

3.2.4 Creation of real fuzzy constraints. In the Sub-section 3.1, we see output of system is a membership value. Satisfaction of each constraint increase output \bar{d}_k (equation (19)). On the other hand, if one constraint is not satisfied output decrease proportional to value of unsatisfying of constraint. In real world, we expect if a constraint is not satisfied we pay little penalty for it. We can easily see this property in the proposed method a real fuzzy constraint.

3.2.5 Study of constraint strength. After training of constraints, we can study effect of each constraint. Increasing penalty for each unsatisfied constraint is seen in the output of system. Slope of output variation (slope of penalty value) shows importance of it, which we study it in the next section with examples.

3.2.6 Increasing importance to constraints. Importance degree of constraints is other note in OR. Expert user can give knowledge about magnitude of constraints and interact with the system for receiving optimum results. He/she can emphasis over some constraints with inserting of weights to those in the proposed scheme to easy from.

4. SVM constraint modeling using ANFIS

As we refer in the introduction, SVM is suitable classifier, which is based on optimization technique. Soft margin is the one main problem in the SVM. Using the proposed approach we can enter importance of each training samples and a soft margin is obtained also missing sample/loss data are modeled as well. One main problem in the SVM is number of training samples, for large or huge number of samples, training of SVM is not performed correctly but we solve it using proposed method.

The mathematical formulation of SVM classifier, the concept of soft margin and the way to overcome the problem of large number of samples are discussed in the following three sub-sections.

4.1 Support vector machine formulation

Let $S = \{(x_i, d_i)\}_{i=1}^n$ be a set of n training samples, where $x_i \in R^m$ is an m -dimensional sample in the input space, and $d_i \in \{-1, 1\}$ is the class label of x_i . SVM finds the optimal separating hyper plane with the minimal classification errors. Let w_0 and b_0 denote the optimum values of the weight vector and bias, respectively. The hyper plane can be represented as:

$$w_0^T x + b_0 = 0 \quad (23)$$

where $w = [w_1, w_2, \dots, w_m]^T$ and $x = [x_1, x_2, \dots, x_m]^T$. w is the normal vector of the hyper plane, and b is the bias that is a scalar. The optimal hyper plane can be obtained by solving the following optimization problem (Ling and Wang, 2002):

$$\begin{aligned} & \text{Minimize} \quad \frac{1}{2} \|w\|^2 + c \sum_{i=1}^n \xi_i \\ & \text{Subject to :} \quad d_i(w^T x_i + b) \geq 1 - \xi_i, \\ & \quad \xi_i \geq 0, \quad i = 1, \dots, n, \end{aligned} \quad (24)$$

where ξ_i is slake variable for obtaining soft margin. But c determine effect of slake variable and margin increases by decreasing value of c .

4.2 Soft margin in the SVM

Adding weights to each sample or increase the degree of any sample needs to define nonlinear constraint, which is presented in equation (25):

$$J(w, b, \xi_i, \beta_i, \gamma_i, \theta_i, \hat{\theta}_i) = \frac{1}{2} \|w\|^2 + c \sum_{i=1}^n \xi_i + \beta_i g_i(\xi_i - 1 - d_i(w^T x_i + b) + \theta_i^2) - \gamma_i(\hat{\theta}_i - \xi_i) \quad (25)$$

where β_i, ξ_i, γ_i are Lagrange multipliers, $\hat{\theta}_i, \theta_i$ are values for achieving equal constraints. But $g_i(\cdot)$ is user defined function for obtaining desired effect over constraints. But user knows about the degree of importance of each constraint, he/she can assign weights to constraints but cannot define a nonlinear function. The proposed approach in Sub-section 3.1 help us with defining the emphasis values over constraints. α_i in equation (20) is used for defining the degree of importance of each constraints.

4.3 Increasing training samples in the SVM

In equation (25), number of constraints is equal to number of training samples (n) that with increasing n , solving problems is difficult. But in the proposed approach all constraints are entered in ANFIS and we know ANFIS clusters data and generates Gaussian MF over each cluster. So, this problem is solved using the proposed method and miss data and noisy samples can be solved in the SVM with this idea.

5. Experimental results

Three examples are considered first, then SVM is studied as an application work.

Example 1. Consider the following convex parametric problem[3]:

$$\begin{aligned} \text{Minimize } z(\theta) &= (\theta^2 - \theta - 3)x_1 + (\theta^2 + 2\theta - 6)x_2 \\ \text{Subject to : } x_1 &\leq 4 + \theta^2 \quad 3x_1 + 2x_2 \leq 18 - 2\theta^2 \\ x_1, x_2 &\geq 0 \quad \theta \in [0, 3] \end{aligned} \quad (26)$$

First, constraints are converted to ANFIS model. For this purpose, we have:

$$\hat{h}_1(x, \theta) = \begin{cases} 4 + \theta^2 - x_1 & \text{if } 4 + \theta^2 - x_1 > 0 \\ 0 & \text{elsewhere} \end{cases} \quad (27)$$

$$\hat{h}_2(x, \theta) = \begin{cases} 18 - 2\theta^2 - 3x_1 - 2x_2 & 18 - 2\theta^2 - 3x_1 - 2x_2 > 0 \\ 0 & \text{elsewhere} \end{cases} \quad (28)$$

$$\hat{h}_3(x, \theta) = \begin{cases} x_1 & x_1 > 0 \\ 0 & \text{elsewhere} \end{cases} \quad (29)$$

$$\hat{h}_4(x, \theta) = \begin{cases} x_2 & x_2 > 0 \\ 0 & elsewhere \end{cases} \quad (30)$$

Then:

$$\vec{F}_k = \{\hat{h}_1(x, \theta), \hat{h}_2(x, \theta), \hat{h}_3(x, \theta), \hat{h}_4(x, \theta)\}_k, \quad k = 1, \dots, 1,000$$

is constructed using Monte-Carlo simulation, x_1, x_2 and θ are computed according to:

$$x_1 = \beta_1(\nu_1 - 0.5), \quad x_2 = \beta_2(\nu_2 - 0.5), \quad \theta = \beta_3(\nu_3 - 0.5) \quad (31)$$

where $\beta_1, \beta_2, \beta_3$ determine the range of variables, and ν_1, ν_2, ν_3 are random numbers. Random variables are between (0, 1), so are between $(-0.5\beta_1, 0.5\beta_1), (-0.5\beta_2, 0.5\beta_2), (-0.5\beta_3, 0.5\beta_3)$, respectively.

In this example, $\beta_1, \beta_2, \beta_3$ are selected equal to 20. Then, \vec{d}_k is obtained according to equation (19), with α_i equal to one, and ANFIS model of $N(\vec{F}_k, \vec{d}_k)$ is generated. Obtained input MFs are shown in Figure 3 for second constraint $\hat{h}_2(x, \theta)$. This figure shows this constraint has values near -60 to 0 and around -180 which are clustered to form of mf1, ..., mf4. Output MF is obtained as a linear function of input constraints, as described in Section 2 and Jayadeva and Khemchandani (2005) and Lee and Gardner (2006) to following form:

$$z_k = -0.0041 \sum_{i=1}^4 \hat{h}_i(x, \theta), \quad k = 1, \dots, 4 \quad (32)$$

Generated rules are:

- (1) If $\hat{h}_1(x, \theta)$ is $Mf_1(\hat{h}_1(x, \theta))$ and $\hat{h}_2(x, \theta)$ is $Mf_1(\hat{h}_2(x, \theta))$ and $\hat{h}_3(x, \theta)$ is $Mf_1(\hat{h}_3(x, \theta))$ and $\hat{h}_4(x, \theta)$ is $Mf_1(\hat{h}_4(x, \theta))$. Then, output is z_1 .

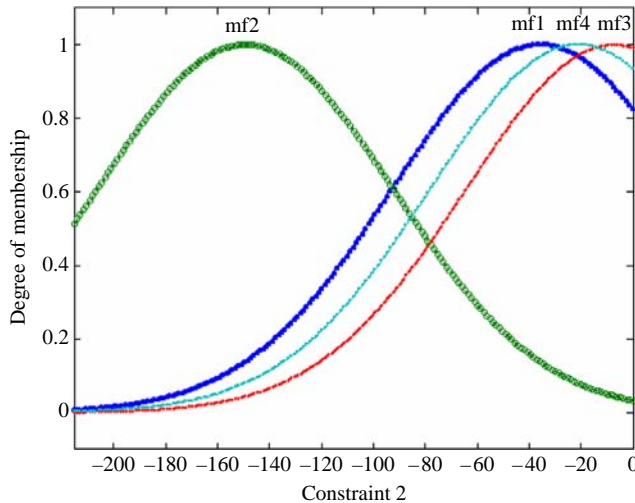
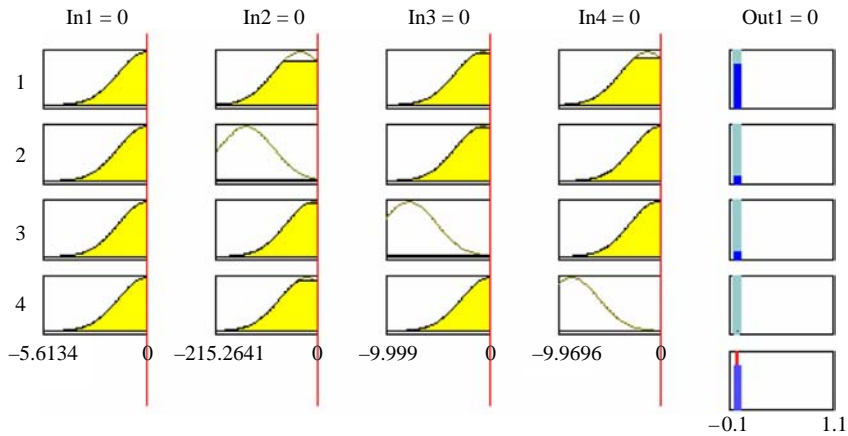


Figure 3.
Obtained input MFs for
second constraint $\hat{h}_2(x, \theta)$

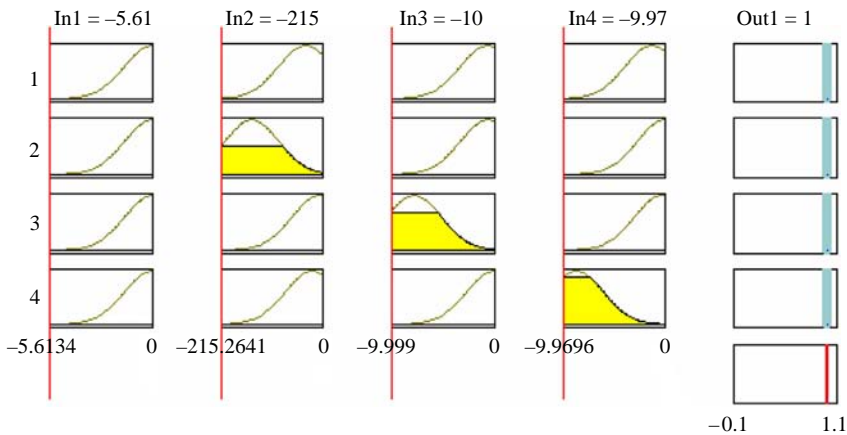
Notes: mf1 (or $Mf_1(\hat{h}_2(x, \theta))$); mf2 (or $Mf_2(\hat{h}_2(x, \theta))$) and mf3 (or $Mf_3(\hat{h}_2(x, \theta))$) are labels of MFs

- (2) If $\hat{h}_1(x, \theta)$ is $Mf_2(\hat{h}_1(x, \theta))$ and $\hat{h}_2(x, \theta)$ is $Mf_2(\hat{h}_2(x, \theta))$ and $\hat{h}_3(x, \theta)$ is $Mf_2(\hat{h}_3(x, \theta))$ and $\hat{h}_4(x, \theta)$ is $Mf_2(\hat{h}_4(x, \theta))$. Then, output is z_2 .
- (3) If $\hat{h}_1(x, \theta)$ is $Mf_3(\hat{h}_1(x, \theta))$ and $\hat{h}_2(x, \theta)$ is $Mf_3(\hat{h}_2(x, \theta))$ and $\hat{h}_3(x, \theta)$ is $Mf_3(\hat{h}_3(x, \theta))$ and $\hat{h}_4(x, \theta)$ is $Mf_3(\hat{h}_4(x, \theta))$. Then, output is z_3 .
- (4) If $\hat{h}_1(x, \theta)$ is $Mf_4(\hat{h}_1(x, \theta))$ and $\hat{h}_2(x, \theta)$ is $Mf_4(\hat{h}_2(x, \theta))$ and $\hat{h}_3(x, \theta)$ is $Mf_4(\hat{h}_3(x, \theta))$ and $\hat{h}_4(x, \theta)$ is $Mf_4(\hat{h}_4(x, \theta))$. Then, output is z_4 .

Constraints fire rules as shown in Figure 4. This figure shows two states of rule operation for boundary conditions. Figure 4(a) shows for constraints ($\hat{h}_i(x, \theta)$, $i = 1, \dots, 4$) with



(a)



(b)

Notes: in1, in2, in3, in4 are four constraints (in1) $\hat{h}_1(x, \theta)$, (in2) $\hat{h}_2(x, \theta)$, (in3) $\hat{h}_3(x, \theta)$; (a) constraints are satisfied and penalty term (Out1) is zeros; (b) all constraints are not satisfied (worst case) and penalty term is maximum)

Figure 4. Rule operation when constraints are selected boundary values

values zero (unsatisfying) which degree of membership of output is zero and for full satisfaction of constraints membership is one (Figure 4(b)).

The importance degree of each constraint can be clearly seen from Figure 5. If constraint $\hat{h}_2(x, \theta)$ is not satisfied, big penalty value is generated and search algorithm moves towards satisfaction of this constraint. But other constraints have minor emphasis ($\{\hat{h}_1(x, \theta), \hat{h}_3(x, \theta), \hat{h}_4(x, \theta)\}$). This can be seen in Figure 6, which output or same penalty value is shown for these constraints. An interesting note in these figures is the linearity of output per inputs, in spite of nonlinearity of constraints per (x, θ) .

Another ability of system is weighting capability over constraints with user. If user wants to increase the importance of one constraint, this is easily implemented in the proposed approach. For example, in above example, we can increase the effect of the first constraint with a new weights selection, for example, $\alpha_1 = 0.9, \alpha_2 = 0.01, \alpha_3 = 0.05, \alpha_4 = 0.04$. Part of the results is shown in Figure 7. This figure shows that the first constraint is not met which causes considerable penalty value relative to second constraint with normal effect.

Above problem is solved for quantized θ with resolution 0.1, i.e. $\theta \in \{0, 0.1, 0.2, \dots, 2.9, 3.0\}$ include 31 solution points. The optimal solution is shown in Figure 8.

Example 2. Consider the following convex parametric programming problem:

$$\begin{aligned} \text{Minimize } z(x_1, x_2) &= -x_1^2 - 3x_2^2 \\ \text{Subject to: } x_1 + x_2 &\leq 6 - \theta \\ -x_1 + 2x_2 &\leq 6 + \theta \quad x_1, x_2 \geq 0 \quad \theta \in [0, 4] \end{aligned} \tag{33}$$

The obtained solution using the proposed method is shown in Figure 9. Importance or efficacy of constraints can be seen in Figure 10. Constraints $x_1, x_2 \geq 0$ are same effect that have not illustrated in this figure. The efficacy of the first and the second constraints are shown in this figure. These figure shows that these constraints have approximately the same effect. In our method for modeling constraints, the effect of each constraint can be studied.

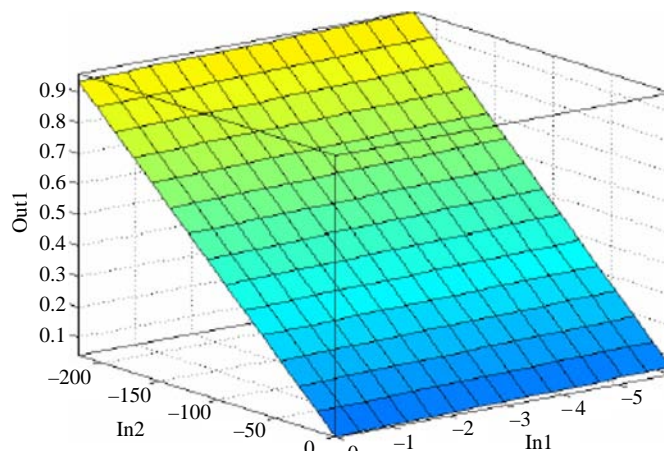
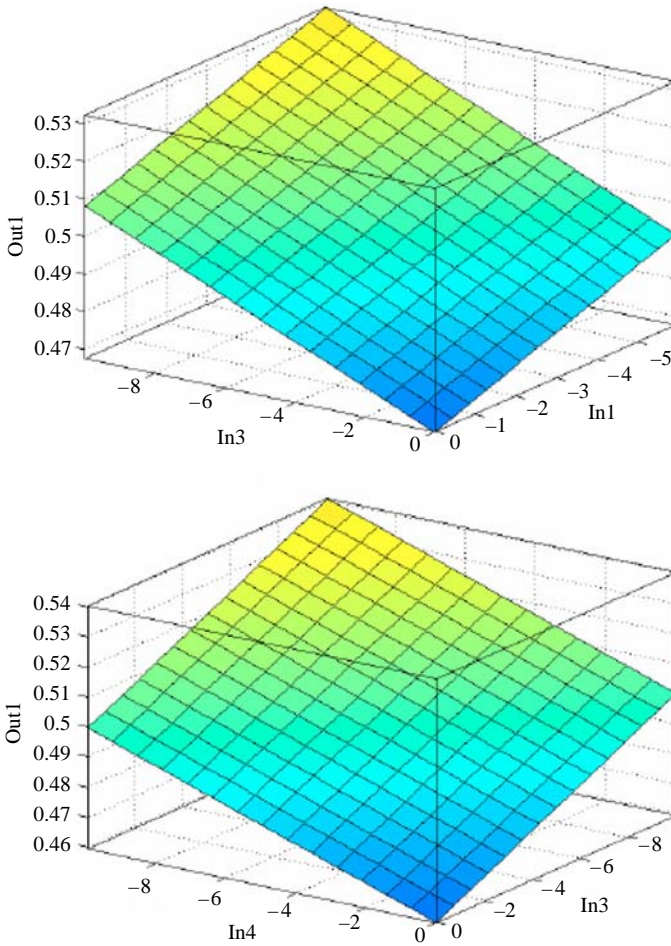


Figure 5.
Output per pair of
constraints for Example 1

Notes: Out1 is output, In1 is $\hat{h}_1(x, \theta)$; In2 is $\hat{h}_2(x, \theta)$



Notes: Out1 is output, In1 is $\hat{h}_1(x, \theta)$; In2 is $\hat{h}_2(x, \theta)$; In3 is $\hat{h}_3(x, \theta)$; In4 is $\hat{h}_4(x, \theta)$

Figure 6. Output per pair of constraints

For more emphasis on linearity of relation between membership of penalty value and constraints, we study following example which have nonlinear constraints.

Example 3. Consider the following constraints in one parametric programming problem.

Constraints:

$$x_1^2 \leq 4 + \theta^2 \quad 3x_1^2 + 2x_2 \leq 18 - 2\theta^2 \quad x_1, x_2 \geq 0 \quad \theta \in [0, 3] \quad (34)$$

The output of ANFIS after constraints learning using Monte-Carlo simulation (membership of penalty value) is shown in Figure 11. We can easily see that a linear relation there exists between the membership penalty value of constrains which is pre-typified but nonlinearity of membership of penalty value and x_1, x_2 is perceptible and is shown in Figure 12. Therefore, one of suitability of the proposed approach in constraint learning is that the effect of nonlinear constraints appears in the form of linear

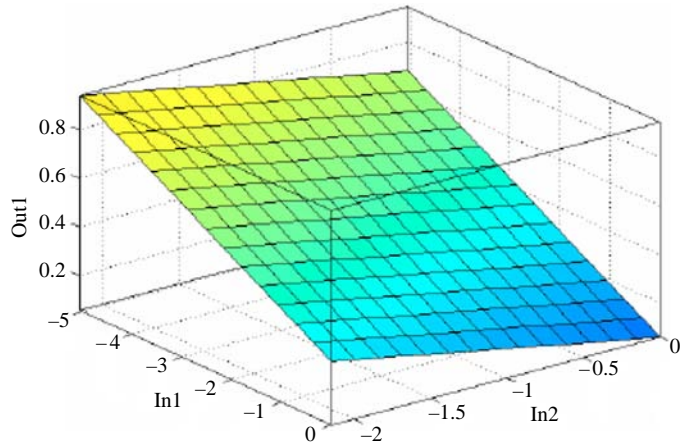


Figure 7.
Studying the effect of weights in constraints

Notes: For selected state $\alpha_1 = 0.9, \alpha_2 = 0.01, \alpha_3 = 0.05, \alpha_4 = 0.04$;
Out1 is output; In1 is $h_1(x, \theta)$; In2 is $h_2(x, \theta)$; output per pair of constraints

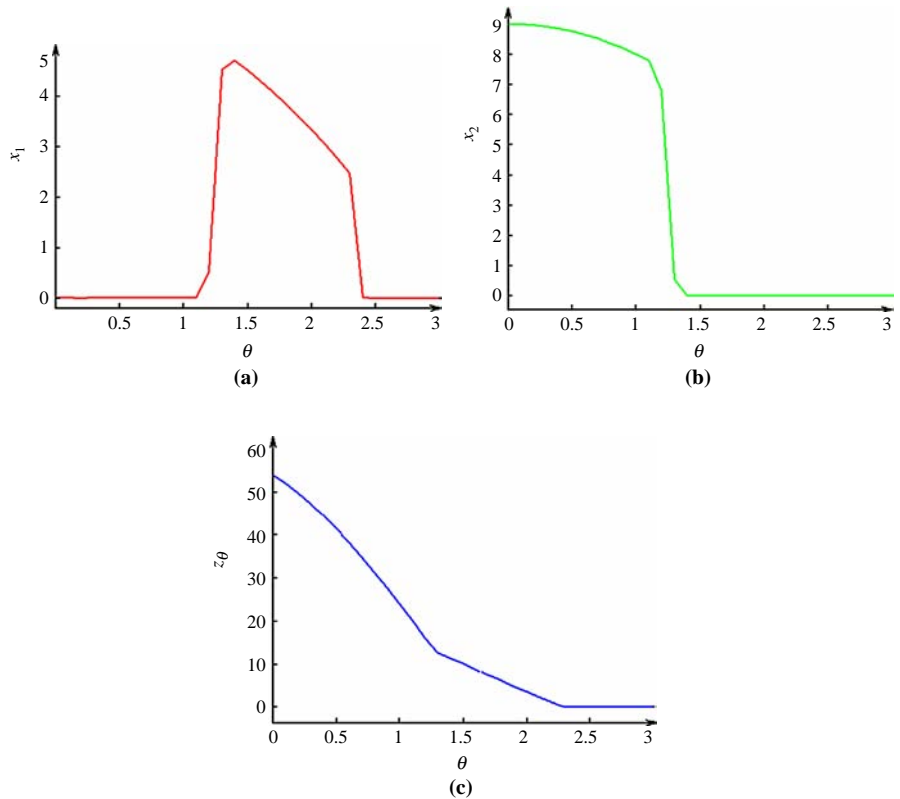


Figure 8.
Optimal solution for Example 1

Notes: (a) x_1 in terms of θ ; (b) x_2 in terms of θ ; (c) z_θ in terms of θ

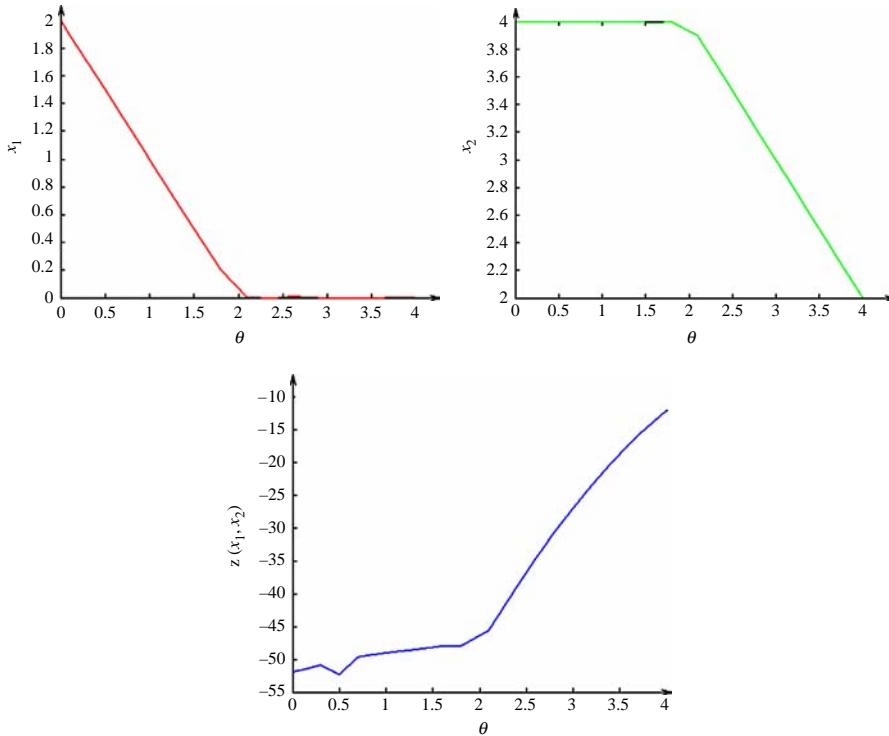
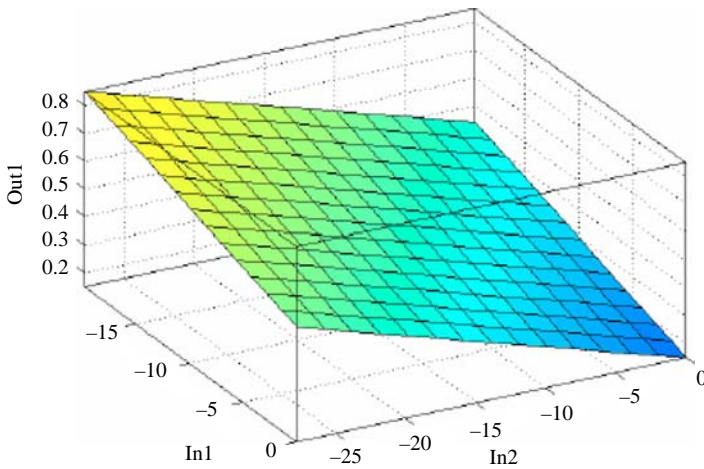


Figure 9. Optimal solution for Example 2



Notes: Out1 is output; In1 is $\hat{h}_1(x, \theta)$; In2 is $\hat{h}_2(x, \theta)$

Figure 10. Output per two constraints for Example 2

Figure 11.
Output of ANFIS after
constraints learning
in Example 3

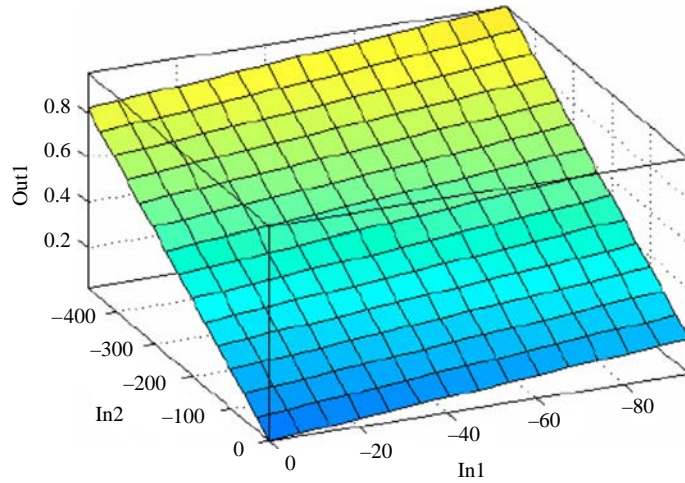
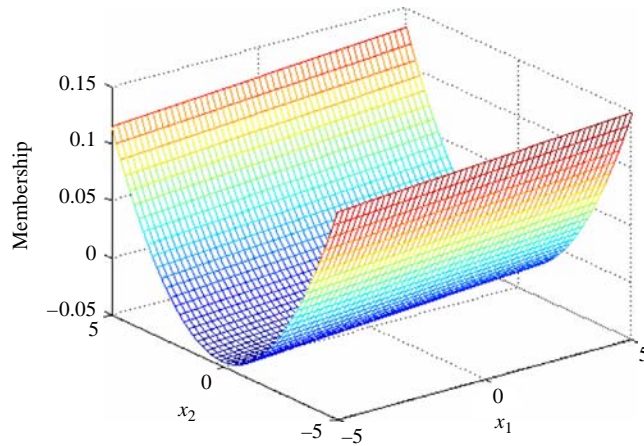


Figure 12.
Membership of penalty
value for x_1, x_2

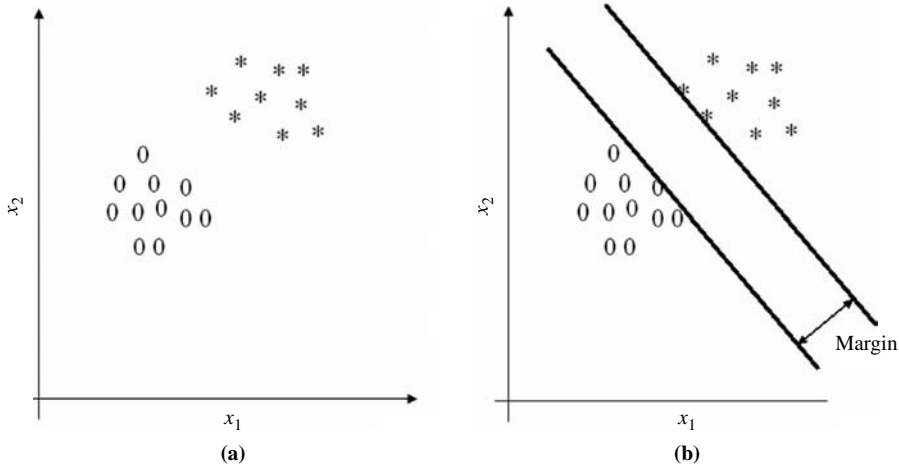


penalty and better convergence is achieved. Also with change of equation (19), user can obtain nonlinear effect of penalties.

Example 4. Consider the constraints modeling in the SVM.

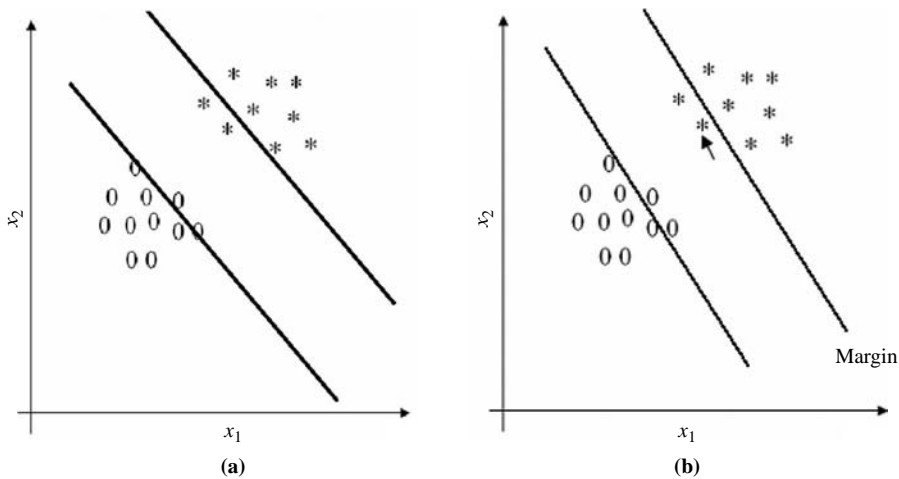
Figure 13(a) shows two classes of data, which are going to be separated by an SVM. Using equation (24), if c tends toward infinity, classification task is performed as shown in Figure 13(b) and the two classes are well separated.

However, decreasing c can only increase margin as can be seen in Figure 14(a). But putting more emphasis on the pointed sample using the proposed method leads to the classification, which is shown in Figure 14(b); clearly the slope of optimum hyperplane has changed.



Notes: (a) Input space; (b) margin

Figure 13. Classification task using SVM



Notes: (a) Effect of increasing c; (b) emphasis over pointed sample

Figure 14.

6. Conclusions and future work

In this paper, we studied a new model for dealing with the lack of precision in a vague nature in the formulation of parametric programming problems. The foundation of this model was based on constraint learning using ANFIS. N-constraints were converted to n-dimensional feature vector and feasible region was applied to form of k-training samples to ANFIS. Simulation results show that each parametric programming problems with properties of non-differentiability, discontinuity and nonlinearity can be trained using ANFIS. Understanding of constraints after generation of ANFIS model was one of sensible features. Some of other features of the proposed approach were global view of constraints, constraints generality, clustering of similar constraints, creation of real fuzzy constraints, study of constraint strength, increasing importance

to constraints. Defining of constraint effect with the proposed idea was an important rule. We changed effect of constraints and visualized results using ANFIS facilities in the MATLAB software.

Also we proposed for applying this method over SVM, which is a robust method in the artificial systems and presented some of its properties. But this idea over SVM needs to great operations, which we follow it in the future work. Learning of fuzzy constraints, effect of support vectors and changing this effect, the study of noisy samples over the proposed method in the field of SVM are other future research orientations.

Notes

1. Convex optimization is a sub-field of mathematical optimization. Given a real vector space X together with a convex, real-valued function $f : X \rightarrow \mathbb{R}$ defined on a convex subset X of X , the problem is to find the point x^* in X for which the number $f(x)$ is smallest, i.e. the point x^* such that $f(x^*) \leq f(x)$ for all $x^* \in X$. The convexity of X and f make the powerful tools of convex analysis applicable: the Hahn-Banach theorem and the theory of sub-gradients lead to a particularly satisfying and complete theory of necessary and sufficient conditions for optimality, a duality theory comparable in perfection to that for linear programming, and effective computational methods.
2. Monte-Carlo methods are a class of computational algorithms that rely on repeated random sampling to compute their results. Monte-Carlo methods are often used when simulating physical and mathematical systems. Because of their reliance on repeated computation and random or pseudo-random numbers, Monte-Carlo methods are most suited to calculation by a computer. Monte-Carlo methods tend to be used when it is infeasible or impossible to compute an exact result with a deterministic algorithm. Monte-Carlo simulation methods are especially useful in studying systems with a large number of coupled degrees of freedom. More broadly, Monte-Carlo methods are useful for modeling phenomena with significant uncertainty in inputs. These methods are also widely used in mathematics. The term Monte-Carlo method was coined in the 1940s by physicists working on nuclear weapon projects in the Los Alamos National Laboratory.
3. Convex parametric programming is discussed in Boyd and Vandenberghe (2004).

References

- Abraham, A. (2005), "Adaptation of fuzzy inference system using neural learning, fuzzy system engineering: theory and practice", *Studies in Fuzziness and Soft Computing*, Vol. 3, pp. 53-83.
- Assaleh, K. (2007), "Extraction of fetal electrocardiogram using adaptive neuro-fuzzy inference systems", *IEEE Trans. Biomedical Engineering*, Vol. 54 No. 1, pp. 59-68.
- Boyd, S. and Vandenberghe, L. (2004), *Convex Optimization*, 1st ed., Cambridge University Press, New York, NY.
- Chu, L. and Wu, C. (2004), "A fuzzy support vector machine based on geometric model", *Proceedings of the 5th World Congress on Intelligent Control and Automation, Hangzhou, People's Republic of China*, pp. 1843-6.
- Çivicioglu, P. (2007), "Using uncorrupted neighborhoods of the pixels for impulsive noise suppression with ANFIS", *IEEE Trans. Image Processing*, Vol. 16 No. 3, pp. 759-73.
- Daoming, G. and Jie, C. (2006), "ANFIS for high-pressure water jet cleaning prediction", *Surface & Coatings Technology*, Vol. 201, pp. 1629-34.

- Depari, A., Flammini, A., Marioli, D. and Taroni, A. (2007), "Application of an ANFIS algorithm to sensor data processing", *IEEE Trans. Instrumentation and Measurement*, Vol. 56 No. 1, pp. 75-9.
- Effati, S. and Jafarzadeh, M. (2006), "A new nonlinear neural network for solving a class of constrained parametric optimization problems", *Appl. Math. Comput.*, Vol. 186 No. 1, pp. 814-19.
- Herrera, F. and Verdegay, J.L. (1995), "Three models of fuzzy integer linear programming", *European Journal of Operational Research*, Vol. 83, pp. 581-93.
- Huang, M.L., Chen, H.Y. and Huang, J.J. (2007), "Glaucoma detection using adaptive neuro-fuzzy inference system", *Expert Systems with Applications*, Vol. 32, pp. 458-68.
- Jang, J.S.R. (1993), "ANFIS: adaptive-network-based fuzzy inference system", *IEEE Trans. System Man and Cybernetics*, Vol. 23 No. 5, pp. 665-85.
- Jang, J.S.R., Sun, C.T. and Mizutani, E. (1997), *Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice-Hall, Englewood Cliffs, NJ.
- Jayadeva, R. and Khemchandani, S. (2005), "Fuzzy linear proximal support vector machines for multi-category data classification", *Neurocomputing*, Vol. 67, pp. 426-35.
- Jimenez, F., Cadenas, J.M., Verdegay, J.L. and Sanchez, G. (2003), "Solving fuzzy optimization problems by evolutionary algorithms", *Information Sciences*, Vol. 152, pp. 303-11.
- Kishor, N., Singh, S.P. and Raghuvanshi, A.S. (2007), "Adaptive intelligent hydro turbine speed identification with water and random load disturbances", *Engineering Applications of Artificial Intelligence*, Vol. 20 No. 6, pp. 795-808.
- Lee, K.C. and Gardner, P. (2006), "Adaptive neuro-fuzzy inference system (ANFIS) digital predistorter for RF power amplifier linearization", *IEEE Trans. Vehicular Technology*, Vol. 55 No. 1, pp. 43-51.
- Leon, T. and Vercher, E. (2004), "Solving a class of fuzzy linear programs by using semi-infinite programming techniques", *Fuzzy Sets and Systems*, Vol. 146, pp. 235-52.
- Lin, C.F. and Wang, S.D. (2002), "Fuzzy support vector machine", *IEEE Trans. Neural Networks*, Vol. 13 No. 2, pp. 464-71.
- Lin, C.F. and Wang, S.D. (2004), "Training algorithms for fuzzy support vector machines with noisy data", *Pattern Recognition Letters*, Vol. 25, pp. 1647-56.
- Lin, C.-T. and Lee, C.S.G. (1996), *Neural Fuzzy Systems: A Neuro-fuzzy Synergism to Intelligent Systems*, Prentice-Hall, Upper Saddle River, NJ.
- Liu, Y.H. and Chen, Y.T. (2007), "Face recognition using total margin-based adaptive fuzzy support vector machines", *IEEE Trans. Neural Networks*, Vol. 18 No. 1, pp. 178-92.
- Maity, K. and Maiti, M. (2007), "Possibility and necessity constraints and their defuzzification – a multi-item production-inventory scenario via optimal control theory", *European Journal of Operational Research*, Vol. 177, pp. 882-96.
- Mula, J., Polera, R. and Garcia-Sabater, J.P. (2006), "Material requirement planning with fuzzy constraints and fuzzy coefficients", *Fuzzy Sets and Systems*, Vol. 158 No. 7, pp. 783-93.
- Noureddin, A., El-Shafie, A. and Tahab, M.R. (2007), "Optimizing neuro-fuzzy modules for data fusion of vehicular navigation systems using temporal cross-validation", *Engineering Applications of Artificial Intelligence*, Vol. 20, pp. 49-61.
- Nuno, A.I., Arcay, B., Cotos, J.M. and Varela, J. (2005), "Optimization of fishing predictions by means of artificial neural networks, ANFIS, functional networks and remote sensing images", *Expert Systems with Applications*, Vol. 29, pp. 356-63.

- Qin, H. and Yang, S.X. (2007), "Adaptive neuro-fuzzy inference systems based approach to nonlinear noise cancellation for images", *Fuzzy Sets and Systems*, Vol. 158 No. 10, pp. 1036-63.
- Rommelfanger, H. (1996), "Fuzzy linear programming and applications", *European Journal of Operational Research*, Vol. 92, pp. 512-27.
- Sadoghi Yazdi, H. and Effati, S. (2007), "Eigenvalue spread criteria in the particle swarm optimization algorithm for solving of constraint parametric problems", *Applied Mathematics and Computation*, Vol. 192 No. 1, pp. 40-50.
- Ubeyli, E.D. and Guler, I. (2006), "Adaptive neuro-fuzzy inference system to compute quasi-TEM characteristic parameters of micro shield lines with practical cavity sidewall profiles", *Neurocomputing*, Vol. 70, pp. 296-304.
- Vapnik, V. (1995), *The Nature of Statistical Learning Theory*, Springer, New York, NY.
- Wang, T.Y. and Chiang, H.M. (2007), "Fuzzy support vector machine for multi-class text categorization", *Information Processing & Management*, Vol. 43, pp. 914-29.
- Wang, Y., Wang, S. and Lai, K.K. (2005), "A new fuzzy support vector machine to evaluate credit risk", *IEEE Trans. Fuzzy Systems*, Vol. 13 No. 6, pp. 820-31.

About the authors



Hadi Sadoghi Yazdi was born in Sabzevar, Iran, in 1971. He received the BS degree in Electrical Engineering from the Ferdowsi University of Mashhad, Iran in 1994, and then he received to the MS and PhD degrees in Electrical Engineering from the Tarbiat Modarres University of Tehran, Iran in 1996 and 2005, respectively. He has been with Engineering Department of Tarbiat Moallem University of Sabzevar in 2005-2008. He is currently an Assistant Professor at the Department of Computer Engineering, Ferdowsi University of Mashhad, Iran. His research interests include adaptive filtering, image and video processing. Hadi Sadoghi Yazdi is the corresponding author and can be contacted at: sadoghi@sttu.ac.ir



Reza Pourreza was born in Iran in 1982. He holds a BS and a MS in Electrical Engineering, received from the Ferdowsi University of Mashhad in 2005 and 2008, respectively. He is currently pursuing his PhD Program in Computer Engineering at the Department of Computer Engineering, Ferdowsi University of Mashhad, Iran. His research interests include machine vision, biomedical image processing and pattern recognition.



Mehri Sadoghi Yazdi was born in Iran in 1985. She received the BS degree in Computer Engineering from the Azzahra University, Iran in 2007. She is currently pursuing his MS Program in Computer Engineering at the Department of Electrical and Computer Engineering, Shahid Beheshti University of Tehran, Iran. Her research interests include artificial intelligence algorithm and image processing.