Proceedings of the World Congress on Engineering 2010 Vol I
WCE 2010, June 30 - July 2, 2010, London, U.K.

Sparse Classifier Design Based on the Shapley Value

Prashanth Ravipally and Dinesh Govindaraj *

Abstract—**Handheld devices like mobile phones have very restricted memory and computation environment. Softwares for Face recognition, Speech recognition, Handwriting recognition etc which are developed for mobile phones need to occupy very less space and should do recognition in real-time. We design a sparse classifier which aids these applications by storing very few vectors, recognizing real-time and still generalizing well. Our Sparse Classifier learns a function that is a weighted sum of basis functions, by sequentially appending functions to an initially empty basis, so that the learnt function minimizes the norm of squared loss to approximate a target function. Selection of a smaller set of basis functions from a dictionary of functions is based on Shapley value, which is a well known solution concept from game theory. The basis functions are selected based on the importance in approximating the decision boundary. We show how our sparse classification model built on kernel-based solutions effectively controls the sparsity of the solution. Experimental comparison with SVMs demonstrate that the proposed model shows comparable accuracy in benchmark datasets with very few basis functions resulting in more than 80% reduction in the number of stored vectors.**

Keywords: sparse classifier, support vector machine, shapley value, mobile applications, generalization

1 Introduction

Handheld devices like mobile phones have very restricted memory and computational environment. Machine learning applications such as Face recognition, Speech Recognition, Handwriting recognition are getting popular in devices like iPhone and Android. For example Google App added *search by voice* feature which enables searching the web using voice queries runs on iPhone, Blackberry and other mobile phones. Clearly these applications which runs on mobile phones needs to occupy very less space and should do recognition in real-time. We attempt to speedup the mobile machine learning applications by designing a spare classifier. Our sparse classifier will store very few vectors, recognize in real-time and still generalize well.

*1. Oracle Corporation India Email: ravipally.reddy@oracle.com and 2. Bell Labs Research India Email: dinesh.govindaraj@alcatel-lucent.com

Most of the above mentioned applications follow supervised learning paradigm during the training phase. In supervised learning we are given a set of examples of input vectors $\{x_n\}_{n=1}^{l}$ along with their corresponding targets $\{y_n\}_{n=1}^{l}$, the latter of which might be real values (in regression) or class labels {+1, -1} (binary classification). From this training set we wish to learn a model with the objective of making accurate predictions of $\mathbf{y}$ for unseen values of $\mathbf{x}$. Typically, our predictions on unseen data is based upon some function $f(x)$ defined over the input space, and learning is the process of inferring (perhaps the parameters of) this function.

The *support vector machines* [14] (SVM) makes predictions based on the function of form:

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i K(x, x_i) + \theta_0 \tag{1}$$

where $\{\alpha_i\}$ are the positive Lagrange multipliers, $K(\cdot,\cdot)$ is a *kernel* function. A key feature of SVM is that, in classification case, its target function attempts to minimize the number of errors made on the training set while simultaneously maximizing the 'margin' between the two classes (in the feature space implicitly defined by the kernel). However, despite its success, we can identify a number of significant and practical disadvantages of the support vector learning methodology: 1. Although relatively sparse, SVMs make liberal use of basis functions. The number of support vectors grow linearly with the size of training set and the time for computing the target function $f(x)$ is also proportional to the number of support vectors. This makes SVM highly unsuitable for handheld devices like mobile phones. So sparsity is an important issue, both for computational efficiency, storage efficiency and generalization performance. 2. It is necessary to estimate the error/margin trade-off parameter 'C'. This generally requires a cross-validation procedure, results in increasing the computational time of the model. 3. The kernel function $K(\cdot,\cdot)$ must satisfy Mercer's condition.

Another flexible and popular set of candidates for $f(x)$ is of the form:

$$f(x, \beta) = \sum_{i=1}^{m} \beta_i \Phi(x, x_i) + \beta_0 \tag{2}$$

where $m \leq l$, $\{\beta_i\}$ are the model weights and $\Phi(\cdot,\cdot)$ is

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

a basis function. The basis functions are more general compared to the kernel functions as they need not satisfy the Mercer's condition[1]. Analysis of functions of type 2 is easy since the weight parameters $\{\beta_i\}$ is linear and the objective is to find 'good' values for these. In this paper we propose a novel Sparse Classification Model(SCM) of the form 2 which avoid all the above mentioned disadvantages of SVM. The key feature of this model is that the model requires relatively fewer basis functions as well as offering good generalization performance,

Outline of the paper is as follows: Section 2 describes related work, Section 3 and 4 presents our approach and design of sparse classifier. Section 5 shows our experimental results and improvements we achieved over SVM classifier. Section 6 concludes the paper.

2 Related Work

There are several methods to build a sparse kernel classifier. In [3], a method is described to speed up the classification process by approximating the solution using a smaller number of vectors. Their method proposes to determine n reduced set vectors $z_1, z_2, \ldots, z_n$, whose linear combination approximates the kernel classifier built in advance. The "reduced set"of vectors determined by Burges' method are generally not support vectors and can in some cases be computed analytically.

In [5], the reduced support vector machine (RSVM) algorithm was proposed, and uses a randomly selected subset of the data to obtain a nonlinear separating surface. But the obtained classifier may not guarantee good classification performance always because the support vectors are chosen at random. In the relevance vector machine proposed in [4] a prior on the expansion coefficients favors sparse solutions.

Our approach of adopting cooperative game theory techniques to the problem of *relevant vector selection* is primarily inspired from [2]. It proposes the Contribution selection algorithm (CSA) for feature selection. The algorithm is based on the Multi-perturbation Shapley Analysis (MSA), a framework which relies on game theory to estimate usefulness of a feature. It iteratively estimates the usefulness of the features and selects them accordingly, using either forward selection or backward elimination.

3 Our Approach

Our aim is to build a sparse classifier with good generalization performance. The classification surface is determined by the selected points known as *relevant vectors* which come from the training samples. In this work, we introduce a Sparse Classification Model(SCM) designed using Shapley value. The Shapley value divides the collective value of the game among the players according to their marginal contribution. That is, the higher the Shapley value of a player, the more important the player in the game.

The proposed SCM, is a general, greedy, sparse approximation scheme with the squared error loss, which iteratively adds new functions (basis functions) to the linear expansion. These basis functions are centered on the training instances and their selection in each iteration (*relevant vector selection*) is based on the Shapley value of the corresponding training instances. The relevant vector selection refers to the problem of selecting input vectors, otherwise called training instances, that are relevant in predicting a target function.

We convert the concepts of cooperative game theory and shapley value to relevant vector selection problem. Here each training vector will now become the player and Shapley value of a vector now represents the importance or contribution of that training vector in classification. Selection of training vectors is based on the contribution values of those vectors. Unlike SVM, the non zero weights in the SCM are not associated with examples close to the decision boundary, but rather represent prototypical examples of classes.

The important features of the proposed SCM over SVM are: 1. It typically requires fewer basis functions without loss in the generalization performance when compared to an equivalent SVM and also allow us to directly control the sparsity of the solution. 2. Selection of relevant vectors on which basis functions are centered, is based on the Shapley value, a novel approach different from traditional ones. 3. It removes the requirement that kernel function should satisfy Mercer's condition for kernel-based solutions. 4. It uses the traditional squared error and also allows other differentiable loss functions.

Next section gives brief overview of Shapley Value.

3.1 Cooperative Game and the Shapley value

Cooperative game theory [8] is a coalitional game where group of players is associated with the payoff and the competition here is between coalitions. Formally, a coalitional game is defined by a pair (N, v) where $N = \{1, \ldots, n\}$ is the set of all players and $v(S)$, where $S \subseteq N$, is a value/payoff of the coalition S. It is important in such a game to divide the value of the coalition to the players in the coalition. The value naturally should correspond to the contribution of the player in achieving a high payoff of the coalition. The Shapley value assigns value for each player in the coalition game based on the contribution.

The Shapley value is defined as follows. Let the marginal

[1]Mercer's condition tells us whether or not a prospective kernel is actually a dot product in some space

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

contribution/importance of the player i to coalition S, with $i \notin S$, be $\triangle_i(S) = v(S \cup \{i\}) - v(S)$. Then the Shapley value defined by the payoff

$$\phi_i(v) = \frac{1}{n!} \sum_{\pi \in \Pi} \triangle_i(S_i(\pi)) = \frac{1}{n!} \sum_{\pi \in \Pi} [v(S_i(\pi) \cup i) - v(S_i(\pi))] \tag{3}$$

where Π is the set of permutations over N and $S_i(\pi)$ is the set of players appearing before the i'th player in permutation π. The Shapley value of a player is a mean of a marginal value over all possible subsets of players.

The calculation of the Shapley value requires sum over all possible subsets of players, which is impractical in the case of relevant vector selection problem. An unbiased estimator for the Shapley value by uniformly sampling permutations from Π is presented by Keinan *et al.* [1] . The bounded estimated contribution value becomes

$$\varphi_i(v) = \frac{1}{|\Pi_d|} \sum_{\pi \in |\Pi_d|} \triangle_i(S_i(\pi)) \tag{4}$$

where Π_d is the set of sampled permutations on subsets of size d. The resulting estimated Shapley value is an efficient value, since the sum of the marginal importance of all players in any permutation is $v(N)$. For a detailed discussion of the MSA framework and its theoretical background see [1].

Relevant vector selection attempts to estimate the contribution of each training vector in generating a classifier. The players are mapped to the training vectors and the payoff is represented by a real-valued function $v(S)$, which measures the performance of a classifier generated using the set of training vectors S. This setting turns classification in to a coalitional game among the training vectors. Higher the Shapley value of a training vector implies that it is more *relevant* in predicting the target function.

3.2 Regularized Least-Squares Learning Model and the Representer Theorem

Given a set of examples $\{x_n\}_{n=1}^l$ along with corresponding targets $\{y_n\}_{n=1}^l$, which have been drawn i.i.d from an unknown joint probability distribution $P(X, Y)$, the aim is to find a function f which minimizes the following risk $R[f] = E(f(X) - Y)^2$. We can use any loss function for estimating the risk, but here we use the traditional *squared loss*. Since $P(X, Y)$ is unknown, we have to look for the function f which minimizes the empirical risk $R_{emp}[f] = \sum_{i=1}^l (f(x_i) - y_i)^2$. This problem is ill-posed and a classical way to turn it into a well-posed one is to use regularization theory [9]. In this context, the solution of the problem is the function $f \in H$ that minimizes the regularized empirical risk :

$$R_{reg}[f] = \frac{1}{n} \sum_{i=1}^{l} (f(x_i) - y_i)^2 + \lambda \Omega(f) \tag{5}$$

where H is the hypothesis space, Ω is measures the smoothness of f and λ a regularization parameter [10]. Under general conditions on H (Reproducing Kernel Hilbert Space), the solution of this minimization problem is of the form: $f(x, \beta) = \sum_{i=1}^l \beta_i K(x, x_i)$ where K is the reproducing kernel of H and has to satisfy Mercer's condition [6]. The above result is proposed in [11] and known as the *representer theorem*. The question of the sparsity of the solution f can be addressed in two different ways. The first approach is to use a regularization term in equation 5 that imposes sparsity of β whereas the second one is based on a stepwise method consisting of adding functions from a dictionary. The bias-variance problem involves several parameters, especially the kernel parameters and the hyper-parameter trading between goodness-of-fit and regularization.

Our proposed model is based on the second approach i.e., the step-wise method of adding functions from a dictionary. We use a greedy constructive strategy similar to Matching Pursuit proposed by [12] for approximating the target function.

4 The Sparse Classification Model(SCM)

Given a set of examples of input vectors $\{x_n\}_{n=1}^l$ along with corresponding targets $\{y_n\}_{n=1}^l$, and a basis function $\Phi : R^d \times R^d \rightarrow R$, we use as our dictionary the basis functions centered on the training points: $D = \{d_i = \Phi(., x_i) | i = 1..l\}$. The sparse decision functions which we are interested are expansions of the form

$$f_N(x) = \sum_{n=1}^{N} \beta_n \Phi(x, x_{\gamma_n})$$

where N is the number of basis functions (sparsity control parameter) in the solution and $N << l$. Here $\gamma_{1..N}$ are the indexes of vectors which are *relevant* in predicting the target function. Selection of relevant vectors $\{x_{\gamma_n}\}_{n=1}^N$ among the set of input vectors, for a given number N of allowed basis functions is in general an NP-complete problem. So the SCM proceeds in a greedy constructive, fashion:

1. It starts at stage 0, with sparse decision function $f_0 = 0$
2. At each stage n, the vector x_{γ_n} is selected with the highest contribution value (highest estimated Shapley value)
3. The basis function centered on the relevant vector is appended to the linear expansion.
   Given f_{n-1} we build,

$$f_n = f_{n-1} + \beta_n \Phi(., x_{\gamma_n})$$

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Algorithm 1: Sparse-Classification-Model(SCM) Algorithm(S; t, d)

Data: Dataset $\{(x_1, y_1), \ldots, (x_l, y_l)\}$, Dictionary of functions $D = \{\Phi(., x_1), \ldots, \Phi(., x_l)\}$, Desired number of basis function N
Result: Sparse Classifier Function $f(x)$
begin
  Labels vector L, dictionary matrix D and $selected \leftarrow \phi$.
  for $n = 1..N$ **do**
    for $s \in S$ **do**
      1. Sample permutations set $\{\pi_1, \ldots, \pi_t\}$ over $S \backslash selected$
      2. $C_s := Contribution(s, selected; d) = \phi_s(v)$
    $\gamma_n \leftarrow argmax_{s \in S} \{C_s\}$
    $selected \leftarrow selected \cup \gamma_n$
    $R \leftarrow L(selected)$
    $\beta_n \leftarrow argmin_\beta \left\{\|R - \beta\Phi(., x_{\gamma_n})\|^2\right\}$
    $R \leftarrow R - \beta_n \Phi(., x_{\gamma_n})$
end

4. Selects $\beta_n \in R$ that minimize the squared norm of the residue $\|R_n\|^2 = \|f_n - L\|^2$

$$\beta_n \leftarrow argmin_\beta \left\{\|f_{n-1} + \beta\Phi(., x_{\gamma_n}) - L\|^2\right\}$$

where $f_{n-1} = \sum_{i=1}^{n-1} \beta_i \Phi(., x_{\gamma_i})$

5. Finally, after N iterations the solution will be of the form

$$f_N(x) = \sum_{n=1}^{N} \beta_n \Phi(x, x_{\gamma_n})$$

We have not yet specified how to choose N (i.e. when to stop), but we shall rather use the error estimated on an independent validation set to decide when to stop. In any case, it can be seen as primary capacity-control parameter of the algorithm. The pseudo-code for the corresponding algorithm is given in Algorithm 1.

In Algorithm 1, S represents the input set of data vectors. t, d and N are hyperparameters: $t = |\Pi_d|$ is the number of permutations sampled (see equation 4), d is the maximal permutation size for calculating the contribution values, and N is a capacity-control parameter.

The *contribution* subroutine calculates the contribution values φ_s by equation 4 where the payoff function $v(S)$ is simply the validation accuracy of any general classifier[2] $F_S(x)$, generated based only on the training vectors from set S. More precisely, $v(S)$ is the accuracy of new decision surface constructed corresponding to the training vectors which are in the set S.
Given set S, payoff $v(S)$ is computed as follows

[2]this classifier is independent of the classifier f

Algorithm 2: Payoff function $v(S)$

Data: Dataset $\{(x_1, y_1), \ldots, (x_l, y_l)\}$, Dictionary of functions $D = \{\Phi(., x_1), \ldots, \Phi(., x_l)\}$, Desired number of basis function H
Result: $f_H(x) = \sum_{i=1}^{H} \alpha_i \Phi(x, x_{\gamma_i})$
$v(S) = \frac{|\{x | f_H(x) = y, (x,y) \in V\}|}{|V|}$
begin
  Labels vector L, dictionary matrix D and $selected \leftarrow \phi$.
  for $n = 1..H$ **do**
    $\gamma_n \leftarrow \max_{k=1..H} \left|\frac{\langle \Phi(., x_k), L\rangle}{\|\Phi(., x_k)\|}\right|$
    $\alpha_n \leftarrow \left|\frac{\langle \Phi(., x_{\gamma_n}), L\rangle}{\|\Phi(., x_{\gamma_n})^2\|}\right|$
    $L \leftarrow L - \alpha_n \Phi(., x_{\gamma_n})$
end

1. $S := S \cup selected$
2. Generate a classifier $F_S(x)$ from the training set.
3. Evaluate $F_S(x)$ for all examples of the validation set.
4. Return the accuracy level, defined as $v(S) = \frac{|\{x | F_s(x) = y, (x,y) \in V\}|}{|V|}$

The case $S = \phi$ is handled by returning the fraction of majority class instances. The maximal permutation size d and number of permutations $|\Pi_d|$ have an important role in deciding the contribution values of the different vectors, and should be selected in a way that ensures that different combinations of data vectors that interact together are inspected.

However, due to computational constraints we have focused on fast classification algorithm similar to Kernel Matching Pursuit algorithm [12]. The pseudo-code for the payoff computation algorithm is given in Algorithm 2

In Algorithm 2, S is the input set of training vectors. $|S|$ is the training set size and $v(S)$ is the payoff or accuracy rate on validation set with KMP algorithm.

Here we would like to emphasize the fact that for both the algorithms, dictionary of basis functions gives a lot of additional flexibility, as it is possible to include any function into it: 1. There is no restriction on the shape of the kernel (no positive-definiteness constraint, could be asymmetrical, etc). 2. Dictionary could include more than a single fixed kernel shape. 3. For huge datasets, a reduced subset can be used as the dictionary to speed

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

up the training. 4. The dictionary can incorporate non-kernel based functions.

The other important motivation for this work, is that sparsity of SCM is directly controlled by controlling the sparsity of the solution, i.e. the number N of allowable basis functions, whereas the sparsity of SVMs is controlled through the error/margin trade-off parameter C, which has an indirect and hardly controllable influence on sparsity.

5 Experimental Results

In all the experiments, we restrict ourselves to using the following Gaussian kernel as the basis function

$$K(x_i, x_j) = e^{\frac{-\|x_i - x_j\|^2}{\sigma^2}}$$

where x_i is the ith example and σ determines width of the gaussian kernel.

5.1 2D Example : Banana Dataset

We first use two dimensional Banana dataset[3] to illustrate graphically the selection of relevant vectors and the decision boundary that the proposed SCM generates based on these vectors.

Hyperparameters (the σ of the kernel function, the margin/error trade-off parameter C for soft margin SVM and the number of relevant vectors N for the SCM algorithm) were chosen using cross-validation technique. Figure 1 illustrates the dependency between the validation set accuracy and test set accuracy for the Banana dataset as a function of number of selected relevant vectors. As it can be seen from Figure 1 that maximum accuracy on validation set is achieved with 35 relevant vectors. Since the SCM usually stops when the accuracy on the validation set stops improving, the algorithm halts once it selects 35 relevant vectors if $35 \leq N$, otherwise it halts selecting N relevant vectors. The corresponding weight parameters of the kernel functions centered on these selected relevant vectors are calculated simultaneously. The performance of the classification improves on the validation set as the algorithm selects new relevant vectors, while the contribution values of the selected relevant vectors decrease. The behavior for the other datasets is similar. Decision boundaries of the target function approximated using SVM and SCM and the corresponding error rates are shown in Figure 2 and Figure 3.

5.2 Benchmark Datasets

To test the proposed SCM empirically we ran a number of experiments on six real-world datasets. Table 4.1 gives the description of datasets we have used.

[3]This and all other data sets we have are publicly available online at http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm.

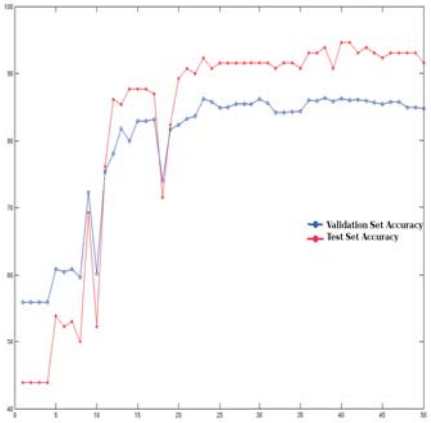

Figure 1: Test set and validation set accuracy v/s number of selected relevant vectors.

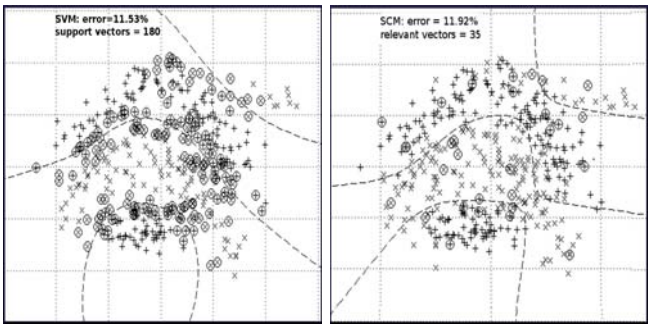

Figure 2: Decision boundary of Banana training set by SVM and SCM. Relevant vectors are shown circled.

Table 1: Description of datasets and parameters used with the **SCM**

Name	M	d	t
Heart	50	30	100
Image	65	100	60
Banana	50	45	100
German	60	75	70
Diabetes	50	75	75
B.Cancer	40	45	100

For all the above data sets, training and testing size follows [4] and we used the following procedure: two out of three equi-sized random partitions of training dataset was used for training and remaining for validation. The SCM is prone to over-fitting on the validation set; when the classifier performance is evaluated on a possibly small validation set. In order to solve this problem one can use 10-fold cross validation; where the training set will be split into several parts and then payoff function evaluated by averaging a classifiers performance on the whole training set.

In order to examine the effect of different parameters val-

[4]http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm.

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

ues on the algorithm, we ran SCM on all datasets with different values of d (size of subsets analyzed) and values of t (number of permutations in each iteration). For small values of d, not enough interactions between different training points are considered. As d increases the performance on the dataset increased until it reaches a critical value. For very small values of t, the performance of the algorithm is limited. But as t grows to values a little higher, the performance (accuracy) grows as well until it becomes stable. Parameters N and H (number of desired relevant points or kernel functions in the target function), which control the sparsity of the solution is chosen using cross validation on the dataset. For each dataset, the parameters used with sparse classification algorithm are shown in Table 1. σ of kernel function and C the margin/error trade-off parameter of SVM for each dataset were determined by applying three-fold cross validation on the training set. The best parameters were then used in implementing the algorithm.

This procedure was repeated over 20 times over 20 different random splits of the dataset into train/validation. Table 2 summarizes the average number of vectors selected in each of the experiments and the corresponding classifier's average accuracy rate on the test set. As it can be seen from these experiments, the error rates obtained are comparable, but SCM appears to require fewer support points (kernel functions) than SVMs. On the average, there are more than 80% reduction in number of support vectors than SVMs which is very significant and well suited for mobile applications.

Table 2: Comparison of error rates and av. # of vectors selected in the different datasets.

Dataset	SVM	SCM	#s.v.	#r.v.
Heart	15.95%	17.94%	105	33
German	23.61%	24.56%	425	32
Image	6.39%	7.95%	420	38
Diabetis	24.53%	23.46%	293	16
Banana	11.58%	12.43%	145	30
B.Cancer	27.54%	24.96%	120	15

6 Conclusion

The sparse classification model presented in this paper views the task of selecting relevant vectors in the context of coalitional games. It uses a novel ranking method that is based on the Shapley contribution values of the training vectors. Examples in this paper have effectively demonstrated that the proposed SCM can attain a comparable level of generalization accuracy as that of the well-established SVM, while at the same time utilizing very few kernel functions. By storing fewer kernel functions we can achieve much faster recognition with less space and achieve comparable accuracy with start-of-art classifiers like SVM.

References

[1] A. Keinan, B. Sandbank, C. Hilgetag, I. Meilijson, E. Ruppin. Fair attribution of functional contribution in artificial and biological networks. *Journal of Neural Computation*, 16(9):1887-1915, (2004).

[2] Shay Cohen, Gideon Dror, Ruppin. Eytan. Feature Selection Via Coalitional Game Theory. *Journal of Neural Computation*, 19(7):1939-1961, (2007).

[3] C. J. C. Burges. Simplified support vector decision rules. *Proceedings of 13th International Conference on Machine Learning*, Bari, Italy, 1996, pp. 71-77.

[4] M. E. Tipping, Sparse Bayesian learning and the relevance vector machine, *Journal of Machine Learning Research*, 1, 211-244, 2001.

[5] Y. J. Lee and O. L. Mangasarian, RSVM: reduced support vector machines, *CD Proceedings of the First SIAM International Conference on Data Mining*, Chicago, 2001, CD-ROM.

[6] V. N. Vapnik. The Nature of Statistical Learning Theory. Springer, 1995.

[7] R.B. Myerson. Game Theory: Analysis of Conflict. *Harvard University Press*, (1997).

[8] P.D. Straffin. Game Theory and Strategy. *The Mathematical Association of America*, (1993).

[9] A. N. Tikhonov and V. A. Arsenin (1977). *Solutions of Ill-posed Problems. Winston and Sons*, Washington.

[10] G. Wahba (1990). *Spline Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM.

[11] G. Kimeldorf and G. Wahba (1971). Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic.*, 33, 82?95.

[12] S. Mallat and Z. Zhang (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12), 3397?3415.

[13] M. Shubik (1985). *Game theory in the social sciences*. Cambridge, MA: MIT Press

[14] Burges, C. J. C. *A Tutorial on Support Vector Machines for Pattern Recognition*. Data Mining and Knowledge Discovery 1998, VOL 2; NUMBER 2, pages 121-168

ISBN: 978-988-17012-9-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)