Evolutionary Training of SVM for Classification Problems with Self-Adaptive Parameters

Ángel Kuri-Morales[1], Iván Mejía-Guevara[2]

[1] Departamento de Computación, Instituto Tecnológico Autónomo de México,
Río Hondo No. 1,
01000 D. F., México
akuri@itam.mx
[2] Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México, IIMAS, Ciudad Universitaria,
04510 D. F., México
imejia@uxmcc2.iimas.unam.mx

Abstract. In this paper we describe a methodology to train Support Vector Machines (SVM) where the regularization parameter (C) is determined automatically via an efficient Genetic Algorithm (Vasconcelos' GA or VGA) in order to solve classification problems. We call the kind of SVMs where C is determined automatically from the application of a GA a "Genetic SVM" or GSVM. In order to test the performance of our GSVM, we solved a representative set of problems. In all of these the algorithm displayed a very good performance. The relevance of the problem, the algorithm, the experiments and the results obtained are discussed.

1 Introduction

Support Vector Machines have recently received increasing attention from the scientific community due to their underlying mathematical foundation. As opposed to more informal (and traditional) alternatives to neural network development, SVMs rely on well understood mathematical properties which, in effect, allow us to theoretically prove that, for example, perceptron networks (PN) or radial basis function (RBF) ensembles are all encompassed by them. Architectural issues such as the number of hidden layers and the number of neurons in such layers are dispensed with. A number of parameters the user has to heuristically estimate (such as the learning rate in PNs or the number of centers in RBFs) are not present. One key issue in this kind of networks, however, has to do with the so-called "regularization parameter" which, in effect, determines the accuracy of the SVM in terms of possible misclassification of sample elements unknown during the training phase. "C", as of today, has been traditionally determined on a case basis and, although some prior efforts to automate its value do exist [1] there has not been a reliable report of it systematic case-independent automated calculation. In this paper we propose the use of evolutionary computation techniques which help us solve the problem of C's determination; particularly, we focus on classification problems. In Section 2 we discuss some theoretical issues re-

© A. Gelbukh, R. Monroy. (Eds.)
Advances in Artificial Intelligence Theory
Research on Computing Science 16, 2005, pp. 207-216

garding SVMs, specifically emphasizing the importance of regularization parameter C. In section 3 we discuss how the methodology of VGA can be used to train this kind of NN and show how to determine automatically the regularization parameter from its application to the dual problem. We also argue that this methodology is appropriate to solve constrained optimization problems, such as these. In section 4 we present four problems we analyzed to show how the GSVM may solve Classification Problems and the resulting level of accuracy. Three of these data sets were obtained from the University of California at Irvine Machine Learning Repository (UCI-MLR); a fourth was derived theoretically. In section 5 we discuss the experiments and results. Finally, in Section 6 we offer our conclusions and point to future lines of research.

2 Support Vector Machines

SVM is a supervised neural network that has been used successfully for classification and nonlinear regression problems [2][3][4]. In what follows we use the notation "x_i" and "w" to denote the independent variable vectors and the weight vectors respectively. A training sample $\tau = \{(x_i, d_i)\}_{i=1}^{N}$ (where x_i is the input pattern for the *i*th example and d_i is the target output) represents two classes in the case of pattern classification and a set of *N* independent variables with *N* dependent variable (d_i) in the case of nonlinear regression.

When attempting pattern classification, the objective is to find a surface that allows the separation of the objects in the sample in two classes: the first class should be on one side of the surface $(d_i = 1)$ and the second class on the other side $(d_i = -1)$. The distance between the nearest points of both classes is called the margin of separation and the optimal surface is found when that margin is maximized.

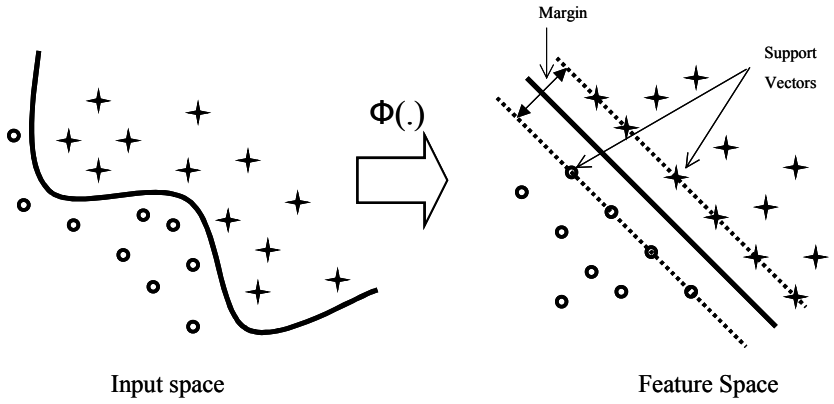

Fig. 1. Transformation from input space to higher-dimensional feature space.

The form of the surface depends of the linear separability characteristics of τ, i. e., when τ is "linearly separable" the optimal surface corresponds to a hyperplane that is

called "Optimal Hyperplane" (OHP) and when τ is "nonlinearly separable", the optimal surface is not a hyperplane in the input space. The introduction of kernel functions is made in order to deal with non-linear decision surfaces. This implies mapping the data to a higher dimensional feature space which allows the construction of an OHP in this space that adequately separates the two classes. In Figure 1, the class 1 (squares) and the class 2 (stars) are non-linearly separable in the input space. In the feature space, however, both classes are separated from each other with a hyperplane.

The kernel functions are used to map vectors in the input space into vectors in the feature space. These functions must satisfy certain known conditions to be admissible as kernels in a SVM. Specifically they must satisfy Mercer´s condition [5][6]. Many functions may be used as kernels [7], but the most popular are: a) Polynomial learning machines (PLM), b) Radial-basis function networks (RBF) and c) Two-layer perceptron networks (LP) [8]. Since the theory allows for any of the above, we used PLM and RBF kernels due to their proven simplicity.

2.1 Primal and dual forms

As mentioned above, we want to find the OHP which maximizes the margin of separation between the two classes that constitute the training set. This gives rise to a constrained optimization problem which has to be solved to get the OHP. The form of the problem depends on linearly separable characteristics of the training set. The Quadratic Programming (QP) problem for linearly separable patterns is formulated as follows:

$$\begin{aligned} &\underset{w,b}{Min}\ \Phi = \frac{1}{2} w^T w \\ &\text{subject to :} \\ &d_i\left(w^T x_i + b\right) \geq 1 \quad \text{for } i = 1, 2, ..., N \end{aligned} \tag{1}$$

The solution of this problem requires the search of w and b that minimize an objective convex function subject to a set of linear constrains. In the case of nonlinear patterns, a set of slack variables is introduced $\{\xi_i\}_{i=1}^{N}$ in order to control the level of misclassification for some elements of τ [9]. In this case the QP problem is:

$$\begin{aligned} &\underset{w,b,\xi}{Min}\ \Phi = \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i \\ &\text{subject to :} \\ &d_i\left(w^T x_i + b\right) \geq 1 - \xi_i \quad \text{for } i = 1, 2, ..., N \end{aligned} \tag{2}$$

Equations (1) and (2) correspond to primal problems for classification of linearly and nonlinearly separable classes, respectively. However, it is possible to define the dual problem. The optimal value for both problems is the same [10]. In both problems, (1) and (2), the solution of the dual form corresponds with the Lagrange Multipliers (LMs) of the QP problem and the LMs different from zero correspond to the support vectors [11]. The dual form for nonlinearly separable patterns is:

$$Max\, Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j d_i d_j K(x_i, x_j)$$
$$\text{subject to:} \quad \sum_{i=1}^{N} \alpha_i d_i = 0 \quad 0 \le \alpha_i \le C \quad \text{for } i = 1, 2, ..., N \tag{3}$$

The dual form for separable patterns is essentially the same, except for $\alpha_i, i = 1,..., N$ which are not bounded; here, C is the upper bound on α_i. It is important to note that a kernel is included in the dual form ($K(\cdot,\cdot)$), a fact which permits us to construct a decision surface that is nonlinear in the input space but whose image in the feature space is linear.

Regularization Parameter. The upper bound C for the LMs in a nonlinearly separable QP problem is known as "Regularization Parameter" [12]. This parameter controls the trade-off between the complexity of the machine and the level of misclassification allowed. When C is low, a higher proportion of errors is allowed in the solution, while few errors are permissible for high C values.

Automatic determination of C via GA. "C" is traditionally selected by the user. It may be estimated experimentally or analytically [13]. The analytical option relies on the calculation of Vapnik-Chervonenkis (VC) dimension for the problem at hand. VC dimension is, however, extremely difficult to calculate in practice and, in effect, disallows the analytical approach. Therefore, the main goal of this paper is to propose a method to estimate automatically the optimal value of this parameter using a GA without the practical limitations mentioned above. In our approach C's value is in the genome and induces a new constraint. This possibility is exclusive of the evolutionary approach (and perhaps a few other meta-heuristics) and explains our choice.

3 Genetic Algorithms

GAs are nowadays commonly used to solve complex optimization problems [14]. It is natural to tackle the problem of finding a good value of "C" with one. In what follows we briefly discuss the methodology.

3.1 Training a SVM using GAs

Several commercial optimization libraries can be used to solve the quadratic programming problem. However, these libraries are of limited used. The memory requirements of the QP problem grow with the square of the size of the training sample [15]. For that reason, in real-life applications, the QP problem cannot be solved by straight forward use of a commercial optimization library. Some optimization tech-

niques can be directly applied to QP problems. However, many of them require that the kernel matrix is stored in memory, implying that the space complexity is quadratic in the sample size. For large size problems, these approaches can be inefficient, and should therefore be used in conjunction with other techniques [16]. In this paper, we use GAs to tackle the QP problem.

GAs as optimization tool. The application of GAs to SVMs differs substantially from previous approaches to train NNs because the dual QP problem presented above is used to find the support vectors directly. In previous experiences the support vectors have been determined from the application of Lagrange Multipliers which neatly adjust to this problem (which satisfies Karush-Kuhn-Tucker conditions) but which are not applicable to search for "C" [13]. In fact, GAs are used here to solve the constrained QP. One advantage of using GAs for this kind of problems is that restrictions are not imposed in the form of the objective function: neither the objective function nor the constrains of the problem must be derivable in order to solve de problem properly.

3.2 Relative optimality of VGA

Although GAs were originally designed to solve unconstrained optimization problems, they can be adapted to tackle the constrained cases [17] as will be shown.

The first step is the selection of the population's size. In this work we considered a population of size P = 100 for all of the problems; the initial population was randomly generated; weighted binary fixed point representation was used. Each individual represents a LM (α_i, i=1,...,N), where N is the number of points in the training set for the dual SVM problem. Every variable is to be expressed in fixed point format with one sign bit (0→+; 1→-), 8 integer bit and 20 decimal bits as shown in figure 2.

α_i		
Sign	**Int**	**Dec**
1 bit	8 bits	20 bit

Fig. 2. Fixed point representation

With this representation: $-2^8+2^{-20} \le \alpha_i \le +2^8-2^{-20}$. The genome's size is (N+1)x29, where N is the number of training data (N) and the (N+1)th point corresponds to the value of C. Once the initial population is generated, VGA [18] is used with P_m=0.05 (probability of mutation) and P_c=0.9 (probability of crossover). The evaluation function is constructed following the methodology of SVMs but we modify it by transforming the constrained original problem to a non-constrained one. To do this, we have chosen the penalty function (*F(x)*) [19]:

$$F(x) = \begin{cases} \left[Z - \sum_{i=1}^{s} \frac{Z}{t} \right] - f(x) & s \neq t \\ 0 & otherwise \end{cases} \quad (4)$$

where Z is a large constant [$O(10^9)$], t is the number of constraints, s is the number of these which have been satisfied and $f(x)$ corresponds to the fitness at point x. The GA operation was terminated after 250 generations.

4 Training SVMs for classification problems

We have applied the methodology to the problem of determining whether an object from a sample belongs to one of two groups. This may be easily extended to N groups [20]. SVMs have traditionally been designed to deal with binary classification, but a lot of real world problems have more than two classes. In this paper we deal with both, binary and multi-class problems. In the case of multiple class problems, one-versus-one classifier and one-versus-all classifier [21] were used. In one-versus-one classifier, a SVM model is built for each pair of classes. This results in $p(p-1)/2$ (p is the number of classes in a specific problem) SVM classifiers. In one-versus-all classifier, p classifiers are used. The ratio between the number of classifiers in one-versus-one classifier and one-versus-all classifier is $(p-1)/2$, which is significant when p is large. On the other hand, all N observations are used in each classifier in one-versus-all classifier.

4.1 Problems

A set of classification problems is presented here in order to illustrate the classification efficiency of the method. The set of problems are:

Lung Cancer Database. The data for this problem describes 3 types of lung cancers. The Authors give no information on the individual variables nor on where the data was originally used[1]. A total of 32 instances are considered in the original data. Since it has 5 missing attributes only 27 were considered. The data have 56 predictive nominal (values 0-3) attributes. Three classes are considered in this problem with: 9 observations for class 1, 13 for class 2 and 10 for class 3. It is important to mention that the problem has few instances (27) and a lot of attributes (55). For this reason we decided to use natural splines [22] to interpolate and enrich the data. The new (interpolated) data set consisted of 100 objects: 85 were used for training and 15 for testing.

Wine Recognition Database. These data are the result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultures[2]. It corresponds to three types of wines with a total of 178 instances: 59 for wine class 1, 71 for class 2 and 48 for class 3. A total of 13 continuous attributes for each object was considered.

[1] UCI-MLR [http://www.ics.uci.edu/~mlearn/MLRepository.html]
[2] Idem

Iris Plant Database. This is perhaps the best known database to be found in the pattern recognition literature. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant[3]. One class is linearly separable from the other two, the latter are not linearly separable from each other. Four attributes are in this database: sepal length, sepal width, petal length and petal width (all of these measured in cm).

Functions. Two classes are defined in this problem with the help of algebraic and trigonometric functions. A total of 88 points with 5 attributes was defined for each class, where these values were randomly generated. The range for each point was $[0,\pi]$. The functions sin(.), cos(.), tan(.), ln(.) and sqrt(.) were applied to attributes 1, 2, 3, 4 and 5, respectively, for each instance in the case of class 1. Likewise, the functions senh(.), cosh(.), tanh(.), exp(.) and sqr(.) were applied to each object of class 2. Classes 1 and 2 were defined as the sum of their respective functions and the outputs for class 1 and 2 were set to 1 and –1, respectively. The number of objects in the sample was 176: 150 for training and 26 for testing. We believe the contribution of these functions is to prove the accuracy of this method in functions that have not any particular pattern, since the values for the selected attributes were randomly generated.

5 Experiments and Results

In the column "problem" of Tables 1, 2, 3 and 4 the codes i_j correspond to the results of one-versus-one classifier for i=1,2 and j=2, 3 (i=1 and j=2 in the case of Table 4). In the case of one-versus-all classifier, i=1,2,3 and j=A ("All"). For instance, 1_2 means "class 1 vs. class 2"; 3_A means "class 3 vs. all", etc.

5.1 Lung Cancer

Because 3 classes are considered in this problem, one-versus-one classifier is used in order to test the methodology proposed here. The result of the application of this classifier is shown in Table 1. Results were: 91.2% of average accuracy for training data and 88.9% for test data where splines were applied and 92.3% of average accuracy when the natural spline interpolation was not applied.

5.2 Wine Recognition

One-versus-one and one-versus-all classifiers were used in this problem. As mentioned above, 3 classes are considered in this problem. Hence, this results in 3 SVM classifiers for each alternative. The comparison between them is shown in Table 2. The accuracy of both classifiers is good, but the one-versus-one classifier has a better accuracy with an average of 93.6% for training data and 94.3% for test data. For one-

[3] Idem

versus-all classifier the average accuracy for training data is 80.2% and 84.6% for test data.

Table 1. Results for Lung Cancer Classification Problem

	Problem	Sample			C	Rest 0	Training Accuracy	Test Accuracy	Kernel
		Total	Train	Test					
Original	1_2	27	27		16.005	0.181	92.6%		RBF 2
	2_3	19	19		4.063	0.460	84.2%		RBF 2
	1_3	17	17		16.000	-0.160	100.0%		RBF 2
Splines	1_2s	100	85	15	128.000	-0.010	84.7%	93.3%	RBF 2
	2_3s	70	60	10	144.000	0.004	90.0%	80.0%	RBF 2
	1_3s	100	85	15	192.001	-0.004	98.8%	93.3%	RBF 2

Table 2. Results for Wine Recognition Problem

Problem	Sample			C	Rest 0	Training Accuracy	Test Accuracy	Kernel
	Total	Train	Test					
1_2	130	111	19	160.016	-0.007	95.5%	94.7%	RBF 2
1_3	107	91	16	192.500	-0.004	100.0%	100.0%	RBF 2
2_3	119	102	17	192.000	-0.009	85.3%	88.2%	RBF 2
1_A	176	150	26	160.000	-0.001	89.3%	88.5%	RBF 2
2_A	176	150	26	128.000	-0.010	58.0%	69.2%	RBF 2
3_A	176	150	26	224.000	0.005	93.3%	96.2%	RBF 2

5.3 Iris Plant

One-versus-one and one-versus-all classifiers were used for this recognition problem. The results for this classification problem are shown in Table 3. Because one of the classes is linearly separable, one-versus-one classifier offers a better accuracy than one-versus-all classifier. The reason is that the linearly separable class (iris setosa) shows a 100% accuracy when compared with each of the other classes. The average accuracy for training set was 96.1% and for testing set was 93.3% in the case of one-versus-one. In the case of one-versus-all, 90.9% for both training sample and testing sample.

5.4 Functions

This is a binary classification problem. The results are shown in Table 4. The accuracy for training set was 97.7% and 100% for testing set.

Table 3. Results of GSVM for Iris Plant Classification Problem.

Problem	Sample			C	Rest 0	Training Accuracy	Test Accuracy	Kernel
	Total	Train	Test					
2_3	100	85	15	176.000	-0.001	88.2%	80.0%	Poly 2
1_3	100	85	15	192.000	0.000	100.0%	100.0%	RBF 2
1_2	100	85	15	64.000	-0.006	100.0%	100.0%	RBF 2
1_A	150	128	22	128.000	0.000	100.0%	100.0%	RBF 2
2_A	150	128	22	172.125	0.005	86.7%	86.4%	Poly 2
3_A	150	128	22	128.000	0.000	85.9%	86.4%	Poly 2

Table 4. Results for the Functions problem.

Problem	Sample			C	Rest 0	Training Accuracy	Test Accuracy	Kernel
	Total	Train	Test					
1_2	150	128	22	224.000	0.001	97.7%	100.0%	RBF 2

6 Conclusions

A GSVM classifier is presented in this paper. The application of this algorithm to a set of test problems resulted in a very good performance. The application of a VGA allows us to tackle an extremely complex constrained optimization problem (if judged from the traditional point of view) in a very simple and straightforward way. Consider that every one of the data vectors determines a constraint. For example, in a typical problem the number of constraints is larger than 150. VGA has to determine the band of feasible values out of a potentially infinite set. However, the most important issue is that the value of the regularization parameter was quasi-optimally found trough the algorithm rather than by hand. The reported work seems to indicate that VGA (along with proper constraint handling) is an efficient way to optimize C by including it in the genome. In the past, the difficulty of properly determining the value of C was usually interpreted, simply put, as changing one typical problem in NNs (the determination of the most adequate architecture) into another, perhaps more difficult, one (the best determination of the regularization parameter). If C's determination may be automated as we have shown, then the theoretical advantages of SVMs may be fully exploited and the negative criticism mentioned above may be eliminated.

Acknowledgement

We want to thank UCI-MLR for the use of Lung Cancer Database, Wine Recognition Database and Iris Plant Database.

References

1. Jordaan, E. M. & Smits, G. F.: Estimation of the regularization parameter for support vector regression. Proc. of World Conference on Computational Intelligence I. Honolulu, Hawaii. (2002) 2785-2791.
2. Schmitdt, M., Grish, H.: Speaker identification via support vector classifiers. Proc. of International Conference on Acoustics, Speech and Signal Processing, (1996) 105-108.
3. Drucker, H., Wu, D., Vapnik, V.: Support vector machine for spam categorization. Trans. on Neural Networks, Vol. 10. IEEE, (1999) 1048-1054.
4. Vapnik, V., Golowich, S., Smola A.: Support vector method for function approximation, regression, estimation and signal processing.Adv.Neural Inform.Process.Syst., Vol.9. (1996) 281-287
5. Haykin, S.: Neural Networks. A comprehensive foundation. 2nd ed. Prentice Hall, New Jersey (1999)
6. Mercer, J.: Functions of positive and negative type, and their connection with the theory of integral equations. Transactions of the London Philosophical Society (A), Vol.209.(1909) 415-446.
7. Shawe-Taylor, J., Cristianini, N,: Kernel Methods for Pattern Analysis. Cambridge University Press, UK (2004).
8. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press, UK (2000).
9. Burges, C. J. C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2, Vol. 2. (1998) 121-167.
10. Cristianini, N., Shawe-Taylor, J., op. cit. pp. 79-92.
11. Cristianini, N., Shawe-Taylor, J., op. cit. pp. 93-124.
12. Haykin, S., op. cit., pp. 318-350.
13. Haykin, S., op. cit., pp. 326-329.
14. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge, Massachusetts (1996).
15. Burges, C. J. C.: op. cit. 121-167.
16. Cristianini, N., op. cit., pp. 93-124.
17. Kuri, A., Gutiérrez, J.: Penalty Functions Methods for Constrained Optimisation with Genetic Algorithms.A Statistical Analysis.Lecture Notes in Artificial Intelligence, Vol.2313. Springer-Verlag, (2002) 108-117.
18. Kuri, A., Mejía, I.: Determination of the Regularization Parameter for Support Vector Machines via Vasconcelos' Genetic Algorithm. Transactions on Circuits and Systems, Issue 4, Vol. 4, WSEAS, (2005) 281-286.
19. Kuri, A., Gutiérrez, J., op. cit., pp.109-111.
20. Fung, G., Mangasarian, O. L.: Multicategory proximal support vector machine classifiers. Data Mining Institute Technical Report, (2001) 01-06.
21. Allwein, E. L., Shapire, R. E., Singer, Y.: Reducing multiclass to binary: A unifying approach for margin classifiers.Journal of Machine Learning Research, Vol.1, (2000) 113-141.
22. Bojanov, B., Hakopian, H., Sahakian, B.: Spline Functions and Multivariate Interpolations. Springer-Verlag, (1993).