Determination of the Regularization Parameter for Support Vector Machines via Vasconcelos' Genetic Algorithm

ANGEL KURI-MORALES[1], IVÁN MEJÍA-GUEVARA[2]
[1]Departamento de Computación
Instituto Tecnológico Autónomo de México
[2]Posgrado en Ciencia e Ingeniería de la Computación
Universidad Nacional Autónoma de México
[1]Río Hondo No. 1
México 01000, D.F.
[2]IIMAS, Ciudad Universitaria
México 04510, D.F.
MÉXICO

Abstract: - This paper presents a genetic algorithm (GA) methodology for training a support vector machine (SVM). The SVM method may be viewed as a quadratic optimization problem with linear constraints, where the objective is to minimize the error learning rate and the Vapnik-Chervonenkis (VC) dimension in order to get an Optimal Separating Hyperplane (OSH) that classifies two sets of data. A SVM is a very good tool for classification problems which displays an excellent generalization ability. In order to test our method we solve the XOR problem, a canonical nonlinearly separable problem. We used a genetic algorithm (GA) called Vasconcelos' GA (VGA). The genome was selected to solve the dual SVM problem, where each individual corresponds to a Lagrange multiplier. Our interest lay in finding the "best" value of *C* (the so-called "regularization" parameter); *C* reflects a tradeoff between the performance of the trained SVM and its allowed level of misclassification. We solved the problem in two ways: (a) We provided *C*, as is traditional in the normal treatment of the problem; (b) We implemented a complementary approach, wherein *C* is also included in the genome. In case (b) VGA finds *C*'s value freeing the user from having to find it from heuristics. We report an exact solution for case (a) and, importantly, encouraging results in which the error in the solution for case (b) is practically zero.

Key-Words: - Support vector machine, Genetic algorithms, Quadratic optimization, Nonlinear separability.

1 Introduction

Neural Networks (NN) have been a subject of interest in a wide variety of fields [1][2][3][4][5]. Their usefulness is today beyond doubt. However, the term "NNs" encompasses many different approaches in which the common denominator is the high connectivity (in some sense) of simple computing elements. A NN has to be "trained" using and *ad hoc* method which, typically, defines the sort of NN at hand. In this regard, NNs may be very broadly classified as "supervised" and "non supervised" depending on whether the training algorithm optimizes its results from known examples or not. In the supervised sort of NNs a relatively new theory (pioneered by Vapnik and coworkers[6][7]) has given formal foundation to the so-called Support Vector Machines (SVM) which, unlike better known NNs (i.e. feedforward multi-layer perceptrons) do not depend on heuristics to define neither their architecture nor the training parameters. This is achieved at the cost of losing the attractive sense of analogy between a perceptron and a physical neuron; in SVMs the "neurons" are, simply stated, members of a class of functions which comply with certain mathematical properties. Historically the usefulness of a given NN scheme has been tested by solving a canonical nonlinearly separable problem which single perceptrons are unable to tackle.

On the other hand, every training algorithm is essentially one of optimization. The variety of such training algorithms atests to the underlying difficulty of the optimization task. For the last two decades the application of a new breed of optimization tools (namely, evolutionary optimization techniques) has permeated almost every field of applied mathematics, and NNs have been no exception. But, as of today, one very interesting problem regarding the proper selection of a fundamental parameter in SVMs has been overlooked: the optimum selection of the "regularization" parameter (typically denoted with C [8]).

Training a SVM constitutes a quadratic optimization problem, where C controls the capability of network to generalize and minimize the number of misclassification errors in problems with linearly nonseparable patterns. In the case of linearly separable patterns, the C parameter is not necessary because, in principle, there is no missclassification: a hyperplane may separate exactly two sets of data.

As mentioned above, a set of evolutionary optimization techniques has been successfully applied in the past. These techniques may be employed to train a SVM and, particularly, to find the optimum values for certain parameters (such as C) instead of having to introduce them by hand or from heuristics. The particular breed of genetic algorithm (GA) we used is the so-called Vasconcelos' GA (or VGA) which has been proven to display superior relative performance in a large set of problems [9].

In Section 2 we show why the *XOR* problem is a nonlinearly separable pattern problem which can be solved with a SVM. In Section 3 we discuss some theoretical issues regarding SVMs. We also point out how the methodology of VGA can be used to train this kind of NN. In section 4 we present the results of training a SVM with the help of VGA. This results are obtained, first, by introducing the appropriate values for C parameter in the genome. This result is equivalent to exchanging the classical Lagrange multipliers' method with a genetic algorithm. More interestingly, we solve the problem of determining the best value of C automatically. This issue is not solvable with classical techniques. Finally, in Section 5 we offer our conclusions and point to future lines of research.

2 Exclusive OR (XOR) problem

The *XOR* problem is the canonical example of nonlinear separability. This problem has beeen used in the past to test the efficiency of certain pattern classification methods. Following Minsky and Pappert [10] we point out that single-layer perceptron ensembles are unable to solve the XOR problem. We know now that this is so [11] because this sets do not have hidden neurons and, consequently, do not have the necessary ability to map the original problem to higher dimensional spaces, which are indispensable to classify input patterns that are nonlinearly separable.

More sophisticated methods have been developed which can solve the *XOR* problem (and, in general, ANY [12] nonlinearly separable problem) with relative ease. For instance: feedforward multilayer perceptrons networks, radial-basis function networks and support vector machines [13].

The *XOR* problem is defined as a function of two boolean variables, as follows: $0 \oplus 0 = 0$, $1 \oplus 1 = 0$, $1 \oplus 0 = 0$ and $0 \oplus 1 = 1$, where the $\oplus$ denotes the *XOR* Boolean function operator.

In terms of classification, we can see that the values resulting from the application of $\oplus$ to inputs (0,0) and (1,1) are in class 0; while the outputs corresponding to (1, 0) and (0, 1) are in class 1. The problem here is that inputs in class 0 cannot be linearly separated from inputs in class 1. From Fig. 1 we can see that the 4 values of the boolean function cannot be separated by a line; the inputs [(1, 0), (0, 1)] are in opposite corners but they belong to class 1. The same occurs with the input [(0, 0), (1, 1)].

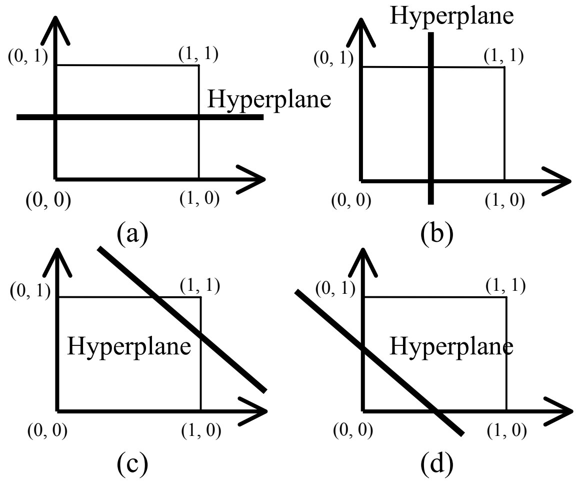

Fig. 1. The XOR problem as an example of a nonlinear separable pattern problem.

Using SVMs we illustrate how this kind of problems may be successfully tackled, as we discuss in what follows.

3 SVM and GA

3.1 Support Vector Machines

As mentioned above, SVMs were pioneered by Vapnik and coworkers and are based in statistical learning theory [14][15]. SVMs have been successfully used for pattern classification in many practical applications [16][17][18][19][20]. SVMs can also be used for solving nonlinear regression models. The idea of SVM is to find an hyperplane that separates two sets of data in order to adequately classify their elements. The nearest points on both sides of the separating hyperplane are called the support vectors (SVs). The smallest distance between any two SVs is called the margin of separation; the hyperplane is known as the optimal separating hyperplane (OSH) if the margin of separation is maximized.

In order to find the OSH, a SVM follows the principle of structural risk minimization, where a good generalization capability is achieved when the smallest VC (Vapnik-Chevonenkis) dimension and training error rate are selected.

It is important to mention that the specification of a SVM problem depends of the separable characteristic of data, i. e. in the case of separable patterns the training error rate is zero, but for nonseparable patterns there is a tradeoff between complexity of the machine and the number of nonseparable points [21]. Parameter *C* is used to control the tradeoff in pattern classification problems. Traditionally, the value of this parameter is selected by the user. The problem of nonseparable patterns is illustrated in Fig. 2. The so-called primal problem specification of SVM for nonseparable patterns is as follows:

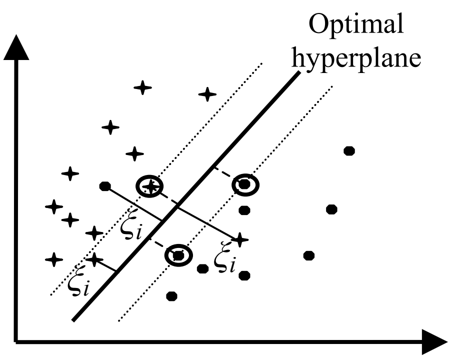

Fig. 2. Classification of data by SVM for nonlinear separable patterns.

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

In the figure, the circled points are the support vectors because by considering those points the margin of separation between them is maximized. As can be noted in the figure, the slack variables ξ_i represent the size of misclassification and the objective of the primal problem is to find the values of w and ξ_i that minimize the error of classification and the training error simultaneously.

In order to achieve a solution for nonlinear separable patterns it is necessary to work in an alternate feature space, rather than the original input space. This can be done if we select an inner-product kernel $K(x,x_i)$ that is valid only if it satisfies Mercer´s theorem [22]. The election of a kernel allows the SVM to construct a decision surface that is nonlinear in the input space but whose image in the feature space is linear. This is so because the kernel constitutes a mapping of a data set from the input space into a feature space. The SVM dual problem clearly illustrates this fact. It is formulated as follows:

$$
\begin{aligned}
&Max\ \mathrm{Q}(\alpha)=\sum_{i=1}^{N}\alpha_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j d_i d_j K(x_i,x_j)\\
&\text{subject to}: \qquad\qquad (2)\\
&\sum_{\mathrm{i}=1}^{\mathrm{N}}\alpha_{\mathrm{i}} d_i = 0\\
&0\leq\alpha_i\leq C \quad \text{for } i=1,2,\ldots,N
\end{aligned}
$$

The advantage of working with the dual problem is that it is only necessary to find the Lagrange multipliers to maximize the objective function. Otherwise, we would require to select a set of slack variables, the vector of weights and the value of *b* of the primal problem. We, therefore, solve the dual form for the XOR problem. However, we introduce the use of a GA to train the SVM. The kernel selected for solving this problem was $\left(\mathrm{x}^{\mathrm{T}}\mathrm{x}_{\mathrm{j}}+1\right)^{\rho}$ and the *C* parameter was specified, consecutively, in two different ways: a) By directly providing it and b) Calculating it with the help of VGA.

It is important to emphasize that the primal and dual cases constitute quadratic constrained optimization problems. It is because of their inherent complexity that classical optimization techniques are not, in general, applicable and we must resort to a GA to train the SVM.

3.2 Genetic Algorithms

GAs have been successfully designed to solve numerical optimization problems. Even though GAs were designed to solve unconstrained optimization problems, they can be adapted to tackle the constrained cases [23] as we explain below.

To use GAs one has to consider a set of issues which must be determined for the proper functionality of the algorithm: genome representation, fitness function, initial population, selection method, genetic operators for crossover and terminating criteria [24][25] among others. Our selection of these parameters is explained below.

The first step is the selection of the population's size. In this work we considered a population of size P = 200. The initial population was randomly generated. Weighted binary fixed point representation was used. Each individual represents a Lagrange multiplier (α_i, i=1,...,4) for the dual SVM problem. Every variable is to be expressed in fixed point format with one sign bit (0→+; 1→-), 1 integer bit and 20 decimal bits. With this representation $-2+2^{-20} \leq \alpha_i \leq +2-2^{-20}$. A fixed point format has been used because it is an excellent representation in constrained optimization problems [26]. Fig. 3 illustrates the form of this representation.

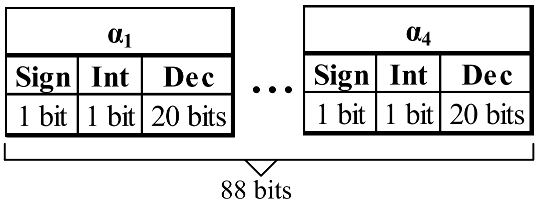

Fig. 3. Fixed point representation

With this representation, the genome's size is 88. When *C* was genetically determined, it was also included in the genome with a similar format. In those cases the size of each genome is 110 instead of 88.

Once the initial population (of size P) is generated, Vasconcelos´s model is used. This model considers full elitism and deterministic coupling, as follows. The genome is considered to be a ring of size ℓ. Individuals *i* and *n-i+1* are selected. A random number is generated; if it is smaller than Pc (the probability of crossover) then a semi-ring of size $\ell/2$ is taken from each of the two parents; the resulting genomes pass on to the next population. Otherwise, the individuals are passed to the next population untouched. Uniform mutations occur with probability Pm = 0.05.

The evaluation function is constructed following the methodology of SVMs. The solutions of *XOR* problem using the dual form are 0.25, 0.083 and 0.0125 for the objective function ($Q(\alpha)$) and 0.125, 0.0417 and 0.0125 for the Lagrange multipliers (α_i ,i=1,..,4) with ρ as 2, 3 and 4, respectively. However, we must remember that the dual problem is a constrained optimization problem. For this reason, it is necessary to modify the problem in order for it to be solved with a GA. The modification consists of transforming the constrained original problem a non-constrained one. To do this, we have chosen a penalty function that has been used [27] effectively in the solution of constrained optimization problems. The penalty function is:

$$p(x) = \begin{cases} \left[K - \sum_{i=1}^{s} \frac{K}{p} \right] - f(x) & s \neq p \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where *K* is a large constant [O(10^9)], *p* is the number of constraints and *s* is the number of these which have been satisfied.

Finally, the execution of the GA is terminated after a predefined number of generations; this bound was set to 150.

4 Experiments

We designed two algorithms to solve the XOR problem employing a SVM. In the first, *C* = 1; in the second *C* is included in the genome. It is important to mention that *C* was always positive.

In both experiments, a probability of mutation of P_m=0.05 and a probability of crossover of P_c=0.9 are set. The best n individuals in the population (n is the number of individuals at the offset) are selected to survive in each generation.

For the kernel selection it is necessary to specify the ρ parameter of the polynomial learning machine. This parameter is chosen as 2, 3 or 4. When this selection is made, the dual form for the solution of XOR problem is as follows:

$$
\begin{aligned}
&Max\ \mathrm{Q}(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}(3^\rho \alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 \\
&+ (-1)^\rho 2\alpha_1\alpha_4 + 3^\rho \alpha_2^2 + (-1)^\rho 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 3^\rho \alpha_3^2 \\
&- 2\alpha_3\alpha_4 + 3^\rho \alpha_4^2) \qquad \rho = 2, 3\ or\ 4. \\
&\text{subject to:} \\
&\alpha_1 d_1 + \alpha_2 d_2 + \alpha_3 d_3 + \alpha_4 d_4 = 0 \qquad (4) \\
&0 \le \alpha_i \le C \quad \text{for } i = 1, .., 4.
\end{aligned}
$$

Fig. 4 illustrates the correct solution of the *XOR* problem. The three dimensional graph in Fig. 4 shows the effect of mapping data from the sample into a higher dimensional feature space. In this last space the *XOR* function becomes a linearly separable classification problem, i. e., the inputs (0, 0) and (1,1) may be separated from (0, 1) and (1, 0) in the feature space with a hyperplane.

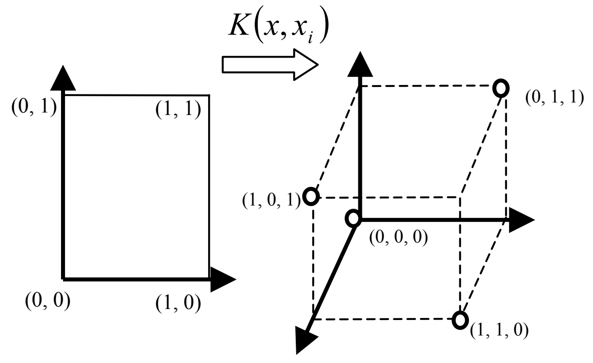

Fig. 4. The effect of kernel function in the input space.

4.1 Solution with C not included in the genome.

The results choosing *C* as 1 and consecutively selecting ρ as 2, 3 and 4 are shown in table 1. This solution was obtained after 150 generations and with a population of 200 individuals.

ρ	α_1	α_2	α_3	α_4
2	0.1250	0.1250	0.1250	0.1250
3	0.0417	0.0418	0.0416	0.0416
4	0.0127	0.0127	0.0124	0.0125

ρ	Fitness	$\sum \alpha_i d_i$
2	0.2500	0.0000
3	0.0833	0.0001
4	0.0250	0.0000

Table 1. Optimal solution with C set by hand .

The values of the α_i are shown for different selections of ρ (ρ 's values determine the degree of the polynomial kernel). The theoretical values for ρ= 2, 3, 4 are 1/8, 1/24 and 1/80 respectively. The differences between the calculated and theoretical values are, therefore, negligible. We also point out that the values for the constraints of the dual problem are reached very closely.

4.2 Solution with C included in the genome.

A second and novel experiment (since *C* is included in the genome) was also performed with good results. In fact, its results turned out to be as good as those of the previous problem. They are shown in Table 2.

ρ	α_1	α_2	α_3	α_4
2	0.1250	0.1250	0.1250	0.1250
3	0.0417	0.0417	0.0417	0.0417
4	0.0125	0.0125	0.0125	0.0125

ρ	Fitness	$\sum \alpha_i d_i$	C
2	0.2500	0.0000	0.3276
3	0.0833	0.0000	1.6028
4	0.0250	0.0000	1.0206

Table 2. Optimal solution with C included.

Notice that VGA finds the proper values for *C* and relieves the user from the task of guessing it. As before, the constraints are satisfied and the support vectors agree with the theoretical values closely. The optimal solution considers the total points of the training data (4) as support vectors (as required) because all optimal multipliers are positive and misclassification is selectively penalized by the method.

5 Conclusions

Although the XOR problem is, perhaps, the simplest prototype of a linearly nonseparable problem, it has canonically been used in the past to test the effectiveness of neural networks to the problem of classification in such cases. SVMs have been successfully applied to this kind of classification problems, but the election of parameters like *C* and the kernel function has traditionally been the user's responsibility (for previous efforts to apply evolutionary techniques see, for instance, [28]). In this paper, we have shown that a particularly efficient breed of a GA (VGA) can be used to optimize a SVM's parameters by including them in the GA's genome. It is to be expected, from theoretical considerations, that the VGA will also provide us with reasonable values for other parameters also usually put in by hand. The direction of our research will be to a) Include other relevant parameters and b) To attack more complex problems. We expect to report soon on these issues.

References:

[1] Widrow, B., D. E. Rumelhart and M.A. Lehr, Neural networks: applications in industry, business and science, *Communications of the ACM*, Vol. 37, 1994, pp. 93–105.

[2] Werbos, P. J., Neural networks, system identification, and control in the chemical process industries, *On Handbook of Intelligent Control,* Van Nostrand Reinhold, New York, 1992.

[3] Morgan, H. and H. Bourlard, Continuous Speech Recognition using Multi-Layer Perceptrons with Hidden Markov Models, *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, 1990, pp. 413-416.

[4] Chang, W. Lee and Jung-A Park, Assessment of HIV/AIDS-related health performance using an artificial neural network, *Information & Management*, Vol. 38, Issue 4, 2001, pp. 231-238.

[5] Zhang, Peter G., B. Eddy Patuwo and Michael Y. Hu, A simulation study of artificial neural networks for nonlinear time-series forecasting, *Computers & Operations Research*, Vol. 28, Issue 4, 2001, pp. 381-396.

[6] Cortes, C. and V. Vapnik, Support vector networks. *Machine Learning*, Vol. 20, 1995, 273-293.

[7] Boser, B. E., I.M. Guyon and V. N. Vapnik, A training algorithm for optimal margin classifiers, *Proc. 5th Annual ACM Workshop on Computational Learning Theory*, 1992, pp. 144–152.

[8] Haykin, S., *Neural Networks. A comprehensive foundation*, 2nd Edition, Prentice Hall, 1999.

[9] Kuri, A., A Methodology for the Statistical Characterization of Genetic Algorithms, *Lectures Notes in Artificial Intelligence,* Vol. 2313, 2000, pp. 79-89.

[10] Minsky, L. M. and S. A. Papert, *Perceptrons*, MIT Press, 1969.

[11] Haykin, S., op. cit., pp. 156-178.

[12] Haykin, S., op. cit., pp. 175-178, 282-284, 335-337.

[13] Haykin, S., op. cit. pp. 156-350.

[14] Vapnik, V., *The Nature of Statistical Learning Theory*, Springer-Verlag, 1995.

[15] Vapnik, V., *Statistical Learning Theory*, John Wiley and Sons., Inc., 1998.

[16] Vapnik, V., S. Golowich and A. Smola, Support vector method for function approximation, regression estimation and signal processing, *Adv. Neural Inform. Process. Syst.,* Vol. *9,* 1996, pp. 281–287.

[17] Schmidt, M. and H. Grish, Speaker identification via support vector classifiers. *In The Proceedings of the International Conference on Acoustics, Speech and Signal Processing,* 1996, pp. 105-108.

[18] Weston, J. and C. Watkins, Support vector machines for multi-class pattern recognition, *in: Proceedings of ESANN'99*, 1999, pp. 219–224.

[19] Drucker, H., D. Wu and V. Vapnik, Support vector machines for spam categorization. *IEEE Trans. On Neural Networks*, Vol. 10, 1999, pp. 1048-1054.

[20] Scholkopf, B., C. Burges and V. Vapnik, Extracting support data for a given task. *In proceedings, First International Conference on knowledge Discovery and Data Mining (Fayyad, U. M. and Uthurusamy, R., eds)*, 1995, pp. 252-257.

[21] Burges, C. J. C., A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery 2,* Vol. 2 ,1998, pp. 955–974.

[22] 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, pp. 415-446.

[23] Kuri, A. and J. Gutiérrez., Penalty Function Methods for Constrained Optimization with Genetic Algorithms: a Statistical Analysis, *Lectures Notes in Artificial Intelligence*, No. 2313, 2002, pp. 108-117.

[24] Goldberg, D., *Genetic algorithms in search, optimization and machine learning*, Addison Wesley, 1989.

[25] Mitchell, M., *An Introduction to Genetic Algorithms*, MIT Press, 1996.

[26] Kuri, A., *A Comprehensive Approach to Genetic Algorithms in Optimization and Learning. Theory and Applications, Vol. 1: Foundations*, Instituto Politécnico Nacional, 1999.

[27] Kuri, A. and J. Gutiérrez., op. cit, pp. 111.

[28] Samanta, B., Al-Balushi K. R. & Al-Arami S. A., Artificial neural networks and support vector machines with genetic algorithms for bearing detection, *Ingeneering Aplications of Artifitial Intelligence,* Vol. 16, 2003, pp. 657-665.