PERGAMON

AVAILABLE AT
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Neural Networks 16 (2003) 763–770

Neural Networks

www.elsevier.com/locate/neunet

2003 Special issue

Quantum optimization for training support vector machines

Davide Anguita*, Sandro Ridella, Fabio Rivieccio, Rodolfo Zunino

DIBE—Department of Biophysical and Electronic Engineering, University of Genoa, Via Opera Pia 11A 16145 Genova, Italy

Abstract

Refined concepts, such as Rademacher estimates of model complexity and nonlinear criteria for weighting empirical classification errors, represent recent and promising approaches to characterize the generalization ability of Support Vector Machines (SVMs). The advantages of those techniques lie in both improving the SVM representation ability and yielding tighter generalization bounds. On the other hand, they often make Quadratic-Programming algorithms no longer applicable, and SVM training cannot benefit from efficient, specialized optimization techniques. The paper considers the application of Quantum Computing to solve the problem of effective SVM training, especially in the case of digital implementations. The presented research compares the behavioral aspects of conventional and enhanced SVMs; experiments in both a synthetic and real-world problems support the theoretical analysis. At the same time, the related differences between Quadratic-Programming and Quantum-based optimization techniques are considered.

Keywords: Quantum optimization; Support vector machine; Quadratic-programming; Robust classification

1. Introduction

The support vector machine (SVM) (Cortes & Vapnik, 1995) is a well-known and effective method for regression and pattern classification, and often leads to outstanding performances in real-world applications. The success of SVMs mainly derives from setting up the training process so as to optimize the run-time generalization performances of the resulting classifiers.

The key feature of Vapnik's formulation (Vapnik, 1998) lies in posing the maximum-margin search process as a quadratic-programming (QP) optimization problem. In spite of the intricacies brought about by highly constrained (and often poorly conditioned) QP, effective tools are available for fast QP optimization (Chang & Lin, 2003). This has ultimately boosted the practical impact of SVM classifiers.

At the same time, a vast literature in the area of Computational Learning Theory reports the search for newer, tighter bounds to the classifiers' generalization errors. In this respect, sample-based methods that use maximal-discrepancy techniques to estimate model complexity (Bartlett, Boucheron, & Lugosi, 2002; Bartlett & Mendelson, 2002) seem to represent a promising line of research. The notably tight generalization bounds attained in (Bartlett & Mendelson, 2002) result from combining two specialized approaches: a Rademacher estimate of model complexity and an advanced, nonlinear criterion for weighting empirical classification errors.

The research presented in this paper exploits these recent achievements as a single basic approach to SVM training. The paper first demonstrates the advantage of the error-weighting criterion for SVM training: the overall classifier is made robust to peculiar distributions that might divert the conventional error-weighting criterion.

On the other hand, a crucial issue raised by the nonlinear error-weighting approach is that the SVM training process cannot any longer be formulated as a conventional QP problem. Several optimization methods exist for the general case (Fletcher, 1987; Powell, 1989), yet the lack of an efficient algorithm such as QP can turn optimization into a problem with NP complexity.

This scenario leads one to envision to exploit novel technologies for effective optimization. Quantum computing (QC) (Nielsen & Chuang, 2000) represents a promising paradigm, whose importance has increased very rapidly in the last decades, mainly for the recent definition of specialized algorithms to solve complex problems, such as large-number factorization and exhaustive search.

A basic feature that makes quantum approaches appealing to applied research is that QC involves a digital representation of processed information. This proves especially useful in training SVMs for two reasons: first, the overall problem is inherently digital in both quantum

* Corresponding author. Fax: +39-010-353-2175.
E-mail address: anguita@dibe.unige.it (D. Anguita).

0893-6080/03/$ - see front matter © 2003 Elsevier Science Ltd. All rights reserved.
doi:10.1016/S0893-6080(03)00087-X

and classical computers; secondly, the optimization process has to scan exhaustively the set of possible bit configurations in the search space.

Therefore, the paper explores the possibility of using quantum-optimization algorithms for SVM training when conventional QP techniques are no longer applicable. The effectiveness of QC-based optimization is evaluated in synthetic and real-world problems, and the performances are compared with those of a Montecarlo random-search method.

2. Error weighting for training SVM classifiers

The reason that justifies the success of the SVM model lies in its structural approach. SVM training aims to find a function capable of incurring few errors on the training sample, while featuring a promising generalization ability.

Let Φ be a mapping function from the input space X into a higher-dimensional space; then the general form for a hyperplane in the mapped space is:

$$f(\mathbf{x}) = \sum_{i=1}^{\mathrm{np}} \alpha_i y_i \Phi(\mathbf{x}_i)\Phi(\mathbf{x}) + b \tag{1}$$

where the function is expressed as a weighted sum of the input samples, $\{\mathbf{x}_i,\ i = 1,\ldots,\mathrm{np}\}$, and the α_i are positive bounded quantities. If we regard Eq. (1) as a classification surface and label by y_i the class associated with each input point $\mathbf{x}_i$, the general statement of the problem sketched above is:

$$\min_{\mathbf{w},\xi,b}\left\{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{\mathrm{np}}\xi_i\right\}$$

subject to (2)

$$y_i(\mathbf{w}\Phi_i + b) = 1 + \mu_i - \xi_i \qquad \forall i = 1\ldots\mathrm{np}$$

$$\mu_i, \xi_i \geq 0$$

where

$$\mathbf{w} = \sum_{i=1}^{\mathrm{np}} \alpha_i y_i \Phi_i \tag{3}$$

C is a constant, μ_i are used to balance the equation in the case of a correct classification, and ξ_i is an analog measure of the error on each data-point. One of the main results of Statistical Learning Theory (Vapnik, 1998) is that the first term considered in Eq. (2) is proportional to the VC-dimension, hence its minimization enhances the generalization ability of the hyperplane in Eq. (1).

It is possible to demonstrate (Fletcher, 1987) that Eq. (2) has the same solution as a constrained QP optimization problem (the dual one) with respect to the α_i; in such a problem these variables are bounded by C and linearly constrained by the following relation:

$$\sum_{i=1}^{\mathrm{np}} \alpha_i y_i = 0 \tag{4}$$

In order to find the solution of the dual problem it is crucial to meet the Karush–Kuhn–Tucker (KKT) conditions:

$$\begin{cases} \alpha_i \mu_i = 0 \\ (C - \alpha_i)\xi_i = 0 \end{cases} \qquad \forall i = 1\ldots\mathrm{np} \tag{5}$$

The formulation of the dual problem only involves the computation of the inner product of the Φ_i. The functions for which

$$k(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_j) \tag{6}$$

are called *Kernel Functions*. These functions, together with the α_i, define the SVM expansion in Eq. (1), which becomes

$$f(\mathbf{x}) = \sum_{i=1}^{\mathrm{np}} \alpha_i y_i k(\mathbf{x_i}, \mathbf{x}) + b \tag{7}$$

thus allowing a non-linear class separation.

Among the several available methods to estimate the classification error, the ones with the tightest bounds seem to be those performing a sample-based complexity estimation (Bartlett & Mendelson, 2002). This estimate involves the computation of the *Rademacher Complexity*, as follows:

$$R_{\mathrm{np}}(F) = E_{\mathrm{P}(X)}\left[E_\sigma\left[\sup_{f\in F}\left|\frac{2}{\mathrm{np}}\sum_{i=1}^{\mathrm{np}}\sigma_i f(\mathbf{x}_i)\right| \left|\mathbf{x}_i,\ldots,\mathbf{x}_{\mathrm{np}}\right.\right]\right] \tag{8}$$

where F is a class of functions mapping the domain of the input samples into $\Re$, $E_{\mathrm{P}(X)}$ is the expectation with respect to the probability distribution of the input data, and E_σ is the expectation with respect to σ_i, which are independent uniform random variables taking the values $\{+1, -1\}$.

One can regard $R_{\mathrm{np}}(F)$ as a measure of the ability of the class to which Eq. (7) belongs to classify the input samples if associated with a random class: as the fitting ability of the function increases, so does its complexity.

Eq. (8) can be used to derive the following bound to the generalization error of a classifier (Bartlett & Mendelson, 2002); this bound holds with probability $(1 - \delta)$:

$$P(y{\cdot}f(\mathbf{x}) \leq 0) = \hat{E}_{\mathrm{np}}h(y{\cdot}f(\mathbf{x})) + 2L{\cdot}R_{\mathrm{np}}(F) + \sqrt{\frac{\ln(2/\delta)}{2\mathrm{np}}} \tag{9}$$

where $\hat{E}_{\mathrm{np}}$ is the error on the input data measured through a *loss function* $h(\cdot)$ having the Lipschitz constant L, that is:

$$\hat{E}_{\mathrm{np}}h(y{\cdot}f(\mathbf{x})) = \frac{1}{\mathrm{np}}\sum_{i=1}^{\mathrm{np}} h(y_i{\cdot}f(\mathbf{x}_i)) \tag{10}$$

The loss function $h(\cdot)$ is used to shape the relative weight of the analog error as follows:

$$\xi_i = h(y_i{\cdot}f(\mathbf{x}_i)) \tag{11}$$

In the classical SVM formulation by Vapnik, the form of the $h(\cdot)$ function is:

$$h_{\mathrm{V}}(y{\cdot}f(\mathbf{x})) = \begin{cases} 0 & \text{if } y{\cdot}f(\mathbf{x}) \geq 1 \\ 1 - y{\cdot}f(\mathbf{x}) & \text{if } y{\cdot}f(\mathbf{x}) \leq 1 \end{cases} \tag{12}$$

Bartlett and Mendelson (2002) suggested the following better function to account for classification errors:

$$h_{\mathrm{BM}}(y{\cdot}f(\mathbf{x})) = \begin{cases} 0 & \text{if } y{\cdot}f(\mathbf{x}) \geq 1 \\ 1 - y{\cdot}f(\mathbf{x}) & \text{if } 0 \leq y{\cdot}f(\mathbf{x}) \leq 1 \\ 1 & \text{if } y{\cdot}f(\mathbf{x}) \leq 0 \end{cases} \tag{13}$$

which has $L = 1$ and saturates to one for any misclassified pattern. Obviously, $h_{\mathrm{V}}(u) \geq h_{\mathrm{BM}}(u) \forall u$. As an important consequence, the formulation of the loss function as per Eq. (13) inhibits the use of well-known linearly constrained Quadratic Programming algorithms.

3. A case study on the effects of outliers

We introduce a one-dimensional dataset for the purpose of illustrating the effects of the linear penalty ξ_i used in Eq. (2) to take into account possible classification errors.

The dataset is built as follows: the points belonging to one class, say, the positive one, are concentrated in the origin; the negative-labeled ones are concentrated in $x = -1$. One outlier for the latter class is also present in $x = +\lambda$ (Fig. 1). Supposing the negative class to be composed of N_1 samples plus the outlier and the positive class to be composed of N_2 samples, the dataset can be described as follows:

$$\begin{aligned} &y_n = -1 \Rightarrow x_n = -1 \wedge x_n = +\lambda \quad && n = 1, \ldots N_1 + 1 \\ &y_p = +1 \Rightarrow x_p = 0 && p = 1, \ldots N_2 \end{aligned} \tag{14}$$

Writing Eq. (2) for this specific setting and restricting our analysis to the one-dimensional case of a linear kernel (Φ is the identity) gives:

$$\min_{w,b} \frac{w^2}{2} + C(N_1\xi_1 + N_2\xi_2 + \xi_3)$$

subject to

$$\begin{cases} w - b = 1 + \mu_1 - \xi_1 \\ b = 1 + \mu_2 - \xi_2 \\ -w\lambda - b = 1 + \mu_3 - \xi_3 \end{cases} \tag{15}$$

From the definition Eq. (3) it follows that:

$$w = N_1\alpha_1 - \lambda\alpha_3 \tag{16}$$

The goal is now to find when the analog error caused by the outlier is large enough to draw the separation threshold beyond the positive class, thus causing its misclassification. The required conditions are $w = 0$ and $b = -1$. From Eq. (15) it follows:

$$\begin{cases} -b = 1 + \mu_1 - \xi_1 \\ b = 1 + \mu_2 - \xi_2 \\ -b = 1 + \mu_3 - \xi_3 \end{cases} \tag{17}$$

and from Eq. (16) we have:

$$N_1\alpha_1 = \alpha_3\lambda \tag{18}$$

The positiveness of the constraints on μ_l and ξ_i provides:

$$\begin{cases} \mu_1 = \xi_1 = \mu_3 = \xi_3 = \mu_2 = 0 \\ \xi_2 = 2 \end{cases} \tag{19}$$

and the KKT conditions require:

$$\alpha_2 = C = \frac{N_1\alpha_1 + \alpha_3}{N_2} \tag{20}$$

where the last equality follows from Eq. (4).

Using relations Eqs. (18) and (20), we rewrite the inequality constraints on α_i as:

$$\begin{cases} \alpha_3 = C\dfrac{N_2}{1+\lambda} \leq C \\ \alpha_1 = \dfrac{\alpha_3\lambda}{N_1} = \dfrac{\lambda}{1+\lambda}\dfrac{N_2}{N_1}C \leq C \end{cases} \tag{21}$$

thus the relations that allow the feasibility of the solution are:

$$\begin{cases} N_2 \leq 1 + \lambda \\ \lambda N_2 \leq (1+\lambda)N_1 \end{cases} \tag{22}$$

The present problem has been maliciously set in order to focus the reader's attention on the difference between minimizing the (integer) number of errors and minimizing a bound on the number of errors. Indeed, under the conditions in Eq. (21), the classical SVM fails to find a reasonable solution. It is straightforward to prove that by using the loss function defined in Eq. (13) the optimal solution can be

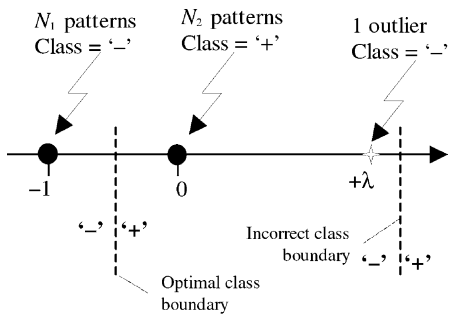

Fig. 1. Outliers might affect class-decision surfaces during SVM training.

obtained with $w = 2$ and $b = 1$ when:

$$\begin{cases} 2 \leq C(N_2 - 1) \\ 2(1 + \lambda^{-2}) \leq C(N_1 - 1) \end{cases} \tag{23}$$

Section 4 illustrates how to approach the NP-hard problem of minimizing the number of errors by the computational paradigm of QC.

4. Quantum computing for SVM training

4.1. Quantum-based representation of states

Research on QC has experienced an enormous growth in the last decades. Due to both the subject complexity and the huge amount of existing literature, the following, synthetic treatment of QC can only provide an informal hint at the current scenario.

The state of a classical computer is supported by the mutually exclusive binary quantities '0'and '1'; any consistent state of the machine has to be represented digitally. In QC, instead, a machine may exist in a 'superposition of states' (Nielsen & Chuang, 2000), provided it is allowed to evolve undisturbed. Thus one states that a single digital quantity (a 'qbit'), ψ, can take on both states '0' and '1' at the same time. Each state is characterized by a complex number giving the probability amplitude of the state:

$$|\Psi\rangle = \sum_{i=1}^{n} \omega_i |\Psi_i\rangle \tag{24}$$

where Dirac's conventional 'ket' notation is used for the state qbits ψ_i; the probability amplitudes, ω_i, must satisfy:

$$\sum_{i=1}^{n} \|\omega_i\|^2 = 1 \tag{25}$$

The property by which the internal status of a system is described in a nontrivial probabilistic fashion is called *coherence*; according to quantum-mechanics laws, a machine can persist in a coherent state only if it does not interact in any way with the external environment. Incidentally, such a requirement is probably the major obstacle to the physical realization of quantum computers (Di Vincenzo, 1995).

As inspecting the internal system state inevitably involves some physical interaction with the system itself, an important consequence of the indetermination principle is that any measurement operation on a quantum computer disrupts coherence.

This property also allows indirect interpretation of Eqs. (24) and (25). Assume that only one qbit encodes the overall system state. The possible outcomes of a measurement operation (called 'eigenstates') can just be $|0\rangle$ and $|1\rangle$, and the state itself is written as $|\Psi\rangle = \varpi_0|0\rangle + \varpi_1|1\rangle$ according to Eq. (24). Thus $\|\varpi_0\|^2$ and $\|\varpi_1\|^2$ give the probabilities that, after the measurements, the system will be found in the related states. The extension of this property to the multi-qbit case is straightforward by combining all possible elementary eigenstates into a quantum register $|\Psi\rangle$; each binary configuration will exhibit an associated probability.

Therefore, the crucial representation advantage in QC is that for a system with n state bits, a coherent quantum computer just uses n qbits to represent the whole system state, whereas, a classical computer requires $N = 2^n$ locations for storing all possible configurations. The fact that a quantum computer can hold simultaneously and linearly the exponential number of states of a classical machine seems to hint at the fact that QC might tackle NP-problems by providing P-complex solutions.

4.2. Quantum algorithms

Such a powerful representation paradigm also called for specific computational paradigms and algorithms. A quantum computer transforms the internal, usually superposed states; thus its functioning can be formally described by a set of state transformations, and every quantum operator ('gate') can be analytically expressed in the form of a matrix.

The basic constraint on any admissible quantum gate is the unitary nature of the associated matrix; this derives from the coherence requirement, and ultimately relates to the possibility of reversible computation (Deutsch, 1985).

From the most general perspective that is adopted in the present treatment, within the above constraints most quantum-computing algorithms follow a few common steps.

First, one initializes all qbit registers into a predetermined classical state (typically, '0'). In the specific case of quantum approaches to optimization problems, qbits store the optimized variables. Applying the Walsh-Hadamard operator (Nielsen & Chuang, 2000) prepares the initial state of the quantum machine as

$$|\Psi_0\rangle = \frac{1}{\sqrt{2^n}} \bigotimes_{i=1}^{n} (|0\rangle + |1\rangle) \tag{26}$$

where $\otimes$ denotes the state direct product (Nielsen & Chuang, 2000). Thus $|\Psi_0\rangle$ comprehends all possible states, which are equally probable.

Then one feeds the cost-function algorithm with $|\Psi_0\rangle$, thus obtaining a superposition of all possible cost values. The computing machine can work out the cost-function values by using the same logic circuitry that supports classical computers. Indeed, basic results from quantum theory ensure that any computation feasible on digital Turing Machines can also be performed on Quantum Computers (Deutsch, 1985). Such a computational approach points out the basic advantage deriving from superposed states: a Quantum Computer can explore all of the cost configurations in a single computational run. By contrast,

a classical computer would face an exponential computational overhead.

In the subsequent (possibly iterated) steps, unitary operators modify quantum registers, and alter the probabilities of the various states. The art in developing quantum algorithms thus consists in cleverly designing proper unitary operators, such that the sought-for solutions progressively emerge as the most likely states in the registers of the quantum machine.

The last step of quantum algorithms involves inspecting quantum registers. This operation is highly critical, as the measurement operation lets register qbits collapse into classical binary quantities, but also projects the quantum state into a new state and actually loses coherence and the representational power conveyed by superposition. Due to these critical issues, the timing of the measuring phase must be carefully designed and selected during the algorithm progress.

4.3. Quantum optimization

Grover's algorithm (Grover, 1996) tackles the (NP-complete) problem of searching an input string within an unsorted database. It is one of the best-known QC techniques proposed so far, and exhibits all of the above features. At start-up, Grover's approach requires a single computation of the matching function on the superposition of all equally probable input entries prepared as per Eq. (26). Then an iterative process makes the sought-for input entry emerge progressively from among other entries. The process uses a series of special transformations of the quantum-machine state that are repeated for a finite number of steps.

The repetitions involved in Grover's algorithm proceed at the internal clock rate of the quantum machine, and cannot be compared with the conventional number of iterations of optimization procedures. For an input string including n bits and $N = 2^n$ possible states, the number of repetitions grows as $O(\sqrt{N})$.

Thus Grover's method does not break the NP-completeness barrier, yet it has represented a popular basis for a large variety of algorithms. For the purposes of the research presented here, a quantum method for minimization is described in (Durr & Hoyer, 1996). The number, R, of repetitions for that algorithm to convergence is given by:

$$R = 22.5\sqrt{N} + 1.4\lg^2 N \sim 22.5\sqrt{N} \tag{27}$$

Theory shows that a single run of the minimization algorithm (Durr & Hoyer, 1996) finds out a valid solution with probability at least 1/2. Therefore, to increase the success probability one just applies the basic algorithm in a series of $k > 1$ different runs. With this approach, the total number of repetitions, i.e., the computational cost for the quantum machine, is:

$$R^{(k)} = k \cdot R \tag{28}$$

and the associated probability of success becomes:

$$P_q^{(k)} \geq 1 - \frac{1}{2^k} \tag{29}$$

To sum up, to use QC for SVM training first requires one to express the SVM model in a digital representation (including both the free parameters and the cost-function computation). The set of digital parameters to be optimized are stored in as many associated qbits, that are prepared in an initial, equally probable superposition Eq. (26). Feeding the initial state to the cost-function supports an exhaustive scanning of the cost space. The resulting optimization problem is eventually solved by the minimization algorithm, whose quantum computational cost and success probability are given by Eqs. (28) and (29), respectively.

Now one might want to analyze the specific advantages of the quantum approach. The comparison involves the quantum-minimization algorithm and a Montecarlo random-search process, which represents the ultimate resort for NP-complete problems in the lack of effective optimization techniques.

Assume that the minimization problem has M different solutions; in a search space with N possible configurations, the probability of success of a Montecarlo search after r test iterations is expressed as:

$$P_m^{(r)} = 1 - \left(1 - \frac{M}{N}\right)^r \tag{30}$$

To compare the two optimization methods on a fair basis, one should try a Montecarlo search for $R^{(k)}$ times; the quantum approach exhibits an advantage whenever $P_m^{(R(k))} < P_q^{(k)}$; by using Eqs. (29) and (30) one easily obtains:

$$22.5M < \sqrt{N} \ln 2 \tag{31}$$

Interestingly, the expression (31) does not depend on the number, k, of test runs of the quantum algorithm. Rather, the resulting condition exclusively depends on the specific problem complexity, involving the number of solutions and the search-space extension. In particular, one observes that condition (31) is most often fulfilled in common practice, as the measure of the number of optimized bits is usually much larger than that of the problem minima.

As a consequence, the applicative interest in QC paradigms also stems from the fact that their relative effectiveness increases with the difficulty of the specific problem at hand.

5. Experimental results

The different effects of using the loss functions (12) and (13) have been studied experimentally by testing a linear classifier on two non-linearly separable datasets. Each testbed exhibits a different displacement of the misclassified patterns from the separating plane; thus one can analyze how such patterns do affect the eventual classifier

configuration in two different cases: in the presence of outliers and when errors lie close to the separation surface.

To inspect the practical effectiveness of a quantum digital implementation, one defines the Digital Cost Function (*DCF*) as per Eq. (2), where the loss function is $h_{\mathrm{BM}}(u)$:

$$\mathrm{DCF} = \frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{\mathrm{np}} h_{\mathrm{BM}}(y_i(\mathbf{w}\cdot\mathbf{x}_i + b)) \tag{32}$$

In order to define the number, M, of acceptable solutions of the optimization process, those associated with a DCF value lower than 101% of the best achievable cost have been considered as 'good solutions'.

5.1. Iris dataset

This testbed is the non-linearly separable version, including the 'Virginica' and 'Versicolor' classes respectively, each represented by 50 2-D patterns. First, a linear SVM was trained by using the algorithm (Chang & Lin, 2003), featuring the classic $h_V(u)$ loss function with an error penalization $C = 1000$; the resulting separating plane scored six errors.

Then, a digital SVM was trained under the same parameter $C = 1000$ to analyze the performance of a linear classifier implementing the loss function $h_{\mathrm{BM}}(u)$. The weights and bias were coded as 10-bit. Such a digital SVM scored three errors; Fig. 2 shows the separating planes resulting from the two different implementations.

Although the dataset does not contain any outlier, the solutions differ in terms of both the number of misclassified patterns and the weight set $\mathbf{w}$: the only errors lie within the overlapping zone between the two classes. This difference can be explained by examining the two terms composing Eq. (2).

The SVM implementing $h_V(u)$ tends to maximize margin and the related errors lie as close as possible to the separation surface; indeed, the specific loss function linearly penalizes the distance from the separating surface. As a result, the solution minimizes complexity and weakly penalizes each of the six errors, which are embedded into the margin itself.

By contrast, the lower error scored by the digital implementation derives from the non-linear weighting of errors that lie away from the margin: in this case, the loss function saturates no matter how distant the error is, thus allowing the better decision surface.

As to quantum optimization, with the notation adopted in Section 4 one has $N = 1,073,741,824$ and $M = 1$; the probability of finding an acceptable solution is $M/N = 9.3 \times 10^{-10}$, thus meeting condition (31).

In a different, intriguing experiment, one might modify the original dataset to introduce the presence of outliers. To this purpose, one flips the classes of two patterns lying far from the separation surface (this simulates the presence of noise in the measurement of the targets).

The resulting artificial dataset was used to train the two SVMs under Eqs. (12) and (13), yielding the classifiers as per Fig. 3. The graph confirms the robustness of $h_{\mathrm{BM}}(u)$, as the flipped targets do not alter the position of the separation line, whereas the classic SVM classifier is significantly affected.

5.2. Breast cancer dataset

This dataset includes 699 patterns with 9 features; each pattern is classified as benign or malignant. After excluding 16 samples featuring missing values, all features were normalized in the range $[-1,1]$.

Previous results (Drago & Ridella, 1998) show that the information carried by the whole dataset is well represented by just two features, i.e. features 6 and 8. Therefore, the present analysis reduces the Breast Cancer dataset to a 2-D problem, which allows useful information to be drawn from a visual inspection. A classical SVM Eq. (7) with a linear

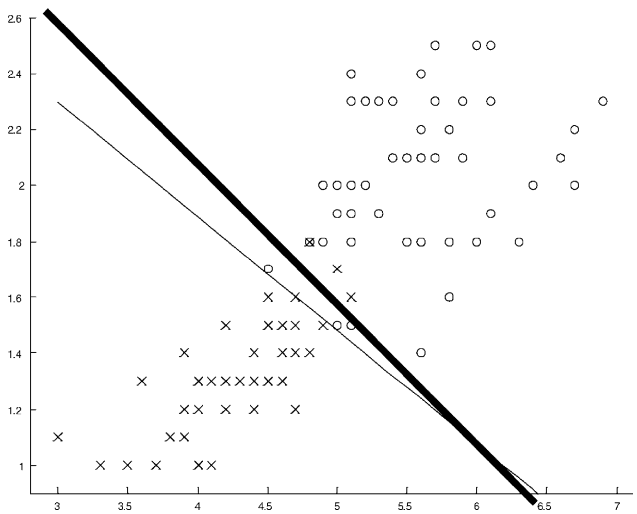

Fig. 2. Iris dataset. The thick-line separation is obtained by a classical SVM implementation while the thin-line represent the solution found by a digital implementation having $h_{\mathrm{BM}}(u)$ as loss function.

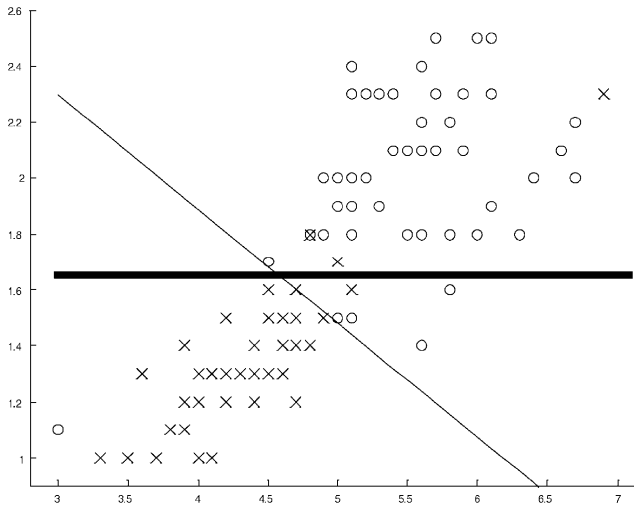

Fig. 3. Modified Iris dataset. The thick-line separation is obtained by a classical SVM implementation; the thin line represents the solution found by a digital implementation having $h_{\mathrm{BM}}(u)$ as loss function.

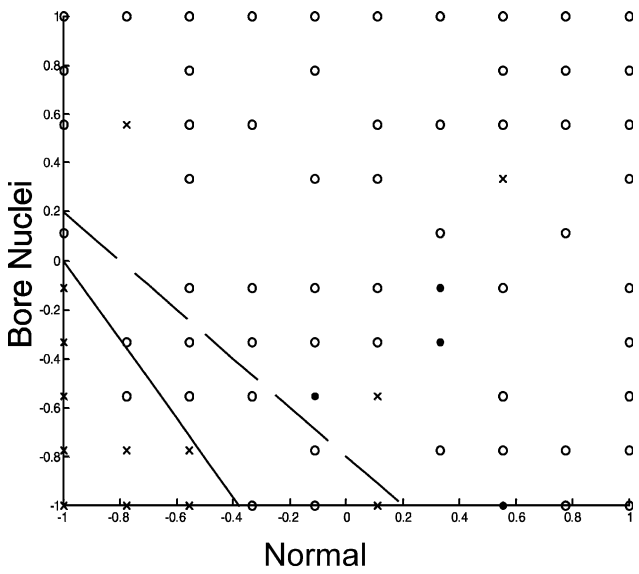

Fig. 4. BCancer. In the ' × ' marks, patterns malignant cases exceed benign ones (viceversa for circles). Filled circles indicate tie cases. The dashed-and solid boundaries relate to a classic and digital-cost SVM, respectively.

kernel and $C = 1000$ scored an empirical error rate of 5.124%; the separation surface is shown in Fig. 4. This result was obtained by using the algorithm (Chang & Lin, 2003), and had to be compared with that obtained by using the loss function $h_{\mathrm{BM}}(u)$.

To that purpose, the analysis considered a digital SVM implementation, whose weights and bias were coded as 8-bit values. The optimization process minimizes DCF with respect to $\mathbf{w}$ and b; Fig. 4 presents the class-separation boundary associated with the minimum of Eq. (32), scoring a misclassification error of 4.25%. An histogram-based analysis of the results (Fig. 5) shows that about 6% of the DCF values are less than twice the minimum of the DCF itself.

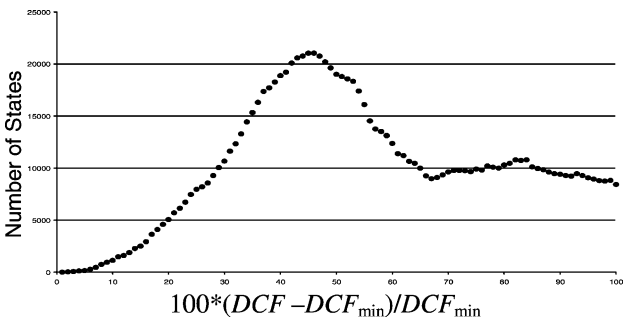

Fig. 5. Breast cancer dataset. Histogram of the values for the digital cost function.

When implementing the digital optimization problem on a quantum machine, in the notation adopted in Section 4, one has $N = 16,777,216$, $M = 6$. The latter value was obtained by keeping valid solutions those which did not displace more than 1% from the minimum of DCF. Thus the probability of finding a good solution is $M/N = 3.6 \times 10^{-7}$, which fulfils condition Eq. (31).

6. Conclusions

The power and effectiveness of SVMs in their original formulation as a general pattern-processing paradigm is not being questioned. However, computer based implementations that take into account the digital nature of represented quantities, and refined formulations that shrink generalization bounds can invalidate the applicability of efficient QP training algorithms.

Scanning a digital bit space without a gradient-based method can turn the optimization task into an NP-complete exhaustive-search problem. This ultimately shifts the interest toward novel and promising computational paradigms such as QC.

The main *pro* for such an approach derives from the principle of quantum superposition of states, which enables an inherent parallelism in information processing that is not achievable by classical computing machinery. On the other hand, two *cons* seem to hold back an excessive enthusiasm for quantum approaches: first, quantum machinery is reportedly not a mature technology yet, hence one should not expect to have quantum optimization available for practical purposes in the near future. Secondly, no proof has been given so far that QC can break the NP-completeness barrier in a real, interesting problem.

In view of these issues, a basic conclusion might anyway be drawn from the research presented in this paper: QC can yet prove effective for an important problem such as training SVMs for digital implementations. The reported analysis also shows that the computational benefits conveyed by quantum optimization increase when the problem complexity increases.

The presented simulations on a real-world problem open new vistas over the possibility of tuning SVM classifiers that are apt to direct and effective realizations in digital circuitry.

References

Bartlett, P., Boucheron, S., & Lugosi, G. (2002). Model selection and error estimation. *Machine Learning*, *48*(1–3), 85–113.

Bartlett, P., & Mendelson, S. (2002). Rademacher and Gaussian complexities: risk bounds and structural result. *Journal of Machine Learning Research*, *3*, 463–482.

Chang, C. C., Lin, C. J., (2003). *LIBSVM: a library for Support Vector Machines*. Retrieved on Mar'03 from: http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf

Cortes, C., & Vapnik, V. (1995). Support vector networks. *Machine Learning*, *20*, 273–297.

Deutsch, D. (1985). Quantum theory, the church-turing principle, and the universal quantum computer. *Proceedings of the Royal Society of London*, *A400*, 97–117.

Di Vincenzo, D. (1995). Two-bit gates are universal for Quantum Computation. *Physics Review A*, *51*, 1015–1022.

Drago, G. P., & Ridella, S. (1998). Pruning with interval arithmetic perceptron. *Neurocomputing*, *18*(1–3), 229–246.

Durr, C., Hoyer, P., (1996). *A Quantum Algorithm for Finding the Minimum*. Retrieved March, 4, 2003 from: http://arxiv.org/abs/quant-ph/9607014.

Fletcher, R. (1987). *Practical methods of optimization* (2nd ed). New York: Wiley.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. *Proceedings of the 28th Annual ACM Symposium Theory of Computing*, 212–219.

Nielsen, M. A., & Chuang, I. L. (2000). *Quantum computation and quantum information*. Cambridge, GB: Cambridge University Press.

Powell, M. J. D., (1989). *TOLMIN: A fortran package for linearly constrained optimization calculations*. DAMTP Report NA2, University of Cambridge, England.

Vapnik, V. (1998). *Statistical learning theory*. Chichester, GB: Wiley.