Neurocomputing 72 (2009) 2824–2832

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Finding optimal model parameters by deterministic and annealed focused grid search

Álvaro Barbero Jiménez, Jorge López Lázaro, José R. Dorronsoro *

Dpto. de Ingeniería Informática and Instituto de Ingeniería del Conocimiento, Universidad Autónoma de Madrid, 28049 Madrid, Spain

ARTICLE INFO

Available online 16 April 2009

Keywords:
Grid search
Optimal parameters
Annealing
CMA-ES
Evolutive algorithms

ABSTRACT

Optimal parameter model finding is usually a crucial task in engineering applications of classification and modelling. The exponential cost of linear search on a parameter grid of a given precision rules it out in all but the simplest problems and random algorithms such as uniform design or the covariance matrix adaptation-evolution strategy (CMA-ES) are usually applied. In this work we shall present two focused grid search (FGS) alternatives in which one repeatedly zooms into more concentrated sets of discrete grid points in the parameter search space. The first one, deterministic FGS (DFGS), is much faster than standard search although still too costly in problems with a large number of parameters. The second one, annealed FGS (AFGS), is a random version of DFGS where a fixed fraction of grid points is randomly selected and examined. As we shall numerically see over several classification problems for multilayer perceptrons and support vector machines, DFGS and AFGS are competitive with respect to CMA-ES, one of the most successful evolutive black-box optimizers. The choice of a concrete technique may thus rest in other facts, and the simplicity and basically parameter-free nature of both DFGS and AFGS may make them worthwile alternatives to the thorough theoretical and experimental background of CMA-ES.

1. Introduction

Successful data mining applications usually rely on the appropriate choice of several parameters for the modelling paradigm that is to be used. A very well-known case is that of multilayer perceptrons (MLPs; [1]), where parameters to be chosen are the number of hidden layers and of units in each layer, the learning rate and momentum factor for gradient descent learning or the penalty factor for weight regularization. Other examples are support vector machines (SVMs; [2,3]), where a margin slack penalty factor has to be chosen for nonlinearly separable models, to which one has to add the kernel width if Gaussian kernels are to be used or the value of the ε-insensitivity parameter in SV regression. However, there may be other reasons that force the selection of modelling parameters. For instance, some sort of input preprocessing is quite often needed, such as the selection of the percentage of variance to be retained if principal components are used for input dimensionality reduction or, in time series applications, the number of delays or the modelling window size to be used.

Some ad-hoc approaches can be applied in concrete settings. For instance, leave-one-out error estimation can be used to derive parameter-dependent error bounds for SVMs [4] which, in turn, make it possible to find optimal SVM parameters by gradient descent on the resulting error model. On the other hand, a relatively standard technique (see [5], Chapter 14) in the design of experiments field is to test the model performance in a few points of the parameter space and to fit a first or second order empirical error surface over the results obtained, which is then taken as an approximation to the real underlying error landscape. If it exists, this surface's minimum is used to provide optimal model parameters.

In any case, the previous situations are the exception rather than the rule. More often (and quite particularly for hybrid systems), several rather different and competing modelling techniques are to be used so that an adequately chosen combination offers less variance than that of its individual components. In such a situation one cannot count on a direct knowledge of the effectiveness of each individual model for the problem at hand and it is thus clear that no general parameter setting procedure exists other than some kind of search on the parameter space. There are then two extreme approaches: a more or less exhaustive, deterministic parameter search [6,7] and, on the other hand, a stochastic metamodel search, typically using genetic algorithms [8,9] or genetic programming [10].

Deterministic parameter search is usually done over a grid obtained by the discretization of each individual parameter's allowable range. This discretization appears naturally for integer

* Corresponding author. Tel.: +34 91 497 2329; fax: +34 91 497 2334.
E-mail address: jose.dorronsoro@uam.es (J.R. Dorronsoro).

0925-2312/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.09.024

valued parameters (such as, for instance, the number of hidden layers or units of an MLP) and, for a continuous parameter α in a range $[a,b]$, a resolution limit δ is fixed and one explores the $\{a, a+\delta \cdots a+H\delta\}$ parameter values, where $H=(b-a)/\delta$. We may assume that $H=2^K$ and call K the depth of the search. An obvious way of exploring the resulting grid $\mathcal{G}$ associated to an M parameter model would be to perform a linear search over all the grid's parameter values. However, the complexity of essentially any model selection procedure is determined by the number of the different models to be built (in our classification examples the fitness function will be the model errors on a validation subset) and here its cost would be $(1+2^K)^M \simeq 2^{KM}$, which rules out its application outside very low values of M and K.

The simplest way out of this is to introduce the possibility of a random evolution in the search procedure. A first example somewhat related to grid search is to fill the parameter space using uniform experimental design [11], where a number L of patterns is chosen so that the square norm discrepancy between their empirical cumulative distribution and that of the uniform distribution is small. Another widely used option is to stochastically explore the parameter space in an evolutionary setting. The well-known (μ,λ) covariance matrix adaptation-evolution strategy (CMA-ES), proposed by Hansen and Ostermeier [12,13], is one of the most effective black-box random optimizers. Briefly, (μ,λ) CMA-ES produces from a population $X_l^t, 1 \leqslant l \leqslant \mu$, of μ individuals a number λ of new individuals with genotypes $X_l^{t+1}=m^t+\xi_l^t$ where m^t is the mass centre of the previous population and the perturbations ξ_l^t are independent realizations of an M dimensional Gaussian distribution $\mathcal{N}(0,C^t)$. The μ offspring with the best fitness are then selected to form the new population to be used at step $t+1$. Moreover, at each step the covariance matrix C^t is adaptively updated in such a way that the probability of applying again previous steps that lead to larger gains is maximized. The complexity of CMA-ES based model selection is clearly λ times the number of generations explored.

While being in general quite effective, these approaches are not problem-free. For instance, finding directly the appropriate space filling points in uniform design (UD) is a difficult endeavor. Tables exist that provide good preselected values, but the number of their points may be too small if a moderate to large number of parameters has to be set. On the other hand, CMA-ES is a rather complicated and difficult to parametrize and implement procedure. Moreover, it must start from a single point in parameter space and its exploration may therefore be less thorough.

As a simpler alternative, we shall propose in this paper two grid based procedures in which starting from the outer grid points, successively narrow the search to smaller half-size grids centred at the point giving the best fitness value in the previous iteration. Because of this we will term them focused grid searches (FGS). The first one is a deterministic FGS (DFGS) for which we will begin by discussing a simple general version that requires $K3^M$ model constructions for a grid depth of K and an M parameter model; we will also show how to refine it to achieve an extra $K+1$ depth precision while requiring the same number $K3^M$ of model trainings in the best case and $K(3^M+2^M)$ in the worst case. Notice, however, that the exponential cost in M remains and to alleviate it we shall also introduce an evolution strategy on the previous focused grid search, in which we will only consider a fixed number Λ of points instead of the 3^M points used in DFGS. These points will be selected through a simulated annealing-like procedure and the one giving the best fitness will be the centre of a new half-size grid; we shall call the resulting procedure annealed FGS (AFGS). While the complexity of DFGS is essentially fixed once we choose the grid depth K, we can control it in AFGS by fixing the number Λ of outer grid points to be randomly examined. Thus, for a K depth search, AFGS will require at most $K\Lambda$ model constructions and its cost will then be comparable to that of CMA-ES whenever the latter performs more than $K\Lambda/\lambda$ generations.

This paper is organized as follows. In Section 2 we will briefly review standard linear grid search, uniform design parameter value selection and, in more detail, the (μ,λ) CMA-ES algorithm that we will use it for comparisons with our deterministic and annealed FGS procedures. We will present our DFGS and AFGS proposals in Section 3 and we will compare them against CMA-ES in Section 4 over several classification problems which will be solved using MLPs and SVMs. MLPs will have a standard 1-hidden layer structure and will be trained in a batch setting by conjugate gradient minimization. We shall train SVMs for classification using the MDM algorithm [14] with quadratic margin slack penalties and Gaussian kernels. We will consider two comparison settings. First we will deal with what we may call "small parameter set" problems. By this we mean selecting the optimal number of hidden units and the weight decay parameter for MLPs and the Gaussian kernel width $2\sigma^2$ and penalty factor C in SVMs; that is, we will select just two optimal parameters for both MLPs and SVMs. On the other hand, we shall also consider "large parameter set" problems for SVMs, where we will seek, besides the penalty factor C, the best parameters of an anisotropic Gaussian kernel. The number of parameters is now $1+D$, with D the pattern dimension in the original feature space. This makes DFGS too costly and in this second situation we shall just compare AFGS and CMA-ES searches. (We observe that in [15] the anisotropic SVM parameter estimation problem is considered applying first a preliminary grid search to obtain an initial $2\sigma^2$ choice and tuning it then using CMA-ES.)

As our results in Section 4 will illustrate, there is no clear winner among the various approaches considered here in any of the parameter number scenarios, and the choice of one particular technique may rest on other facts. For instance, while practical only for models with few parameters, DFGS avoids the parametrization required by stochastic searches and AFGS, with few and very simple parameters, can be used over more complex models. On the other hand, while CMA-ES has undergone a very thorough theoretical and experimental analysis and good implementations in several programming languages are available in [16], it is still a fairly complex procedure that does not allow for an easy autonomous implementation and whose own parametrization is difficult. In contrast, DFGS and AFGS are procedurally much simpler and, thus, easier to implement. This paper will end with a brief discussion and conclusions section.

2. Previous work

In this section we will briefly describe standard linear grid search, establishing in part the notation to be used in the next section, and, then, review the application of uniform design to optimal parameter selection and, in more detail, the (μ,λ) CMA-ES procedure that we shall use in our experimental comparisons.

Linear grid search is the simplest (but also costliest) way to find optimal model parameters. A grid structure lends itself naturally to search for discrete parameters while continuous parameters must be discretized, typically by setting a parameter range and a minimum resolution level. Assuming for simplicity all parameters $\alpha_i, i=1,\ldots,M$, to be continuous, recall that we will discretize them to values in a range $\{a_i, a_i+\delta_i \cdots a_i+H\delta_i=b_i\}$ where we assume the same $H=2^K$ for all parameters and $\delta_i=(b_i-a_i)/2^K$. An exhaustive search would perform a number $1+2^K$ of model evaluations on each of the M parameter dimensions and its cost in number of models to be built would then be a prohibitive $(1+2^K)^M \simeq 2^{KM}$.

A simple way to lower this is to consider just a moderately small number of points that are randomly selected. Uniform design is such a procedure to obtain random points that, in particular, has been used in [11] for SVM σ^2 and C parameter selection. The basic idea in UD consists of evaluating the model at hand in a set of points that are uniformly scattered in the parameter range, in a quasi-Monte Carlo fashion. In order to produce a uniform scattering, one possible approach is to minimize the L_2-discrepancy, a measure of the difference over the search domain between the cumulative uniform distribution function and the empirical cumulative distribution function of the generated points. Thus, by minimizing it, the uniformity of the generated points is closer to the ideal one. Nevertheless, finding UDs with minimum L_2-discrepancy is an NP-hard problem, so this measure is modified to the so-called centred L_2-discrepancy, which is a stricter measure that also considers the uniformity of the scattering over several projections of the search domain [17]. Once the search region has been fixed, one can use predefined tables for the UD centred L_2-discrepancy available in [18] to carry out the selection of the points to be examined.

Although the results in [11] show that UD produces competitive parameter choices, there are some problems with this approach. In particular, for moderate to high numbers of parameters, the number of values in the predefined tables may be too small, so that the points are indeed uniform, but also too sparse. This would then require to set up a uniform point generation procedure, something which may turn out to be quite difficult. An evolutionary algorithm such as the (μ,λ) CMA-ES method, which is described next, would work better in this situation.

CMA-ES (see [12,19] for more details) selects at generation g the μ patterns $x^g_{i:\lambda}$, $1\leqslant i\leqslant\mu$, with the best fitness among a population of $\lambda>\mu$ vectors x^g_j. These $x^g_{i:\lambda}$ are ordered by decreasing fitness and the new generation of λ offspring is produced according to the sampling formula:

$$x^{g+1}_k = m^g + \sigma^g y^{g+1}_k, \quad k=1,\ldots,\lambda, \tag{1}$$

where the y^{g+1}_k vectors follow a $\mathcal{N}(0,C^g)$ distribution. In this equation λ and μ are external parameters. As proposed in [19], we shall take as default values $\lambda = 4+\lfloor 3\log M\rfloor$, $\mu=\lceil(\lambda-1)/2\rceil$. Notice that $\lambda>\mu$ so that population selection takes place. The vector m^g is a weighted average of the selected $x^g_{i:\lambda}$ patterns,

$$m^g = \sum_1^\mu w_i x^g_{i:\lambda}, \quad \sum w_i = 1, \quad w_1\geqslant w_2\geqslant\cdots\geqslant w_\mu>0,$$

and the quantity $\mu_{eff}=(\sum_1^\mu w_i^2)^{-1}$ is called the variance effective selection mass. For the weight values and other concrete parameter selections we refer to [16].

To obtain the new covariance C^{g+1}, CMA-ES starts at $C^0=I_M$, the M-dimensional identity matrix, and adapts the previous covariance C^g according to a combination of a rank-μ update and a cumulation strategy, as given in the following formula:

$$C^{g+1} = (1-c_{cov})C^g + \frac{c_{cov}}{\mu_{eff}}p^{g+1}(p^{g+1})^t + c_{cov}\left(1-\frac{1}{\mu_{eff}}\right)R_\mu. \tag{2}$$

Here R_μ is a rank-μ update defined as $R_\mu=\sum_1^\mu w_i y^{g+1}_{i:\lambda}(y^{g+1}_{i:\lambda})^t$, where $y^{g+1}_{i:\lambda}$ denotes now the y vectors ordered again according to decreasing fitness. On the other hand, the p^{g+1} vectors try to capture the evolution path up to generation $g+1$ and to somehow retain the most successful previous steps. Starting at $p^0=0$, p^{g+1} is obtained by exponential smoothing according to the formula

$$p^{g+1} = (1-c_{cum})p^g + \sqrt{c_{cum}(2-c_{cum})\mu_{eff}}\langle y\rangle_w, \tag{3}$$

where $\langle y\rangle_w=\sum_1^\mu w_i y^{g+1}_{i:\lambda}$. Finally, we have to set up the σ^g value that controls the update step size in (1). To do so, a new conjugate evolution path p^g_σ is constructed, starting at $p^0_\sigma=0$, and we compute σ^g at the $g+1$-generation as

$$p^{g+1}_\sigma = (1-c_\sigma)p^g_\sigma + \sqrt{c_\sigma(2-c_\sigma)\mu_{eff}}(C^g)^{-1}\langle y\rangle_w, \tag{4}$$

$$\sigma^{g+1} = \sigma^g \exp\left\{\frac{c_\sigma}{d_\sigma}\left(\frac{\|p^{g+1}\|}{D_M}-1\right)\right\}, \tag{5}$$

where $D_M=\sqrt{2}\Gamma((M+1)/2)/\Gamma(M/2)=E[\|\mathcal{N}(0,I_M)\|]$. Good general choices for the values of the c_σ and d_σ parameters, as well as those of c_{cov} and c_{cum}, are suggested in [16].

While being probably one of the best black-box optimizers, the implementation of CMA-ES can be rather complicated. In the next section we will go back to a grid search setting and propose first deterministic focused grid search, a procedure with a cost much smaller than that of a full search but still too big for problems with a large number of parameters. Our second proposal, annealed focused grid search, will alleviate this by introducing a random selection of grid points.

3. Deterministic and annealed focused parameter grid search

3.1. Deterministic focused grid search

Our starting option will be a simple parameter grid search, where we will start at a 3^M point initial outer grid $\mathcal{O}_0$ made up of vectors of the form $\alpha^0_i=c^0_i+\gamma\Delta_i$, where we assume parameter ranges $[a_i,b_i]$, $\Delta_i=b_i-a_i$, $c^0_i=a_i+\Delta_i/2$ are the coordinates of the centre c^0 of the parameter hypercube to be explored and $\gamma\in\{-\frac{1}{2},0,\frac{1}{2}\}$. In principle, the point $\tau^0=((\alpha^0_1)^*,\ldots,(\alpha^0_M)^*)$ giving a better fitness will be taken as the centre c^1 of a new smaller grid $\mathcal{O}_1$ of 3^M points of the form $\alpha^1_i=c^1_i+\gamma\Delta_i/2$ with γ as before. We will repeatedly proceed in this way, considering at each step smaller outer grids $\mathcal{O}_k$ centred at points c^k and with side length $\Delta_i/2^k$ at the i-th coordinate, and stopping after $K+1$ iterations when the final parameter precision is reached. Note that in order not to get out of bounds, a centre c^{k+1} may be moved if it lies on a border or a corner of the grid $\mathcal{O}_k$ (refer to the details in Eq. (6)).

It is straightforward to check that this simple procedure requires $(K+1)3^M$ model evaluations. Moreover, it is very easy to implement and, as we shall see in Section 3.2, lends itself quite naturally to a less costly annealed FGS. In any case, a generalization of a proposal made in [7] allows to refine it and to achieve an extra $K+1$ depth precision while requiring the same number $K3^M$ of model trainings in the best case and $K(3^M+2^M)$ in the worst case. For this two grids are used in [7]: an outer one which is built in the same way that has been described above, and an inner one whose corners are placed between the outer grid's centre and corners. The reasoning for doing this is that by using just an outer grid the parameter space may not be sampled in enough detail, so that the centering decision might not be very appropriate. The inner grid helps to sample in a more uniform way, increasing thus the probability of centering around a good point. Besides, more points can be reused, as will be shown next.

The algorithm starts as before, fixing the initial centre c^0 with coordinates $c^0_i=a_i+\Delta_i/2$ and defining the left and right limits $\alpha^L_i=c^0_i-\Delta_i/2$, $\alpha^R_i=c^0_i+\Delta_i/2$ of the initial outer grid $\mathcal{O}_1$ and, similarly, the left and right limits $\beta^L_i=c^0_i-\Delta_i/2^2$, $\beta^R_i=c^0_i+\Delta_i/2^2$ of the initial inner grid $\mathcal{I}_1$. The outer grid $\mathcal{O}_2$ is then the set of the 3^M points $(p_1,\ldots,p_M)$ where $p_i\in\{\alpha^L_i,c^0_i,\alpha^R_i\}$, and the inner grid $\mathcal{I}_2$ is then the set of the 2^M points $(q_1,\ldots,q_M)$, where $q_i\in\{\beta^L_i,\beta^R_i\}$. These initial grids are depicted in Fig. 1. Now the model is evaluated at each of the 3^M+2^M grid points, and the point τ^1 giving the best fitness is used to define the new centre c^1as

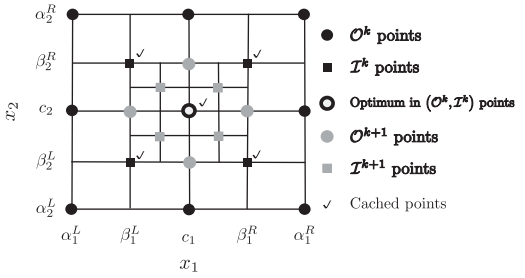


Fig. 1. Example of an iteration of the algorithm in the best case with two parameters. We show the initial configuration of the inner and outer grids $\mathcal{O}^k$ and $\mathcal{I}^k$, as well as the new grids which are built in the following iteration around the optimum and the cached points.

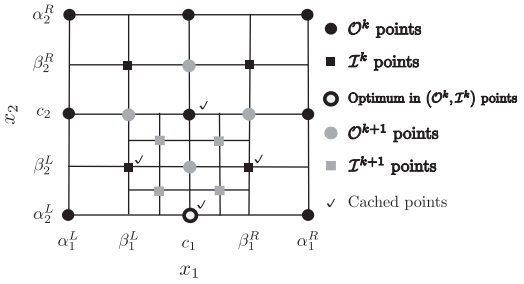


Fig. 2. Example of an iteration of the algorithm in the worst case with two parameters. We show the initial configuration of the inner and outer grids $\mathcal{O}^k$ and $\mathcal{I}^k$ as well as the new grids which are built in the following iteration around the optimum and the points that can be reused from the cache.

follows:

$$\begin{aligned} c_i^1 &= \tau_i^1 + \tfrac{1}{4}\Delta_i \quad \text{if } \tau_i^1 = \alpha_i^L \\ &= \tau_i^1 - \tfrac{1}{4}\Delta_i \quad \text{if } \tau_i^1 = \alpha_i^R \\ &= \tau_i^1 \quad \text{otherwise} \end{aligned} \tag{6}$$

(recall that we want the new point to be the centre of an inner grid). The new centre c^1 is then used to define the new outer grid limits $\alpha_i^L = c_i^1 - \Delta_i/2^2$, $\alpha_i^R = c_i^1 + \Delta_i/2^2$ as well as the new inner grid limits $\beta_i^L = c_i^1 - \Delta_i/2^3$, $\beta_i^R = c_i^1 + \Delta_i/2^3$. A new optimal parameter vector τ^2 is then found and the next grid centre c^2 is defined as before. Notice that the minimum resolution at the first iteration is $\rho_i^1 = \Delta_i/4 = 2^{K-2}\delta_i$. Thus, the procedure will perform $K-1$ iterations until it reaches the limit resolution $\rho_i^{K-1} = 2^{K-1-(K-1)}\delta_i = \delta_i$. The final vector τ^{K-1} will give the optimal parameter values. While at the first iteration $3^M + 2^M$ model evaluations have to be done to arrive at τ^1, notice that they can be suitably cached when selecting the next optimal parameter vector τ^2. This can be done at successive iterations avoiding thus model re-evaluation at the cached points that belong to the successive outer and inner grids. For instance, assuming the grid centre to be the optimal parameter vector, we do not have to re-evaluate the model at the checked points shown in Fig. 1.

We analyze next the procedure's computational cost assuming for simplicity that $\delta_i = \delta$ and also that $\Delta_i = \Delta = 2^K\delta$ for all $i = 1, \ldots, M$. The refined grid search performs $K-1$ iterations. In the first one 3^M outer and 2^M inner grid evaluations must be performed. The number of evaluations in subsequent iterations depends on the location of the optimal parameter vector τ^k. The best situation takes place, for instance, when τ^k coincides with the outer grid centre. As shown in Fig. 1 for two-dimensional parameter vectors, the next iteration can reuse all the 2^M inner grid corners and its centre, and will then require $3^M - 1$ new model evaluations. If we assume an optimal position for the new grid centre at each iteration (for instance, if it occupies each new grid's geometrical centre), the same will happen for all iterations except the last one, which only requires 2^M model evaluations (see Fig. 3). The overall number of model evaluations will then be

$$\begin{aligned} 3^M + 2^M + (3^M - 1) \times (K-3) + 2^M &= K \times 3^M + 2 \times 2^M \\ &\quad - 2 \times 3^M - (K-3) \\ &\simeq K \times 3^M, \end{aligned} \tag{7}$$

which is essentially half of that of the simpler initial deterministic FGS. The worst case for this refined DFGS is depicted in Fig. 2 for two-dimensional parameters, where the next grid iteration will require five model evaluations, one more than for the optimal two-dimensional case. To simplify the somewhat complicated general analysis, we can assume $3^M + 2^M$ evaluations being done on each iteration (this would also be the case if no previous model evaluations have been cached). The overall number of model evaluations would then be

$$(3^M + 2^M) \times (K-1) \simeq K \times (3^M + 2^M).$$

While about twice larger than the best case complexity, this worst case is again much better than exhaustive search but still too large for moderate M.

3.2. Annealed focused grid search

The simplest way to lower the number of model trainings in grid search is to replace the complete exploration of its points by a stochastic one. In this context it is more natural to consider our first grid search instead of the refined one: it is certainly simpler and the advantage of working with a finer grid would probably be lost because of the inherently coarser nature of any stochastic procedure. Thus the following discussion will be made assuming that only the 3^M point outer grids are used.

We shall apply a simulated annealing-like procedure that repeats the following annealing loop in order to optimize a fitness function:

(1) Generate randomly a new neighbour of the current parameter vector according to a certain neighbourhood function.
(2) Train a model at the new point and evaluate its fitness.
(3) Decide whether to remain at the current point or move to the new one depending on the current temperature value and the fitness of the current and new points.
(4) Decrease the temperature according to a certain annealing schedule.

Recall that the general aim of an annealing schedule is to allow for worse fitness points when annealing begins, while gradually enforcing better fitness as the algorithm progresses. More precisely, in the original simulated-annealing algorithm [20], the third loop operation considers two possibilities. If the neighbour's fitness is better than the current one, the current point is replaced by the neighbour. However, if it is worse, there is still a possibility of moving to the new point, given by a Maxwell–Boltzmann distribution, namely

$$p(F_C, F_N, T) = \min\{1, e^{(F_C - F_N)/T}\},$$

where F_C, F_N and T stand for the current point's fitness, the neighbour's fitness and the current temperature. In this formula we have $p(F_C, F_N, T) = 1$ if $F_N < F_C$. On the other hand, the temperature is gradually decreased towards zero in the fourth loop operation, so that we have $p \simeq 0$ when $F_N > F_C$ as the annealing progresses.

Starting at the centre of a given outer grid $\mathcal{O}_k$, we will iterate the annealing loop $\Lambda - 1$ times, so that only Λ out of 3^M grid points are evaluated. We define next the neighbour generating

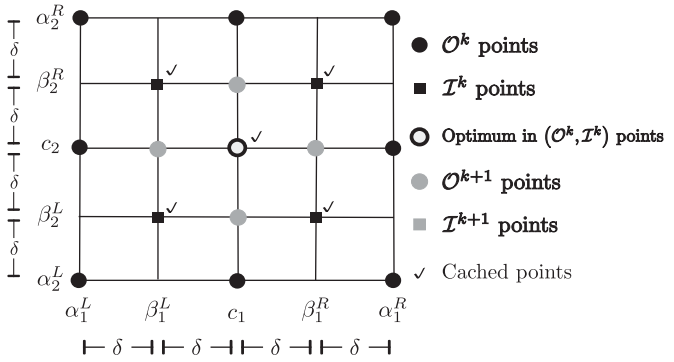


Fig. 3. The two last iterations of the algorithm with two parameters. Since we are in the precision limit δ after the approximation of the grid points to the discrete parameters more points can be reused. After the iterations shown the algorithm will stop, because a hypothetical next iteration would not be able to evaluate new points.

function we will use. Given the current point $\alpha^k = (\alpha_1^k, \ldots, \alpha_M^k)$ in $\mathcal{O}_k$ in the annealing loop, we will do the following:

(1) Select uniformly a parameter index i in $[1, M]$.
(2) If $\alpha_i^k = c_i^k$, toss a coin. If we get heads, then we change it to $\alpha_i^k = c_i^k - \Delta_i/2^{k+1}$, whereas if we get tails, we set $\alpha_i^k = c_i^k + \Delta_i/2^{k+1}$.
(3) Otherwise, if $\alpha_i^k = c_i^k \pm \Delta_i/2^{k+1}$ we set $\alpha_i^k = c_i^k$, i.e., we bring it back to the middle point.

Notice that possible values for α_i^k are the centre point's i-th coordinate c_i^k and the values $c_i^k - \Delta_i/2^{k+1}$ and $c_i^k + \Delta_i/2^{k+1}$ of the grid points "below" and "above" it (Fig. 3). In this way, only one of the coordinates of the current point α^k is changed: if its value is the middle one, then we move it randomly to the left or right ones; if it is either at the left or right bounds, its value is changed to the middle one.

To define the annealing schedule to be used in the loop's fourth step, notice that the number of neighbours to be tested is $\Lambda - 1$. Thus, a straightforward option is to start from an initial temperature T_0 and decrease it uniformly after evaluating a new point, so that it becomes 0 at the loop's end. Therefore, the temperature at the i-th neighbour, $0 \leqslant i \leqslant \Lambda - 1$, will be

$$T_i = T_0\left(1 - \frac{i}{\Lambda - 1}\right). \tag{8}$$

Once the loop has finished, we will choose from the Λ observed points the one giving the overall best fitness as the centre of the new grid, and we zoom in on the next finer grid. The initial T_0 is a parameter to be decided upon. In order to choose it, recall that the temperature establishes how frequent fitness worsening movements will be. Taking the Maxwell–Boltzmann distribution formula, if the current and neighbour fitnesses F_C and F_N lie in the interval $(0, 1)$ (as this is our case because the fitnesses are validation errors), then the exponential term lies in $[e^{-1/T_0}, e^{1/T_0}]$. By choosing $T_0 = 1$ the probability of moving in the worst case is $\frac{1}{e} \approx 0.368$. A usual heuristic is to use an initial temperature that somewhat decreases this probability, but that it allows, anyway, for any worsening movement with a significant probability. Because of this, in all our experiments we will use the value $T_0 = 0.8$.

While simple in principle, there are some subtleties in AFGS that might be overlooked at first glance. For instance, we would like to reuse already-evaluated points, which we can do as in DFGS using now a cache of size 2Λ that includes the points from the previous and current grids. However, reusing points may be undesirable for we may end up with less than Λ *new* grid points if the neighbourhood function re-selects already-evaluated neighbour points. This can also arise when $\Lambda \approx 3^M - 2^M$ (something that may happen if, as in some of our examples, M is small), because then the annealing loop has to examine nearly all the grid points while the random neighbourhood function may miss some of them. In order to overcome these situations, we decrease the temperature by using (8) when the loop seems to be stuck (i.e. when a certain number S of successive neighbours have been previously evaluated) and make the annealing loop exit when the temperature becomes zero, even if less than Λ new points have been evaluated.

4. Numerical experiments

We will illustrate the preceding techniques over Rãtsch's classification datasets available in [21] and listed in Table 1 together with their number of patterns and attributes. These datasets are given as 100 train–test pairs; we shall use these splits in our experiments. Here we shall work with both Gaussian kernel SVMs and single hidden layer multilayer perceptrons. For SVMs we shall allow for margin slacks with quadratic penalties; that is, we use a criterion function of the form

$$\frac{1}{2}\|W\|^2 + \frac{C}{2}\sum \xi_i^2;$$

SVMs will be trained by the simple 1-vector MDM algorithm given in [22]. Our MLPs will have linear outputs and will be trained by conjugate gradient optimization with a weight decay penalty term of the form $\rho\|W\|^2/nW$, where ρ denotes the normalized penalty factor, W the overall MLP weight vector and nW the total number of weights.

We will consider two different situations. In the first one, that we shall call "small number of parameters" problems, we will try to find, for MLPs, the optimal number of hidden units and the ρ value while for SVM models we will try to find the optimal slack margin penalty C and the Gaussian kernel width $2\sigma^2$. Note that the number of model parameters to be optimized is 2 in both cases and all the previously described search methods will be considered. Recall that the complexity of any parameter finding procedure is determined by the number of different models to be built and tested. As a consequence, in this small number of parameters case we shall discuss two searching scenarios, a first one where a low number of different models are considered and a second one where we will allow the search procedures to consider a larger but still moderate number of models. In the first setting we will work with AFGS and CMA-ES, while in the second we will compare DFGS and CMA-ES allowing for a larger number of generations. Finally, and just for comparison purposes, we will

Table 1
Datasets used, number of examples and number of attributes.

Dataset	# Patterns	# Attributes
Thyroid	215	5
Heart disease	170	13
Breast cancer	277	9
Pima diabetes	768	8
German credit	1000	20
Flare	1066	9
Titanic	2201	3
Image	2310	18
Splice	3175	60
Waveform	5000	21
Banana	5300	2
Ringnorm	7400	20
Twonorm	7400	20

consider optimal parameter selection when a substantial number of generations are done, working only with CMA-ES.

On the other hand, we shall also consider "large number of parameters" problems, for which we will work only with SVM models with an anisotropic kernel of the form

$$g(X, X', \Sigma) = \exp\left(-\sum_{1}^{D} \frac{(x_i - x_i')^2}{2\sigma_i^2}\right)$$

with D the number of data attributes and for which we will try to find, besides the previously considered C, the optimal width vector $\Sigma = (2\sigma_1^2, \ldots, 2\sigma_D^2)$. Here DFGS would be too costly and we shall only compare the performance of annealed FGS with that of CMA-ES under three searching scenarios where optimal parameters are chosen after a small, moderate and large number of models have been built.

Finally, for both models the fitness function will be the averaged test error on the first five splits for each dataset; that is, the test subsets of the first five splits will be used as validation subsets. Once the presumably optimal parameters for a given model have been chosen, they will be tested by cross validation over the full 100 splits and we will report in our tables the averaged test errors.

4.1. Results for problems with a small number of parameters

We shall work in this section with both DFGS and AFGS and also CMA-ES, comparing separately their performance in both MLPs and SVMs. As mentioned in the introduction we shall discuss two searching scenarios, a first one where a small number of different models are examined and a second one where we will allow the search procedures to consider a larger but still moderate number of model evaluations. Given that we want to optimize fitness values over $M = 2$ parameters, our CMA-ES parameter choices imply that the population size $\lambda = 4 + \lfloor 3 \log M \rfloor$ of each generation will just be 6, from which we will select a parent population of $\mu = \lceil (\lambda - 1)/2 \rceil = 3$ patterns. In particular, a CMA-ES run over Q generations requires $6Q$ model constructions. While this makes it quite easy to decide on the number of CMA-ES generations to be allowed, we cannot, on the other hand, determine in advance the number of models that will be explored in deterministic and annealed FGS.

We have thus to decide experimentally how to pair CMA-ES and the grid search runs so that they have comparable complexity.

Table 2
Average number of runs per dataset of 5-pattern offspring AFGS and DFGS for MLP and SVM optimal parameter selection.

Dataset	5-AFGS MLP	DFGS MLP	5-AFGS SVM	DFGS SVM
Thyroid	16	44	47	77
Heart disease	10	44	42	74
Breast cancer	10	45	45	78
Pima diabetes	16	44	46	81
German credit	22	43	45	76
Flare	16	44	44	73
Titanic	10	40	44	81
Image	10	43	47	75
Splice	20	40	42	80
Waveform	16	43	43	82
Banana	17	42	39	80
Ringnorm	10	41	25	80
Twonorm	17	45	36	73
Average	15	43	42	78

The last row gives the overall averages over all datasets.

Table 3
Average test classification errors and their standard deviation for MLP optimal parameter selection for small (cols. 2 and 3) and moderate (cols. 4 and 5) number of model evaluations.

Dataset	Lower complexity		Moderate complexity		50-CMA-ES
	5-AFGS	5-CMA-ES	DFGS	10-CMA-ES	
Thyroid	4.9 ± 2.7	4.9 ± 2.7	$3.7 \pm 2.0^{1,2,3}$	4.4 ± 2.2	4.4 ± 2.4
Heart dis.	27.8 ± 3.8	19.7 ± 3.3^{1}	19.6 ± 3.1	19.6 ± 3.1	19.5 ± 3.2
Br. cancer	29.1 ± 5.0	29.3 ± 5.1	29.2 ± 4.8	29.3 ± 5.1	29.1 ± 4.8
Pima diabetes	26.5 ± 2.0	24.8 ± 2.4^{1}	$23.9 \pm 1.9^{1,2,3}$	24.8 ± 2.4	24.8 ± 2.4
German credit	26.6 ± 2.7	26.4 ± 2.7	24.8 ± 2.2^{2}	25.0 ± 2.3	24.6 ± 2.1
Flare	$33.9 \pm 1.7^{1,3}$	34.7 ± 1.8	$33.8 \pm 1.7^{1,3}$	34.7 ± 1.8	34.7 ± 1.8
Titanic	22.4 ± 1.2	22.4 ± 1.1	22.4 ± 1.2	22.5 ± 1.3	22.5 ± 1.3
Image	3.4 ± 0.5	3.6 ± 0.8	3.6 ± 0.7	3.6 ± 0.8	3.6 ± 0.8
Splice	17.3 ± 1.0	17.3 ± 1.2	17.2 ± 1.1	17.3 ± 1.2	17.2 ± 1.1
Waveform	$13.1 \pm 0.8^{1,3}$	13.7 ± 1.4	$13.2 \pm 0.9^{1,3}$	13.7 ± 1.4	13.7 ± 1.4
Banana	11.2 ± 0.6	10.9 ± 0.5^{1}	10.8 ± 0.5	10.8 ± 0.5	10.8 ± 0.4
Ringnorm	15.8 ± 1.5	15.3 ± 1.6^{1}	15.3 ± 1.6	15.3 ± 1.6	15.3 ± 1.6
Twonorm	3.9 ± 0.7	3.9 ± 0.7	3.7 ± 0.6	3.9 ± 0.7	3.9 ± 0.7

Column 6 reports average accuracies for 50-generation CMA-ES.

To do so we have first fixed three generation numbers for CMA-ES, 5, 10 and 15, requiring, respectively, 30, 60 and 90 models and have adjusted experimentally the depth of 5-AFGS and DFGS so that they result in similar numbers of model constructions. Table 2 gives for each dataset the number of models that the FGS require as well as their average over all datasets. As it can be seen, for MLPs 5-AFGS requires an average of 15 models (versus 30 for 5-generation CMA-ES) and DFGS an average of 43 (versus 60 for 10-generation CMA-ES), while for SVMs 5-AFGS requires an average of 42 models (versus 60 for 10-generation CMA-ES) and DFGS an average of 78 (versus 90 for 15-generation CMA-ES). Thus, for MLPs we will pair 5-AFGS with 5-CMA-ES and DFGS with 10-CMA-ES and, for SVMs, we will pair 5-AFGS with 10-CMA-ES and DFGS with 15-CMA-ES.

Comparison results are given in Table 3, which gives for each dataset the average test classification errors over the 100 train–test splits mentioned above. In the table a ^{1} upper index means an average test accuracy significant at the 10% level on a Wilcoxon rank test when comparing columns 2 and 3 (i.e., a relatively low complexity scenario), and 4 and 5 (i.e., a moderate complexity one), respectively. Similarly, a ^{2} upper index means a significant difference for the same test between the best values in columns 2–3 and in columns 4–5. Table 3 also gives in column 6 large complexity results for 50-generation CMA-ES, which requires 300 model constructions; a ^{3} upper index marks the best significant value between the accuracies in columns 2–5 and the average accuracies of 50 generation CMA-ES. Recall that a Wilcoxon rank sum test checks for the null hypothesis that two paired distributions have the same mean. An advantage when compared to a t test is that it does not assume that the distributions are normal, so it is more robust whenever the distributions (here the errors) are not Gaussian.

As it can be seen, for the lower complexity case, 5-point AFGS gives in two datasets significantly better results than 5-generation CMA-ES while the latter gives better results in four datasets. Considering that 5-AFGS carries out about half as many evaluations as 5-CMA-ES, it seems that AFGS can give good results if properly tuned. However, sometimes it may end too quickly, as it is the case for the Pima and heart problems. For the moderate complexity case, DFGS gives better results than 10-generation CMA-ES in four datasets while the latter does not win on any dataset. This is remarkable, since DFGS is seen in Table 2 to be less costly in terms of the number of model evaluations. Finally, the best of the previous models gives a better test accuracy than

50-generation CMA-ES in four datasets while the latter does not give better results on any problem.

In summary, it is seen that DFGS beats 50-CMA-ES whenever it also beats 10-CMA-ES and whenever 5-AFGS beats 5-CMA-ES. This seems to imply that sometimes CMA-ES may get stuck at not too good points (note that error values for 5- and 50-CMA-ES are quite close). Since DFGS performs a bit better than CMA-ES and AFGS is slightly worse than its CMA-ES counterpart, it also looks likely that it would be possible to tune the Λ parameter in AFGS so that its cost lies somewhere between that of 5-AFGS and DFGS and gives a similar performance to that of CMA-ES but with less work than both DFGS and CMA-ES.

Similar results are given in Table 4 for SVM optimal parameter selection. Here for a small number of model evaluations, 5-point AFGS does not give better results than 5-generation CMA-ES on any dataset while the latter gives better results in four datasets. This fact is partly due to 5-CMA-ES being a bit fortunate in datasets Thyroid and Waveform, but here AFGS performance is anyway worse, since it executes more runs and nevertheless ends up with higher error rates. For a larger number of model evaluations, DFGS gives better results than 15-generation CMA-ES in one dataset while the latter gives better results in another one. Thus they seem to be in a tie, DFGS needing slightly less runs than CMA-ES. Finally, the best of the previous models gives a better test accuracy than 50-generation CMA-ES in two datasets, while the latter never gives the best results.

As it can be seen, the SVM situation is different from the MLP one. While for the latter focused grid search seems to be better than CMA-ES, for SVMs it is not so, and both methods perform similarly. Analyzing the results in more detail it turns out that CMA-ES almost always yields a lower validation error than either DFGS or AFGS. Therefore, considered as a black-box optimizer CMA-ES is clearly a better technique, but an excessive optimization of the validation error for MLPs may result in worse test errors and CMA-ES seems to be more prone to some overfitting. This is why the simpler grid searches may perform better in some tests. Besides, in the MLP case CMA-ES treats the number of hidden units as a continuous parameter, needing thus to deal with an additional difficulty that does not affect grid search. Conversely, CMA-ES can examine points that grid search cannot, since the latter is confined to the points specified by the resolution level.

Table 4
Average test classification errors and their standard deviation for SVM optimal parameter selection for small (cols. 2 and 3) and moderate (cols. 4 and 5) number of model evaluations.

Dataset	Lower complexity		Moderate complexity		50-CMA-ES
	5-AFGS	5-CMA-ES	DFGS	15-CMA-ES	
Thyroid	5.0 ± 2.5	$3.8 \pm 2.0^{1,2,3}$	4.3 ± 1.8	4.3 ± 1.8	4.3 ± 1.8
Heart dis.	15.8 ± 3.4	15.8 ± 3.1	15.8 ± 3.2	15.8 ± 3.1	15.8 ± 3.1
Br. cancer	26.2 ± 4.7	25.8 ± 4.6	26.5 ± 4.8	26.5 ± 4.8	26.5 ± 4.9
Pima diabetes	23.3 ± 1.7	23.2 ± 1.7	23.3 ± 1.7	23.2 ± 1.7	23.1 ± 1.7
German credit	23.6 ± 2.1	23.5 ± 2.1	23.8 ± 2.2	23.6 ± 2.1	23.6 ± 2.1
Flare	33.7 ± 1.7	33.6 ± 1.7	33.7 ± 1.7	33.5 ± 1.7	33.5 ± 1.7
Titanic	22.7 ± 1.1	22.7 ± 1.2	22.5 ± 0.9	22.7 ± 1.2	22.8 ± 1.2
Image	2.8 ± 0.5	2.9 ± 0.5	3.0 ± 0.5	2.9 ± 0.5	2.9 ± 0.5
Splice	10.9 ± 0.6	10.9 ± 0.6	10.6 ± 0.8	10.6 ± 0.7	10.6 ± 0.7
Waveform	10.5 ± 0.5	$10.0 \pm 0.4^{1,3}$	$9.9 \pm 0.4^{1,3}$	10.3 ± 0.5	10.3 ± 0.5
Banana	19.5 ± 1.7	10.4 ± 0.5^{1}	10.4 ± 0.5	10.4 ± 0.5	10.4 ± 0.5
Ringnorm	2.0 ± 0.2	1.5 ± 0.2^{1}	1.6 ± 0.1	$1.5 \pm 0.1^{1,2}$	1.4 ± 0.1
Twonorm	2.4 ± 0.1	2.4 ± 0.1	2.4 ± 0.1	2.4 ± 0.1	2.4 ± 0.1

Column 6 reports average accuracies for 50-generation CMA-ES.

4.2. Results for problems with a large number of parameters

As mentioned before, we repeated the previous experiments in some of the previous datasets (see Table 5) using now a SVM with an anisotropic kernel. With this kind of kernel $D+1$ parameters have to be optimized, with D corresponding to the number of input attributes of the dataset and the additional one corresponding to the C SVM penalization factor. DFGS is now ruled out because of its too high complexity and three different runs per dataset were performed for CMA-ES and AFGS, corresponding to a small, moderate and large number of model evaluations. Columns 2, 4 and 6 in Table 5 give the number of model constructions for CMA-ES when 5, 50 and 250 generations are considered. To get approximately similar complexities for AFGS, the heart disease problem was chosen to select the number Λ of grid point evaluations in AFGS that would result in about the same complexity for that dataset. This results in a choice of 5, 50

Table 5
Number of runs per dataset and average values of 5-, 50- and 250-pattern offspring CMA-ES and 5-, 50- and 275-AFGS for anisotropic kernel SVM parameter selection.

Dataset	5-CMA-ES	5-AFGS	50-CMA-ES	50-AFGS	250-CMA-ES	275-AFGS
Heart disease	55	54	550	553	2750	2750
Breast cancer	50	38	500	422	2500	2127
Pima diabetes	50	36	500	314	2500	1739
German credit	65	87	650	353	3250	2242
Flare	50	35	500	308	2500	1658
Waveform	65	39	650	403	3250	2180
Ringnorm	65	41	650	630	3250	1832
Twonorm	65	33	650	419	3250	2206
Average	58	45	581	425	2906	2092

Table 6
Average test classification errors and their standard deviation for an anisotropic kernel SVM optimal parameter selection for small (cols. 2 and 3), moderate (cols. 4 and 5) and large (cols. 6 and 7) number of model evaluations.

Dataset	5-CMA-ES	5-AFGS	50-CMA-ES	50-AFGS	250-CMA-ES	275-AFGS
Heart dis.	15.7 ± 3.2	15.7 ± 3.1	15.8 ± 3.4	15.7 ± 3.2	15.9 ± 3.4	15.7 ± 3.3
Br. cancer	26.9 ± 4.8	26.2 ± 4.7	26.6 ± 4.7	25.9 ± 4.5	25.7 ± 4.5	25.9 ± 4.5
Pima diabetes	23.2 ± 1.7	23.3 ± 1.7	23.3 ± 1.7	23.3 ± 1.7	23.3 ± 1.7	23.3 ± 1.7
German credit	23.6 ± 2.1	23.5 ± 2.2	23.6 ± 2.1	23.6 ± 2.2	23.6 ± 2.1	23.6 ± 2.1
Flare	33.5 ± 1.7	33.7 ± 1.7	33.5 ± 1.7	33.7 ± 1.7	33.5 ± 1.7	33.7 ± 1.7
Waveform	10.9 ± 0.6	10.6 ± 0.5^{1}	10.2 ± 0.5^{1}	10.6 ± 0.5	10.1 ± 0.6^{1}	10.6 ± 0.5
Ringnorm	1.5 ± 0.1^{1}	2.8 ± 0.4	1.4 ± 0.1^{1}	2.4 ± 0.3	1.5 ± 0.1^{1}	2.0 ± 0.2
Twonorm	2.4 ± 0.1	2.4 ± 0.1	2.4 ± 0.1	2.4 ± 0.1	2.4 ± 0.1	2.4 ± 0.1

and 275 point evaluations per grid and columns 3, 5 and 7 in Table 5 give the corresponding number of model constructions required by AFGS for all datasets considered now. As it can be seen, while more or less the same for the heart disease problem, the AFGS complexity is in general much smaller than that of CMA-ES.

Accuracy results are reported in Table 6. Again, a ^{1} upper index means a significant improvement for a method over its counterpart with about the same complexity, quantified by a Wilcoxon rank sum test at the 10% confidence level. No significant difference is observed in six of the eight datasets and only in the high complexity cases of the Waveform and Ringnorm datasets the CMA-ES performance is significantly better than that of AFGS. These results point out that AFGS can also be effectively used in a large parameter number scenario, although CMA-ES may have the advantage in some problems.

5. Discussion and conclusions

Present day machine learning offers a very wide range of modelling methods and while for a given problem, the ability to select the best one would be highly desirable, this same high number of options makes that choice quite difficult. Besides, even when one particular model is chosen, it is not an easy task to come up with good values for that model's specific parameters. On the other hand, if the model is itself general enough to cover a wide range of problems, an appropriate choice of parameters may make its performance comparable to that of the best method for a given problem.

In this work, we have considered binary classification problems and two general models, MLPs and SVMs, have been studied. In order to find good parameters for both, we have explored two metamodels, a purely stochastic one, CMA-ES and two types of focused grid search, deterministic and annealed. Our experiments cover two scenarios. In the first one, when the number of parameters is small, all methods have comparable computational costs. However, when it is not so, the exponential cost of DFGS forbids its application, whereas AFGS and CMA-ES are able to handle this case.

Regarding metamodel performances, CMA-ES turns out to be a better black-box optimizer than either DFGS or AFGS. Nonetheless, the function to be minimized in the parameter search (validation error) is only an approximation to the function that we actually want to minimize (test error). This fact makes CMA-ES more prone to overfitting, yielding sometimes worse parameters than its grid counterparts. Besides, CMA-ES assumes that all the parameters are continuous; if some are discrete, its performance seems to be worse. Conversely, DFGS and AFGS discretize all the parameters, so their choices for continuous parameter values are limited.

Anyway, whichever the metamodel, if the number of model constructions is large enough, the parameters that are obtained with the three methods produce competitive models and a decision among them may have to be made on other grounds. For instance, we would recommend CMA-ES in a general minimization setting, as it has implementations in several programming languages that are publicly available. However, it is certainly a complex procedure and if for some reason it does not perform well with the default choices for its parameters and formulæ, it may be quite hard to tune; moreover, for classification problems where fitness is measured as accuracy over validation sets, CMA-ES may produce some overfitting. On the other hand, while having a similar performance on most of the datasets considered, DFGS or AFGS is much simpler, and the second one can also be used when there is a considerable number of parameters to be chosen. Moreover, in DFGS we only have to select the parameter boundaries and the resolution levels and, while AFGS needs additional metaparameters to be tuned, they are more intuitive than the ones in CMA-ES. Thus, while there is no clear-cut final decision on what procedure should be applied, the simplicity and effectiveness of DFGS and AFGS may make them a worthwhile choice, particularly in applications where third party solutions cannot be used and a proprietary ad-hoc implementation has to be developed.

Acknowledgements

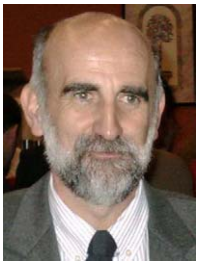
With partial support of Spain's TIN 2004–07676 and TIN 2007–66862. The first author is kindly supported by the FPU–MEC grant reference AP2006–02285.

References

[1] R. Duda, P. Hart, D. Stork, Pattern Classification, Wiley, New York, 2000.
[2] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[3] B. Schölkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond, Machine Learning, MIT Press, Cambridge, MA, 2002.
[4] S.S. Keerthi, Efficient tuning of SVM hyperparameters using radius/margin bound and iterative algorithms, IEEE Transactions on Neural Networks 13 (5) (2002) 1225–1229.
[5] D.C. Montgomery, Design and Analysis of Experiments, Wiley, New York, 1976.
[6] C.-W. Hsu, Ch.-Ch. Chang, Ch.-J. Lin, A practical guide to support vector classification ⟨www.csie.ntu.edu.tw/~cjlin/libsvmtools⟩.
[7] C. Staelin, Parameter selection for support vector machines, Technical Report HPL-2002-354, HP Laboratories, Israel, 2000.
[8] C. Harpham, C.W. Dawson, M.R. Brown, A review of genetic algorithms applied to training radial basis function networks, Neural Computing & Applications 13 (3) (2004) 193–201.
[9] L.E. Kuo, S.S. Melsheimer, Using genetic algorithms to estimate the optimum width parameter in radial basis function networks, in: Proceedings of the American Control Conference, Baltimore, MD, June 1994.
[10] T. Howley, M.G. Madden, The genetic kernel support vector machine: description and evaluation, Artificial Intelligence Review 24 (3–4) (2005) 379–395.
[11] C.M. Huang, Y.J. Lee, D. Lin, S.Y. Huang, Model selection for support vector machines via uniform design, Computational Statistics and Data Analysis 52 (1) (2007) 335–346.
[12] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary Computation 9 (2) (2001) 159–195.
[13] A. Auger, N. Hansen, Performance evaluation of an advanced local search evolutionary algorithm, in: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, September 2005, pp. 1777–1884.
[14] B.F. Mitchell, V.F. Dem'yanov, V.N. Malozemov, Finding the point of a polyhedron closest to the origin, SIAM Journal on Control 12 (1974) 19–26.
[15] F. Friedrichs, C. Igel, Evolutionary tuning of multiple SVM parameters, Neurocomputing 64 (2005) 107–117.
[16] N. Hansen, The CMA evolution strategy: source code ⟨http://www.bionik.tu-berlin.de/user/niko/cmaes_inmatlab.html⟩.
[17] K.T. Fang, D.K.J. Lin, Uniform experimental designs and their applications in industry, Handbook of Statistics 22 (2003) 131–170.
[18] The Uniform Design Association of China, Uniform design tables ⟨http://www.math.hkbu.edu.hk/UniformDesign/⟩.
[19] N. Hansen, The CMA evolution strategy: a tutorial ⟨http://www.bionik.tu-berlin.de/user/niko/cmatutorial.pdf⟩.
[20] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimization by simulated annealing, Science 4598 (1983) 671–680.
[21] G. Rätsch, Benchmark repository ⟨http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm⟩.
[22] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, K.R.K. Murthy, A fast iterative nearest point algorithm for support vector machine classifier design, IEEE Transactions on Neural Networks 11 (1) (2000) 124–136.

A. Barbero received the Computer Scientist degree from the Universidad Autónoma de Madrid (UAM) in 2006, and is currently a student in a Master/Ph.D. degree in Computer Science at the same university. At present, he is working at the UAM under a National Predoctoral Grant, in collaboration with the Instituto de Ingeniería del Conocimiento. His research interests are in pattern recognition, kernel methods and wind power forecasting.

J. López received his Computer Engineering degree from the Universidad Autónoma de Madrid in 2006, where he got an Honorific Mention as the best student. Currently he is attending the Postgraduate Programme organized by the Computer Engineering Department of the same university. His research interests concentrate on support vector machines, but also cover additional machine learning and pattern recognition paradigms.



J.R. Dorronsoro received the Licenciado en Matemáticas degree from the Universidad Complutense de Madrid in 1977 and the Ph.D. degree in Mathematics from Washington University in St. Louis in 1981. Currently he is Full Professor in the Computer Engineering Department of the Universidad Autónoma de Madrid, of which he was the head from 1993 to 1996. His research interest is in neural networks, image processing and pattern recognition.