A Note on Parameter Selection for Support Vector Machines

Igor Braga, Laís Pessine do Carmo,
Caio César Benatti, and Maria Carolina Monard

Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo – São Carlos, SP – Brasil
{igorab,mcmonard}@icmc.usp.br, {lais.carmo,caiocesar}@usp.br

Abstract. Parameter selection greatly impacts the classification accuracy of Support Vector Machines (SVM). However, this step is often overlooked in experimental comparisons, for it is time consuming and requires familiarity with the inner workings of SVM. Focusing on Gaussian RBF kernels, we propose a grid-search procedure for SVM parameter selection which is economic in its running time and does not require user intervention. Based on probabilistic assumptions of standardized data, this procedure works by filtering out parameter values that are not likely to yield reasonable classification accuracy. We instantiate this procedure in the popular WEKA data mining toolbox and show its performance on real datasets.

1 Introduction

Support Vector Machines (SVM) [1] is a supervised learning method that has been used to achieve state-of-the-art classification results in many domains of application. It is usually among the methods that are experimentally evaluated when a new application or learning method is being proposed.

A crucial step when applying SVM is the selection of its parameters. In order to do it properly and efficiently, it is required an understanding of how these parameters affect SVM classification. Hence, users not familiarized with SVM tend to skip parameter selection, often resorting on default parameters of the implementations of their choice. The problem with this approach is that there are no default parameters for SVM.

There has been some effort to introduce procedures for SVM parameter selection that are both easy-to-use and principled [2,3]. However, they currently involve checking a large range of parameter candidates, which can be quite time consuming. What is more, there is an emphasis in fine-grained parameter selection, which may be overkill.

In this work we intend to make SVM parameter selection automatic and economic in its running time. In order to do that, we try to exclude parameter values that are not likely to provide good classification accuracy. After that, we try to investigate parameter candidates that are essentially different, that is, lead to different classification results. A key step to our procedure is data standardization, which is a common data pre-processing task that is also useful for bringing all data features (attributes) to the same scale.

The machine learning community has much aided users by making available a wide variety of learning algorithms through open source packages. A popular environment which is widely used by machine learning experts and non-experts is the Waikato Environment for Knowledge Analysis (WEKA) [4]. It contains a wide variety of machine learning methods and also provides graphical user interfaces for easy access. Due to its popularity, we use it in this work to instantiate and illustrate the proposed SVM parameter selection procedure.

The remainder of this paper is organized as follows. In Section 2 we review SVM parameters and how they affect classification performance. In Section 3 we express caution on the use of default parameters in SVM. In Section 4 we present the proposed parameter selection procedure for SVM. In Section 5, we instantiate the proposed procedure in WEKA and conduct illustrative experiments. We conclude in Section 6.

2 Essential Parameters of SVM

This section reviews the parameters that most affect the generalization ability of Support Vector Machines[1]. It is not intended to be a tutorial on SVM classification, as good references on the subject already exist [5].

Given n training examples $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$, where $\boldsymbol{x} \in R^d$ and $Y = \{+1, -1\}$, the optimization problem that emerges from SVM is usually written as follows

$$\begin{aligned} \max_{\alpha_1,\ldots,\alpha_n} \quad & \sum_{i=1}^{n} \alpha_i - \tfrac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \, k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{subject to} \quad & \sum_{i=1}^{n} \alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C, \end{aligned}$$

where the constant $C > 0$ and the kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ are parameters to be defined. After solving this problem for fixed C and k, the output variables α_i are used to derive the function $f(\boldsymbol{x})$ used to classify unseen data

$$f(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i y_i \, k(\boldsymbol{x}, \boldsymbol{x}_i) + b \right), \tag{1}$$

where b is also computed from α_i.

Let us first tackle the so-called *generalization parameter C*. When a learning algorithm selects a function $f(\boldsymbol{x})$ from a set of functions $\mathcal{F}$ using training examples, it does so by means of an inductive principle. In the case of SVM, induction is performed by the Structural Risk Minimization principle [6], which controls

[1] Besides the parameters discussed in this section, there may be other parameters related to specific SVM solvers that do not greatly affect generalization.

two factors: 1) the number of training errors made by $f(\boldsymbol{x})$ and 2) the capacity (diversity) of the set of functions $\mathcal{F}$. In order to obtain a function $f(\boldsymbol{x})$ that delivers good classification accuracy on test examples, both factors should be small. As these factors are contradictive, a balance has to be found by controlling the parameter C.

The effect of C in SVM is clearer from its primal optimization problem, which for simplicity we show for the linear case

$$\begin{array}{ll} \min\limits_{\boldsymbol{w},b,\xi_1,\ldots,\xi_n} & \|\boldsymbol{w}\|^2 + C\sum\limits_{i=1}^{n}\xi_i \\ \text{subject to} & y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{array}$$

Note that large values of C put more emphasis on the minimization of the slack variables ξ_i, which leads to correct separation of the training examples but risks overfitting. On the other hand, small values of C put more emphasis on the minimization of $\|\boldsymbol{w}\|$, which leads to maximization of the margin of separation and, consequently, minimization of the capacity of the set of functions being considered [6]. This latter case, however, may lead to underfitting. Figure 1 illustrates this trade-off.

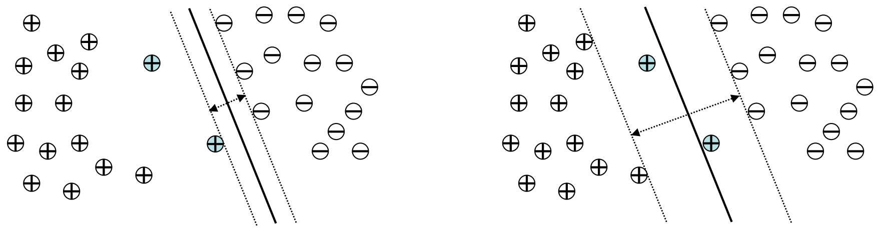


Fig. 1. Typical linear SVM scenario. The value of C used in the left is larger than the one used in the right.

We now turn our attention to the kernel function parameter $k(\boldsymbol{x},\boldsymbol{x}')$, which induces the set of functions $\mathcal{F}$ from where the function $f(\boldsymbol{x})$ is picked. Two kinds of kernel functions have been extensively employed: the Gaussian RBF and the linear types. Since most domains require non-linear classification, the former is often more appropriate[2]. Another important property of the Gaussian RBF type is its universal function approximation ability [7].

In fact, Gaussian RBF kernels form a family of kernel functions parameterized by a parameter $\gamma > 0$

$$k(\boldsymbol{x},\boldsymbol{x}') = \exp\left(-\gamma\,\|\boldsymbol{x}-\boldsymbol{x}'\|^2\right).$$

[2] An exception occurs when the number of dimensions d (features) of the training examples greatly exceeds the number of training examples n ($d \gg n$).

This means that by choosing γ, we are effectively choosing a kernel function for SVM. The effect of choosing increasingly larger values of γ is to make the kernel an identity evaluator, that is

$$k(\boldsymbol{x}, \boldsymbol{x}') = 1, \text{ if } \boldsymbol{x} = \boldsymbol{x}',$$
$$k(\boldsymbol{x}, \boldsymbol{x}') \approx 0, \text{ if } \boldsymbol{x} \neq \boldsymbol{x}'.$$

On the other hand, choosing very small values of γ has the effect of making $k(\boldsymbol{x}, \boldsymbol{x}') \approx 1$ for any $\boldsymbol{x}$ and $\boldsymbol{x}'$. Figure 2 illustrates the effect of γ in SVM. Note how a large value of γ allows for more diversity in the set of functions implemented by SVM.

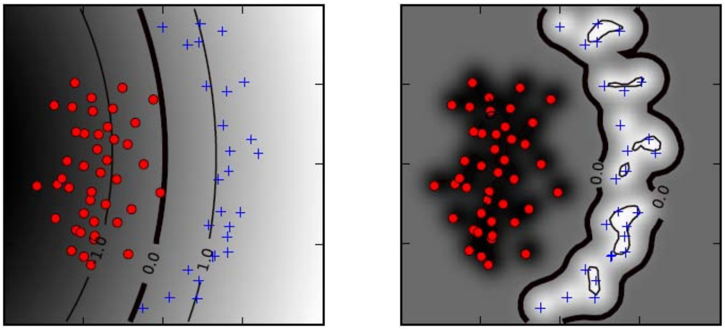


Fig. 2. SVM decision function using Gaussian RBF kernels. The value of γ used in the right is larger than the one used in the left. (Source: [3])

3 A Word of Caution about Default Parameters

The aim of machine learning algorithms is the autonomous building of models from datasets. However, learning algorithms do not necessarily provide good results without being properly tuned. In other words, besides choosing an appropriate learning algorithm for the specific learning domain, it is also necessary to set the algorithm parameters. Nevertheless, manual tuning of the algorithm parameters can frequently be very time-consuming, as well as requiring good knowledge of the learning algorithm.

As there is an increasing number of non-expert users of machine learning tools who require off-the-shelf solutions, these environments always offer default parameter values to execute the algorithms. However, while the default parameter values may be reasonably appropriate for some learning algorithms such as decision trees, in which, for example, the generalization default parameter (confidence limit[3]) is usually set to 25%, this is not the case for most learning algorithms. For example, most implementations of the standard k-NN algorithm usually take $k = 1$ as the default number of neighbors. As the learning domain

[3] The smaller the confidence limit, the higher the chances of pruning the tree.

is not known in advance and $k = 1$ entails less running time, this is the usual default value used in most implementations, although other values can dramatically improve the results.

This is also the case of other algorithms, such as SVM, which also strongly depends on parameter selection. It is worth noting that quite different default values are used in different implementations. For example, in WEKA and LIBSVM[4][8,2] the default value of C is 1. In SVMlight[5] the default value is the average of $(\boldsymbol{x} \cdot \boldsymbol{x})^{-1}$, while in SVMTorch[6][9], the default value is 100. Moreover, WEKA and SVMlight use the linear kernel as default, while LIBSVM uses RBF.

In other words, default values are offered such that non-expert users can initially experiment how the software performs right out of the box. However, a typical machine learning scientist would at least try to improve the expected performance of a learning algorithm over a few parameters. In case there is a need of an optimal model, the search should be conducted over all possible parameters.

To illustrate our point, let us consider UCI's Monk-1 and Monk-2 artificial datasets [10], both from the same domain, with no missing feature values. Each dataset consists of 432 examples described by 7 features and classified as $\oplus$ or $\ominus$. Monk-1 is a balanced dataset (50% $\oplus$, 50% $\ominus$), while Monk-2 is slightly unbalanced (67% $\oplus$, 33% $\ominus$). Thus, the error of the simplest algorithm that always predicts the majority class will be 0.50 for Monk-1 and 0.33 for Monk-2. Using WEKA, we executed John C. Platt's Sequential Minimal Optimization (SMO) algorithm for training a support vector machine [11]. SMO was executed with its default parameters, which are the linear kernel and $C = 1$. We randomly select 2/3 of the corresponding dataset as a training set and 1/3 as a test set. The results are shown in Table 1, where Error is the error rate (proportion of correctly classified instances); B.Error is the balanced error (mean of the positive and negative error rates), and AUC is the area under the ROC curve [12]. Recall that AUC = 0.5 correspond to random guessing. Thus, no realistic classifier should have an AUC less than 0.5.

Table 1. Results of SVM classification using the default parameters in WEKA — linear kernel and $C = 1$

Dataset	Error	B.Error	AUC
Monk-1	0.340	0.341	0.659
Monk-2	0.340	0.500	0.500

It can be observed that the results are poor for Monk-1 and extremely poor for Monk-2. Trying to improve these results, several other values of the parameter

[4] A Library for Support Vector Machines
`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

[5] `http://svmlight.joachims.org/`

[6] `http://www.torch.ch/`

Table 2. Results of SVM classification using the Gaussian RBF kernel and parameters $(C = 64, \gamma = 0.1)$

Dataset	Error	B.Error	AUC
Monk-1	0.088	0.087	0.913
Monk-2	0.150	0.172	0.828

C in $[10^{-6}, 10^{-5}, \dots, 10^{6}]$ were tested, but without success. The reason is that for both datasets the relationship between class values and features is nonlinear. This situation cannot be handled by a linear kernel.

In such situations, a kernel which nonlinearly maps examples into a higher dimensional space must be used. Several nonlinear kernels have been proposed in the literature. Among them, the Gaussian RBF kernel is a reasonable choice as explained in Section 2. Table 2 shows the results of using the RBF kernel in WEKA with the assignment of parameters $C = 64$ and $\gamma = 0.1$.

Comparing the results from Table 1 and 2, the improvement obtained by using RBF with a good pair of (C, γ) values is extremely high. Observe that for Monk-1, the error went from 0.340 down to 0.088, 3.86 times lower. For Monk-2 the error went from 0.340 down to 0.150, 2.27 times lower. Moreover, for Monk-1 and Monk-2 respectively, the B.Error was 3.92 and 2.91 times lower. Furthermore, the AUC improved 38% and 67% respectively.

Considering that the RBF kernel can handle datasets having linear as well as non-linear relationship between class values and features, non-expert SVM users may be tempted to report SVM classification results using the RBF kernel with default parameter values. To illustrate the significant differences while using default and optimized SVM-RBF parameters (C, γ), we execute SMO again on datasets Monk-1 and Monk-2, and add two other real world binary datasets: Parkinson and Ionosphere [10]. The first one consists of 197 examples (25% $\oplus$, 75% $\ominus$) described by 23 features, and the second one consists of 351 examples (64% $\oplus$, 36% $\ominus$) described by 34 features. Thus, the majority error rate will be 25% for Parkinson and 36% for Ionosphere.

As before, the experiments were carried out using 2/3 of the corresponding dataset as training set and 1/3 as test set. Table 3 shows the results obtained by executing SMO using the RBF kernel with WEKA default parameters, as well as with other parameters that showed better results.

Table 3. Results of SVM classification using the Gaussian RBF kernel with the default values in WEKA $(C = 1, \gamma = 0.01)$ and with other parameter values

Dataset	Error	B.Error	AUC	Error	B.Error	AUC	(C, γ)
Monk-1	0.510	0.493	0.507	0.088	0.087	0.913	$(64, 0.1)$
Monk-2	0.340	0.500	0.500	0.150	0.172	0.828	$(64, 0.1)$
Parkinson	0.212	0.500	0.500	0.091	0.120	0.838	$(64, 10)$
Ionosphere	0.322	0.424	0.576	0.076	0.082	0.918	$(2, 1000)$

As it can be observed, the improvement obtained by SMO using RBF with other parameters (C, γ) is exceedingly good. For Monk-1 and Monk-2, using the RBF kernel with WEKA default parameters yields an accuracy that is not better than the majority error classifier. For Monk-1, the error went from 0.510 down to 0.088, 5.80 times lower, and the B.Error went from 0.493 down to 0.087, 5.67 times lower. For Monk-2 the error went from 0.340 down to 0.150, 2.27 times lower, and the B.error went from 0.500 down to 0.172, 2.91 times lower. Moreover, for Monk-1 and Monk-2 the AUC improved 80% and 67% respectively.

Also for Parkinson and Ionosphere, SMO with default parameters does not do better than the majority error classifier. However, the classification results yielded by the other parameters are very good. For Parkinson, the error and the B.Error were 2.33 and 4.24 times lower respectively, while the AUC improved 67%. For Ionosphere, the error and the B.Error were 2.66 and 4.24 times lower, while the AUC improved 59%

4 Economic Grid-Search Procedure

It was shown in the last section that using default SVM parameters may yield unsatisfactory classification accuracy. This fact is well-known within the SVM community. As a consequence, for a fixed kernel function, it is desirable to investigate the accuracy provided by SVM with respect to several parameter candidates of C. As to the kernel function, the Gaussian RBF is usually recommended, as explained in Section 2. This way, it is also desirable to investigate the accuracy provided by different candidates of the RBF parameter γ.

The most used method for selecting parameter candidates in the SVM-RBF context is the grid-search procedure with k-fold cross-validation [2,3]. Given a set $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of m candidates of C and a set $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_\ell\}$ of ℓ candidates of γ, an accuracy figure is obtained on the training data for each parameter combination $(C_i, \gamma_j) \in \mathcal{C} \times \Gamma$ through k-fold cross-validation. The pair with the highest cross-validation accuracy is then selected and an SVM classifier trained with this parameter combination is used for classifying unseen data.

Despite its popularity in the SVM community, the described procedure poses a computational problem, since $k \times m \times \ell$ calls to an SVM solver (*e.g* SMO) need to be made, with k generally equal to 10. The problem is worsened when each call to the solver is already expensive, for instance when tens of thousands of examples are available and/or data is very high-dimensional. This means that the candidate sets should be chosen very carefully as to have as few effective candidates as possible.

In what follows, we propose sets of 5 candidates each, which amounts to 25 parameter combinations. Computationally speaking, this is a much better scenario than what is usually considered. For instance, in [2] the following candidate sets are suggested for a coarse grid-search procedure: $\mathcal{C} = \{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ and $\Gamma = \{2^{-15}, 2^{-13}, \ldots, 2^{3}\}$, which amounts to 110 combinations.

4.1 Candidates of γ

In order to choose candidates of γ, the distribution of the values $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$ should be taken into consideration. One possible approach is to pick candidates that revolve around the *inverse* of the mean value of $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$. This way, we avoid choosing a too small or too large candidate of γ. In this work we take advantage of a data pre-processing step to approximate such mean value using a formula, thus avoiding its calculation from data.

When employing SVM classification using RBF kernels, it is very desirable to bring all features of the dataset to the same scale, since features ranging in larger intervals tend to dominate the calculation of distances in the RBF kernel. A popular procedure to tackle the scaling problem is data *standardization*: for each feature, the mean and standard deviation value is computed; then, each feature value is subtracted by the corresponding mean and divided by the corresponding standard deviation. The effect of conducting this procedure on data is that every feature will have an average value of $\hat{\mu} = 0$ and standard deviation $\hat{\sigma} = 1$. By assuming that each feature follows a Normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$, we can come up with an approximation for the mean value of $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$.

Proposition. *Let $\boldsymbol{x} = (x^1, \ldots, x^d)$ and $\boldsymbol{z} = (z^1, \ldots, z^d)$ be two random vectors such that x^j and z^j follow a Normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$ (standard Normal variables). Then the random variable $\|\boldsymbol{x} - \boldsymbol{z}\|^2$ follows a Gamma distribution with shape parameter $s = \frac{d}{2}$ and scale parameter $\theta = 4$.*

Proof. If x^j and z^j are standard normal variables, then the random variable $(x^j - z^j)$ follows a Normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 2$. Thus, $\left(\frac{x^j - z^j}{\sqrt{2}}\right)$ is also a standard normal variable. By definition, the sum of the squares of d independent standard normal variables follows a Chi-squared distribution with d degrees of freedom — $\chi^2(d)$. Thus, the random variable $\mathcal{A} = \sum_{j=1}^{d} \left(\frac{x^j - z^j}{\sqrt{2}}\right)^2$ is $\chi^2(d)$. Note that

$$\|\boldsymbol{x} - \boldsymbol{z}\|^2 = \sum_{j=1}^{d} \left(x^j - z^j\right)^2 = 2\mathcal{A}.$$

The Chi-squared distribution and the Gamma distribution are related in the following way: if $\mathcal{B}$ is $\chi^2(\nu)$ and $c > 0$, then $c\mathcal{B}$ is distributed according to a Gamma distribution with shape parameter $s = \frac{\nu}{2}$ and scale parameter $\theta = 2c$. As $\|\boldsymbol{x} - \boldsymbol{z}\|^2 = 2\mathcal{A}$ and $\mathcal{A}$ is $\chi^2(d)$, it follows that $\|\boldsymbol{x} - \boldsymbol{z}\|^2$ is distributed according to a Gamma distribution with $s = \frac{d}{2}$ and $\theta = 4$. $\square$

When a random variable follows a Gamma distribution, its mean value corresponds to the product $s\theta$. Under the conditions stated in the aforementioned proposition, it follows that the mean value of $\|\boldsymbol{x} - \boldsymbol{z}\|^2$ is $2d$. Thus, whenever we standardize the features of our dataset, we expect the average value of $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$ to be about twice the number of features in the dataset.

Considering $u = \frac{1}{2d}$, we propose as candidates of γ the values $\frac{u}{4}, \frac{u}{2}, u, 2u, 4u$. For $d \geq 4$, this range of values is enough to cover at least 90% of the distribution of $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$ when the features are standardized. This means that our parameter candidates effectively iterates over the range of most probable values of $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$.

4.2 Candidates of *C*

Now, let us consider the candidates of C. It can be observed from the general SVM optimization problem in Section 2 that this parameter is an upper bound on the values of the α_i variables. In Equation (1), we see that these α_i variables are used to calculate the decision function. Note that if C is too small, say 2^{-5}, the product $\alpha_i y_i \, k(\boldsymbol{x}, \boldsymbol{x}_i)$ will also be very small, since $0 \leq k(\boldsymbol{x}, \boldsymbol{x}_i) \leq 1$ when $k(\boldsymbol{x}, \boldsymbol{x}_i)$ is a Gaussian RBF. Under usual conditions, the sum over i will also be small, that is, far from either $+1$ or -1. Thus, the SVM solver will set b so as to classify every training example to the majority class, which results in underfitting. That way, it makes sense to remove from our candidate list small values of C.

We propose considering $C = 2^0$ as the first candidate, and the subsequent candidates $2^1, 2^2, 2^3, 2^4$. We stop at 2^4 for the following reason. If we used $C = 2^4$, then, for any $\boldsymbol{x}_j$ and $\boldsymbol{x}_i$ such that $k(\boldsymbol{x}_j, \boldsymbol{x}_i) = \exp(-\gamma \, \|\boldsymbol{x}_j - \boldsymbol{x}_i\|) \geq 2^{-4}$, it would be feasible to have the product $\alpha_j y_j \, k(\boldsymbol{x}_j, \boldsymbol{x}_i)$ close to either $+1$ or -1. Considering the probabilistic setting described previously and the proposed candidate set $\mathit{\Gamma}$, we would expect $k(\boldsymbol{x}_j, \boldsymbol{x}_i) \geq 2^{-4}$ to occur at least 90% of the time when $d \geq 4$. Thus, we would see little difference in the way the training examples are classified by taking $C = 2^5$ or a larger value.

5 Illustrative Experiments in Weka

In what follows, we instantiate our procedure for SVM parameter selection in the WEKA data mining environment[7]. Given training examples $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$ and test data $\boldsymbol{x}^*_1, \ldots, \boldsymbol{x}^*_m$, where $\boldsymbol{x} \in R^d$ and $Y = \{+1, -1\}$, the two steps of the procedure are:

Data standardization. For each feature value x_i^j of the example $\boldsymbol{x}_i$ in the training set, the corresponding standardized value is computed as $\frac{x_i^j - avg(x^j)}{std(x^j)}$. For each feature value x_i^{*j} of $\boldsymbol{x}^*_i$ in the test set, the corresponding standardized value is computed as $\frac{x_i^{*j} - avg(x^j)}{std(x^j)}$. Note that the scaling factors used to standardize the test set are those of the training set.

Grid-search. Given $\mathcal{C} = \{2^0, 2^1, 2^2, 2^3, 2^4\}$ and $\mathit{\Gamma} = \{\frac{u}{4}, \frac{u}{2}, u, 2u, 4u\}$, where $u = \frac{1}{2d}$, a grid-search procedure with cross-validation is performed on the candidate pairs in $\mathcal{C} \times \mathit{\Gamma}$.

[7] We use version 3.6.9 of WEKA.

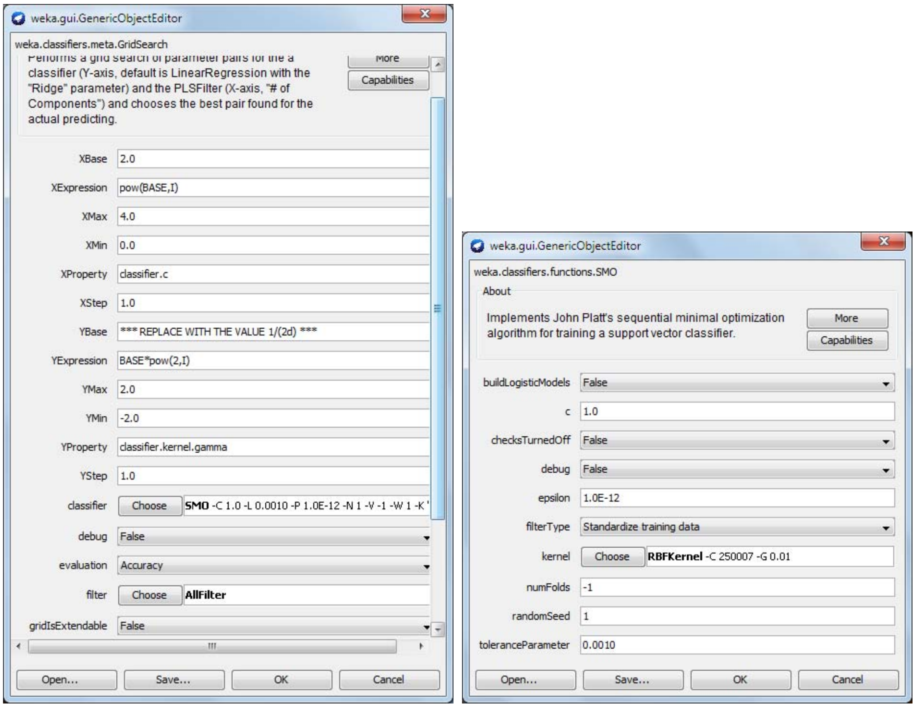
These two steps can be easily carried out using WEKA on a wide variety of datasets (including datasets with categorical features, which are automatically transformed into numerical ones by the SMO implementation in WEKA). After loading the training and the test set into WEKA, we choose the meta-classifier `GridSearch`. Figure 3 shows how to configure `GridSearch` to perform our parameter selection procedure for SVM. Among these settings, there is an option that enables data standardization as described previously.

Alternatively, the user may download the configuration file at

“`http://www.icmc.usp.br/~igorab/gridSearchSMO.conf`”.

It can be loaded using the `Open` button on the configuration window of `GridSearch`. After that, it remains to set the option `YBASE` to the value $\frac{1}{2d}$, where d is the number of features (excluding the class attribute) in his/her dataset.

Now, we illustrate our procedure for parameter selection in WEKA. For comparison, we run the same experiments using the candidate values suggested in [2] for a coarse grid-search procedure over a large range of values. The candidates



weka.classifiers.meta.GridSearch

Performs a grid search of parameter pairs for the a classifier (Y-axis, default is LinearRegression with the "Ridge" parameter) and the PLSFilter (X-axis, "# of Components") and chooses the best pair found for the actual predicting.

XBase	2.0
XExpression	pow(BASE,I)
XMax	4.0
XMin	0.0
XProperty	classifier.c
XStep	1.0
YBase	*** REPLACE WITH THE VALUE 1/(2d) ***
YExpression	BASE*pow(2,I)
YMax	2.0
YMin	-2.0
YProperty	classifier.kernel.gamma
YStep	1.0
classifier	Choose **SMO** -C 1.0 -L 0.0010 -P 1.0E-12 -N 1 -V -1 -W 1 -K '
debug	False
evaluation	Accuracy
filter	Choose **AllFilter**
gridIsExtendable	False

weka.classifiers.functions.SMO

About

Implements John Platt's sequential minimal optimization algorithm for training a support vector classifier.

buildLogisticModels	False
c	1.0
checksTurnedOff	False
debug	False
epsilon	1.0E-12
filterType	Standardize training data
kernel	Choose **RBFKernel** -C 250007 -G 0.01
numFolds	-1
randomSeed	1
toleranceParameter	0.0010

Fig. 3. Configuration options for instantiating the proposed procedure of SVM parameter selection using the `GridSearch` meta-classifier. Note that both standardization and the RBF kernel should be enabled on SMO configuration window.

Table 4. Results of SVM classification after parameter selection using the coarse grid-search suggested in [2] (110 candidate pairs) and the procedure proposed in Section 4 (25 candidate pairs)

	Coarse Grid-Search				Our Procedure			
Dataset	Error	B.Error	AUC	(C,γ)	Error	B.Error	AUC	(C,γ)
Monk-1	0.000	0.000	1.000	$(2^{15}, 2^{-03})$	0.000	0.000	1.000	$(2^{4}, 1.67\,\mathrm{E}^{-1})$
Monk-2	0.184	0.246	0.754	$(2^{15}, 2^{-01})$	0.150	0.172	0.828	$(2^{4}, 3.33\,\mathrm{E}^{-1})$
Parkinson	0.076	0.100	0.900	$(2^{15}, 2^{-03})$	0.076	0.100	0.900	$(2^{4}, 9.10\,\mathrm{E}^{-2})$
Ionosphere	0.067	0.071	0.929	$(2^{15}, 2^{-03})$	0.101	0.110	0.890	$(2^{4}, 1.47\,\mathrm{E}^{-2})$
Monk-3	0.000	0.000	1.000	$(2^{15}, 2^{-03})$	0.007	0.006	0.994	$(2^{4}, 8.33\,\mathrm{E}^{-2})$
Arcene	0.191	0.193	0.807	$(2^{15}, 2^{-15})$	0.191	0.189	0.811	$(2^{4}, 1.25\,\mathrm{E}^{-5})$
Breast-colon	0.019	0.019	0.981	$(2^{01}, 2^{-15})$	0.019	0.019	0.981	$(2^{3}, 1.14\,\mathrm{E}^{-5})$
Lung-uterus	0.129	0.130	0.870	$(2^{15}, 2^{-15})$	0.118	0.118	0.882	$(2^{4}, 1.14\,\mathrm{E}^{-5})$

are $\mathcal{C} = \{2^{-5}, 2^{-3}, \ldots, 2^{15}\}$ and $\Gamma = \{2^{-15}, 2^{-13}, \ldots, 2^{3}\}$, which amounts to 110 parameter pairs. Everything else is kept intact, including data standardization.

In addition to the datasets used in Section 3, we use another four datasets, all of them with no missing attribute values: Monk-3, from the same domain as Monk-1 and Monk-2, consisting of 432 examples (47% $\oplus$,53% $\ominus$) described by 7 features; Arcene [10], consisting of 200 examples (44% $\oplus$, 56% $\ominus$) described by 10000 features; Breast-colon, consisting of 630 examples (54% $\oplus$, 46% $\ominus$), and Lung-uterus, consisting of 250 examples (50% $\oplus$, 50% $\ominus$), both described by 10937 features and fetched from the Gene Expression Machine Learning Repository (GEMLeR[8]). As before, each dataset is randomly split in a training set containing 2/3 of the examples and a test set containing 1/3 of the examples.

Note from Table 4 that in most datasets our procedure of parameter selection yields practically equal or better accuracy than the coarse grid-search procedure of [2]. Only in one dataset, Ionosphere, does our procedure perform worse. On the other hand, our procedure performs better in Monk-2. The small improvements observed in Arcene and Lung-uterus can be credited to the fine-grained nature of our parameter search.

The selected value of γ for each dataset is not so different across the two selection procedures, which shows that our strategy for selecting candidates of γ is working well. However, for Ionosphere, our strategy did not include a candidate of γ as good as the best one found by the coarse grid-search procedure. Nevertheless, when dealing with large datasets, our strategy is justified even if the results are not optimal, since a more exhaustive search may be unfeasible.

It is important to observe that in [2] it is advocated to perform a fine-grained grid-search with candidates revolving around the best parameter found by the coarse grid-search procedure. What we are effectively showing with our results is that such a coarse grid-search procedure, which is time consuming, can be avoided by previously selecting the candidates through a careful analysis of how the parameters affect SVM classification.

[8] `http://gemler.fzv.uni-mb.si/`

6 Conclusion

In this work we proposed an economic grid-search procedure for SVM parameter selection using Gaussian RBF kernels. This procedure works by first standardizing the data and then using probabilistic assumptions to select five candidates of the generalization parameter C and five candidates of the RBF parameter γ. By considering all combinations of candidates, our procedure investigates 25 candidate pairs, which is a small number compared to the number of candidate pairs usually suggested in the literature.

We instantiated our procedure in the WEKA data mining toolbox, making it very easy to be used by non-experts. Using the same environment, we conducted experiments on several datasets. The results show that our procedure for parameter selection was as effective as a traditional grid-search procedure, though using less computational resources.

As future work, we intend to investigate a smarter way of selecting parameter candidates of C. Currently, the candidates of C do not change when we consider different candidates of γ. However, we believe it is possible to select better and fewer candidates of C if this selection is done after we fix a candidate of γ.

Acknowledgments. We would like to thank the anonymous reviewers for their helpful comments. This work is supported by grant #2009/17773-7, São Paulo Research Foundation (FAPESP).

References

1. Cortes, C., Vapnik, V.: Support vector networks. Machine Learning 20(3), 273–297 (1995)
2. Hsu, C.W., Chang, C.C., Lin, C.J.: A practical guide to support vector classification. Technical report (2003), `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`
3. Ben-Hur, A., Weston, J.: A user's guide to support vector machines. Methods in Molecular Biology 609, 223–239 (2010)
4. Witten, I.H., Eibe, F., Hall, M.H.: Data Mining: Practical Machine Learning Tools and Techniques, 3rd edn. Morgan Kaufmann Publishers Inc. (2010)
5. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2(2), 121–167 (1998)
6. Vapnik, V.: Statistical Learning Theory. Adaptive and Learning Systems for Signal Processing, Communications, and Control. Wiley-Interscience (1998)
7. Steinwart, I.: On the influence of the kernel on the consistency of support vector machines. Journal of Machine Learning Research 2, 67–93 (2001)
8. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology
9. Collobert, R., Bengio, S.: SVMTorch: Support vector machines for large-scale regression problems. Journal of Machine Learning Research 1, 143–160 (2001)
10. Bache, K., Lichman, M.: UCI machine learning repository (2013)
11. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. MIT Press (1999)
12. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters 27(8), 861–874 (2006)