Pattern Recognition Letters 88 (2017) 6–11

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Empirical comparison of cross-validation and internal metrics for tuning SVM hyperparameters

Edson Duarte, Jacques Wainer*

Computing Institute, University of Campinas, Av. Albert Einstein 1251 Cidade Universitaria Zeferino Vaz, Campinas, SP 13083-852, Brazil

ARTICLE INFO

Article history:
Received 1 September 2016
Available online 16 January 2017

Keywords:
SVM
Internal metrics
Cross validation
Hyper-parameter tuning
Model selection

ABSTRACT

Hyperparameter tuning is a mandatory step for building a support vector machine classifier. In this work, we study some methods based on metrics of the training set itself, and not the performance of the classifier on a different test set - the usual cross-validation approach. We compare cross-validation (5-fold) with Xi-alpha, radius-margin bound, generalized approximate cross validation, maximum discrepancy and distance between two classes on 110 public binary data sets. Cross validation is the method that resulted in the best selection of the hyper-parameters, but it is also the method with one of the highest execution time. Distance between two classes (DBTC) is the fastest and the second best ranked method. We discuss that DBTC is a reasonable alternative to cross validation when training/hyperparameter-selection times are an issue and that the loss in accuracy when using DBTC is reasonably small.

1. Introduction

Support Vector Machines (SVMs) are commonly used in classification problems with two or more classes. In its general formulation, SVM works by mapping input data (x) into a high-dimensional feature space ($\phi(x)$), and building a hyperplane ($f(x) = w \cdot \phi(x) + b$) to separate examples from two classes. For a L1 soft-margin SVM, this hyperplane is defined by solving the primal problem:

$$\min \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to } y_i(w \cdot \phi(x_i) + b) \geq 1 - \xi_i$$
$$\text{with } \xi_i \geq 0 \text{ and } 1 \leq i \leq n \quad (1)$$

where x_i is a data example, and $y_i \in \{-1, 1\}$ its label/class. Computationally this problem is solved on its dual form:

$$\min \frac{1}{2}\alpha^T K\alpha - e^T\alpha$$
$$\text{subject to } y^T\alpha = 0$$
$$\text{with } 0 \leq \alpha_i \leq C \text{ and } 1 \leq i \leq n \quad (2)$$

where e is a vector of ones and $K_{ij} = k(x_i, x_j) = \phi(x_i)^T \cdot \phi(x_j)$ is a kernel function that performs the ϕ mapping. The Gaussian Radial Basis Function (RBF) kernel is a common choice for the kernel function

$$k(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2).$$

The hyper-parameters C and γ must be defined before solving the minimization in Eq. (2) and must be carefully chosen to obtain a good accuracy. Choosing C and γ, known as *hyper-parameter tuning* or *model selection*, is usually done by performing a grid search over pairs of C and γ and testing each pair using some cross-validation procedure. Examples of cross validation procedures are: k-fold, repeated k-fold, bootstrap, leave one out, hold-out, among others.

Formally, if $\mathcal{D}$ is the data set, cross validation procedures will define a set of pairs of sets TR_i and TE_i, called train and test, such that:

$$TR_i \cap TE_i = \emptyset$$
$$TR_i \cup TE_I \subseteq \mathcal{D}$$

Let us use the notation

$$\text{acc}(\mathcal{B}|\mathcal{A}, C, \gamma)$$

to indicate the accuracy on a data set $\mathcal{B}$ of a SVM trained on data set $\mathcal{A}$ with hyperparameters C and γ.

Then the cross-validation of pair C, γ (for a data set $\mathcal{D}$ under some cross validation procedure) is:

$$acc_{cv}(C, \gamma) = \text{mean}_i \quad \text{acc}(TE_i|TR_i, C, \gamma)$$

The best set of hyper-parameters, from a set of candidates S is:

$$\text{argmax}_{C,\gamma \in S} \quad acc_{cv}(C, \gamma) \quad (3)$$

* Corresponding author.
E-mail addresses: e145892@dac.unicamp.br, wainer@ic.unicamp.br (J. Wainer).

http://dx.doi.org/10.1016/j.patrec.2017.01.007
0167-8655/

The expression $C, \gamma \in S$ indicates that we are selecting the best C an γ from a pre-defined set of candidates (see Section 2.2).

This usual cross-validation procedure may be too costly, and there has been many proposals to consider measures of the training set itself as the ones that are maximized or minimized in order to select one or both hyper-parameters. We will call them *internal metrics*[9]; call them *performance measures*, [3] call then *in-sample methods*, [1] call them *methods based on the Statistical Learning Theory*.

If $\psi(\mathcal{D}, C, \gamma)$ is one of these internal metrics, when applied to the set $\mathcal{D}$ with hyper-parameters C and γ, then Eq. (3) would be:

$$\text{argmax}_{C,\gamma \in S} \quad \psi(D, C, \gamma)$$

Again we are selecting the best hyperparameters from a pre-defined set of pairs S.

One of the ideas behind using internal metrics to select hyper-parameters is that the cost of selecting of hyper-parameters will probably be lower. For each pair of hyper-parameters, the cross-validation method has to learn a SVM for each of the TR_i sets, while the internal metrics method requires only one learning step, for the whole D. Furthermore, some of the internal metrics are convex functions on the hyper-parameters, and thus a gradient descent method can be used to select the hyper-parameters, instead of a grid search. This may also reduce the execution time of the whole learning process.

The aim of this paper is to replicate and update the works of [9] and [1] (discussed below), which tested some internal metric procedures against cross-validation procedures on a few data sets (5 and 13 respectively). We will perform a similar comparison of 6 internal metrics, one of which has not been used by the previous research, on 110 data sets. We will compare not only the quality of the hyper-parameter selection but also the execution time for such procedure.

1.1. Related literature

Duan et al. [9] compare the following internal metrics:

- Xi-Alpha bound [12]
- Generalized approximate cross-validation [19]
- Approximate span bound [18]
- VC bound [17]
- Radius-margin bound [5]
- Modified radius-margin bound [7]

with a standard cross validation procedure to select hyper-parameters on 5 different data sets. Each data set was split into a training subset, with 400 to 1300 data points. The quality of the choice of hyper-parameters was the accuracy of the classifiers on the remaining test set.

Duan et al. [9] concluded that the cross validation procedures result in better classifiers. They also concluded that Xi-Alpha results in reasonable choices of the hyper-parameters in the sense that the resulting classifier had an accuracy close to that of the cross-validation classifier, but the choices of hyper-parameters were not close to the ones chosen by the cross-validation. They also concluded that approximate span and VC bound do not result in high accuracy classifiers, and that (unmodified) radius bound also do not result in good hyper-parameter selections. The two remaining methods, modified radius-margin and generalized approximate cross-validation result in choices worse than that of Xi-Alpha.

Anguita et al. [1] analyzed 5 cross-validation procedures (100 repetitions of bootstrap, 10 repetitions of bootstrap, 10-fold CV, leave-one-out, and 70%/30% split for training/test), and two internal metrics (compression bound [11] and maximal discrepancy [4]). They tested the procedures on 13 data sets, and discovered that all cross-validation procedures and maximal discrepancy metric were able to select appropriate hyper-parameters.

This research replicates those researches, but we use 110 binary data sets from the UCI; we do not test the VC bound, approximate span, and unmodified radius margin bounds, given the negative results of [9], and we include maximal discrepancy [4] and distance between two classes [16] as new internal metrics to be compared. We also compare the execution time of the procedures, since one of the intended goals of internal metrics methods is a faster selection procedure for hyper-parameters.

We must point out that this research does not cover all proposed internal metrics to select hyperparameters. As discussed above, we removed from consideration some of the older proposal that were shown by other experiments to perform worse than the others such as VC bounds, approximate span, and unmodified radius margin bounds. We also do not cover a recent internal metric proposed in [14] based on stability of the models[14]. show the efficacy of the metric in selecting hyperparameters in the cases where $d \gg n$, that is, high dimensional data sets with few data points, which are not the cases tested in this paper. Another proposal not tested herein was the heuristics to select C and γ proposed in [13]: γ is selected based on the Euclidian distance (in the data space) between random samples of both classes, and C is selected from the distribution of values $\frac{1}{K(x_i,x_j)}$ for a small sample of data.

In the remaining of this paper we show a brief review of the methods in Section 2; the results are presented in Section 3 and conclusions in Section 4.

2. Methods

Next, we briefly review the internal metrics and the experimental setup.

2.1. Selection metrics

2.1.1. Cross-validation

Cross-validation (CV) is the most common method for tuning the hyper-parameters. In these experiments, we used 5-fold as the cross-validation procedure.

We performed the 5-fold CV procedure on a grid of 110 pairs of C and γ. We followed the traditional grid used in LIBSVM [6]:

$$\begin{aligned} C &= 2^{\{-5,-3,-1,1,3,5,7,9,11,13,15\}} \\ \gamma &= 2^{\{-15,-13,-11,-9,-7,-5,-3,-1,1,3\}} \end{aligned} \tag{4}$$

2.1.2. Distance between two classes

Distance between two classes (DBTC) [16] is a method that uses the idea that the hyper-parameters C and γ are independent, and thus can be chosen separately. The method proposes selecting the kernel hyper-parameter (γ) as to maximize the distance between the mean from each of the classes (calculated in the transformed feature space). Once γ is selected to maximize DBTC, C is chosen using a cross-validation procedure. The DBTC is defined as:

$$\begin{aligned} \text{DBTC} = &\frac{1}{n_1^2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_1} k(x_{1i}, x_{1j}) + \frac{1}{n_2^2}\sum_{i=1}^{n_2}\sum_{j=1}^{n_2} k(x_{2i}, x_{2j}) \\ &- \frac{2}{n_1 n_2}\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} k(x_{1i}, x_{2j}) \end{aligned} \tag{5}$$

where n_i is the number of examples, and x_{ij} is one of the examples of the ith class.

We used the γ grid in Eq. (4) to select a $\hat{\gamma}$ that maximizes Eq. (5), and with that $\hat{\gamma}$ we used a 5-fold CV to select the C hyper-parameter.

2.1.3. Maximal discrepancy

Maximal discrepancy (MD) [4] works by dividing the original training set D equally into 2 subsets S_1 and S_2. For the S_2 subset, the labels are flipped, so a data originally classified as $+1$ receives the label -1 (and -1 receives a label $+1$) on S_2. Let us define the error rate of the training data itself T as:

$$\epsilon(T) = \frac{1}{n}\sum_{i\in T} l(f_T(x_i), y_i)$$

where the dataset T has n data points, (x_i, y_i) are the data points of T, $f_T(x_i)$ is the prediction of the SVM trained using the data T when applied to the data x_i, and $l(f_T(x), y)$ is a loss function such that $l(f_D(x), y) = 1$ if $f_D(x)y \leq 0$ and 0 otherwise.

The maximum discrepancy computes two error rates, one for the original dataset D and one for the union of S_1 and S_2.

$$MD = \epsilon(D) - 2\epsilon(S_1 \cup S_2) \tag{6}$$

Note: the term $1 - 2\epsilon(S_1 \cup S_2)$ is called the maximum discrepancy, and is a term that should be maximized, but for simplicity we named MD the term to be minimized in Eq. (6).

Anguita et al. [2] propose a variation of MD, where the MD is the mean of 100 repetitions of the procedure above. We did implement this variation but with only 10 repetitions, which is refer as MD10.

The MD is calculated for each pair of C and γ in grid defined by Eq. (4), for the whole training subset, and the pair that minimizes the MD is selected.

2.1.4. Generalized approximate cross-validation

Generalized approximate cross-validation (GACV) is a method developed by Wahba et al. [19] as an approximation for the generalized Kullback–Leibler distance (GCKL) that does not require knowing the probability that given a x_i, $y_i = 1$. GACV is defined as:

$$\text{GACV} = \frac{1}{n}\left[\sum_{i=1}^{n}\xi_i + 2\sum_{y_i f(x_i)<-1}\alpha_i K_{ii} + \sum_{y_i f(x_i)\in[-1,1]}\alpha_i K_{ii}\right].$$

The GACV is calculated for each pair of C and γ in the grid defined in Eq. (4), for the whole training subset, and the pair that minimizes the GACV is selected.

2.1.5. Xi-alpha bound

Xi-alpha bound (XAB) is an error estimate proposed byJoachims [12]. It is computed using ξ and α from the solutions of the primal (1) and dual problems (2):

$$\text{XAB} = \frac{1}{n}\text{card}\{i : (2\alpha_i\delta + \xi_i) \geq 1\}$$

where *card* means cardinality and δ is a number such as $1 \leq k(x_i, x_j) \leq 1 + \delta$.

The XAB is calculated for each pair of C and γ in Eq. (4), for the whole training subset, and the pair that minimizes the XAB is selected.

2.1.6. Radius/margin bound

Radius/Margin bound (RMB) [7], is a method that uses an sphere containing all the $\phi(x)$ to calculate the measurement error

$$\text{RMB} = \left(R^2 + \frac{\Delta}{C}\right)\left(||w||^2 + 2C\sum_{i=1}^{n}\xi_i\right)$$

where R is the radius of the smallest sphere containing every $\phi(x)$ and Δ is a constant close to 1.

The RBM is calculated for each pair of C and γ in Eq. (4), for the whole training subset, and the pair that minimizes the RBM is selected.

2.2. Grid search, not gradient descent

Most of the internal metric discussed above were proposed in the context of a gradient descent search for the hyper-parameters, that is, they are smooth functions and the authors proposed and some tested their metric with a gradient descent search. We are not following that approach in this paper. Instead we perform a grid search with γ in the grid defined in Eq. (4).

There is a two-fold reason for using grid search on γ. A gradient descent search will depend on the particular optimization algorithm (steepest descent, Newton–Raphson, conjugate gradient, and so on) and on particular parameters for each of the optimization algorithms (stoping criteria, for all algorithms, and learning rate for the steepest descent, for example) which add variability to analysis—if an internal metric does not perform well is it an intrinsic property of the metric or an inadequacy of the gradient descent algorithm? In our experimental set up we are evaluating only the intrinsic properties of the metric itself, in particular how well it approximates the response surface of the SVM as a function of the hyper-parameters.

The second reason is that we are also measuring the execution time of the different solutions. Gradient descent algorithms could run for a very long time depending on the learning rate and stoping criteria, and that could be unfair for some of the internal metrics.

2.3. Data sets

We used the data sets described in [20]. In summary, the data sets are derived from the ones collected by [10] from the publicly available data sets from UCI. From the 121 data sets from [10], we removed the ones with 100 data points or less, and in this research, we also remove the largest one `miniboone`, resulting in 110 datasets. All data sets that were multi-class were converted to binary (as described in [20]).

2.4. Experimental setup

Each data set was split into 2 equal sized random subsets. We ran the 6 selection procedures in one of the subsets, and computed the accuracy of an SVM that was trained on that subset, with the hyper-parameters selected by the procedures and tested on the other subset. We then repeated the procedure for the other subset. The accuracy of the procedure on that data set was the average of the two measured accuracies.

The experiments were written in the Python programming language using the SVM implementation from Pedregosa et al. [15]. The calculations of each of the internal metrics, with the exception of the Radius margin was implemented in Python. For the radius margin bound we used the implementation from Chung et al. [7] publicly available[1] with some modifications due to their use of deprecated Python features.

2.5. Statistical procedures

To analyze the accuracy and execution times of each selection procedure, we used the method proposed by [8]: a Friedman test to reject the hypothesis that all methods perform equally well, followed by a Nemenyi test to perform the pairwise comparisons.

We also report the ranking of each method regarding both accuracy and execution time. The rank is dense, that is all procedures that have the highest same accuracy receive rank 1, all procedures

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/#radius_margin_bounds_for_svm_model_selection_ .

Table 1
Comparison of the selection procedures. The first column is the selection procedure, the second the mean rank of the accuracy, and the third the mean rank of the execution time.

selection method	mean rank accuracy	mean rank execution time
CV	1.69	6.10
DBTC	2.75	1.13
GACV	3.15	2.98
MD	3.54	2.93
MD10	3.86	6.80
XAB	4.18	3.30
RMB	4.48	4.75

Table 2
P values of the Nemenyi pairwise comparisons of accuracy between selection methods.

	CV	DBTC	GACV	MD	MD10	XAB
DBTC	0.00					
GACV	0.00	0.65				
MD	0.00	0.03	0.75			
MD10	0.00	0.00	0.11	0.90		
XAB	0.00	0.00	0.00	0.23	0.91	
RMB	0.00	0.00	0.00	0.01	0.28	0.93

Table 3
P values of the Nemenyi pairwise comparisons of execution time between selection methods.

	CV	DBTC	GACV	MD	MD10	XAB
DBTC	0.00					
GACV	0.00	0.00				
MD	0.00	0.00	1.00			
MD10	0.21	0.00	0.00	0.00		
XAB	0.00	0.00	0.93	0.88	0.00	
RMB	0.00	0.00	0.00	0.00	0.00	0.00

that have the second highest accuracy receive the rank 2 and so on.

3. Results

Table 1 displays the mean rank regarding accuracy for each of the 6 selection procedures and the mean rank regarding the execution time. The table is ordered by the mean accuracy rank.

The Friedman test of the 6 procedures results in a p-value$<$ $2.2e-16$ (Friedman chi-squared $= 296.05$, df $= 6$). Tables 2 and 3 display the p-values of the Nemenyi pairwise comparisons between the selection methods, for accuracies and for the execution times.

In agreement with the previous literature [1,4,9], CV selection results in highest accuracies, and significantly so. In particular, CV is statistically significantly better than all the other methods (first column of Table 2). The second best method DBTC is not significantly better than the third GACV (p.value=0.65), but it is significantly better than the others (second column of Table 2). Most differences in accuracy regarding the different methods are significant, with a few exceptions.

Interestingly, the second best rated selection procedure, DBTC is the fastest one and the differences are significant in relation to all other methods (third column of Table 3). Therefore we believe DBTC is a reasonable alternative selection procedure if time is of concern.

If one would use DBTC as the selection method, one could expect a median decrease in accuracy of 0.0097, and a mean decrease in accuracy of 0.0322. Fig. 1 is the histogram of the change in accuracy by using DBTC instead of the 5-fold CV. Regarding execution time, the median ratio of the DBTC in relation to 5-fold CV is

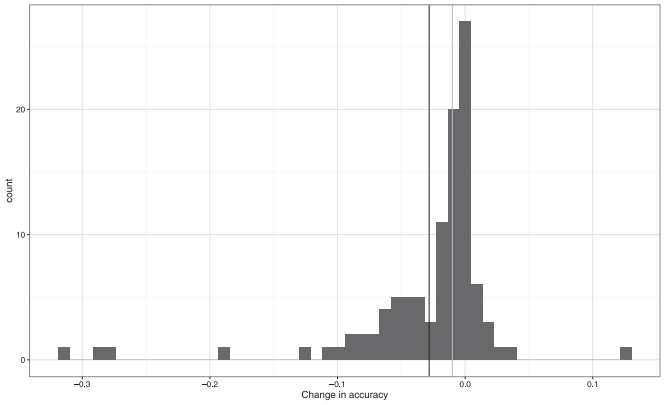

Fig. 1. Distribution of the change on accuracy as a consequence of using DBTC as the selection method. Lighter vertical line is the median and darker vertical line is the mean of the distribution.

0.146, the median is 0.165. That is, on average the DBTC will select the hyper-parameters in 16% of the time needed for the 5-fold CV.

3.1. Further analysis of the methods

In this section we examine the difference between the accuracy achieved by each of the internal metrics and the CV procedure, as it relates to the data set characteristics. The accuracy gain of an internal metric procedure on a particular data set is the accuracy achieved by the procedure minus the accuracy of the CV procedure on that data set.

Fig. 2 plots the accuracy gain of each procedure as a function of the data set size. And Fig. 3 plots the accuracy gain as a function of the difficulty of the dataset itself, measured by the CV accuracy.

Fig. 2 shows that there is a dependency of the accuracy gain of the MD, XAB, and RMB procedures with the data set size; and that these procedures perform better with smaller data sets. This is in agreement with [2] that argues that internal metrics could be an important selection method for smaller data sets, especially in the cases where some of the classes are so small that they cannot be reliable divided into k folds with the same number of classes. In other words, for smaller data sets, the variance of the CV estimate of the accuracy (when selecting the hyperparameters) is so high that internal metrics may be competitive. There seems to be no important dependency of the accuracy gain of DBTC and GACV in relation to the data set size.

There seems to be no dependency of DBTC and XAB on the difficulty of the problem, where as the other methods seems to perform worse as the problem becomes easier. As far as the authors know, this phenomenon has not yet been discussed in the literature. We do not have an explanation for this dependency.

4. Conclusion

In accordance to the previous literature [1,4,9], cross validation is the selection procedure to tune SVM hyper-parameters that will have lower expected error on future data.

When the selection time is a concern, we would recommend the use the distance between two classes (DBTC) method. As described, the method first selects the γ (using a 1D grid) that maximizes Eq. (5) (which represent the distance between the center of each class in the feature space) and then with that γ selects the C using cross validation. If one accepts the arguments of [20] that changes in accuracy bellow 0.0112 should be considered as irrelevant (since this is the median change one should expect in accuracy when a good classifier for the data sets in this experiment

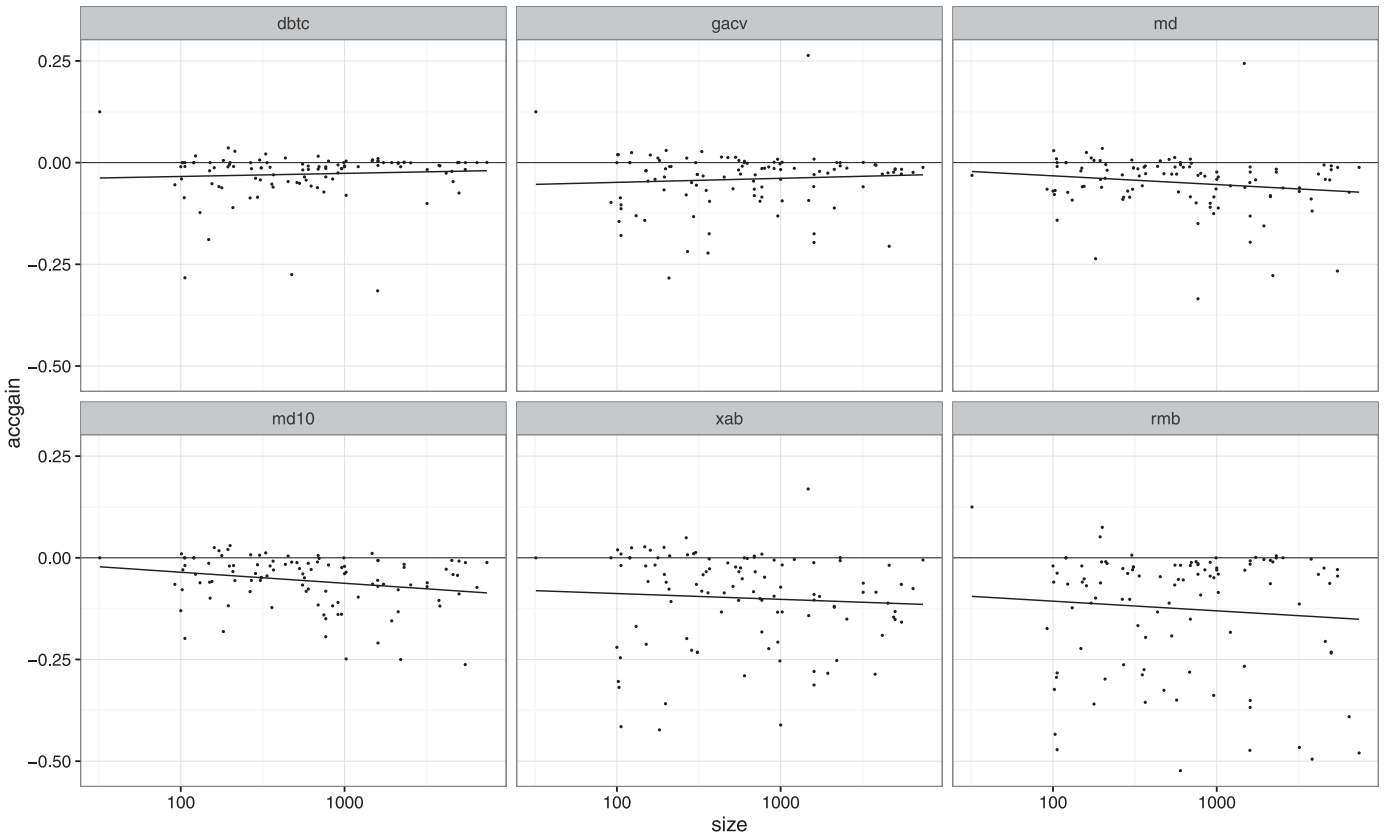

Fig. 2. Accuracy gain of each of the internal metrics in relation to CV, as a function of the data set size. The line represent a linear regression of the data.

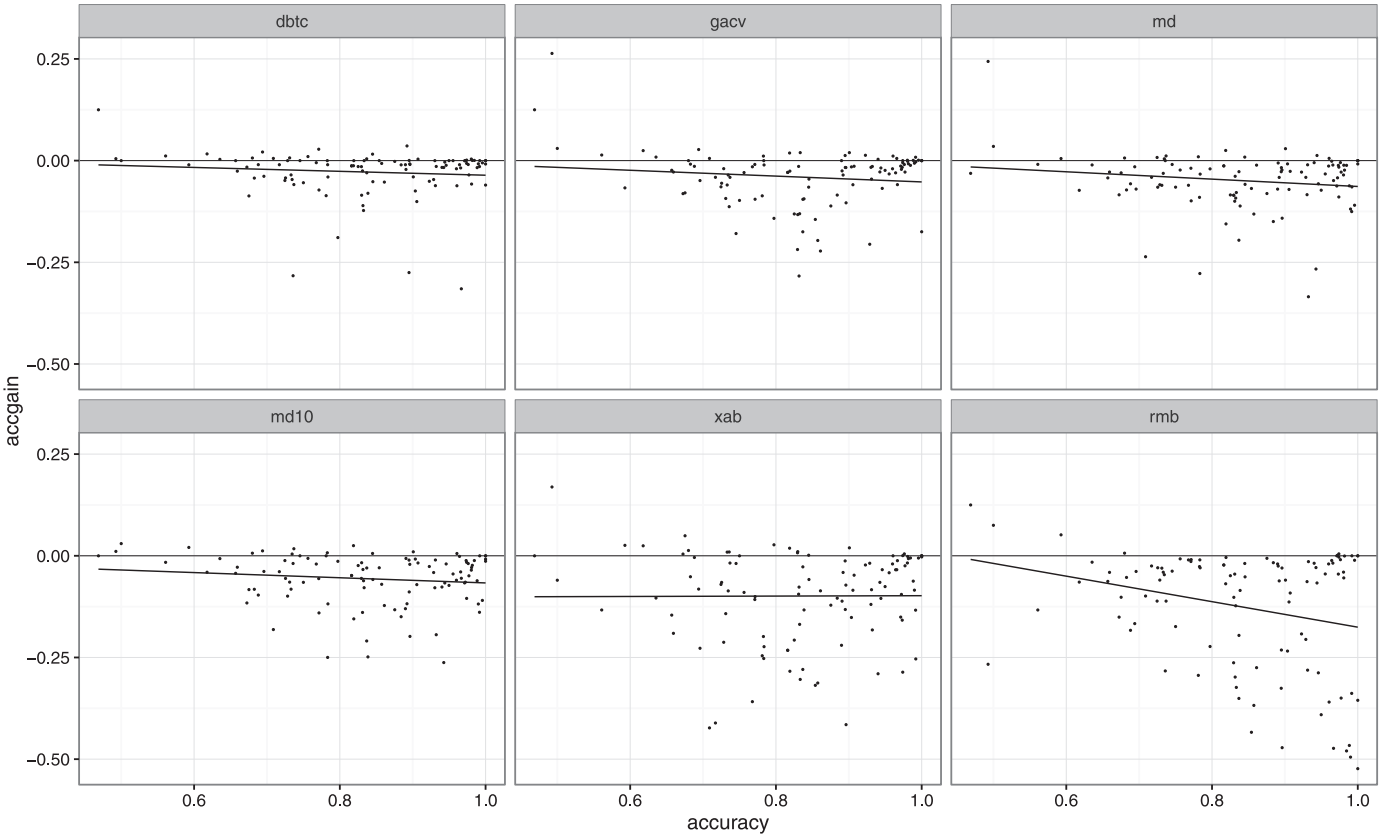

Fig. 3. Accuracy gain of each of the internal metrics in relation to CV, as a function of the data set difficulty as measured by the CV accuracy. The line represent a linear regression of the data.

are faced with new data), then in 50% of the time, the change in accuracy when changing from 5-fold CV to DBTC will be irrelevant.

References

[1] D. Anguita, A. Boni, S. Ridella, F. Rivieccio, D. Sterpi, Theoretical and practical model selection methods for support vector classifiers, in: Support vector machines: theory and applications, Springer, 2005, pp. 159–179.

[2] D. Anguita, A. Ghio, N. Greco, L. Oneto, S. Ridella, Model selection for support vector machines: advantages and disadvantages of the machine learning theory, in: IEEE International Joint Conference on Neural Networks, 2010, pp. 1–8.

[3] D. Anguita, A. Ghio, L. Oneto, S. Ridella, In-sample and out-of-sample model selection and error estimation for support vector machines, IEEE Trans. Neural Netw. Learn. Syst. 23 (2012) 1390–1406.

[4] D. Anguita, S. Ridella, F. Rivieccio, R. Zunino, Hyperparameter design criteria for support vector classifiers, Neurocomputing 55 (2003) 109–134.

[5] C. Campbell, N. Cristianini, J. Shawe-Taylor, Dynamically adapting kernels in support vector machines, Adv. Neural Inf. Process. Syst. 11 (1999) 204–210.

[6] C.C. Chang, C.J. Lin, Libsvm: a library for support vector machines, ACM Trans. Intell. Syst. Technol. (TIST) 2 (2011) 27.

[7] K. Chung, W. Kao, C. Sun, L. Wang, C. Lin, Radius margin bounds for support vector machines with the RBF kernel, Neural Comput. 15 (2003) 2643–2681.

[8] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.

[9] K. Duan, S.S. Keerthi, A.N. Poo, Evaluation of simple performance measures for tuning svm hyperparameters, Neurocomputing 51 (2003) 41–59.

[10] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. 15 (2014) 3133–3181.

[11] S. Floyd, M. Warmuth, Sample compression, learnability, and the Vapnik–Chervonenkis dimension, Mach. Learn. 21 (1995) 269–304.

[12] T. Joachims, The Maximum-Margin Approach to Learning Text Classifiers: Methods, Theory, and Algorithms, Dortmund University, 2000 Ph.d. thesis. doctoral dissertation.

[13] B. Milenova, J. Yarmus, M. Campos, SVMin Oracle database 10g: removing the barriers to widespread adoption of support vector machines, in: 31st International Conference on Very Large Data Bases, VLDB Endowment, 2005, pp. 1152–1163.

[14] L. Oneto, A. Ghio, S. Ridella, D. Anguita, Fully empirical and data-dependent stability-based bounds, Trans. Cybern. 45 (2015) 1913–1926.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[16] J. Sun, C. Zheng, X. Li, Y. Zhou, Analysis of the distance between two classes for tuning svm hyperparameters, IEEE Trans. Neural Netw. 21 (2010) 305–318.

[17] V. Vapnik, Statistical Learning Theory, Wiley New York, 1998.

[18] V. Vapnik, O. Chapelle, Bounds on error expectation for support vector machines, Neural Comput. 12 (2000) 2013–2036.

[19] G. Wahba, G. Wahba, Y. Lin, Y. Lin, Y. Lee, Y. Lee, H. Zhang, H. Zhang, On the Relation Between the GACV and Joachims' Xi-alpha Method for Tuning Support Vector Machines, With Extensions to the Non-Standard Case, Technical report 1039, Statistics Department University of Wisconsin, Madison WI, 2001.

[20] J. Wainer, Comparison of 14 different families of classification algorithms on 115 binary datasets. arxiv e-prints 1606.00930. arxiv:1606.00930, 2016.