BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 74(1), 2026, Article number: e155891
DOI: 10.24425/bpasts.2025.155891

Data structure better than labels? Unsupervised heuristics for SVM hyperparameter estimation

Michał CHOLEWA*, Michał ROMASZEWSKI, and Przemysław GŁOMB

Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Bałtycka 5, 44-100, Gliwice, Poland

Abstract. Classification is one of the main areas of pattern recognition research, and within it, support vector machine (SVM) is one of the most popular methods outside of the field of deep learning – and a de facto reference for many machine learning approaches. Its performance is determined by parameter selection, which is usually achieved by a time-consuming grid search cross-validation procedure (GSCV). That method, however, relies on the availability and quality of labelled examples and thus, when those are limited, can be hindered. To address this problem, several unsupervised heuristics exist that utilise the characteristics of the dataset to select parameters, rather than relying on class label information. While being an order of magnitude faster, they are scarcely used under the assumption that their results are significantly worse than those of grid search. To challenge that assumption, we have surveyed several heuristics for SVM parameter selection and tested them against GSCV on over 30 standard classification datasets. The results demonstrate their high accuracy, with performance in terms of statistical significance comparable to GSCV, opening up an avenue for reliable label-free model defaults in resource-constrained settings, e.g., edge devices or rapid prototyping.

Keywords: SVM; classification; parameters selection; unsupervised.

1. INTRODUCTION

Classification is one of the most frequently encountered problems in the field of pattern recognition. It is utilised, among many other fields, in computer vision [1], document analysis [2], data science [3] and biometrics [4]. The classification itself is a broad area that encompasses both traditional machine learning methods and, more recently, increasingly popular deep learning models. However, even with the formidable results achieved by deep learning approaches, e.g., [5, 6], the classical methods still have a role to play. The high computational cost, large data volume required and the open-ended difficulty of finding a combination of a suitable architecture, hyperparameters and learning algorithm for the deep learning model are prohibitive for many current applications of pattern recognition. This situation occurs, e.g., for Internet of Things devices [7], edge computing [8], medical devices [9] or with limited training labels [10]. Additionally, classical methods – notably support vector machines – are selected for their robustness [11] or theoretical consideration [12].

Support vector machine (SVM) is a supervised classification scheme based on ideas developed by V.N. Vapnik and A.Ya. Chervonenkis in 1960s [13] and later expanded on in works such as [14–16]. It is based on computing a hyperplane that optimally separates training examples and then making classification decisions based on the position of a point in relation to that hyperplane. The SVMs have been consistently used in various roles – as an independent classification scheme, e.g., [17–19], part of more complex engines, e.g., [20, 21] or a detection engine, e.g., [22, 23]. It has also been employed in unsupervised settings, as seen in works such as [24] and [25]. This flexibility allows SVM to be one of the most frequently used machine learning approaches in medicine [26], remote sensing [10], threat detection [27], criminology [28], and is often utilized in photo, text, and time sequence analysis [29]. In numerous studies, SVM is consistently ranked as one of the top-performing methods [11].

The popularity and versatility of SVM are to a large degree due to its controllability by the key hyperparameters. The first is a label error regularization coefficient C, which balances training error and margin width. It allows us to classify non-linearly separable datasets or preserve margin width at the cost of misclassification of some training examples. The second is related to extension with the 'kernel trick' to kernel-SVM, which is much more effective in working with complex data distributions; it introduces a kernel function value computation as an extension of a dot-product. Various kernel functions have been investigated, however, overwhelming majority of applications use Gaussian radial basis function as it provides best classification performances on a large range of datasets [30] and assumes only smoothness of the data, which makes it a natural choice when knowledge about data is limited [31]. Values of these hyperparameters are typically found through supervised search procedures, cross-validation (CV) on the training set and grid-search through a range of predefined parameters [32]. However, major disadvantage of the CV is the $\mathcal{O}(n^2)$ complexity in the number of hyperparameter values to be evaluated, each requiring training of a separate model. This is a burden for performing pattern recognition in distributed edge computing devices in Industry 4.0 [8] or optimization of battery usage for mobile devices with limited connectivity, e.g., in monitoring of ageing people [9].

*e-mail: mcholewa@iitis.pl

Manuscript submitted 2025-05-13, revised 2025-08-25, initially accepted for publication 2025-08-26, published in January 2026.

An alternative for hyperparameter selection is to derive their values from a statistical analysis of the data. Those approaches range from simple 'rule of thumb' statistics, e.g. [33], to more complex approaches involving, e.g., cluster assumptions and graph distances between datapoints [34]. Through these approaches, the values of C and γ can be estimated based on the structure of the entire available dataset, in an unsupervised manner – without the requirement of labels. This is especially useful for applications that acquire a large amount of data with limited supervision, e.g., IoT devices [7]. Additionally, this estimation is a one-pass computation, which is much less intensive than cross-validation, allowing for greater applicability, e.g., in IoT/edge/medical supervision devices. Unsupervised estimation avoids the issues of optimising parameters on the same set as the one used for training, which can lead to overfitting [14]. It is known that in some cases, e.g., where classes indeed conform to the cluster assumption and Gaussian distribution [35], optimal or nearly optimal parameter values can be analytically derived from the data without knowledge of the class labels. This approach is also helpful when training data is very limited and may poorly reflect the true class distributions – a situation typically encountered in semi-supervised hyperspectral classification, e.g., [10]. The robustness of this approach has led unsupervised heuristics to be a default parameter setting in SVM programming libraries, e.g., scikit-learn [36].

In this survey, we benchmark unsupervised heuristics-based hyperparameter estimation for an SVM classifier (UH-SVM) against an SVM tuned via grid-search cross-validation (GSCV-SVM).

1. We evaluate a large set of unsupervised heuristics on a comprehensive collection of balanced and imbalanced datasets. While numerous works investigate individual heuristics, to the best of the authors' knowledge, no work collects them together and compares them with one another.
2. We show that, without specific prior knowledge of a dataset, there is a significantly higher chance of a number of UH-SVM approaches having similar or better accuracy than GSCV-SVM–in terms of statistical significance of the results–than having a worse accuracy. Considering the substantially lower computational cost of UH-SVM with respect to GSCV-SVM, this, in our opinion, validates the conclusion of UH-SVM parameter estimation being in many application cases on par with the grid search.
3. We observe that C-selection heuristics tend to underestimate the value of that parameter, which leads to lower accuracy in classification. To illustrate this, we evaluate an extension of Chapelle's very effective heuristic that increases the C and obtains results practically equivalent (in terms of statistical significance, see Section 3.3) to GSCV.

2. METHODS

In the following section, we will recall both the ideas behind the support vector machines classifier and the heuristics that we include in our experiments. In some cases our unified presentation of them allows us to derive natural generalizations, e.g., a scaling of [34] in high dimensional datasets or correction for [37].

2.1. Kernel SVM

A kernel SVM [14] is a classifier based on the principle of mapping the examples from the input space into a high-dimensional feature space and then constructing a hyperplane in this feature space, with the maximum margin of separation between classes. Let $\mathscr{X} \subset \mathbb{R}^n$ be a set of data and let $\mathbf{x}_i \in \mathscr{X}, i = 1, \ldots, m$ be the set of labelled examples. Let also $\mathscr{Y} = \{-1, 1\}$ be a set of labels. We define a training set as a set of examples with labels assigned to them,

$$\mathscr{L} = \{(\mathbf{x}_i, y_i), i = 1, \ldots, m\}, \quad \mathbf{x}_i \in \mathscr{X}, \quad y_i \in \mathscr{Y}. \tag{1}$$

The SVM assigns an example $\mathbf{x} \in \mathscr{X} \subset \mathbb{R}^n$ into one of two classes using a decision function

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{m} y_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) + b\right). \tag{2}$$

Here, $\alpha_i \geq 0$ and b are coefficients computed through Lagrangian optimization – maximization of margin, or distance from hyperplane to class datapoints on the training set. Training examples $\mathbf{x}_i$ where the corresponding values of $\alpha_i \neq 0$ are called support vectors (SV). Since SVM is inherently a binary classifier, for multi-class problems several classifiers are combined, e.g., using one-against-one method [38].

2.1.1. Kernel function

The function $K : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ is called the kernel function and it is used to compute the similarity between the classified example $\mathbf{x}$ and each training instance $\mathbf{x}_i$. It is a generalization of a dot product operation used in the original linear SVM derivation, i.e., $K(\mathbf{x}, \mathbf{x}_i) = \langle \mathbf{x}, \mathbf{x}_i \rangle$, taking advantage of the 'kernel trick' [14] – a non-linear mapping $\phi : \mathscr{X} \to \mathscr{H}$ to a feature space $\mathscr{H}$ where the dot product is computed by evaluating the value $K(\mathbf{x}, \mathbf{x}_i) = \langle \phi(\mathbf{x}), \phi(\mathbf{x}_i) \rangle$. The kernel trick allows the SVM to be effectively applied in the case where classes are not linearly separable in the data space. A number of positive definite symmetric functions can be used as kernels, such as polynomial $K(\mathbf{x}, \mathbf{x}_i) = (\langle \mathbf{x}, \mathbf{x}_i \rangle + c)^k$, $c \geq 0$, $k = 1, 2, \ldots$; Laplace $K(\mathbf{x}, \mathbf{x}_i) = \exp\left(\frac{-||\mathbf{x}-\mathbf{x}_i||}{\sigma}\right)$ or Gaussian radial basis function (RBF)

$$K(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_i\|^2}{2\sigma^2}\right), \tag{3}$$

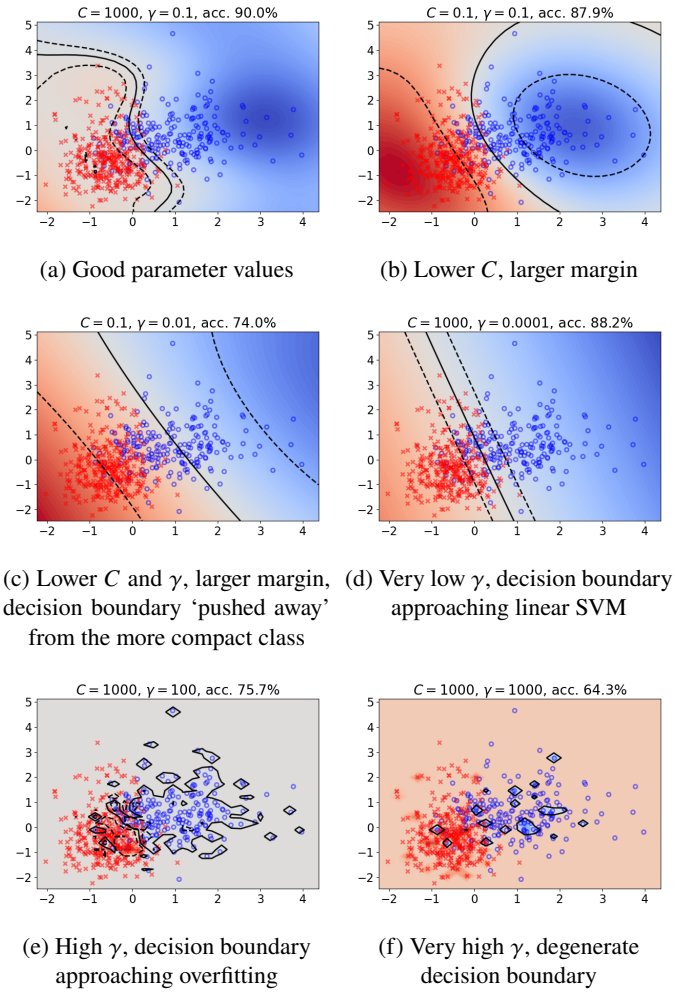
where σ^2 represents the variance of the data and $\|\cdot\|$ is an Euclidean distance in $\mathscr{X} \subset \mathbb{R}^n$. This kernel has been found to be versatile and effective for many different kinds of data [39] and it will be the focus of our research. By substituting $\gamma = \frac{1}{2\sigma^2}$, it can be written

$$K(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2\right), \tag{4}$$

where γ can be viewed as scaling factor, which is one of the parameters of the SVM classifier.

The parameter γ controls the impact of individual SV as the kernel distance between two examples decreases with higher values of γ. Therefore, small values of γ will result in many SV influencing the point under test $\mathbf{x}$, producing smooth separating

hyperplanes and simpler models. Very small values will lead to all SV having a comparable influence, making the classifier behave like a linear SVM. Large values of γ result in more complex separating hyperplanes, better fitting the training data. However, a too high value of γ may lead to overfitting (see Fig. 1).



(a) Good parameter values

(b) Lower C, larger margin

(c) Lower C and γ, larger margin, decision boundary 'pushed away' from the more compact class

(d) Very low γ, decision boundary approaching linear SVM

(e) High γ, decision boundary approaching overfitting

(f) Very high γ, degenerate decision boundary

Fig. 1. Example SVM behaviour on first two features from the 'Breast Cancer Wisconsin Dataset' (wdbc). Red crosses and blue circles mark the position of data points from two classes. Solid line presents the decision boundary, dashed lines denote margin ranges. Presented cases show the example influence of values of C and γ parameters, both for good and bad values

2.1.2. Soft margin

In practice, even using a kernel trick, a hyperplane that separates classes may not exist. Therefore, SVM is usually defined as a soft margin classifier by introducing slack variables to relax constraints of Lagrangian optimisation, which allows some examples to be misclassified. It introduces the soft margin parameter $C > 0$ where $0 < \alpha_i < C$ a constraint on α_i controlling the penalty on misclassified examples and determining the trade-off between margin maximization and training error minimization. Large values of the parameter C will result in small number of support vectors while lowering this parameter results in larger number of support vectors and wider margins (see Fig. 1).

2.2. Setting the SVM parameters

One of the early discussions about SVM parameters was provided in [14]. In chapter 7.8, the authors mentioned the grid search CV (GSCV) as a common method of SVM parameter selection. As an alternative, in order to avoid the CV, the authors suggested a number of general approaches including scaling kernel parameters such as the denominator of the RBF kernel so that the kernel values are in the same range. They also suggested that the value of the parameters C can be estimated as $C \propto 1/R^2$ where R is some measure of data variability such as standard deviation of the examples from their mean, or the maximum/average distance between examples. Model selection by searching the kernel parameter space was later discussed in [40], where authors proposed two simple heuristics based on leave-one-out CV.

Unsupervised heuristics are relatively less discussed than their supervised counterparts. A simple heuristic that estimates γ as an inverse of some aggregate (e.g., a median) of distances between data points has been proposed in a blog post [33]. In fact, when searching the Internet for a method to choose kernel parameters in an unsupervised way, this post – which refers to the idea from a thesis of B. Schölkopf – is a common find. This heuristics is similar to the 'sigest'[1] method [42]. However, even in surveys comparing heuristics for SVM parameter selection [43] when sigest is considered it is applied to the training set and complimented with cross-validation for the value of the C parameter.

Sometimes, unsupervised heuristics supplement more complex methods, e.g. in [34] authors propose a method for parameter selection inspired by the cluster assumption, based on graph distances between examples in the feature space; a heuristic for unsupervised initialisation of SVM parameters is provided as a starting point of a grid search. Another example are initialisation methods used in well-known ML libraries, e.g. scikit-learn[2] employs its own implementation of heuristic for the γ parameter [36]. Shark[3] uses the heuristic from [44] which can also be used in an unsupervised way [37].

2.2.1. Grid search cross validation

As a baseline method for model selection in this article, grid search cross validation (GSCV) [45] is used. This method is based on dividing the dataset into k parts $\{p_1, \ldots, p_k\}$ and then repeat the experiment using parts $\{p_1, \ldots, p_k\} \setminus \{p_i\}$ for training and $\{p_i\}$ for testing and averaging the results. This method allows us to mitigate the variance resulting for random train/test set selection.

In case of this research, the additional layer is used for model selection – called an internal layer. It is designed to detect the best set of parameters (C, γ) from given grid $\mathcal{G} \subset \mathbb{R}^2$. Similarly

[1] Implemented e.g. in R, see [41]

[2] https://scikit-learn.org

[3] http://www.shark-ml.org/

to external layer, each training set $\mathscr{T}_i = \{p_1, \ldots, p_k\} \setminus \{p_i\}$ is divided into t subparts $\{p_i^1, \ldots, p_i^t\}$, with $\{p_i^1, \ldots, p_i^t\} \setminus \{p_i^j\}$ used for training with given parameters from grid $\mathscr{G}$ and $\{p_i^j\}$ used for testing (hence grid search cross validation). The parameters for $\{p_i\}$ are determined by the results of this second level of cross validation.

2.3. Unsupervised heuristics for γ

Unsupervised heuristics usually assume that γ should be relative to 'average' distance (measured by $\|\cdot\|^2$) between the examples from $\mathscr{X}$, so that the two extreme situations – no SV influence or comparable influence of all SV – are avoided. For example, γ can be assigned the inverse of the data variance, which corresponds, e.g., with heuristics described in [36] or [33]). Intuitively then kernel value between two points is a function of how large is the distance between two given points compared to the average distance among the data. Differences between heuristics can be thus reduced to different interpretations of what that average distance is.

2.3.1. γ heuristics for Gaussian-distributed data

Considering a pair of examples $(\mathbf{x}_i, \mathbf{x}_j) \in \mathscr{X} \times \mathscr{X} \subset \mathbb{R}^n \times \mathbb{R}^n$ from Gaussian-distributed data, it has been noted in [35], that the squared Euclidean distance $\|\mathbf{x}_i - \mathbf{x}_j\|^2$ is Chi-squared distributed with a mean of $2n\sigma^2$, assuming that every data feature has variance σ and mean 0. This observation could be used as a heuristic to estimate the value of γ as

$$\gamma = \frac{1}{2n\sigma^2}\,. \tag{5}$$

If we further assume that $\sigma^2 = 1$, this simplifies to $\gamma = \frac{1}{2n}$, as noticed by authors of [46].

This approach relies on an underlying assumption that data covariance matrix is in the form $\mathrm{Cov}(\boldsymbol{X}) = \boldsymbol{I}\sigma^2$, which, in turn, means that in a matrix of examples $\boldsymbol{X} \in \mathbb{R}^{m \times n}$, every feature has an equal variance. In practice, data standardisation is used, which divides each feature by its standard deviation. However, the standard deviations are estimated on the training set, and on the test set will produce slightly varying values that are only approximately equal $\sigma_1 \approx \sigma_2 \approx \cdots \approx \sigma_n$. To take that into account, we use another formula for estimation of the value of γ as

$$\gamma = \frac{1}{2\,\mathrm{Tr}\,(\mathrm{Cov}(\boldsymbol{X}))}\,, \tag{6}$$

where $\mathrm{Tr}(\cdot)$ denotes a trace of a matrix. This heuristic is denoted in the experiments as *covtrace*.

2.3.2. Smola's heuristics

A well-known heuristic for computing the initial value of a parameter γ was provided by A.J. Smola in an article on his website [33]. Given examples $(\boldsymbol{x}_i, \boldsymbol{x}_j) \in \mathbb{R}^n \times \mathbb{R}^n$, he considered a kernel function in the form

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \kappa(\lambda \|\boldsymbol{x}_i - \boldsymbol{x}_j\|), \tag{7}$$

where a scaling factor λ of this kernel is to be estimated and $\kappa : \mathbb{R} \to \mathbb{R}^+$. The Smola's kernel form is consistent with the RBF kernel given by (4) – it as special case of (7), with $\kappa(x) = \exp(-x^2)$, where $x \in \mathbb{R}$ and $\lambda = \sqrt{\gamma}$.

He proposes to select a subset of (e.g., $m = 1000$) available pairs $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and to compute their distances. Then, the value of λ can be estimated as the inverse of q quantile (percentile) of distances where one of three candidates $q \in \{0.1, 0.5, 0.9\}$ is selected through cross-validation. The reasoning behind those values extends the concept of 'average' distance: the value of $q = 0.9$ corresponds to the high value of a scaling factor which results in decision boundary that is 'close' to SV, $q = 0.1$ corresponds to 'far' decision boundary, $q = 0.5$ aims to balance its distance as 'average' decision boundary. The author argues that one of these values in likely to be correct, i.e., result in an accurate classifier. Those three q values are included in the experiments as *Smola_10*, *Smola_50* and *Smola_90*.

2.3.3. Chapelle & Zien γ heuristic

A heuristic for choosing SVM parameters can be found in [34]. Interestingly, to the best of our knowledge it is the only method that estimates both C and γ in an unsupervised setting (see 2.4.1). The heuristics take into account the density of examples in the data space. Authors introduce a generalization of a 'connectivity' kernel, parametrized by $\rho > 0$, which in the case of $\rho \to 0$ defaults to the Gaussian kernel. This kernel proposition is based on minimal ρ-path distance D_{ij}^{ρ} which, for $\rho \to 0$ becomes Euclidean distance, i.e., $D_{ij}^{\rho \to 0} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$.

Authors use the cluster assumption, by assuming that data points should be considered far from each other when they are positioned in different clusters. In [34] authors consider three classifiers: Graph-based, TSVM and LDS. As this approach introduces additional parameters, which would make cross-validated estimation difficult, authors propose to estimate parameters through heuristics. The value of σ (equation (3)) is computed as $\frac{1}{n_c}$-th quantile of $\mathscr{D} = \{D_{ij}^{\rho} : \boldsymbol{X} \times \boldsymbol{X} \in \mathbb{R}^n \times \mathbb{R}^n\}$ where n_c is the number of classes. For Gaussian RBF kernel this results in

$$\gamma = \frac{1}{2\ \mathrm{quantile}_{\frac{1}{n_c}}(\mathscr{D})}\,. \tag{8}$$

Note that we consider only the case $\rho \to 0$, as only under this condition heuristics proposed in [34] are comparable with other heuristics presented in this Section and compatible with our experiment. However, the authors' original formulation allows for other values of ρ. This heuristic, along with the complimentary for the C parameter (see Section 2.4.1) is denoted in the experiments as *Chapelle*.

2.3.4. Jaakkola's and Soares' heuristics

While the original Jaakkola's heuristics, described in [44] and [47], was supervised, in this article we will focus on its unsupervised version proposed in [37].

The original heuristics based on median inter-class distance and is computed as follows: for all training examples $\mathbf{x} \in \mathscr{X} \subset \mathbb{R}^n$ we define $d_{\min}^l(\mathbf{x})$ as a distance to its closest neighbour from a different class. Then a set of all nearest neighbour distances is computed as

$$\mathscr{D}^l = \left\{ d_{\min}^l(\mathbf{x}) : \mathbf{x} \in \mathscr{X} \right\}, \tag{9}$$

and the value of $\sigma = \mathrm{median}(\mathscr{D}^l)$.

This approach, however, has been interpreted differently in [37], which resulted in an unsupervised heuristic based on what was proposed in [44]. The approach to estimate σ is similar, however, it is calculated without any knowledge about labels of examples, which means that not inter-class but inter-vector distances are used. Considering an unlabelled distance $d_{\min}(\mathbf{x})$ of an example $\mathbf{x}$ to its closest neighbour, the set of all neighbour distances is computed as

$$\mathscr{D} = \{d_{\min}(\mathbf{x}) : \mathbf{x} \in \mathscr{X}\}, \tag{10}$$

and the value of $\sigma = \text{mean}(\mathscr{D})$. This heuristic is denoted as *Soares*.

The use of mean instead of median in an approach proposed in [37] results in larger values of γ in the case of outliers in the data space. Therefore, following the reasoning in the original manuscript [44], we propose to compute $\sigma = \text{median}(\mathscr{D})$, which in case of the Gaussian RBF kernel results in

$$\gamma = \frac{1}{2\ \text{median}(\mathscr{D})}. \tag{11}$$

This heuristic is denoted as *Soares_med*.

2.3.5. Gelbart's heuristic

The heuristic used to estimate the initial value of γ in a well-known Python library scikit-learn, was proposed by Michael Gelbart in [36][4]. The scaling factor of Gaussian RBF kernel is computed as

$$\gamma = \frac{1}{n\text{Var}(\mathscr{X})},, \tag{12}$$

where $\mathscr{X} \subset \mathbb{R}^n$ and $\text{Var}(\mathscr{X})$ is a variance of all elements in the data set $\mathscr{X}$. It is easy to see that this heuristic is similar to the one discussed in Section 2.3.1, based on [35]: provided that every data feature has variance σ and mean 0 the value of Gelbart's heuristics is equal to the one described by equation (5). The advantage of this heuristics is its computational performance, and it has the potential to perform well when the variance of elements in the data array reflect the variance of the actual data vectors. This heuristic is denoted in our results as *Gelbart*.

2.4. Unsupervised heuristics for C

Unsupervised heuristics for the C parameter are much less common than for γ; in [14], there is a suggestion that parameter $C \propto 1/R^2$, where R is a measure for a range of the data in feature space and proposes examples of such R as the standard deviation of the distance between points and their mean or radius of the smallest sphere containing the data. However, to the best of our knowledge, the only actual derivation of this idea was presented in [34], which we discuss below.

[4] https://github.com/scikit-learn/scikit-learn/issues/12741

2.4.1. Chapelle & Zien C heuristic

Given a γ value (originally computed as described in Section 2.3.3), [34] calculate the empirical variance

$$s^2 = \frac{1}{m}\sum_{i=1}^{m} K(\mathbf{x}_i,\mathbf{x}_i) - \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} K(\mathbf{x}_i,\mathbf{x}_j), \tag{13}$$

which, with $K(\boldsymbol{x}_i,\boldsymbol{x}_i)$ being the value of RBF kernel (4), under the same $\rho \to 0$ assumption as Section 2.3.3, evaluates to

$$s^2 = 1 - a, \qquad a = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} K(\mathbf{x}_i,\mathbf{x}_j). \tag{14}$$

The C parameter value is then estimated as

$$C = \frac{1}{s^2}. \tag{15}$$

This heuristic is denoted in our experiments as: *Chapelle* when used in combination with authors' γ heuristic (see Section 2.3.3) and $+C$ when used with *covtrace* heuristic.

2.4.2. Mitigating C underestimation: an improvement to the Chapelle & Zien heuristic

Our observations suggest that values of parameter C, when dealing with high-dimensional data such as hyperspectral images, should be higher than estimated with the heuristic proposed in Section 2.4.1. To counter this we decided to additionally test a modified Chapelle & Zien heuristics with modified formula (14). Since in formula (14) the factor $a < 1$, higher values of C can be achieved by substituting $s^2 = 1 - a'$ with $a \le a' < 1$.

The value of a in equation (14) is an average of kernel values for all data points, which, for the RBF kernel, is a function of the average distances between the data points. By selecting a subset of the data points based on values of their distances, we can arbitrarily raise or lower the value of a. We start by considering a set of distances between the data points

$$\mathscr{A} = \left\{\|\mathbf{x}_i - \mathbf{x}_j\| : i,j \le m; \mathbf{x}_i,\mathbf{x}_j \in \mathscr{X}\right\}. \tag{16}$$

Then we define a subset of distances $\mathscr{A}'$ as $\frac{1}{n}$ quantile of $\mathscr{A}$ and we select a relevant set of data points pairs

$$\mathscr{B} = \left\{(i,j) : \|\mathbf{x}_i - \mathbf{x}_j\| \in \mathscr{A}'\right\}. \tag{17}$$

This leads to a modified version of the heuristic

$$s^2 = 1 - a', \qquad a' = \frac{1}{t}\sum_{(i,j)\in\mathscr{B}} K(\mathbf{x}_i,\mathbf{x}_j), \tag{18}$$

with $t = |\mathscr{B}|$. The rationale of using $\frac{1}{n}$ quantile is that with increased dimension n, the proposed condition will restrict the set of pairs $\mathscr{B}$ to the distances between close points. This modified Chapelle's heuristic is denoted as $+MC$, when used with *covtrace* heuristic for γ.

Compared with the original Chapelle & Zien heuristic, this adjustment consistently selected larger C values and, across our evaluated datasets, yielded accuracy that was practically equivalent (by definition of [48]) to GSCV (see Section 3.3).

Table 1
Datasets used in the experiment. Balance is the ratio between the size of the smallest and largest class. OA(0R) denotes the accuracy of a zero-rule, naive classifier that predicts the label of the most frequent class

Name[a]	Examples	Features	Classes	Balance	OA(0R)	Notes or full name
appendicitis	106	7	2	0.25	80.2	
balance	625	4	3	0.17	46.1	Balance Scale DS
banana	5300	2	2	0.81	55.2	Balance Shape DS
bands	365	19	2	0.59	63.0	Cylinder Bands
cleveland	297	13	5	0.08	53.9	Heart Disease (Cleveland), multi-class
glass	214	9	6	0.12	35.5	Glass Identification
haberman	306	3	2	0.36	73.5	Haberman's Survival
hayes-roth	160	4	3	0.48	40.6	Hayes-Roth
heart	270	13	2	0.80	55.6	Statlog (Heart)
hepatitis	80	19	2	0.19	83.8	
ionosphere	351	33	2	0.56	64.1	
iris	150	4	3	1.00	33.3	Iris plants
led7digit	500	7	10	0.65	11.4	LED Display Domain
mammographic	830	5	2	0.94	51.4	Mammographic Mass
marketing	6876	13	9	0.40	18.3	
monk-2	432	6	2	0.89	52.8	MONK's Problem 2
movement-libras	360	90	15	1.00	6.7	Libras Movement
newthyroid	215	5	3	0.20	69.8	Thyroid Disease (New Thyroid)
page-blocks	5472	10	5	0.01	89.8	Page Blocks Classification
phoneme	5404	5	2	0.42	70.7	
pima	768	8	2	0.54	65.1	Pima Indians Diabetes
segment	2310	19	7	1.00	14.3	
sonar	208	60	2	0.87	53.4	Sonar, Mines vs. Rocks
spectfheart	267	44	2	0.26	79.4	SPECTF Heart
tae	151	5	3	0.94	34.4	Teaching Assistant Evaluation
vehicle	846	18	4	0.91	25.8	Vehicle Silhouettes
vowel	990	13	11	1.00	9.1	Connectionist Bench
wdbc	569	30	2	0.59	62.7	Breast Cancer Wisconsin (Diagnostic)
wine	178	13	3	0.68	39.9	
wisconsin	683	9	2	0.54	65.0	Breast Cancer Wisconsin (Original)
yeast	1484	8	10	0.01	31.2	

[a] As the dataset is named in KEEL repository https://sci2s.ugr.es/keel/datasets.php

3. EXPERIMENTS

In this section we will present our method for experimental verification of unsupervised heuristics: the datasets that we use for tests, experimental procedure and finally our approach to statistical testing of obtained results.

3.1. Datasets

Experiments were performed using 31 standard classification datasets obtained from Keel-dataset repository[5], described in [49]. Instances with missing values and features with zero-variance were removed, therefore the number of examples/features can differ from their version in the UCI [50] repository. The datasets were chosen to be diverse in regards to the number of features and classes and to include imbalanced cases. In addition, following [51], the chosen set includes both complex cases where advanced ML models achieve an advantage over simple methods as well as datasets where most models perform similarly. Reference classification results can be found in [52] or through OpenML project [53]. The summary of the datasets used in experiments can be found in Table 1, along with the overall accuracy (OA) results of naive classifier (or zero-rule classifier, 0R) that classifies every point as the member of most frequent class.

Before the experiment, every dataset was preprocessed by centering the data and scaling it to the unit variance. This operation was performed using mean and variance values estimated from the training part of the dataset.

[5] https://sci2s.ugr.es/keel/category.php?cat=clas

3.2. Choosing SVM parameters for a given dataset

The experiments used either one or two stages of cross-validation – 'external' and 'internal' or 'external' only – depending on whether the grid search or heuristics were used. Let the heuristics $h \in \mathscr{H}$ from the set of tested heuristics $\mathscr{H}$ be a function that generates SVM parameters $\{C, \gamma\}$ based on a supplied training set $\mathscr{T}$ i.e. $h : \mathscr{T} \rightarrow \mathbb{R}^2$. We denote by h_0 a heuristic which always returns a pair $\{C, \gamma\} = \{1, 1\}$, which are commonly assumed defaults, and thus a reference values which are not data-dependent. The h_0 heuristic is denoted in our experiments as *default*.

For every training set $\mathscr{T}_i$ corresponding with a given $i-$fold of the external CV, and for every heuristics $h \in \mathscr{H}$ parameters of the SVM were selected in three ways:

1. By performing a grid-search around the initial parameters h_0 and selecting the best model in the internal CV on $\mathscr{T}_i$.
2. By applying the heuristics $h(\mathscr{T}_i)$.
3. By performing a grid-search around the initial parameters $h(\mathscr{T}_i)$ and selecting the best model in the internal CV on $\mathscr{T}_i$.

The range of parameters for GSCV to test is not always easy to determine as different studies propose different ranges – in [54] the range $\{0, 0.1, 0.3, 0.5, 0.7\}$ is taken into consideration for C, while for γ its $\{2^{-4}, 2^{-3}, \ldots 2^4\}$. Authors of [55] propose $C \in \{x10^y : x \in \{1, 2, \ldots, 10\}, y \in \{-2, -1, \ldots, 2\}\}$, $\gamma \in \{x10^y : x \in 1, 2, \ldots, 10, y \in \{-4, -3, \ldots, 1\}\}$ while in research conducted in [56] the selected range was $\{2^{-17}, 2^{-16}, \ldots 2^3\}$ for γ and $\{2^{-3}, 2^{-2}, \ldots 2^{17}\}$ for C. In [57], the authors decided to use the grid of $10^{-6}, \ldots, 10^6$ for both C and γ.

In this research, similar approach was selected, with range of parameters set as $\mathscr{R} = \langle 10^{-5}, 10^{-4}, .., 10^0, .., 10^5 \rangle$, and the parameter grid $\mathscr{G}_h$ for the heuristics h generated as

$$\mathscr{G}_h = \left\{ ri_\gamma : r \in \mathscr{R} \right\} \times \left\{ ri_C : r \in \mathscr{R} \right\}, \tag{19}$$

where $h(\mathscr{T}) = (i_\gamma, i_C)$. For the external CV, the number of folds $k_{\text{external}} = 5$, for the internal CV the number of folds $k_{\text{internal}} = 3$; both were stratified CVs, by which we mean the approach often used towards unbalanced sets which selects training and test sets maintaining similar percentage of datapoints from each class[6].

For assessing classification performance, the balanced accuracy measure [58] (BA) was employed. BA can be expressed as the mean of classification accuracies in classes i.e. the mean between a ratio of correctly classified examples to the total number of examples in every class. Compared to the overall accuracy (OA), which is the ratio between a number of correctly classified examples to the total number of examples in dataset, it less sensitive to unbalance in class size.

The final performance of the classifier in an experiment is the mean BA between external folds. Every experiment was repeated 10 times and the final values of BA were obtained by averaging the performance values of individual runs.

[6] we used implementation provided by https://scikit-learn.org/

3.3. Statistical verification of results

A typical approach to verify statistical significance of results is to use null hypothesis significance testing (NHST). While common, the NHST has several disadvantages explained in detail in [48]. Two particular ones are: the fact that point-wise null hypotheses are usually false, provided that sufficiently large number of data points is available, as in practice no two classifiers have perfectly similar accuracy; NHST does not allow to reach conclusion when the null hypothesis is rejected, which limits its usefulness. As an alternative, authors of [48] propose a new methodology based on Bayesian analysis that was adapted for analysing our results. This methodology compares classifiers by estimating and querying the posterior distribution of their mean difference. The methodology introduces the *region of practical equivalence* (rope) which refers to the value of mean difference that implies that classifiers are practically equivalent, e.g., when their accuracies differ by less than 1%. This allows us to infer the probability $P(\text{classifier}_A < \text{classifier}_B)$ of the mean difference between classifiers being practically negative which implies that classifier_B is more accurate, as well as the probability of the opposite inequality and the probability $P(\text{classifier}_A = \text{classifier}_B)$ that both classifiers are practically equivalent with regard to the rope value. In addition the methodology allows for drawing conclusions through the simultaneous analysis of multiple data sets and it has a dedicated, clear visualisation of test results.

Since we perform experiments using multiple datasets, the approach employing hierarchical models, described in Section 4.3.1 of [48] was employed. Following the suggestion in [48], the value of rope was set to 1%.

3.4. Implementation

SVM implementation was from the scikit-learn library v1.0.2. Bayesian comparison of classifiers [48] and its visualisation was performed using baycomp library v. 1.0.2[7]. Matplotlib and seaborn libraries were used for data visualisation.

4. RESULTS AND DISCUSSION

Our experiments compared the accuracy of the previously discussed UH-SVM approaches, to the GSCV-SVM, on the 31 Keel datasets. For each approach, the individual scores were aggregated into an estimated probability of practical advantage/disadvantage/equivalence of the heuristics and GSCV with regard to classifier accuracy. The summary of results for the balanced accuracy (BA) measure[8] is presented in Table 2. Since most of the heuristics only estimate the γ parameter, and only two of them estimate the C (*Chapelle*, *MC*), we present results as a combination of every γ and C heuristic including the 'default' value of $\gamma = 1$, $C = 1$. The advantage or any disadvantage of any one method corresponds to a sufficiently large difference between means of accuracies over all datasets, as described in

[7] https://github.com/janezd/baycomp

[8] For the reference, results of experiments for OA measure are presented in Appendix 4.

Table 2
Results of experiments – performance of different UH-SVM approaches with respect to GSCV-SVM. The numbers correspond to probabilities computed with the Bayesian analysis with methodology from [48]. Three right columns present probabilities of cross-validation being on average more/ equivalent to / less accurate than heuristics. Results were obtained for the balanced accuracy measure and rope value of 1%. Note that, with *MC* heuristic, several of γ heuristics achieve results close to GSCV

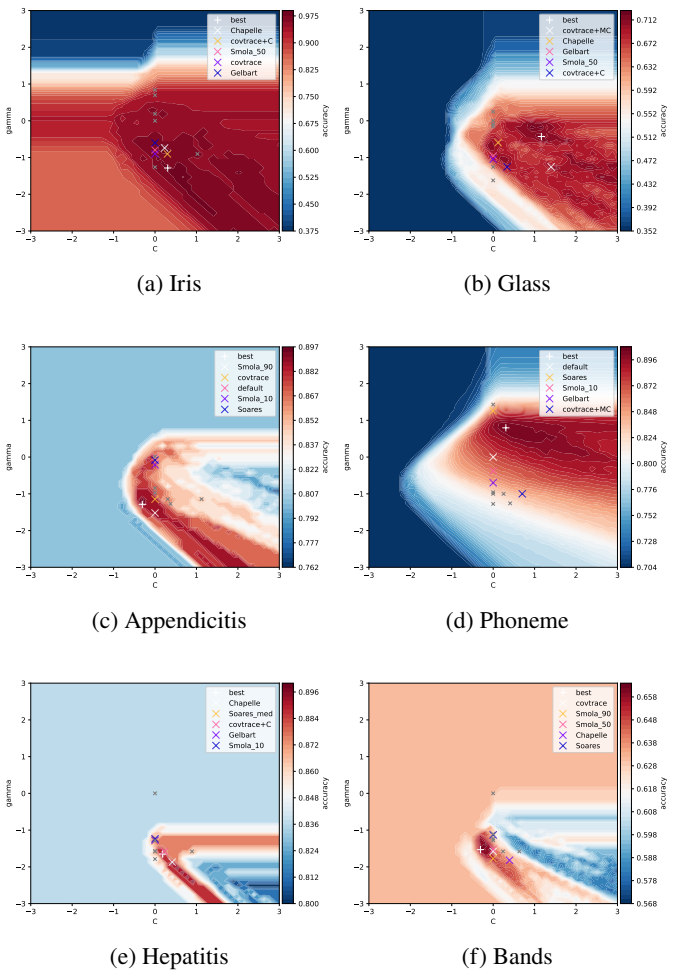
C heuristics	γ heuristics	P(CV > H)	P(CV = H)	P(CV < H)
default	default	1.00	0.00	0.00
	Gelbart	0.85	0.13	0.02
	Smola_10	0.97	0.02	0.01
	Smola_50	0.86	0.11	0.03
	Smola_90	0.99	0.00	0.01
	Soares	1.00	0.00	0.00
	Soares_med	1.00	0.00	0.00
	Chapelle	0.98	0.01	0.01
	covtrace	0.93	0.06	0.01
Chapelle	default	1.00	0.00	0.00
	Gelbart	0.60	0.36	0.04
	Smola_10	0.96	0.03	0.01
	Smola_50	0.68	0.24	0.08
	Smola_90	0.84	0.12	0.04
	Soares	0.99	0.00	0.01
	Soares_med	0.99	0.00	0.01
	Chapelle	0.83	0.14	0.03
	covtrace	0.55	0.41	0.05
MC	default	1.00	0.00	0.00
	Gelbart	0.19	0.76	0.05
	Smola_10	0.68	0.28	0.04
	Smola_50	0.17	0.81	0.02
	Smola_90	0.34	0.60	0.07
	Soares	0.98	0.00	0.02
	Soares_med	0.99	0.00	0.01
	Chapelle	0.21	0.76	0.02
	covtrace	0.12	0.84	0.03

Section 3.3; their practical equivalence corresponds to sufficiently small difference, with regards to rope value of 1%.

When no heuristics or only γ heuristics are used, parameters obtained by GSCV result in significantly higher accuracy. There is only a marginal improvement when *Chapelle* heusitics is used for selecting a *C* parameter value. However, when using the extension that improves *C* estimation, the *MC* heuristics, five of the γ heuristics tested obtained the accuracy very close, or practically equivalent to CV. The combination of *Covtrace+MC* resulted in the highest estimated value of this probability which indicates that on average this heuristics results in classification accuracy no worse than GSCV. Visualisation of results for example heuristics is presented in Fig. 3. The improvement in accuracy arising from the use of the two heuristics (three, if including the default) for the *C* parameter is clearly evident in plots (a–c). Notably, the more effective the heuristic, the more equivalent are the scores of UH-SVM and GSCV-SVM. Plot (d) presents similar results for an overall accuracy (OA) measure compared to the BA in plot (c). The use of OA measure usually results in slightly higher probabilities of practical equivalence between heuristics and GSCV. This suggests the class imbalance negatively affects GSCV performance.

The practical equivalence in the accuracy of classifiers whose parameters were chosen by GSCV and heuristic, is also visible during the inspection of the parameter values obtained from heuristics plotted on the graph showing the relationship between the classifier effectiveness and its parameters (estimated through a dense grid of parameters). In the selected representative examples in Fig. 2, it can be seen that most of these points, especially for the best heuristics, are usually located in areas of high accuracy.

Interestingly, out of *Smola* heuristics, the result of $Smola_{50}$ + *MC* resulted in the BA value most equivalent to GSCV. This

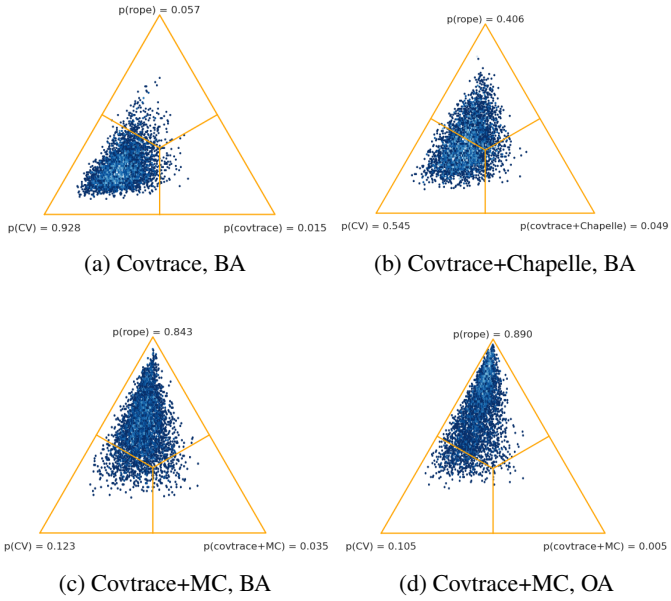


(a) Iris (b) Glass

(c) Appendicitis (d) Phoneme

(e) Hepatitis (f) Bands

Fig. 2. The impact of SVM parameters on its accuracy. Parameter values are presented in logspace. Accuracy values were obtained from experiments with 5-fold CV by sampling each pair of parameters from the 50×50 parameter grid. The highest value of accuracy is denoted as 'best'. Marked points denote results of unsupervised heuristics from this paper, with the five heuristics scoring highest marked with colour



(a) Covtrace, BA (b) Covtrace+Chapelle, BA

(c) Covtrace+MC, BA (d) Covtrace+MC, OA

Fig. 3. Visualisation of Bayesian analysis of results with methodology from [48] for selected cases from Table 2: *covtrace+default*, *covtrace+Chapelle*, *covtrace+MC*. Vertices of the simplex represent decisions with certainty in favour of: CV (lower left), example heuristics (lower right) and rope (top); the latter corresponds to practical equivalence of CV and heuristics accuracy. Points represent Monte Carlo sampling of posterior probabilities in barycentric coordinates. BA denotes balanced accuracy, OA denotes overall accuracy. Note the impact of the C heuristics on the equivalence of UH-SVM and GSCV-SVM

Table 3

Performance of heuristics as ratio of CV/heuristics execution time, i.e., how many times heuristics is faster than CV. Times were estimated from 10 experiments and averaged over all datasets. Note that in almost all cases the speedup is 100–200 times

γ heuristics	C heuristics		
	Default	Chapelle	MC
default	136.88	106.52	97.91
Gelbart	248.72	182.27	149.33
Smola_10	153.92	136.47	121.71
Smola_50	169.13	149.29	131.29
Smola_90	158.75	149.64	131.19
Soares	132.01	105.50	94.59
Soares_med	125.46	101.45	92.51
Chapelle	163.59	148.40	132.23
covtrace	237.38	188.61	154.86

indicates that the median distance between examples in the data space is of particular importance when choosing the γ parameter.

Comparison of execution time for heuristics and GSCV is presented in Table 3. The values express a ratio of mean computation time of an experiment with GSCV parameter selection to experiment with parameters selected with heuristics. The average time was calculated over ten iterations of the experiment across all datasets. The use of heuristics allows, on average, to speed up calculations 100–200 times. Differences in times result not only from calculating the parameter values, but also from the impact of these values on the classifier – increasing the value of the γ and C parameter extends the calculation time.

To summarise, estimation of both parameters, in particular with *Covtrace+MC* heuristics, leads to accuracy practically equivalent (by definition of [48]) to GSCV (see Fig. 3c) with parameters obtained in only ~0.006 of its working time (see Table 3).

The higher results of *Covtrace+MC* approach mean that there is a potential space for improvement in C value estimation, as the conventional and widely used Chapelle& Zien, modified to estimate higher values of C, not only consistently improves the results on tested datasets, but also makes them practically equivalent (as defined in [48]) to GSCV with respect to accuracy. Unsupervised heuristics for SVM parameters are likely effective because the test datasets conform to the clustering assumption, where data space forms structures/clusters useful to the classification problem, and data point distributions reflect class divisions. However, the same assumption is the basis of training set selection with GSCV. As datasets deviate from the clustering assumption, the effectiveness of both approaches decreases, especially when the training data is limited. GSCV is by no means inferior to the heuristics, especially if supplied with a proper number of labelled datapoints. In practice, however, the differences are often very small. Moreover, while it is natural to use GSCV when the standard approach is preferable (i.e., a small number of examples, training time is not an issue), in many scenarios (e.g., processing on edge IoT devices), the proposed heuristics offer practically equivalent accuracy in a fraction of the time.

5. CONCLUSIONS

In this study, we evaluated unsupervised heuristics for SVM parameter selection on over thirty benchmark datasets, comparing their performance with GSCV. We have also proposed a modification to Chapelle & Zien's heuristic for the C parameter, as optimisation of both parameters is vital for accurate classifiers. We compared results using methodology based on Bayesian analysis, described in [48]. Our results indicate that heuristics are usually practically equivalent to GSCV in terms of achieved accuracy of the classifier, i.e., obtained accuracies differ by less than 1% (see Fig. 3c and probabilities of equivalence in Table 2). Moreover, these heuristics offer a reduction in computation time, achieving a 100–200 times speedup (see Table 3). This makes an unsupervised, heuristic approach to parameter selection a compelling alternative for GSCV for rapid SVM calibration.

Additionally, our results presented in Table 4 (*Chapelle* and *covtrace* heuristics) show that estimating C sharply increases the accuracy of the produced classifier. Choosing larger values of C shifts the probabilities of the UH approach to practical equivalence (as defined by [48]) of GSCV.

Table 4

Results of experiments – performance of different UH-SVM approaches with respect to GSCV-SVM, for overall accuracy (a supplement to Table 2). The numbers correspond to probabilities computed with the Bayesian analysis with methodology from [48]. Three right columns present probabilities of cross-validation being on average more / equivalently / less accurate than heuristics. Results were obtained for the rope value of 1%. Note that, with *MC* heuristic, several of γ heuristics achieve results close to GSCV

C heuristics	γ heuristics	P(CV > H)	P(CV = H)	P(CV < H)
Default	default	1.00	0.00	0.00
	Gelbart	0.70	0.25	0.05
	Smola_10	0.94	0.05	0.01
	Smola_50	0.61	0.35	0.04
	Smola_90	0.95	0.03	0.02
	Soares	0.99	0.00	0.00
	Soares_med	1.00	0.00	0.00
	Chapelle	0.94	0.04	0.02
	covtrace	0.74	0.23	0.04
Chapelle	default	1.00	0.00	0.00
	Gelbart	0.56	0.40	0.04
	Smola_10	0.92	0.07	0.01
	Smola_50	0.66	0.27	0.07
	Smola_90	0.85	0.10	0.05
	Soares	1.00	0.00	0.00
	Soares_med	1.00	0.00	0.00
	Chapelle	0.74	0.21	0.04
	covtrace	0.68	0.24	0.08
MC	default	1.00	0.00	0.00
	Gelbart	0.10	0.90	0.01
	Smola_10	0.54	0.44	0.01
	Smola_50	0.13	0.86	0.00
	Smola_90	0.41	0.58	0.02
	Soares	0.99	0.00	0.00
	Soares_med	1.00	0.00	0.00
	Chapelle	0.19	0.80	0.01
	covtrace	0.10	0.89	0.01

REFERENCES

[1] M. Koklu and I. A. Ozkan, "Multiclass classification of dry beans using computer vision and machine learning techniques," *Comput. Electron. Agric.*, vol. 174, p. 105507, 2020.

[2] K. Shah, H. Patel, D. Sanghvi, and M. Shah, "A comparative analysis of logistic regression, random forest and knn models for the text classification," *Augment. Hum. Res.*, vol. 5, pp. 1–16, 2020.

[3] M. Alloghani, D. Al-Jumeily, J. Mustafina, A. Hussain, and A.J. Aljaaf, *A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science*, Cham: Springer International Publishing, 2020, pp. 3–21, doi: 10.1007/978-3-030-22475-2_1.

[4] W. Yang, S. Wang, N.M. Sahri, N.M. Karie, M. Ahmed, and C. Valli, "Biometrics for internet-of-things security: A review," *Sensors*, vol. 21, no. 18, p. 6163, 2021.

[5] L. Li, S. Zhang, and B. Wang, "Plant disease detection and classification by deep learning—a review," *IEEE Access*, vol. 9, pp. 56 683–56 698, 2021.

[6] G. Cheng, X. Xie, J. Han, L. Guo, and G.-S. Xia, "Remote sensing image scene classification meets deep learning: Challenges, methods, benchmarks, and opportunities," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 13, pp. 3735–3756, 2020.

[7] Z. Menter, W.Z. Tee, and R. Dave, "Application of machine learning-based pattern recognition in iot devices: Review," in *Proc. International Conference on Communication and Computational Technologies*, S. Kumar, S.D. Purohit, S. Hiranwal, and M. Prasad, Eds., Singapore: Springer Singapore, 2021, pp. 669–689.

[8] B.B. Gupta, A. Tewari, I. Cvitić, D. Peraković, and X. Chang, "Artificial intelligence empowered emails classifier for internet of things based systems in industry 4.0," *Wireless Netw.*, vol. 28, no. 1, pp. 493–503, Jan 2022, doi: 10.1007/s11276-021-02619-w. [Online]. Available: https://doi.org/10.1007/s11276-021-02619-w

[9] I. M. Pires, N. M. Garcia, N. Pombo, and F. Flórez-Revuelta, "Limitations of the use of mobile devices and smart environments for the monitoring of ageing people," in *Proc. 4th International Conference on Information and Communication Technologies for Ageing Well and e-Health – HSP*, INSTICC, SciTePress, 2018, pp. 269–275, doi: 10.5220/0006817802690275.

[10] M. Romaszewski, P. Głomb, and M. Cholewa, "Semi-supervised hyperspectral classification from a small number of training samples using a co-training approach," *ISPRS – J. Photogramm. Remote Sens.*, vol. 121, pp. 60–76, 2016, doi: 10.1016/j.isprsjprs.2016.08.011.

[11] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, "A comprehensive survey on support vector machine classification: Applications, challenges and trends," *Neurocomputing*, vol. 408, pp. 189–215, 2020.

[12] H.-Y. Huang, R. Kueng, G. Torlai, V.V. Albert, and J. Preskill, "Provably efficient machine learning for quantum many-body problems," *Science*, vol. 377, no. 6613, p. eabk3333, 2022.

[13] V. Vapnik and A.Y. Lerner, "Recognition of patterns with help of generalized portraits," *Avtomat. Telemekh.*, vol. 24, no. 6, pp. 774–780, 1963.

[14] B. Schölkopf and A.J. Smola, *Learning with kernels*, Citeseer, 1998, vol. 4.

[15] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995, doi: 10.1007/BF00994018.

[16] H. Drucker, C.J. Burges, L. Kaufman, A.J. Smola, V. Vapnik *et al.*, "Support vector regression machines," *Adv. Neural Inf. Process. Syst.*, vol. 9, pp. 155–161, 1997.

[17] M.N.A.H. Sha'abani, N. Fuad, N. Jamal, and M.F. Ismail, "kNN and SVM classification for eeg: A review," in *InECCE2019*, A.N. Kasruddin Nasir *et al.*, Eds., Singapore: Springer Singapore, 2020, pp. 555–565, doi: 10.1007/978-981-15-2317-5_47.

[18] P. Głomb, M. Romaszewski, M. Cholewa, and K. Domino, "Application of hyperspectral imaging and machine learning methods for the detection of gunshot residue patterns," *Forensic Sci. Int.*, vol. 290, pp. 227–237, 2018, doi: 10.1016/j.forsciint.2018.06.040.

[19] B. Direito, C. A. Teixeira, F. Sales, M. Castelo-Branco, and A. Dourado, "A realistic seizure prediction study based on mul-

ticlass SVM," *Int. J. Neural Syst.*, vol. 27, no. 03, p. 1750006, 2017, doi: 10.1142/S012906571750006X.

[20] H.-C. Kim, S. Pang, H.-M. Je, D. Kim, and S.Y. Bang, "Constructing support vector machine ensemble," *Pattern Recognit.*, vol. 36, no. 12, pp. 2757–2767, 2003, doi: 10.1016/S0031-3203(03)00175-4.

[21] M. Cholewa, P. Głomb, and M. Romaszewski, "A spatial-spectral disagreement-based sample selection with an application to hyperspectral data classification," *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 3, pp. 467–471, 2019, doi: 10.1109/LGRS.2018.2868862.

[22] M. Ebrahimi, M. Khoshtaghaza, S. Minaei, and B. Jamshidi, "Vision-based pest detection based on SVM classification method," *Comput. Electron. Agric.*, vol. 137, pp. 52–58, 2017, doi: 10.1016/j.compag.2017.03.016.

[23] W.-H. Chen, S.-H. Hsu, and H.-P. Shen, "Application of SVM and ANN for intrusion detection," *Comput. Oper. Res.*, vol. 32, no. 10, pp. 2617–2634, 2005, doi: 10.1016/j.cor.2004.03.019.

[24] S. Lecomte, R. Lengellé, C. Richard, F. Capman, and B. Ravera, "Abnormal events detection using unsupervised one-class svm-application to audio surveillance and evaluation," in *2011 8th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, IEEE, 2011, pp. 124–129, doi: 10.1109/AVSS.2011.6027306.

[25] J. Song, H. Takakura, Y. Okabe, and Y. Kwon, "Unsupervised anomaly detection based on clustering and multiple one-class SVM," *IEICE Trans. Commun.*, vol. 92, no. 6, pp. 1981–1990, 2009, doi: 10.1587/transcom.E92.B.1981.

[26] T. Subashini, V. Ramalingam, and S. Palanivel, "Breast mass classification based on cytological patterns using RBFNN and SVM," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 5284–5290, 2009, doi: 10.1016/j.eswa.2008.06.127.

[27] P. Parveen, Z.R. Weger, B. Thuraisingham, K. Hamlen, and L. Khan, "Supervised learning for insider threat detection using stream mining," in *2011 IEEE 23rd international conference on tools with artificial intelligence*, IEEE, 2011, pp. 1032–1039, doi: 10.1109/ICTAI.2011.176.

[28] P. Wang, R. Mathieu, J. Ke, and H. Cai, "Predicting criminal recidivism with support vector machine," in *2010 International Conference on Management and Service Science*, IEEE, 2010, pp. 1–9, doi: 10.1109/ICMSS.2010.5575352.

[29] X. Li and X. Guo, "A HOG feature and SVM based method for forward vehicle detection with single camera," in *2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 1, IEEE, 2013, pp. 263–266, doi: 10.1109/IHMSC.2013.69.

[30] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3133–3181, 2014.

[31] B. Schölkopf and A.J. Smola, "From regularization operators to support vector kernels," *Adv. Neural Inf. Process. Syst.*, vol. 10, pp. 343–349, 1998.

[32] J. Zhang and S. Wang, "A fast leave-one-out cross-validation for SVM-like family," *Neural Comput. Appl.*, vol. 27, no. 6, pp. 1717–1730, 2016, doi: 10.1007/s00521-015-1970-4.

[33] A.J. Smola, "Easy kernel width choice," 2011, accessed: 10-06-2021. [Online]. Available: https://blog.smola.org/post/940859888/easy-kernel-width-choice

[34] O. Chapelle and A. Zien, "Semi-supervised classification by low density separation," in *Proc. Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS)*, vol. 2005, Citeseer, 2005, pp. 57–64.

[35] M. Varewyck and J.-P. Martens, "A practical approach to model selection for support vector machines with a gaussian kernel," *IEEE Trans. Syst. Man. Cyber. Part B -Cybern.*, vol. 41, no. 2, pp. 330–340, 2010, doi: 10.1109/TSMCB.2010.2053026.

[36] M. Gelbart, "Gamma='scale' in SVC," 2018, accessed: 10-06-2021. [Online]. Available: https://github.com/scikit-learn/scikit-learn/issues/12741

[37] C. Soares, P.B. Brazdil, and P. Kuba, "A meta-learning method to select the kernel width in support vector regression," *Mach. Learn.*, vol. 54, no. 3, pp. 195–209, 2004, doi: 10.1023/B:MACH.0000015879.28004.9b.

[38] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, 2002, doi: 10.1109/72.991427.

[39] G.L. Prajapati and A. Patle, "On performing classification using SVM with radial basis and polynomial kernel functions," in *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, IEEE, 2010, pp. 512–515.

[40] O. Chapelle and V. Vapnik, "Model selection for support vector machines," in *Advances in neural information processing systems*, 2000, pp. 230–236.

[41] N. Carchedi, D. De Mesmaeker, and L. Vannoorenberghe, "Sigest: Hyperparameter estimation for the gaussian radial basis kernel," 2021, rDocumentation, Accessed: 10-06-2021. [Online]. Available: https://www.rdocumentation.org/packages/kernlab/versions/0.9-29/topics/sigest

[42] B. Caputo, K. Sim, F. Furesjo, and A.J. Smola, "Appearance-based object recognition using SVMs: which kernel should i use?" in *Proc. NIPS workshop on Statistical methods for computational experiments in visual processing and computer vision, Whistler*, vol. 2002, 2002.

[43] J. Wainer and P. Fonseca, "How to tune the RBF SVM hyperparameters? an empirical evaluation of 18 search algorithms," *Artif. Intell. Rev.*, pp. 1–27, 2021, doi: 10.1007/s10462-021-10011-5.

[44] T.S. Jaakkola, M. Diekhans, and D. Haussler, "Using the fisher kernel method to detect remote protein homologies," in *Proc. ISMB-99*, vol. 99, 1999, pp. 149–158.

[45] D. Berrar, "Cross-validation," in *Encyclopedia of Bioinformatics and Computational Biology*, S. Ranganathan, M. Gribskov, K. Nakai, and C. Schönbach, Eds., Oxford: Academic Press, 2019, pp. 542–545, doi: 10.1016/B978-0-12-809633-8.20349-X.

[46] X. Wang, F. Huang, and Y. Cheng, "Super-parameter selection for gaussian-kernel SVM based on outlier-resisting," *Measurement*, vol. 58, pp. 147–153, 2014, doi: 10.1016/j.measurement.2014.08.019.

[47] T. Jaakkola, M. Diekhans, and D. Haussler, "A discriminative framework for detecting remote protein homologies," *J. Comput. Biol.*, vol. 7, no. 1-2, pp. 95–114, 2000, doi: 10.1089/10665270050081405.

[48] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, "Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 2653–2688, 2017.

[49] J. Alcalá-Fdez *et al.*, "Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis

framework," *J. Multiple-Valued Logic & Soft Comput.*, vol. 17, p. 255–287, 2011.

[50] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[51] W. Duch, N. Jankowski, and T. Maszczyk, "Make it cheap: learning with o(nd) complexity," in *2012 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2012, pp. 1–4, doi: 10.1109/IJCNN.2012.6252380.

[52] J.G. Moreno-Torres, J.A. Sáez, and F. Herrera, "Study on the impact of partition-induced dataset shift on k-fold cross-validation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1304–1312, 2012, doi: 10.1109/TNNLS.2012.2199516.

[53] M. Feurer *et al.*, "OpenML-Python: an extensible Python API for openML," *J. Mach. Learn. Res.*, vol. 22, no. 100, pp. 1–5, 2021.

[54] M.E. Matheny, F.S. Resnic, N. Arora, and L. Ohno-Machado, "Effects of SVM parameter optimization on discrimination and calibration for post-procedural pci mortality," *J. Biomed. Inf.*, vol. 40, no. 6, pp. 688–697, 2007.

[55] R.M. Schuhmann, A. Rausch, and T. Schanze, "Parameter estimation of support vector machine with radial basis function kernel using grid search with leave-p-out cross validation for classification of motion patterns of subviral particles," *Curr. Dir. Biomed. Eng.*, vol. 7, no. 2, pp. 121–124, 2021.

[56] F. Budiman, "Svm-rbf parameters testing optimization using cross validation and grid search to improve multiclass classification," *Sci. Visualization*, vol. 11, no. 1, pp. 80–90, 2019.

[57] P. Lameski, E. Zdravevski, R. Mingov, and A. Kulakov, "SVM parameter tuning with grid search and its impact on reduction of model over-fitting," in *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing: 15th International Conference, RSFDGrC 2015, Tianjin, China, November 20-23, 2015, Proc.*, Springer, 2015, pp. 464–474.

[58] K.H. Brodersen, C.S. Ong, K.E. Stephan, and J.M. Buhmann, "The balanced accuracy and its posterior distribution," in *2010 20th international conference on pattern recognition*, IEEE, 2010, pp. 3121–3124.