Data structure ≥ labels? Unsupervised heuristics for SVM hyperparameter estimation

Michał Cholewa\textsuperscript{a}, Michał Romaszewski\textsuperscript{a}, Przemysław Głomb\textsuperscript{a}

\textsuperscript{a}Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Bałtycka 5, Gliwice, 44100, 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 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 that problem, there exist several unsupervised heuristics that take advantage of the characteristics of the dataset for selecting parameters instead of using class label information. While 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 proposed an improved heuristics for SVM parameter selection and tested it against GSCV and state of art heuristic on over 30 standard classification datasets. The results show not only its advantage over state-of-art heuristics but also that it is statistically no worse than GSCV.

Keywords: SVM, cross validation, unsupervised heuristics

1. Introduction

Classification is among the most frequently encountered problems within the field of pattern recognition. It is utilized, among many other fields, in computer vision\textsuperscript{Koklu and Ozkan (2020)}, document analysis\textsuperscript{Shah et al.}
The classification itself is a wide area that contains both traditional machine learning methods and, recently increasingly popular, deep learning models. However, even with formidable results achieved by the deep learning approaches, e.g. [Li et al. (2021), Cheng et al. (2020)], 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 a learning algorithm for the deep learning model is prohibitive for many current applications of pattern recognition. This situation occurs e.g. for Internet of Things devices [Menter et al. (2021)], edge computing [Gupta et al. (2022)], medical devices [Pires et al. (2018)] or with limited training labels [Romaszewski et al. (2016)]. Additionally, classical methods – notably Support Vector Machines – are selected for their robustness [Cervantes et al. (2020)] or theoretical consideration [Huang et al. (2022)].

Support Vector Machine (SVM) is a supervised classification scheme based on ideas developed by V. N. Vapnik and A. Ya. Chervonenkis in 1960s [Vapnik and Lerner (1963)] and later expanded on in works such as Schölkopf and Smola (1998b), Cortes and Vapnik (1995) or Drucker et al. (1997). 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 SVM have been consistently used in various roles – as an independent classification scheme e.g. Sha’abani et al. (2020), Glomb et al. (2018), Direito et al. (2017), part of more complex engines e.g. Kim et al. (2003), Cholewa et al. (2019) or a detection engine e.g. Ebrahimi et al. (2017), Chen et al. (2005). It has been also employed in unsupervised setting in works such as Lecomte et al. (2011), Song et al. (2009). This flexibility allows SVM to be one of the most frequently used machine learning approaches in medicine Subashini et al. (2009), remote sensing Romaszewski et al. (2016), threat detection Parveen et al. (2011), criminology Wang et al. (2010), and is often utilized in photo, text, and time sequence analysis Li and Guo (2013). In numerous studies, SVM is consistently marked as one of the top performing method Cervantes et al. (2020).

The popularity and versatility of SVM is 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 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 [Fernández-Delgado et al. (2014)] and assumes only smoothness of the data, which makes it a natural choice when knowledge about data is limited [Schölkopf and Smola (1998a)]. 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 [An et al. (2007) Zhang and Wang (2016)]. 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 a separate model. This is a burden for performing pattern recognition in distributed edge computing devices in Industry 4.0 [Gupta et al. (2022)] or optimization of battery usage for mobile devices with limited connectivity, e.g. in monitoring of ageing people [Pires et al. (2018)].

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. Smola (2011), to more complex approaches involving e.g. cluster assumptions and graph distances between datapoints [Chapelle and Zien (2005)]. Through those approaches, values of $C$ and $\gamma$ can be estimated based on structure of entire available dataset, in an unsupervised way – without the requirement of labels. This is especially useful for applications that acquire large amount of data with limited supervision, e.g. IoT devices [Menter et al. (2021)]. Additionally, this estimation is one-pass computation much less intensive than cross-validation, allowing for much greater applicability, e.g. in IoT/edge/medical supervision devices. Unsupervised estimation avoids the issues of optimizing parameters on the same set as the one used for training, which can lead to overfitting [Schölkopf and Smola (1998b)]. It is known that in some cases, e.g. where classes indeed do conform to the cluster assumption and Gaussian distribution [Varewyck and Martens (2010)], optimal or close to optimal parameter values can be analytically derived from data without knowledge about class labels. This approach is also very helpful when training data is very limited and may poorly reflect true class distributions – a situation typically encountered in semi-supervised hyperspectral classification, e.g. [Romaszewski et al. (2016)]. The robustness of this approach has lead unsupervised heuristics to be a default parameter.
setting in SVM programming libraries, e.g. scikit-learn [Gelbart (2018)].

In this work, we experimentally verify the performance of unsupervised heuristical hyperparameter estimation for an SVM classifier (UH-SVM) against a grid search CV trained SVM (GSCV-SVM). We evaluate a large set of unsupervised heuristics, and propose an extension aimed at improving performance of one of the most general approach – Chapelle’s heuristics [Chapelle and Zien (2005)]. Our experiments show, that without specific prior knowledge of a dataset, there’s 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 to have a worse accuracy. Considering the significantly lesser requirement of computation power 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. As part of results we show that our proposed extension of Chapelle’s heuristics obtains results practically equivalent to GSCV. Additionally, while there are numerous works investigating individual heuristics, to the best of the authors’ knowledge, there is no work that collects them together and compares them with each other.

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 [Chapelle and Zien (2005)] in high dimensional datasets or correction for [Soares et al. (2004)].

2.1. Kernel SVM

A kernel SVM [Schölkopf and Smola (1998b)] 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 \( \mathcal{X} \subset \mathbb{R}^n \) be a set of data and let \( x_i \in \mathcal{X}, i = 1, \ldots, m \) be the set of labelled examples. Let also \( \mathcal{Y} = \{-1, 1\} \) be a set of labels. We define a training set as a set of examples with labels assigned to them,

\[
\mathcal{L} = \{(x_i, y_i), i = 1, \ldots, m\} \quad x_i \in \mathcal{X} \quad y_i \in \mathcal{Y}.
\]
The SVM assigns an example \( x \in \mathcal{X} \subset \mathbb{R}^n \) into one of two classes using a decision function

\[
f(x) = \text{sgn} \left( \sum_{i=1}^{m} y_i \alpha_i K(x, x_i) + b \right).
\]

(2)

Here, \( \alpha_i \geq 0 \) and \( b \) are coefficients computed through Lagrangian optimization – maximization of margin, or distance from hyperplane to classes’ datapoints on the training set. Training examples \( x_i \) 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 Hsu and Lin (2002).

2.1.1. Kernel function

The function \( K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \) is called the kernel function and it is used to compute the similarity between the classified example \( x \) and each training instance \( x_i \). It is a generalization of a dot product operation used in the original linear SVM derivation, i.e. \( K(x, x_i) = \langle x, x_i \rangle \), taking advantage of the ‘kernel trick’ Schölkopf and Smola (1998b) – a non-linear mapping \( \phi : \mathcal{X} \rightarrow \mathcal{H} \) to a feature space \( \mathcal{H} \) where the dot product is computed by evaluating the value \( K(x, x_i) = \langle \phi(x), \phi(x_i) \rangle \). 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

\[
K(x, x_i) = (\langle x, x_i \rangle + c)^k, \quad c \geq 0, \quad k = 1, 2, \ldots
\]

Laplace

\[
K(x, x_i) = \exp \left( -\frac{||x-x_i||}{\sigma} \right)
\]

(3)

where \( \sigma^2 \) represents the variance of the data and \( ||\cdot|| \) is an Euclidean distance in \( \mathcal{X} \subset \mathbb{R}^n \). This kernel has been found to be versatile and effective for many different kinds of data Prajapati and Patle (2010) and it will be the focus of our research. By substituting \( \gamma = \frac{1}{2\sigma^2} \), it can be written:

\[
K(x, x_i) = \exp \left( -\gamma ||x-x_i||^2 \right)
\]

(4)

where \( \gamma \) can be viewed as scaling factor, which is one of the parameters of the SVM classifier.
The parameter $\gamma$ controls the impact of individual SV as the kernel distance between two examples decreases with higher values of $\gamma$. Therefore, small values of $\gamma$ will result in many SV influencing the point under test $x$, 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 $\gamma$ result in more complex separating hyperplanes, better fitting the training data. However, a too high value of $\gamma$ may lead to overfitting (see Figure 1).

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 optimization, which allows some examples to be misclassified. It introduces the soft margin parameter $C > 0$ where $0 < \alpha_i < C$ a constraint on $\alpha_i$ controlling the penalty on misclassified examples and determining the trade-off between margin maximization and training error minimization. Large values to 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 Figure 1).

2.2. Setting the SVM parameters

One of the early discussions about SVM parameters was provided in Schölkopf and Smola (1998b). In the 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 Chapelle and Vapnik (2000), 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 $\gamma$ as an inverse of some aggregate (e.g. a median) of distances between data points has been proposed in a blog post Smola (2011). In fact, when searching the Internet for a method...
Figure 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 $\gamma$ parameters, both for good and bad values.
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’ method, which Caputo et al. (2002) used. However, even in surveys comparing heuristics for SVM parameter selection, Wainer and Fonseca (2021) show that sigest 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 Chapelle and Zien (2005) 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 employs its own implementation of heuristic for the $\gamma$ parameter, while this one is supervised, it can be used in an unsupervised way Soares et al. (2004).

2.2.1. Grid Search Cross Validation

As a baseline method for model selection in this article, Grid Search Cross Validation (GSCV) Berrar (2019) 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 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 $(C, \gamma)$ from given grid $G \subset \mathbb{R}^2$. Similarly to external layer, each training set $T_i = \{p_1, \ldots, p_k\} \setminus \{p_i\}$ is divided into $t$ subparts $\{p_i^1, \ldots, p_i^t\}$, with $\{p_i^1, \ldots, p_i^t\} \setminus \{p_i^i\}$ used for training with given parameters from grid $G$ and $\{p_i^i\}$ 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.

---

1 Implemented e.g. in R, see Carchedi et al. (2021)
2 https://scikit-learn.org
3 http://www.shark-ml.org/
2.3. Unsupervised heuristics for \( \gamma \)

Unsupervised heuristics usually assume that \( \gamma \) should be relative to ‘average’ distance (measured by \( \| \cdot \|^2 \)) between the examples from \( \mathcal{X} \), so that the two extreme situations – no SV influence or comparable influence of all SV – are avoided. For example, \( \gamma \) can be assigned the inverse of the data variance, which corresponds e.g. with heuristics described in Gelbart (2018) or Smola (2011)). 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. \( \gamma \) heuristics for Gaussian-distributed data

Considering a pair of examples \( (x_i, x_j) \in \mathcal{X} \times \mathcal{X} \subset \mathbb{R}^n \times \mathbb{R}^n \) from Gaussian-distributed data, it has been noted in Varewyck and Martens (2010), that the squared Euclidean distance \( \| x_i - x_j \|^2 \) is Chi-squared distributed with a mean of \( 2n\sigma^2 \), assuming that every data feature has variance \( \sigma \) and mean 0. This observation could be used as a heuristics to estimate the value of \( \gamma \) as

\[
\gamma = \frac{1}{2n\sigma^2}.
\]  

(5)

If we further assume that \( \sigma^2 = 1 \), this simplifies to \( \gamma = \frac{1}{2n} \), as noticed by authors of Wang et al. (2014).

This approach relies on an underlying assumption that data covariance matrix is in the form \( \text{Cov}(X) = I\sigma^2 \), which, in turn, means that in a matrix of examples \( X \in \mathbb{R}^{m \times n} \), 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 \( \gamma \) as:

\[
\gamma = \frac{1}{2\text{Tr}(\text{Cov}(X))},
\]  

(6)

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

9
2.3.2. Smola’s heuristics

A well-known heuristics for computing the initial value of a parameter $\gamma$ was provided by A. J. Smola in an article on his website [Smola (2011)]. Given examples $(x_i, x_j) \in \mathbb{R}^n \times \mathbb{R}^n$, he considered a kernel function in the form

$$K(x_i, x_j) = \kappa(\lambda \|x_i - x_j\|),$$  \hspace{1cm} (7)

where a scaling factor $\lambda$ of this kernel is to be estimated and $\kappa : \mathbb{R} \rightarrow \mathbb{R}^+$. The Smola’s kernel form is consistent with the RBF kernel given by Eq. (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 $(x_i, x_j)$ and to compute their distances. Then, the value of $\lambda$ 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 $\gamma$ heuristics

A heuristic for choosing SVM parameters can be found in [Chapelle and Zien (2005)]. Interestingly, to the best of our knowledge it is the only method that estimates both $C$ and $\gamma$ 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 $\rho \rightarrow 0$ defaults to the Gaussian kernel. This kernel proposition is based on minimal $\rho$-path distance $D_{ij}^{\rho}$ which, for $\rho \rightarrow 0$ becomes Euclidean distance i.e. $D_{ij}^{\rho \rightarrow 0} = \|x_i - x_j\|_2$.

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 [Chapelle and Zien (2005)] 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 $\sigma$ (Equation 3) is computed as $\frac{1}{n_c}$-th quantile of $D = \{D_{ij}^\rho : X \times X \in \mathbb{R}^n \times \mathbb{R}^n\}$ where $n_c$ is the
number of classes. For Gaussian RBF kernel this results in

\[ \gamma = \frac{1}{2 \text{ quantile}_{\frac{1}{nc}}(D)} \]  

(8)

Note that we consider only the case \( \rho \to 0 \), as only under this condition heuristics proposed in [Chapelle and Zien (2005)] are comparable with other heuristics presented in this Section and compatible with our experiment. However, the authors’ original formulation allows for other values of \( \rho \). This heuristic, along with the complimentary for the \( C \) parameter (see Section 2.4.1) are denoted in the experiments as Chapelle.

2.3.4. Jaakkola’s and Soares’ heuristics

While the original Jaakkola’s heuristics, described in [Jaakkola et al. (1999)] and [Jaakkola et al. (2000)], was supervised, in this article we will focus on its unsupervised version proposed in [Soares et al. (2004)].

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

\[ D_l = \{ d_{\text{min}}^l(\mathbf{x}) : \mathbf{x} \in \mathcal{X} \}, \]  

(9)

and the value of \( \sigma = \text{median}(D_l) \).

This approach, however, has been interpreted differently in [Soares et al. (2004)], which resulted in an unsupervised heuristic based on what was proposed in [Jaakkola et al. (1999)]. The approach to estimate \( \sigma \) 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 \( d_{\text{min}}(\mathbf{x}) \) of an example \( \mathbf{x} \) to its closest neighbour, the set of all neighbour distances is computed as

\[ D = \{ d_{\text{min}}(\mathbf{x}) : \mathbf{x} \in \mathcal{X} \}, \]  

(10)

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

The use of mean instead of median in an approach proposed in [Soares et al. (2004)] results in larger values of \( \gamma \) in the case of outliers in the data space. Therefore, following the reasoning in the original manuscript [Jaakkola]
et al. (1999), we propose to compute \( \sigma = \text{median}(D) \), which in case of the Gaussian RBF kernel results in:

\[
\gamma = \frac{1}{2 \, \text{median}(D)}.
\]  

(11)

This heuristic is denoted as Soares_med.

2.3.5. Gelbart’s heuristics

The heuristic used to estimate the initial value of \( \gamma \) in a well-known Python library scikit-learn, was proposed by Michael Gelbart in Gelbart (2018). The scaling factor of Gaussian RBF kernel is computed as

\[
\gamma = \frac{1}{n \text{Var}(X)},
\]  

(12)

where \( X \subset \mathbb{R}^n \) and \( \text{Var}(X) \) is a variance of all elements in the data set \( X \). It is easy to see that this heuristic is similar to the one discussed in Section 2.3.1 based on Varewyck and Martens (2010): provided that every data feature has variance \( \sigma \) 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 \( \gamma \); in Schölkopf and Smola (1998b), 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 Chapelle and Zien (2005), which we discuss below.

\[\text{https://github.com/scikit-learn/scikit-learn/issues/12741}\]
2.4.1. Chapelle & Zien C heuristic

Given a $\gamma$ value (originally computed as described in Section 2.3.3), Chapelle and Zien (2005) calculate the empirical variance:

$$s^2 = \frac{1}{m} \sum_{i=1}^{m} K(x_i, x_i) - \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} K(x_i, x_j),$$

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

$$s^2 = 1 - a, \quad a = \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} K(x_i, x_j).$$

The $C$ parameter value is then estimated as

$$C = \frac{1}{s^2}. \quad (15)$$

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

2.4.2. A proposed extension of Chapelle & Zien C 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. Therefore we propose a new version of the heuristic, by modifying the formula 14. Since in formula 14 the factor $a < 1$, higher values of $C$ can be achieved by substituting $s^2 = 1 - a'$ with $a \leq a' < 1$.

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

$$\mathcal{A} = \{\|x_i - x_j\| : i, j \leq m; \ x_i, x_j \in X\}. \quad (16)$$

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

$$\mathcal{B} = \{(i, j) : \|x_i - x_j\| \in \mathcal{A}'\}. \quad (17)$$
This leads to a modified version of the heuristic

$$s^2 = 1 - a', \quad a' = \frac{1}{t} \sum_{(i,j) \in B} K(x_i, x_j),$$

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

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 Alcalá-Fdez et al. (2011). 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 Dua and Graff (2017) 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 Duch et al. (2012), 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 Moreno-Torres et al. (2012) or through OpenML project Feurer et al. (2021). 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 $h \in \mathcal{H}$ from the set of tested heuristics $\mathcal{H}$ be a function that generates SVM parameters $\{C, \gamma\}$ based on a supplied training set $\mathcal{T}$ i.e. $h: \mathcal{T} \rightarrow \mathbb{R}^2$. 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 $\mathcal{T}_i$ corresponding with a given $i$–fold of the external CV, and for every heuristics $h \in \mathcal{H}$ 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 $\mathcal{T}_i$.
2. by applying the heuristics $h(\mathcal{T}_i)$,
3. by performing a grid-search around the initial parameters $h(T_i)$ and selecting the best model in the internal CV on $T_i$.

The range of parameters for GSCV to test is not always easy to determine as different studies propose different ranges - in Matheny et al. (2007) the range $\{0, 0.1, 0.3, 0.5, 0.7\}$ is taken into consideration for $C$, while for $\gamma$ its $\{2^{-4}, 2^{-3}, \ldots, 2^4\}$. Authors of Schuhmann et al. (2021) 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 Budiman (2019) the selected range was $\{2^{-17}, 2^{-16}, \ldots, 2^3\}$ for $\gamma$ and $\{2^{-3}, 2^{-2}, \ldots, 2^{17}\}$ for $C$. In Lameski et al. (2015), the authors decided to use the grid of $10^{-6}, \ldots, 10^6$ for both $C$ and $\gamma$.

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

$$G_h = \{ri_\gamma : r \in R\} \times \{ri_C : r \in R\}, \quad (19)$$

where $h(T_i) = (i_\gamma, i_C)$. 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 Brodersen et al. (2010) (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 Benavoli et al. (2017). 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 Benavoli et al. (2017) 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 then 1%. This allows 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 regards 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 Benavoli et al. (2017) was employed. Following the suggestion in Benavoli et al. (2017), 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 Benavoli et al. (2017) and its visualisation was performed using baycomp library v. 1.0.2. Matplotlib and seaborn libraries were used for data visualisation.
Figure 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 $50 \times 50$ 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.
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 regards 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 \(\gamma\) parameter, and only two of them estimate the \(C\) (Chapelle, MC), we present results as a combination of every \(\gamma\) and \(C\) heuristics 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 Section 3.3; their practical equivalence corresponds to sufficiently small difference, with regards to rope value of 1%.

When no heuristics or only \(\gamma\) heuristics are used, parameters obtained by GSCV result in significantly higher accuracy. There’s only a marginal improvement when Chapelle heuristics is used for selecting a \(C\) parameter value. However, when using our proposed extension, the MC heuristics, five of the \(\gamma\) 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 Figure 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’s performance.

The practical equivalence in the accuracy of classifiers whose parameters were chosen by GSCV and heuristics, is also visible during the inspection of

\(^7\)https://github.com/janezd/baycomp

\(^8\)For the reference, results of experiments for OA measure are presented in the Appendix 4.
the parameter values obtained from heuristics plotted on the graph showing the relationship between the classifier’s effectiveness and its parameters (estimated through a dense grid of parameters). In the selected representative examples on Figure 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 indicates that the median distance between examples in the data space is of particular importance when choosing the $\gamma$ 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 $\gamma$ and $C$ parameter extends the calculation time.

To summarise: estimation of both parameters, in particular with Covtrace+MC heuristics, leads to accuracy practically equivalent to GSCV (see Figure 3c) with parameters obtained in only $\sim$0.006 of its working time (see Table 3). 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 training data is limited.

GSCV is by no means inferior to the heuristics, especially if supplied with proper number of labelled datapoints. In practice, however, the differences are often very small. and while it is natural to use GSCV when standard approach is preferable (small number of examples, training time is not an issue), in many scenarios i.e. processing on edge IoT devices, proposed heuristics offers practically equivalent accuracy in fraction of time.
Figure 3: Visualisation of Bayesian analysis of results with methodology from Benavoli et al. (2017) 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 that the better the C heuristics, the closer to equivalence of UH-SVM and GSCV-SVM. Our proposed extension (MC) provides the best results. The tendency visible in this plot is similar across other well-performing γ heuristics.
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 Benavoli et al. (2017). Three right columns present probabilities of cross-validation being on average more/ equivalently / less accurate than heuristics. Results were obtained for the balanced accuracy measure and rope value of 1%. Note that, with proposed (MC) heuristic, several of \( \gamma \) heuristics achieve results very close to GSCV.

| C heuristics | \( \gamma \) 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       |
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 | 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|

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 heuristics for the $C$ parameter as optimisation of both parameters is vital for accurate classifiers. We compared results with methodology based on Bayesian analysis, described in Benavoli et al. (2017). Our results indicate that heuristics and in particular the proposed $covtrace+MC$, are usually practically equivalent to GSCV i.e. obtained accuracies differ by less than 1% (see Figure 3c and probabilities of equivalence in Table 2). Moreover, these heuristics offer a computation time reduction, achieving a 100-200 times speed-up (see Table 3). This makes unsupervised, heuristic approach to parameters selection a compelling alternative for GSCV for rapid SVM calibration.

References

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F., 2011. Keel data-mining software tool: data set repository,
integration of algorithms and experimental analysis framework. Journal of Multiple-Valued Logic & Soft Computing 17.

Alloghani, M., Al-Jumeily, D., Mustafina, J., Hussain, A., Aljaaf, A.J., 2020. A systematic review on supervised and unsupervised machine learning algorithms for data science. Supervised and unsupervised learning for data science , 3–21.

An, S., Liu, W., Venkatesh, S., 2007. Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression. Pattern Recognition 40, 2154–2162. doi: https://doi.org/10.1016/j.patcog.2006.12.015

Benavoli, A., Corani, G., Demšar, J., Zaffalon, M., 2017. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. The Journal of Machine Learning Research 18, 2653–2688.

Berrar, D., 2019. Cross-validation.

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

Budiman, F., 2019. Svm-rbf parameters testing optimization using cross validation and grid search to improve multiclass classification. Scientific Visualization 11, 80–90.

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

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

Cervantes, J., Garcia-Lamont, F., Rodríguez-Mazahua, L., Lopez, A., 2020. A comprehensive survey on support vector machine classification: Applications, challenges and trends. Neurocomputing 408, 189–215.
Chapelle, O., Vapnik, V., 2000. Model selection for support vector machines, in: Advances in neural information processing systems, pp. 230–236.

Chapelle, O., Zien, A., 2005. Semi-supervised classification by low density separation., in: AISTATS, Citeseer. pp. 57–64.

Chen, W.H., Hsu, S.H., Shen, H.P., 2005. Application of SVM and ANN for intrusion detection. Computers & Operations Research 32, 2617–2634. doi: https://doi.org/10.1016/j.cor.2004.03.019

Cheng, G., Xie, X., Han, J., Guo, L., Xia, G.S., 2020. Remote sensing image scene classification meets deep learning: Challenges, methods, benchmarks, and opportunities. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 13, 3735–3756.

Cholewa, M., Głomb, P., Romaszewski, M., 2019. A spatial-spectral disagreement-based sample selection with an application to hyperspectral data classification. IEEE Geoscience and Remote Sensing Letters 16, 467–471. doi: https://doi.org/10.1109/LGRS.2018.2868862

Cortes, C., Vapnik, V., 1995. Support-vector networks. Machine learning 20, 273–297. doi: https://doi.org/10.1007/BF00994018

Direito, B., Teixeira, C.A., Sales, F., Castelo-Branco, M., Dourado, A., 2017. A realistic seizure prediction study based on multiclass SVM. International journal of neural systems 27, 1750006. doi: https://doi.org/10.1142/S012906571750006X

Drucker, H., Burges, C.J., Kaufman, L., Smola, A.J., Vapnik, V., et al., 1997. Support vector regression machines. Advances in neural information processing systems 9, 155–161.

Dua, D., Graff, C., 2017. UCI machine learning repository. URL: http://archive.ics.uci.edu/ml.

Duch, W., Jankowski, N., Maszczyk, T., 2012. Make it cheap: learning with o(nd) complexity, in: The 2012 International Joint Conference on Neural Networks (IJCNN), IEEE. pp. 1–4. doi: https://doi.org/10.1109/IJCNN.2012.6252380
Ebrahimi, M., Khoshtaghaza, M., Minaei, S., Jamshidi, B., 2017. Vision-based pest detection based on SVM classification method. Computers and Electronics in Agriculture 137, 52–58. URL: https://www.sciencedirect.com/science/article/pii/S016816991631136X, doi:https://doi.org/10.1016/j.compag.2017.03.016.

Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D., 2014. Do we need hundreds of classifiers to solve real world classification problems? The journal of machine learning research 15, 3133–3181.

Feurer, M., van Rijn, J.N., Kadra, A., Gijsbers, P., Mallik, N., Ravi, S., Müller, A., Vanschoren, J., Hutter, F., 2021. OpenML-Python: an extensible Python API for openML. Journal of Machine Learning Research 22, 1–5.

Gelbart, M., 2018. Gamma=’scale’ in SVC. https://github.com/scikit-learn/scikit-learn/issues/12741. Accessed: 10-06-2021.

Głomb, P., Romaszewski, M., Cholewa, M., Domino, K., 2018. Application of hyperspectral imaging and machine learning methods for the detection of gunshot residue patterns. Forensic Science International 290, 227–237. doi:https://doi.org/10.1016/j.forsciint.2018.06.040.

Gupta, B.B., Tewari, A., Cvitić, I., Peraković, D., Chang, X., 2022. Artificial intelligence empowered emails classifier for internet of things based systems in industry 4.0. Wireless Networks 28, 493–503. URL: https://doi.org/10.1007/s11276-021-02619-w, doi:10.1007/s11276-021-02619-w.

Hsu, C.W., Lin, C.J., 2002. A comparison of methods for multiclass support vector machines. IEEE transactions on Neural Networks 13, 415–425. doi:https://doi.org/10.1109/72.991427.

Huang, H.Y., Kueng, R., Torlai, G., Albert, V.V., Preskill, J., 2022. Provably efficient machine learning for quantum many-body problems. Science 377, eabk3333.

Jaakkola, T., Diekhans, M., Haussler, D., 2000. A discriminative framework for detecting remote protein homologies. Journal of computational biology 7, 95–114. doi:https://doi.org/10.1089/10665270050081405.
Jaakkola, T.S., Diekhans, M., Haussler, D., 1999. Using the fisher kernel method to detect remote protein homologies., in: ISMB, pp. 149–158.

Kim, H.C., Pang, S., Je, H.M., Kim, D., Bang, S.Y., 2003. Constructing support vector machine ensemble. Pattern recognition 36, 2757–2767. doi:https://doi.org/10.1016/S0031-3203(03)00175-4

Koklu, M., Ozkan, I.A., 2020. Multiclass classification of dry beans using computer vision and machine learning techniques. Computers and Electronics in Agriculture 174, 105507.

Lameski, P., Zdravevski, E., Mingov, R., Kulakov, A., 2015. 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, Proceedings, Springer. pp. 464–474.

Lecomte, S., Lengellé, R., Richard, C., Capman, F., Ravera, B., 2011. 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. pp. 124–129. doi:https://doi.org/10.1109/AVSS.2011.6027306

Li, L., Zhang, S., Wang, B., 2021. Plant disease detection and classification by deep learning—a review. IEEE Access 9, 56683–56698.

Li, X., Guo, X., 2013. 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, IEEE. pp. 263–266. doi:https://doi.org/10.1109/IHMSC.2013.69

Matheny, M.E., Resnic, F.S., Arora, N., Ohno-Machado, L., 2007. Effects of SVM parameter optimization on discrimination and calibration for post-procedural PCI mortality. Journal of Biomedical Informatics 40, 688–697.

Menter, Z., Tee, W.Z., Dave, R., 2021. Application of machine learning-based pattern recognition in iot devices: Review, in: Kumar, S., Purohit, S.D., Hiranwal, S., Prasad, M. (Eds.), Proceedings of International Conference on Communication and Computational Technologies, Springer Singapore, Singapore. pp. 669–689.
Moreno-Torres, J.G., Sáez, J.A., Herrera, F., 2012. Study on the impact of partition-induced dataset shift on k-fold cross-validation. IEEE Transactions on Neural Networks and Learning Systems 23, 1304–1312. doi:https://doi.org/10.1109/TNNLS.2012.2199516.

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

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

Prajapati, G.L., Patle, A., 2010. 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. pp. 512–515.

Romaszewski, M., Głomb, P., Cholewa, M., 2016. Semi-supervised hyperspectral classification from a small number of training samples using a co-training approach. ISPRS Journal of Photogrammetry and Remote Sensing 121, 60–76. doi:https://doi.org/10.1016/j.isprsjprs.2016.08.011.

Schölkopf, B., Smola, A.J., 1998a. From regularization operators to support vector kernels. Adv. Neural Inf. Process. Syst 10, 343–349.

Schölkopf, B., Smola, A.J., 1998b. Learning with kernels. volume 4. Citeseer.

Schuhmann, R.M., Rausch, A., Schanze, T., 2021. 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. Current Directions in Biomedical Engineering 7, 121–124.
Sha’abani, M., Fuad, N., Jamal, N., Ismail, M., 2020. kNN and SVM classification for EEG: a review. InECCE2019, 555–565. doi: https://doi.org/10.1007/978-981-15-2317-5_47.

Shah, K., Patel, H., Sanghvi, D., Shah, M., 2020. A comparative analysis of logistic regression, random forest and knn models for the text classification. Augmented Human Research 5, 1–16.

Smola, A.J., 2011. Easy kernel width choice. https://blog.smola.org/post/940859888/easy-kernel-width-choice. Accessed: 10-06-2021.

Soares, C., Brazdil, P.B., Kuba, P., 2004. A meta-learning method to select the kernel width in support vector regression. Machine learning 54, 195–209. doi: https://doi.org/10.1023/B:MACH.0000015879.28004.9b.

Song, J., Takakura, H., Okabe, Y., Kwon, Y., 2009. Unsupervised anomaly detection based on clustering and multiple one-class SVM. IEICE transactions on communications 92, 1981–1990. doi: https://doi.org/10.1587/transcom.E92.B.1981.

Subashini, T., Ramalingam, V., Palanivel, S., 2009. Breast mass classification based on cytological patterns using RBFNN and SVM. Expert Systems with Applications 36, 5284–5290. doi: https://doi.org/10.1016/j.eswa.2008.06.127.

Vapnik, V., Lerner, A.Y., 1963. Recognition of patterns with help of generalized portraits. Avtomat. i Telemekh 24, 774–780.

Varewyck, M., Martens, J.P., 2010. A practical approach to model selection for support vector machines with a gaussian kernel. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 41, 330–340. doi: https://doi.org/10.1109/TSMCB.2010.2053026.

Wainer, J., Fonseca, P., 2021. How to tune the RBF SVM hyperparameters? an empirical evaluation of 18 search algorithms. Artificial Intelligence Review, 1–27. doi: https://doi.org/10.1007/s10462-021-10011-5.

Wang, P., Mathieu, R., Ke, J., Cai, H., 2010. Predicting criminal recidivism with support vector machine, in: 2010 International Conference on Management and Service Science, IEEE. pp. 1–9. doi: https://doi.org/10.1109/ICMSS.2010.5575352.
Wang, X., Huang, F., Cheng, Y., 2014. Super-parameter selection for gaussian-kernel SVM based on outlier-resisting. Measurement 58, 147–153. doi:https://doi.org/10.1016/j.measurement.2014.08.019.

Yang, W., Wang, S., Sahri, N.M., Karie, N.M., Ahmed, M., Valli, C., 2021. Biometrics for internet-of-things security: A review. Sensors 21, 6163.

Zhang, J., Wang, S., 2016. A fast leave-one-out cross-validation for SVM-like family. Neural Computing and Applications 27, 1717–1730. doi:https://doi.org/10.1007/s00521-015-1970-4
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 [Benavoli et al. 2017]. 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 proposed (MC) heuristic, several of γ heuristics achieve results very 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      |