Closed-Form Gaussian Spread Estimation for Small and Large Support Vector Classification

Diego Isla-Cernadas, Manuel Fernández-Delgado, Eva Cernadas, Manisha S. Sirsat, Haitham Maarouf, and Senén Barro

***Abstract*— The support vector machine (SVM) with Gaussian kernel often achieves state-of-the-art performance in classification problems, but requires the tuning of the kernel spread. Most optimization methods for spread tuning require training, being slow and not suited for large-scale datasets. We formulate an analytic expression to calculate, directly from data without iterative search, the spread minimizing the difference between Gaussian and ideal kernel matrices. The proposed direct gamma tuning (DGT) equals the performance of and is one to two orders of magnitude faster than the state-of-the art approaches on 30 small datasets. Combined with random sampling of training patterns, it also runs on large classification problems. Our method is very efficient in experiments with 20 large datasets up to 31 million of patterns, it is faster and performs significantly better than linear SVM, and it is also faster than iterative minimization. Code is available upon paper acceptance from this link: http://persoal.citius.usc.es/manuel.fernandez.delgado/papers/dgt/index.html and from CodeOcean: https://codeocean.com/capsule/4271163/tree/v1.**

***Index Terms*— Classification, efficient computing, large-scale datasets, model selection, radial basis kernel, support vector machine (SVM).**

I. Introduction

THE support vector machine (SVM) is a popular classifier that can use several kinds of kernels, being the radial basis function (RBF) very used because of its good behavior in terms of performance [1]. A major configuration issue for the SVM is the tuning of its hyperparameters: regularization or penalty (λ) and spread (σ) of the RBF kernel, which has a specially strong influence on performance. Its tuning has been largely studied in the literature and it will be the focus of this article. Often, σ is selected by searching the value that maximizes the SVM performance from a collection of values (grid-search (GS) approach), thus requiring to train and test the SVM for each σ value. This also happens with random search [2], although the number of values tested is reduced with possible performance loss. Alternative model selection criteria for the SVM were compared in [3] on physiological data, including distance between two classes and expected square distance ratio. Other in-sample statistical approaches were also proposed for model selection and error estimation [4].

Among optimization methods, genetic algorithms (GAs) were used in [5] updating the strategy parameters and the values of λ and σ with the covariance data matrix to maximize the average test accuracy of several SVMs. The method was applied only on small datasets up to 768 patterns and 13 features because SVM training and covariance matrix calculation were slow. Both the hyperparameters were also selected using particle swarm optimization (PSO) in the modeling of Iju deposit mineralization and alteration zones [6], and using dynamic PSO [7] on 14 datasets up to 7000 training patterns, outperforming GS, standard, and chained PSO.

Penalty and spread were also selected using differential evolution [8] to maximize the accuracy on four datasets up to 11 692 patterns. Multiobjective adaptive differential evolution [9] was proposed to minimize the number of support vectors and maximize the generalization capacity in SVM model selection with RBF, polynomial, and Hermite kernels, being validated on 11 datasets up to 4435 patterns. The Bat algorithm [10], inspired in swarm intelligence, outperformed GA and PSO on nine small datasets up to 958 patterns, minimizing the classification error and avoiding the fall into local minima. Tharwat and Gabel [11] combined the social ski driver algorithm and synthetic minority oversampling technique (SMOTE) to select λ and σ maximizing sensitivity. The method outperformed GS and PSO over eight small unbalanced datasets up to 336 patterns and 11 features. Glasmachers and Igel [12] developed a model selection method for 1-norm soft-margin SVM using gradient ascent on a likelihood function of λ and σ based on logistic regression, being validated on 28 datasets up to 5000 patterns. The spread was also selected [13] by maximizing the margin in the feature space, while regularization is calculated analytically using the jackknife estimate of perturbations in the eigenvalues of the kernel matrix. The experiments included 24 datasets up to 7400 patterns. Model selection for multiclass SVM was studied [14] using spherical and elliptical RBF kernels and two criteria that redefine the SVM radius-margin bound in terms of class separability. The method was validated on 13 datasets up to 6435 patterns, where it spent 232 s using spatial GRBF kernel (dag) and criterion II.

Manuscript received 14 February 2023; revised 3 October 2023 and 12 December 2023; accepted 12 March 2024. This work was supported in part by the Consellería de Educación, Universidade e Formación Profesional under Grant ED431G-2019/04 and in part by the European Regional Development Fund (ERDF), through the Centro Singular de Investigación en Tecnoloxías Intelixentes da Universidade de Santiago de Compostela (CiTIUS) as a Research Center of the Galician University System. *(Corresponding author: Manuel Fernández-Delgado.)*

Diego Isla-Cernadas, Manuel Fernández-Delgado, Eva Cernadas, Haitham Maarouf, and Senén Barro are with the Centro Singular de Investigación en Tecnoloxías Intelixentes da USC (CiTIUS), University of Santiago de Compostela, 15782 Santiago de Compostela, Spain (e-mail: manuel.fernandez.delgado@usc.es).

Manisha S. Sirsat is with the Department of Data Management and Risk Analysis, InnovPlantProtect, 7350-478 Elvás, Portugal.

Digital Object Identifier 10.1109/TNNLS.2024.3377370

Several approaches calculate the spread directly from data. Xu et al. [15] formulated a direct formula to set σ using local and global distances, thus requiring to sort the distances between training patterns randomly selected. The method was validated on 13 datasets up to 7400 patterns. Varewick and Martens [16] calculated σ using a simple analytical formula of the input dimensionality and the class dispersion, without any SVM train or test, evaluating its method on 17 datasets up to 8124 patterns. Several alternative ways to calculate σ based on local features, such as soft k-nearest-neighbor, nearest enemy, and redundant fast clustering estimations [17], were compared with tenfold GS on six datasets up to 7400 patterns.

Liu and Xu [18] selected σ maximizing (resp. minimizing) simultaneously between-class (resp. within-class) separability, measured by the cosine similarity in the kernel space. The method used eight small datasets up to 400 patterns. Afterward, Liu et al. [19] proposed an analytical formula valid when the within-class mean distance is below the between-class mean distance, using 17 classification datasets up to 141 691 patterns spending 43.4 s. The random RBF kernel SVM [20] used a modified RBF kernel with random-generated parameters that maximized the SVM accuracy on 18 datasets up to 32 561 patterns. Menezes et al. [21] selected σ by maximizing a dissimilarity function between two classes based on an RBF-based kernel density estimation (KDE). This method was efficient and outperformed GS on a collection of 18 small datasets up to 3210 patterns.

In a previous work [22], we proposed the ideal kernel tuning (IKT), where σ minimizes the difference between RBF and ideal kernel matrices. On 37 datasets up to 1 million of patterns, IKT achieved lower time (up to 384 s) and memory requirements, with performance similar to GS, KDE, GA, Bayesian search, and PSO. Starting from this idea, this article derives an efficient closed-form expression for the inverse γ of spread that minimizes the previous difference, extending it to datasets much larger than the ones in previous approaches. Sections II and III describe the proposed methods and experimental results, respectively, while Section IV reports the conclusions of the current study.

II. Materials and Methods

The ideal kernel J for a classification problem [22] is a function defined as $J(\mathbf{x},\mathbf{y}) = 1$ when $\mathbf{x}$ and $\mathbf{y}$ share the class label, and $J(\mathbf{x},\mathbf{y}) = 0$ otherwise. Let $\{\mathbf{x}_n\}_{n=1}^N$ be the set of training patterns, c_n be the class label of $\mathbf{x}_n$, with $c_n \in \{1,\ldots,C\}$, and C be the number of classes. The ideal kernel matrix $\mathbf{J}$, that is squared of order N, has elements $\{J_{nm}\}_{nm=1}^N$ defined as $J_{nm} = J(\mathbf{x}_n,\mathbf{x}_m) = 1$ when $c_n = c_m$ and $J_{nm} = J(\mathbf{x}_n,\mathbf{x}_m) = 0$ otherwise. To achieve a good performance in classification problems, a kernel should be so similar as possible to the ideal kernel. Specifically, an RBF kernel

$$K(\mathbf{x},\mathbf{y},\sigma) = \exp\left(\frac{-|\mathbf{x}-\mathbf{y}|^2}{2\sigma^2}\right) \tag{1}$$

is expected to provide the best performance when σ minimizes the difference between the RBF and ideal kernels, which can be evaluated as the mean square difference $D(\sigma)$ between their respective matrices

$$D(\sigma) = \frac{1}{M}\sum_{n=1}^{N-1}\sum_{m=n+1}^{N}[K_{nm}(\sigma) - J_{nm}]^2. \tag{2}$$

Here, $K_{nm}(\sigma) = K(\mathbf{x}_n,\mathbf{x}_m,\sigma)$ and the sum is over $m > n$ because $K_{nn} = 1$ and $K_{nm} = K_{mn}$, being $M = N(N-1)/2$ the number of terms in the sum. Our strategy in previous works [22], [23] was to calculate $D(\sigma)$ for several σ values and to select σ minimizing $D(\sigma)$. In the following, we develop a method to estimate σ directly from the training set $\{\mathbf{x}_n, c_n\}_{n=1}^N$. We define

$$\gamma = \frac{1}{2\sigma^2}, \quad d_{nm} = -|\mathbf{x}_n - \mathbf{x}_m|^2, \quad K_{nm}(\gamma) = \mathrm{e}^{\gamma d_{nm}}. \tag{3}$$

Thus, γ is the inverse of double squared kernel spread σ, while d_{nm} is minus the squared distance between $\mathbf{x}_n$ and $\mathbf{x}_m$. The values $\{d_{nm}\}_{nm=1}^N$ compose the $N \times N$-order distance matrix $\mathbf{D}$. The difference $D(\gamma)$ is

$$D(\gamma) = \frac{1}{M}\sum_{nm}(\mathrm{e}^{\gamma d_{nm}} - J_{nm})^2. \tag{4}$$

Deriving $D(\gamma)$, using that $K'_{nm}(\gamma) = \mathrm{e}^{\gamma d_{nm}} d_{nm}$ and equaling to zero, we achieve

$$\frac{dD}{d\gamma} = \frac{2}{M}\sum_{nm}(\mathrm{e}^{\gamma d_{nm}} - J_{nm})\mathrm{e}^{\gamma d_{nm}} d_{nm} = 0. \tag{5}$$

This equation does not allow to calculate an analytic solution for γ, but an estimation might be achieved by supposing that all the distances d_{nm} are equal to their expected or mean value, denoted by d

$$d_{nm} = d = \frac{1}{M}\sum_{pq} d_{pq} \tag{6}$$

where the sum over p, q has the same limits as previously with n, m. Using this hypothesis, (5) becomes

$$\sum_{nm}(\mathrm{e}^{\gamma d} - J_{nm})\mathrm{e}^{\gamma d} d = 0 \rightarrow \mathrm{e}^{\gamma d}\sum_{nm} 1 = \sum_{nm} J_{nm} \tag{7}$$

where we divided by $\mathrm{e}^{\gamma d} d$. Note that $\sum_{nm} 1 = M$. Besides, $\sum_{nm} J_{nm}$ is the number of pairs of patterns (n, m) where $c_n = c_m$. For $k = 1,\ldots,C$, the class k with N_k training patterns has $N_k(N_k-1)/2$ pairs, so that

$$\sum_{nm} J_{nm} = \frac{1}{2}\sum_{k=1}^{C}(N_k^2 - N_k) \tag{8}$$

and (7) becomes

$$M\mathrm{e}^{\gamma d} = \frac{1}{2}\sum_{k=1}^{C}(N_k^2 - N_k). \tag{9}$$

Substituting d from (6), the estimated value of γ is

$$\gamma = \frac{-\ln\left[\frac{1}{2M}\sum_{k=1}^{C}(N_k^2 - N_k)\right]}{\frac{1}{M}\sum_{n=1}^{N-1}\sum_{m=n+1}^{N}|\mathbf{x}_n - \mathbf{x}_m|^2}. \tag{10}$$

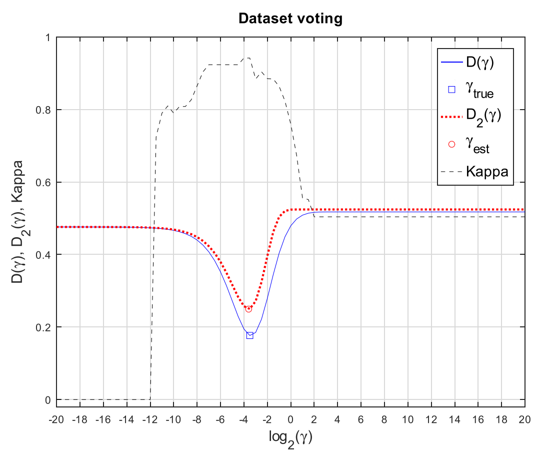


Fig. 1. Values of D, D_2, and kappa varying γ from 2^{-20} to 2^{20}, with the true and estimated γ values minimizing D and D_2 (square and circle, respectively) in dataset `voting`.

Theorem 1: The function $D(\gamma)$ in (4) has a minimum at the value of γ calculated by (10) under hypothesis in (6).

Proof: The function $D(\gamma)$ has a minimum for γ in (10) if $D''(\gamma) > 0$. This second derivative is

$$\frac{d^2D}{d\gamma^2} = \frac{2}{M}\sum_{nm}(2d_{nm}e^{2\gamma d_{nm}} - d_{nm}J_{nm}e^{\gamma d_{nm}})d_{nm}. \qquad (11)$$

Using that $d_{nm} = d$ and $\gamma = (\ln p)/d$, so that $e^{\gamma d} = p$ and $e^{2\gamma d} = p^2$, we achieve

$$\frac{d^2D}{d\gamma^2} = \frac{2}{M}\sum_{nm}(2p^2d^2 - pJ_{nm}d^2)$$
$$= \frac{2}{M}\left(2\ p^2Md^2 - pd^2\sum_{nm}J_{nm}\right) = 2\ p^2d^2 > 0 \qquad (12)$$

where we used that $\sum_{nm} 1 = M$ and $\sum_{nm} J_{nm} = Mp$. □

Note that class populations N_k in (10) require the class labels that define the classification problem, being therefore necessary to calculate the optimal γ. Using d defined by (6) instead of d_{nm} transforms $D(\gamma)$ in (4) to

$$D_2(\gamma) = \frac{1}{M}\sum_{nm}\left(e^{\gamma d} - J_{nm}\right)^2 = e^{2\gamma d} + p\left(1 - 2e^{\gamma d}\right) \qquad (13)$$

where $p = (1/M)\sum_{nm} J_{nm}$. It follows that $D_2'(\gamma) = 0$ and $D_2(\gamma) = p(1-p)$ for $\gamma = (\ln p)/d$ as in (10). Note that $D(0) = 1 - p$ and $D(\infty) = p$. Fig. 1 shows in this case for dataset `voting` that $D(\gamma)$ and $D_2(\gamma)$ are very similar and their minima are very near. Thus, although the distances $\{d_{nm}\}$ are not equal to their mean d, the deviation is somehow compensated and the value of γ that minimizes $D(\gamma)$ can be calculated as if $\{d_{nm}\}$ were equal to their mean. Besides, both the minima are located at the maximum of performance on the validation set, measured by the Cohen kappa statistic, so that minimizing D or D_2 also maximizes performance. The proposed method to estimate γ has been named **direct gamma tuning** (DGT). Algorithm 1 describes DGT for small datasets (DGS), that trains the SVM using the γ calculated using (10) by procedure `gamma` (algorithm 2) on the whole training set $\{\mathbf{x}_n, c_n\}_{n=1}^N$.

Algorithm 1 DGT, Small Datasets.

```
1 Algorithm: $\mathcal{S}$=DGS($\{\mathbf{x}_n, c_n\}_{n=1}^N$, $\lambda$)
    Data: $\{\mathbf{x}_n, c_n\}_{n=1}^N$: training patterns and class labels
          $c_n \in \{1, \ldots, C\}$; $\lambda$: regularization parameter.
    Result: $\mathcal{S}$: trained SVM.
2 $\gamma \leftarrow$gamma($\{\mathbf{x}_n, c_n\}_{n=1}^N$) #procedure gamma in alg. 2
3 $\mathcal{S} \leftarrow$SVMTrain($\{\mathbf{x}_n, c_n\}_{n=1}^N$,$\lambda$,$\gamma$)
```

Algorithm 2 Calculation of γ.

```
1 Algorithm: $\gamma$=gamma($\{\mathbf{w}_h, b_h\}_{h=1}^H$)
    Data: $\{\mathbf{w}_h, b_h\}_{h=1}^H$: training patterns and class labels
          $b_h \in \{1, \ldots, C\}$.
    Result: $\gamma$: inverse of double squared kernel spread $\sigma$.
2 $M \leftarrow \frac{H(H-1)}{2}$; $d \leftarrow \frac{-1}{M}\sum_{h=1}^{H-1}\sum_{m=h+1}^{H}|\mathbf{w}_h - \mathbf{w}_m|^2$
3 $\left\{N_k \leftarrow \sum_{h=1, b_h=k}^{H} 1\right\}_{k=1}^{C}$; $p \leftarrow \frac{1}{2M}\sum_{k=1}^{C}(N_k^2 - N_k)$
4 $\gamma = \frac{1}{d}\ln\left(\frac{p}{M}\right)$
```

When the number N of training patterns is high, the calculation of distances $\{d_{nm}\}_{n=1,m=n-1}^{N-1,N}$ becomes expensive. Besides, the SVM cannot be trained on the whole dataset. The approach proposed in [23] by the fast support vector classifier (FSVC) for large datasets is to use a reduced set of 100 prototypes of each class instead of the whole training set. This low number of prototypes is because the method requires to calculate the distances between these prototypes and the training patterns, which is slow when many prototypes are used. This article is focused on the standard SVM instead of FSVC, and the SVM performance might be very poor using so few training patterns. In addition, a large number of prototypes would slow down the distance computation, so we propose to replace prototypes by training patterns randomly selected, i.e., to perform a random sampling on the training set, for the distance calculation. This sampling must be performed by keeping the relative class populations in the original set.

The number $L < N$ of patterns to be selected for SVM training is very important: low values might reduce performance, and high values slow down the training and may lead to memory failures. Larger datasets require larger training sets, so L must be increasing with N as, e.g., $L = \alpha N$ with $\alpha < 1$. To avoid errors in SVM training for high N, an upper bounded L_0 is required for L. The values of α and L_0 should be carefully set to achieve a good tradeoff between performance and speed. The upper panel of Fig. 2 reports kappa versus time when L raises from 1000 to N (from left to right in each line) for several large datasets used in the experimental work (Section III-B). The time raises with L because the training is slower. Kappa raises slowly in some datasets (`magic`, `letter`, `adult`, and `shuttle`) and faster in others with larger datasets (`chess`, `wisdm`, and `ijcnn1`).

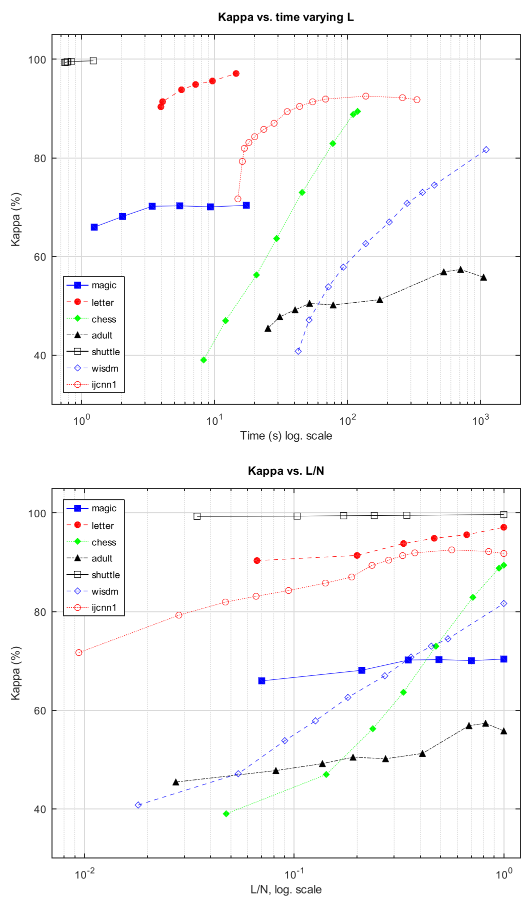


Fig. 2. Upper panel: kappa versus time (log. scale) of several large datasets varying L from 1000 to N from left to right. Lower panel: kappa versus L/N (log. scale) for the same datasets.

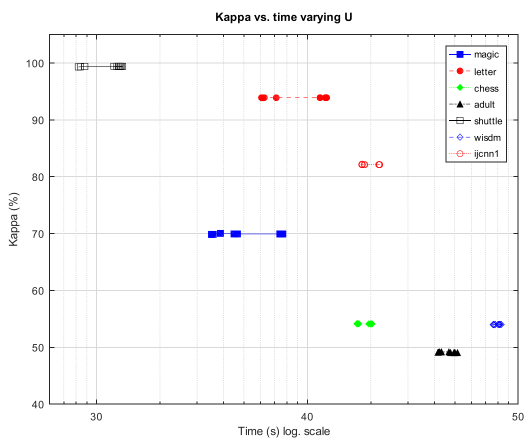


Fig. 3. Kappa versus time in several large datasets varying U from 500 to 5000 with step 500 from left to right.

Algorithm 3 DGT, Large Datasets.

```
1  Algorithm: $\mathcal{S}$=DGL($\{\mathbf{x}_n, c_n\}_{n=1}^N$, $\lambda$)
   Data: $\{\mathbf{x}_n, c_n\}_{n=1}^N$: training patterns and class labels
         $c_n \in \{1, \dots, C\}$; $\lambda$: regularization parameter.
   Result: $\mathcal{S}$: trained SVM.
2  $L_0 \leftarrow$20,000; $\alpha \leftarrow 0.5$; $L \leftarrow \min(L_0, \lfloor \alpha N \rfloor)$
3  $U \leftarrow$1,000; $r \leftarrow \lfloor L/U \rfloor$
4  # $\mathcal{R}(\mathcal{A}, L) \leftarrow$ random sampling of $L$ items from $\mathcal{A}$
5  $\{\mathbf{z}_l, q_l\}_{l=1}^L \leftarrow \mathcal{R}(\{\mathbf{x}_n, c_n\}_{n=1}^N, L)$
6  $\gamma \leftarrow$gamma($\{\mathbf{z}_l, q_l\}_{l=1,r}^L$) #procedure gamma, alg. 2
7  $\mathcal{S} \leftarrow$SVMTrain($\{\mathbf{z}_l, q_l\}_{l=1}^L$,$\lambda$,$\gamma$)
```

The lower panel of Fig. 2 shows plots of kappa versus L/N. In datasets where kappa raises slowly and in ijcnn1, L/N about 0.5 or even lower already provides a good performance. In the second group of datasets, larger L/N values are required to achieve performance near the highest available (for $L/N \to 1$). Therefore, to achieve a tradeoff between performance and speed, we propose to use $\alpha = 0.5$, although in some datasets (e.g., chess and wisdm) the performance may be suboptimal. To set the upper bound L_0 for L, the capability of SVM to train with large datasets must be considered. Datasets with hundreds or thousands of inputs can be discarded for RBF SVM, which does not outperform and is slower than linear SVM. With less than 200 inputs, the SVM is able to train with 10 000–20 000 training patterns, so we propose to set $L_0 = 20\,000$ patterns. Finally, L is calculated as

$$L = \min(L_0, \lfloor \alpha N \rfloor), \quad \alpha = 0.5, L_0 = 20,000. \tag{14}$$

Equation (10) requires the distance matrix $\mathbf{D}$, which is of size L. The computation of $\mathbf{D}$, of complexity $\mathcal{O}(L^2)$, might be too slow, so that a smaller size $U < L$ was used instead. To evaluate the influence of U on performance and speed, Fig. 3 shows kappa versus time for the datasets in Fig. 2 varying U from 500 to 5000 with step 500 from left to right points in each line. The time raises with U, but kappa remains fairly insensitive, so γ can be accurately estimated using a smaller distance matrix. Specifically, we propose to use $U = 1000$ patterns. Algorithm 3 describes the whole DGT procedure for large datasets (DGL), which: selects a random sample $\mathcal{R}$ of size L of the whole training set, keeping the relative class populations; calls procedure gamma (algorithm 2) passing only U of these L patterns in line 6, where l runs from 1 to L with step $r = \lfloor L/U \rfloor$; and trains SVM on the L training patterns.

III. Results and Discussion

The proposed methods DGS and DGL were used to calculate γ and train the SVM. The experiments were performed in a desktop computer with 8 Intel© Core[1]i7-9700K processors at 3.60 GHz, equipped with 64-GB RAM under operative system Linux Kubuntu 20.04. The methodology was four fold cross-validation (CV), using two folds to train the SVM, one for validation (in hyperparameter tuning) and the remaining fold for test. The classification performance was measured using the kappa statistic. The SVM implementation was LibSVM [24], accessed from its Octave binding.[2] The datasets were selected from the UCI [25] and Kaggle[3] repositories.

A. Small Datasets

A first group of experiments were run over a collection of 30 datasets of small size (Table I), with less

[1]Trademarked.
[2]http://www.octave.org (May, 2022).
[3]https://www.kaggle.com

TABLE I
LIST OF SMALL DATASETS (Q <15 000) WITH THEIR NUMBERS OF TOTAL (Q) AND TRAINING (N) PATTERNS, INPUTS (I), AND CLASSES (C)

Original name	Brief name	Q	N	I	C
Promoter gene	promoter	106	56	57	2
Breast tissue	tissue	106	58	9	6
Hepatitis	hepatitis	155	78	19	2
Wine quality	wine	178	90	13	3
Sonar, mines rocks	sonar	208	106	60	2
Seeds	seeds	210	108	7	3
Heart	heart	270	136	28	2
Ionosphere	ion	351	178	33	2
Dermatology	derm	366	186	130	6
Monks-2	monks	432	236	17	2
Congress. voting	voting	435	218	16	2
Breast cancer Wisc.	wdbc	569	286	30	2
Synthetic control	synthetic	600	300	60	6
Australian credit	australian	690	346	43	2
Pima diabetes	pima	768	384	8	2
Energy efficiency	energy	768	386	8	3
Audit	audit	776	390	25	2
Vehicle silhouettes	vehicle	846	426	18	4
Annealing	annealing	898	450	54	5
Tic-tac-toe	tic	958	480	27	2
Mammographic	mgraph	961	482	5	2
German credit	german	1,000	500	65	2
Isolet	isolet	1,559	780	617	26
Image segment.	imseg	2,310	1,162	18	7
Abalone	abalone	4,177	2,090	8	3
Landsat satellite	sat	6,435	3,222	36	6
Musk	musk	6,598	3,302	166	2
Handwritten USPS	usps	9,298	4,650	256	2
Electrical Grid	grid	10,000	5,000	13	2
Nursery	nursery	12,958	6,480	27	4

TABLE II
VALUES OF γ AND $D(\gamma)$, ABSOLUTE DIFFERENCES, AND KAPPA VALUES ON THE VALIDATION SETS ACHIEVED BY DGS AND DMS ON THE SMALL DATASETS

	γ			$D(\gamma)$			Kappa (%)	
Dataset	DGS	DMS	|Diff|	DGS	DMS	|Diff|	DGS	DMS
promoter	0.005	0.004	0.001	0.24	0.25	0.004	57.1	57.1
tissue	0.101	0.500	0.399	0.21	0.14	0.068	57.7	62.2
hepatitis	0.038	0.031	0.007	0.23	0.23	0.002	28.3	28.3
wine	0.041	0.062	0.021	0.15	0.15	0.001	100.0	100.0
sonar	0.006	0.008	0.002	0.27	0.27	0.006	62.0	62.0
seeds	0.078	0.125	0.047	0.14	0.12	0.021	97.2	97.2
heart	0.039	0.031	0.007	0.23	0.24	0.007	78.9	78.9
imseg	0.056	0.125	0.069	0.13	0.09	0.043	94.0	95.2
ion	0.034	0.031	0.003	0.24	0.24	0.002	87.4	95.1
derm	0.052	0.062	0.010	0.13	0.13	0.002	97.3	97.3
monks	0.093	0.125	0.032	0.25	0.26	0.009	100.0	100.0
voting	0.083	0.125	0.042	0.18	0.19	0.007	94.2	94.2
wdbc	0.011	0.016	0.005	0.26	0.26	0.006	94.0	94.0
synthetic	0.082	0.125	0.043	0.11	0.09	0.018	98.4	98.4
australian	0.033	0.031	0.002	0.25	0.26	0.003	64.4	65.5
pima	0.038	0.031	0.007	0.25	0.26	0.005	43.6	43.6
energy	0.061	0.062	0.001	0.17	0.17	0.001	88.2	90.8
audit	0.000	0.000	0.000	0.40	0.36	0.034	83.9	83.9
vehicle	0.039	0.125	0.086	0.24	0.21	0.030	74.3	77.5
annealing	0.028	0.031	0.004	0.25	0.25	0.002	93.4	93.4
tic	0.052	0.062	0.011	0.25	0.26	0.004	98.2	98.2
mgraph	0.065	0.125	0.060	0.26	0.25	0.004	65.0	66.6
german	0.017	0.016	0.002	0.25	0.25	0.001	44.2	44.2
isolet	0.003	0.004	0.001	0.03	0.03	0.001	93.6	93.6
abalone	0.067	0.250	0.183	0.30	0.27	0.027	31.3	31.4
sat	0.023	0.062	0.039	0.15	0.10	0.044	87.2	89.6
musk	0.005	0.004	0.001	0.21	0.21	0.002	97.9	97.9
grid	0.024	0.031	0.007	0.23	0.24	0.006	98.1	98.1
usps	0.000	0.000	0.000	0.13	0.13	0.000	96.4	97.0
nursery	0.109	0.125	0.016	0.21	0.21	0.001	100.0	100.0
Average	—	—	0.037	—	—	0.012	80.2	81.0

than 15 000 patterns. The γ of SVM was estimated: 1) using DGS; 2) using GS, with values in the set $\Gamma = \{2^i\}_{i=-20}^{20}$; and 3) selecting the value in Γ that minimizes $D(\gamma)$ in (4), named D-minimization for small datasets (DMS), equivalent to IKT [22]. The regularization λ was always tuned in the set $\{2^i\}_{i=-7}^{15}$ to maximize kappa on the validation set.

The hypothesis in (6) is evaluated in Table II by comparing γ, $D(\gamma)$, and kappa on the validation set achieved by DGS and DMS. For most datasets, DGS achieves γ very near to the right value selected by DMS (column 3), with an average difference (last row, column 4) of 0.037. The difference in $D(\gamma)$ between DGS and DMS (column 7) is also very low, 0.012 on average. The kappa values of DGS and DMS in columns 8 and 9 are also very similar in all the datasets, with an average difference of 0.8%, so the difference in γ between DGS and DMS does not reduce very much the SVM validation performance.

Table III reports the kappa and times (in s excluding λ tuning) spent by DGS, GS, and DMS on test sets. The kappa of DGS is very similar to GS and DMS in all the cases, achieving the best result in seven of them. In the last line, the average kappa of DGS (79.7%) is only 0.6 and 1 point below GS and DMS (80.3% and 80.7%, respectively). Columns 5 and 6 report the average time (T_1, in s) per fold spent by DGS or DMS to calculate γ. The values of DMS are one to three orders of magnitude higher than DGS. Column 6 of the last line reports the average of $T_1(\text{DMS})/T_1(\text{DGS})$, being the latter one order of magnitude (16.3 times) faster than the former. This is expectable because DGS calculates γ using (10). Although this requires to calculate the distances between the N training patterns, they must also be calculated for DMS, which additionally requires to repeat the calculation of $D(\gamma)$ for the 41 values of γ in the set Γ. This ratio raises very fast with the dataset size, with values above 20 in the 11 last datasets. The time T_2 (columns 7 and 8 in Table III) is the average total execution time per fold for DGS and GS. The last line of column 8 reports that DGS is one order of magnitude (10.9 times on average) faster than GS, reaching ratios $T_2(\text{GS})/T_2(\text{DGS})$ about 20–40 in the last datasets.

B. Large Datasets

Experiments were also conducted on a collection of 20 large datasets (more than 15 000 training patterns, Table IV) up to 31 million of patterns and 122 inputs. Our method DGL was compared to the ones below.

TABLE III

KAPPA (IN %) AND TIMES T_1, T_2 (IN S, SEE TEXT FOR DETAILS) OF SVM USING DGS, GS, AND DMS ON THE TEST SETS FOR SMALL DATASETS. BEST KAPPA VALUES ARE IN BOLD

	Kappa (%)			T_1		T_2	
Dataset	DGS	GS	DMS	DGS	DMS	DGS	GS
promoter	59.1	50.2	**64.1**	0.0101	0.006	0.027	0.03
tissue	50.4	55.7	**60.6**	0.0004	0.003	0.004	0.03
hepatitis	31.5	**35.4**	34.1	0.0004	0.006	0.005	0.04
wine	97.5	**98.3**	97.5	0.0005	0.007	0.005	0.04
sonar	66.0	72.2	**73.9**	0.0030	0.012	0.020	0.08
seeds	87.8	**91.2**	89.2	0.0007	0.010	0.005	0.04
heart	69.9	67.7	**71.3**	0.0031	0.019	0.012	0.09
imseg	96.1	**96.2**	96.2	0.0886	2.054	0.199	2.68
ion	83.8	**87.6**	83.1	0.0043	0.031	0.020	0.12
derm	**96.2**	95.5	95.9	0.0040	0.030	0.055	0.47
monks	**100.0**	100.0	100.0	0.0042	0.044	0.037	0.36
voting	**89.4**	88.0	88.0	0.0043	0.071	0.015	0.14
wdbc	87.1	**88.7**	85.6	0.0060	0.073	0.031	0.22
synthetic	98.2	96.4	**99.2**	0.0066	0.093	0.059	0.52
australian	71.1	68.4	**71.3**	0.0078	0.118	0.045	0.45
pima	44.6	**46.0**	45.5	0.0085	0.164	0.051	0.38
energy	91.1	**92.7**	91.1	0.0135	0.154	0.039	0.36
audit	84.6	87.1	**89.7**	0.0065	0.129	0.033	0.42
vehicle	**78.3**	76.2	74.1	0.0155	0.194	0.057	0.55
annealing	96.7	97.5	**98.3**	0.0117	0.293	0.064	1.01
tic	96.3	**97.9**	95.6	0.0128	0.271	0.071	0.99
mgraph	63.3	65.4	**67.3**	0.0131	0.292	0.156	1.20
german	41.9	**42.2**	41.8	0.0147	0.346	0.073	0.72
isolet	94.5	**94.7**	92.5	0.0436	0.888	3.544	33.04
abalone	**33.5**	32.5	32.9	0.4177	8.735	13.476	16.39
sat	89.1	**90.2**	89.7	0.8454	21.025	1.642	29.54
musk	**99.1**	99.1	99.0	0.9003	22.004	3.465	72.11
grid	98.0	**99.4**	97.6	2.0299	49.801	2.597	26.78
usps	96.2	**96.9**	96.2	1.8367	43.289	5.752	255.58
nursery	**100.0**	100.0	100.0	3.2652	80.506	8.215	160.11
Average	79.7	80.3	80.7	—	16.3	—	10.9

1) Minimization of $D(\gamma)$ for large datasets, named DML, with $\gamma \in \Gamma$, equivalent to IKT with random sampling of L training patterns, see (14) with $\alpha = 0.5$ and $L_0 = 20\,000$, and a reduced distance matrix of size $U = 1000$.
2) Linear kernel SVM (LSVM), implemented by the Liblinear library [26], because RBF kernel SVM cannot be executed on such large datasets. The regularization parameter λ was not tuned, given the dataset size, but set to 100, a value widely used in the literature.

All the experiments used the same seed for the random number generator to guarantee the reproducibility of the results. To evaluate whether the initialization of the random number generator during the training set sampling influences performance of DGL, Table V reports the mean and standard deviations of the kappa achieved by DGL on the first seven large datasets using 20 different random seeds. The deviations are very low compared with the mean kappa values. This means that performance is very similar in all the cases, and that influence of randomness on performance is not significant.

Table VI reports kappa and times T_1 and T_2 of DGL, DML, and LSVM, and time T_3 required only by the disk reading, on the large datasets. In 12 of 20 datasets, DGL achieves the best kappa, being globally higher than DML and much higher than LSVM, which runs out-of-memory in three cases.

TABLE IV

LIST OF LARGE DATASETS (Q >15 000 PATTERNS) SORTED BY $Q \cdot I$

Original name	Brief name	Q	N	I	C
Magic gamma	magic	19,020	9,510	10	2
Letter recogn.	letter	20,000	10,018	16	26
Shuttle	shuttle	58,000	43,483	9	5
Chess(rook-pawn)	chess	28,056	14,044	40	18
IJCNN 2001	ijcnn1	141,691	70,848	22	2
Adult(census)	adult	48,842	24,422	105	2
Wisdm	wisdm	73,803	55,380	92	18
Connect-4	conn-4	67,557	33,782	126	3
Poker hand	poker	1,025,010	512,506	10	2
Covertype	covtype	581,012	435,768	54	7
Record linkage	record	5,749,132	4,311,846	11	2
SUSY	susy	5,000,000	3,750,000	18	2
KDD Cup 1999	kddcup	4,000,000	3,638,724	122	23
Wesad	wesad	31,470,603	23,602,947	8	4
Hepmass	hepmass	10,500,000	7,000,000	28	2
Higgs	higgs	11,000,000	8,249,997	28	2
Heter. activ.	har	29,097,887	21,823,410	14	6
Human activ.	human	13,956,557	10,467,372	36	42
Detection IoT	baiot	7,062,606	5,296,869	115	11
Kitsune netw.	kitsune	21,017,597	8,646,375	115	2

TABLE V

MEAN AND STANDARD DEVIATION OF KAPPA (IN %) ACHIEVED BY DGL OVER 20 DIFFERENT RANDOM SEEDS ON SOME LARGE DATASETS

Kappa (%)	magic	letter	chess	adult	shuttle	wisdm	ijcnn1
Mean	70.1	95.4	74.1	52.5	99.7	70.7	87.0
Std. Dev.	0.21	0.13	0.24	0.20	0.05	0.11	0.18

TABLE VI

KAPPA (IN %), TIMES T_1 AND T_2 (IN S) OF DGL, LSVM, AND DML, AND DATASET READ TIME T_3 IN THE LARGE DATASETS

	Kappa (%)			T_1		T_2		T_3
Dataset	DGL	LSVM	DML	DGL	DML	DGL	LSVM	
magic	**70.0**	52.5	69.8	0.022	0.9	1.4	1.0	0.2
letter	**95.4**	52.9	95.3	0.045	0.8	2.0	4.3	0.3
shuttle	**99.7**	46.8	99.6	0.030	1.4	1.1	3.4	0.6
chess	74.0	26.9	**74.0**	0.042	0.7	12.4	3.0	0.6
ijcnn1	**87.0**	22.0	86.9	0.030	2.9	7.7	15.8	3.4
adult	52.0	**55.6**	50.9	0.028	1.4	74.9	26.4	5.8
wisdm	70.8	25.0	**74.5**	0.038	0.7	78.7	227.8	7.2
conn-4	**60.8**	44.4	60.7	0.025	0.6	235.2	18.6	4.1
poker	**22.1**	0.1	21.9	0.025	9.6	128.6	19.9	11.1
covtype	62.2	52.7	**63.0**	0.026	4.3	104.5	130.9	19.1
record	**95.9**	1.8	95.6	0.027	98.1	82.2	457.3	70.9
susy	**58.6**	54.3	58.0	0.039	48.1	403.7	439.7	96.9
kddcup	92.6	**92.7**	92.6	0.031	17.3	215.4	1455.7	207.3
wesad	**20.8**	-2.2	17.2	0.030	212.7	474.7	1333.4	243.6
hepmass	66.8	**67.2**	65.5	0.050	91.1	1078.7	435.7	314.3
higgs	**34.1**	—	33.1	0.030	89.6	1930.2	—	328.4
har	9.8	—	**13.0**	0.048	82.4	2304.6	—	448.9
human	16.7	**23.6**	16.7	0.044	0.8	5113.3	8195.0	662.5
baiot	**85.8**	-8.2	26.6	0.061	9.8	1774.0	39722.0	726.2
kitsune	**77.4**	—	76.7	0.024	286.7	2793.0	—	2167.5
Average	62.6	35.8	59.6	—	907.2	—	3.2	
p-value	—	0.0032	0.7					

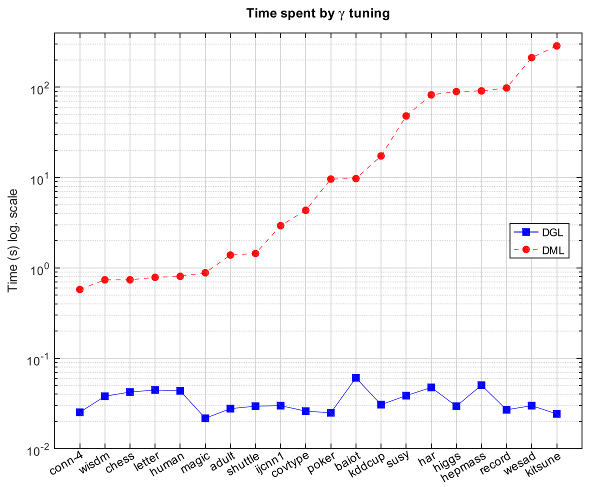


Fig. 4. Time spent by DGL and DML to find γ for large datasets.

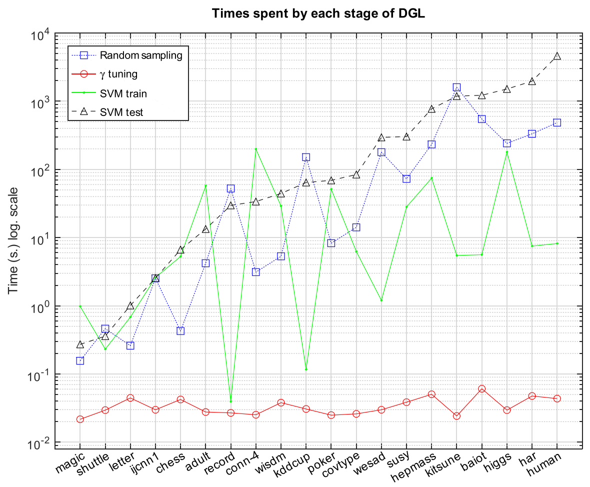


Fig. 5. Time spent by DGL in each stage.

On average, the kappa of DGL (62.6%) is 27 and 3 points higher than LSVM and DML, respectively. Thus, direct calculation of γ (10) combined to random sampling seems to be effective performing the RBF tuning for the SVM on large datasets. According to a Wilcoxon rank-sum test, the difference between DGL and LSVM is statistically significant ($p = 0.0032$), while the difference between DGL and DML is not statistically significant ($p = 0.7$).

Considering the time T_1 spent to calculate γ (columns 5 and 6), DGL is much faster (907 times, almost three orders) than DML. The difference between their times raises with the dataset size, from one to even three to four orders, so direct calculation of γ in DGL really allows a large time saving compared with DML. Considering the total execution time T_2 (columns 7 and 8), DGL is faster than LSVM in 10 of 17 datasets where LSVM does not fail, and on average DGL is 3.3 times faster than LSVM, despite that LSVM uses linear kernel and is designed for large datasets. Note also that in the largest datasets, the ratio $T_2(\text{LSVM})/T_2(\text{DGL})$ is much higher than 3.3, e.g., 22.4 in `baiot`. The time T_3 required by file reading is near T_2 for DGL in some datasets, e.g., in `record`, `kddcup`, `baiot`, and `kitsune`, so the execution of DGL is very fast and most of the time is spent just in data reading.

Fig. 4 shows plots of the time spent by DGL and DML to calculate and search, respectively, γ on the large datasets. DML is much slower than DGL because it requires to calculate $D(\gamma)$ for the 41 values in the collection $\{2^i\}_{-20}^{20}$. Besides, the difference between their times raises from left to right starting in one order of magnitude up to three to four orders for the largest datasets, so DGL allows a large time saving compared with DML.

Fig. 5 shows plots of the times spent in the large datasets by the four stages of DGL: random sampling of training patterns, γ tuning using (10), training, and test of the SVM. The sampling, train, and test times raise with the dataset size from left to right, although the train times oscillating strongly due to SVM training may be faster or slower depending on the complexity of the training set. However, the tuning time is very stable and remains constant with the dataset size, being much slower than the times of the other stages.

C. Comparison With Existing Approaches

Compared with the methods evaluated in [22], the average performance of DGS or DGL over the datasets also used in the current work (78.3%) is similar to IKT (78.6%), KDE (78.1%), genetic (79.7%), PSO (78.4%), and Bayesian (80%). IKT spends 384 s in its largest training set (`ijcnn1`, Table III in [22]), while DGL spends only 7.7 s (T_2 in Table VI). DGL is faster than IKT due to: 1) the closed-form expression in (10), faster than iterative minimization and not limited to a predefined collection of γ values; and 2) use of random sampling instead of class prototypes in IKT. Thus, DGL comes up to training sets of 8 million of patterns (`kitsune`), being a step ahead to extend the standard SVM for large-scale datasets. The FSVC [23] achieves kappa of 27.9% and 59.8% spending 6015 and 599 s in datasets `kitsune` and `hepmass`, respectively, while DGL achieves 77.4% and 66.8% spending 2793 and 1078 s, so DGL outperforms FSVC and is faster in the first case. The indefinite core vector machine [27] achieves kappa[4] of 28.5%, 59.2%, and 73% with times 115.26, 170.36, and 88.0 s in datasets `magic`, `nursery`, and `grid`, respectively, while DGL achieves 70%, 100%, and 98% with times 1.4, 8.21, and 2.59 s in the same datasets, so DGL outperforms ICVM and is much faster.

Table VII shows comparison of the test error and execution time T_2 (or T_1 when asterisk is present), of DGS (or DGL with †), and several related approaches referenced in Section I on common datasets. Note that the experimental methodology can be different to the current one, which may lead to differences in performance. In general, DGS achieves a test error similar to the competing approaches, outperforming them in 13 of 26 datasets. Besides, DGS is much faster, and the differences reach often several orders of magnitude. In dataset `ion`, the Bat algorithm [10] spends 645.3 s while DGS spends 0.02 s, a large difference even considering the difference in computing power since 2017. In `adult`, dynamic PSO [7] achieved 15.55% error using tenfold CV spending 5287 s, while DGL achieves 20.44% using fourfold CV in 74.9 s. The DGS also compares well to

[4]https://www.techfak.uni-bielefeld.de/~fschleif/software.xhtml

TABLE VII
TEST ERROR (IN %) AND TIME T_2 (* MEANS T_1) OF DGS († MEANS DGL) AND OTHER APPROACHES IN THE LITERATURE

		Approach		DGS	
Approach	Dataset	Error(%)	T_2	Error(%)	T_2
Tharwat [10]	ion	2.1	645.3	7.1	0.02
Kapp [7]	adult	15.55	5,287	20.44†	74.9†
	sat	8.06	6,355	8.8	1.64
Varewyck [16]	sat	8.19	77.47	8.8	1.64
Liu [18]	ion	4.77	0.3*	7.1	0.004*
Liu [19]	ijcnn1	1.83	43.46*	2.48†	2.9*†
Xu [15]	imseg	5.55	4.9	3.3	0.199
	adult	14.95	639.24	20.44†	74.9
	ijcnn1	1.21	501.52	2.48†	7.7†
Lázaro [17]	imseg	2.16	0.68*	3.3	0.089*
	pima	24.42	0.09*	23.7	0.009*
Zhang [8]	imseg	2.88	155.35	3.3	0.199
Ding [20]	australian	31.9	11.83	14.5	0.045
(RRBF)	heart	22.25	1.33	14.8	0.012
	ion	7.91	2.86	7.1	0.02
Chang [13]	heart	23.3	0.1	14.8	0.012
(DML+M+JC)	ion	6.3	0.4	7.1	0.02
	sonar	15.3	0.3	16.8	0.02
	vehicle	23.4	1.4	16.3	0.057
	voting	38.3	0.3	5.1	0.015
	wdbc	4.0	0.7	6.0	0.031
	wine	2.5	0.1	1.7	0.005
Wang [14]	derm	3.53	44.56	3.0	0.055
(Criterion II,	vehicle	24.11	95.71	16.3	0.057
Elips. GRBF,	sat	9.85	962.87	8.8	1.64
dag)	usps	4.93	–	0.6	5.752

nearest enemy [17], calculating γ about ten times faster with similar error. Differential evolution [8] is almost three orders slower than DGS (155.35 versus 0.199 s), and the random RBF SVM [20] gets more error than DGS being two orders slower. Overall, DGS and DGL are between one and three orders faster than the existing approaches with performances similar to or higher than the state-of-the-art, confirming the value of DGT.

IV. CONCLUSION

We propose a method to calculate the inverse γ of the double squared RBF kernel spread σ for the SVM directly from training data in classification problems. The DGT for small (DGS) or large (DGL) datasets uses a closed-form expression that only requires the average distance between training patterns. This expression is an approximated analytical solution for the minimization of the mean squared difference $D(\gamma)$ between the RBF and ideal kernel matrices. It avoids to repeat the calculation of $D(\gamma)$ for different γ values, as in our previous approach IKT [22], being two orders of magnitude faster with similar performance. In a collection of 30 small size datasets, the SVM with γ tuned by DGS is very efficient, one order of magnitude faster than GS and D-minimization (DMS), equivalent to IKT, and achieves kappa (79.5% on average) virtually equal to DMS and GS (80.4% and 80.1%, respectively). In large datasets, both the calculation of the distances between patterns required by the γ estimation and SVM training become slow, so DGL performs a random sampling of the training set both for distance calculation and SVM training. The experiments prove that a small distance matrix, with a low number of randomly sampled training patterns, is enough for an accurate estimation of γ. The performance of DGL (62.6%) on 20 large-scale datasets up to 31 million of patterns clearly outperforms the linear kernel SVM (35.8%), which fails in the largest datasets, with statistical significance. Besides, DGL also outperforms D-minimization (DML), equivalent to IKT with random sampling, which achieves kappa of 59.6%. On average, DGL is 907 times (two to three orders) faster than DML and even three times faster than LSVM, which is specially suited for large datasets. These results prove that the proposed analytical formula for γ always achieves the optimal value and, combined with random sampling of the training set, allows to extend the SVM with RBF kernel to arbitrary large classification problems. Future work includes to generalize the proposed method to other nonlinear kernels, such as polynomial or sigmoid, with one or even two tunable parameters.

REFERENCES

[1] M. 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, pp. 3133–3181, Jan. 2014.

[2] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

[3] M. Choi and J. J. Jeong, “Comparison of selection criteria for model selection of support vector machine on physiological data with inter-subject variance,” *Appl. Sci.*, vol. 12, no. 3, p. 1749, Feb. 2022.

[4] D. Anguita, A. Ghio, L. Oneto, and S. Ridella, “In-sample and out-of-sample model selection and error estimation for support vector machines,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 9, pp. 1390–1406, Sep. 2012.

[5] F. Friedrichs and C. Igel, “Evolutionary tuning of multiple SVM parameters,” *Neurocomputing*, vol. 64, pp. 107–117, Mar. 2005.

[6] M. Abbaszadeh, S. Soltani-Mohammadi, and A. N. Ahmed, “Optimization of support vector machine parameters in modeling of Iju deposit mineralization and alteration zones using particle swarm optimization algorithm and grid search method,” *Comput. Geosci.*, vol. 165, Aug. 2022, Art. no. 105140.

[7] M. N. Kapp, R. Sabourin, and P. Maupin, “A dynamic model selection strategy for support vector machine classifiers,” *Appl. Soft Comput.*, vol. 12, no. 8, pp. 2550–2565, Aug. 2012.

[8] J. Zhang, A. Niu, K. Li, and G. Irwing, “Model selection in SVMs using differential evolution,” in *Proc. World Congr. Int. Fed. Automat. Control*, 2011, pp. 14717–14722.

[9] C. E. D. S. Santos, R. C. Sampaio, L. D. S. Coelho, G. A. Bestard, and C. H. Llanos, “Multi-objective adaptive differential evolution for SVM/SVR hyperparameters selection,” *Pattern Recognit.*, vol. 110, Feb. 2021, Art. no. 107649.

[10] A. Tharwat, A. E. Hassanien, and B. E. Elnaghi, “A BA-based algorithm for parameter optimization of support vector machine,” *Pattern Recognit. Lett.*, vol. 93, pp. 13–22, Jul. 2017.

[11] A. Tharwat and T. Gabel, “Parameters optimization of support vector machines for imbalanced data using social ski driver algorithm,” *Neural Comput. Appl.*, vol. 32, no. 11, pp. 6925–6938, Jun. 2020.

[12] T. Glasmachers and C. Igel, “Maximum likelihood model selection for 1-norm soft margin SVMs with multiple parameters,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 8, pp. 1522–1528, Aug. 2010.

[13] C.-C. Chang and S.-H. Chou, “Tuning of the hyperparameters for $L2$-loss SVMs with the RBF kernel by the maximum-margin principle and the jackknife technique,” *Pattern Recognit.*, vol. 48, no. 12, pp. 3983–3992, Dec. 2015.

[14] L. Wang, P. Xue, and K. L. Chan, “Two criteria for model selection in multiclass support vector machines,” *IEEE Trans. Syst., Man, Cybern., B, Cybern.*, vol. 38, no. 6, pp. 1432–1448, Dec. 2008.

[15] Z. Xu, M. Dai, and D. Meng, “Fast and efficient strategies for model selection of Gaussian support vector machine,” *IEEE Trans. Syst., Man, Cybern., B, Cybern.*, vol. 39, no. 5, pp. 1292–1307, Oct. 2009.

[16] M. Varewyck and J.-P. Martens, “A practical approach to model selection for support vector machines with a Gaussian kernel,” *IEEE Trans. Syst., Man, Cybern., B, Cybern.*, vol. 41, no. 2, pp. 330–340, Apr. 2011.

[17] M. Lázaro-Gredilla, V. Gómez-Verdejo, and E. Parrado-Hernández, “Low-cost model selection for SVMs using local features,” *Eng. Appl. Artif. Intell.*, vol. 25, no. 6, pp. 1203–1211, Sep. 2012.

[18] Z. Liu and H. Xu, “Kernel parameter selection for support vector machine classification,” *J. Algorithms Comput. Technol.*, vol. 8, no. 2, pp. 163–177, Jun. 2014.

[19] Z. Liu, M. Zuo, X. Zhao, and H. Xu, “An analytical approach to fast parameter selection of Gaussian RBF kernel for support vector machine,” *J. Inf. Sci. Eng.*, vol. 31, no. 2, pp. 691–710, 2015.

[20] X. Ding, J. Liu, F. Yang, and J. Cao, “Random radial basis function kernel-based support vector machine,” *J. Franklin Inst.*, vol. 358, no. 18, pp. 10121–10140, Dec. 2021.

[21] M. V. F. Menezes, L. C. B. Torres, and A. P. Braga, “Width optimization of RBF kernels for binary classification of support vector machines: A density estimation-based approach,” *Pattern Recognit. Lett.*, vol. 128, pp. 1–7, Dec. 2019.

[22] Z. Akram-Ali-Hammouri, M. Fernández-Delgado, A. Albtoush, E. Cernadas, and S. Barro, “Ideal kernel tuning: Fast and scalable selection of the radial basis kernel spread for support vector classification,” *Neurocomputing*, vol. 489, pp. 1–8, Jun. 2022.

[23] Z. Akram-Ali-Hammouri, M. Fernández-Delgado, E. Cernadas, and S. Barro, “Fast support vector classification for large-scale problems,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6184–6195, Oct. 2022.

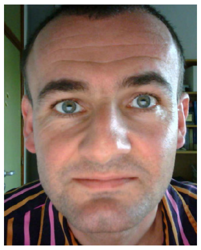
[24] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, Apr. 2011.

[25] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml

[26] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “LIBLINEAR: A library for large linear classification,” *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.

[27] F.-M. Schleif and P. Tino, “Indefinite core vector machine,” *Pattern Recognit.*, vol. 71, pp. 187–195, Nov. 2017.

Diego Isla-Cernadas was born in A Coruña, Spain, in 1981. He received the B.S. degree from the Polithecnic University of Valencia, Valencia, Spain, in 2011, and the M.S. degree in automatic and electronic engineering from the Polithecnic University of Madrid, Madrid, Spain, in 2018. He is currently pursuing the Ph.D. degree in machine learning, neural networks, and classification with the Centro Singular de Investigación en Tecnoloxías Intelixentes da USC (CiTIUS), Santiago de Compostela, Spain.



Manuel Fernández-Delgado received the B.S. degree in physics and the Ph.D. degree in computer science from the University of Santiago de Compostela (USC), Santiago, Spain, in 1994 and 1999, respectively.

He is a Lecturer of computer science and researcher at Centro Singular de Investigación en Tecnoloxías Intelixentes da USC (CiTIUS), Santiago de Compostela, Spain, with a focus on neural computation, machine learning, and pattern recognition of the CiTIUS.

Eva Cernadas was born in A Coruña, Spain, in 1969. She received the B.S. and Ph.D. degrees in physics from the University of Santiago de Compostela (USC), Santiago, Spain, in 1992 and 1997, respectively.

She is a Lecturer of computer science and researcher at Centro Singular de Investigación en Tecnoloxías Intelixentes da USC (CiTIUS), Santiago de Compostela, Spain, with a focus on computer vision and machine learning, specially in food technology, medical, and biological domains.

Manisha S. Sirsat received the Ph.D. degree in computer science from the University of Santiago de Compostela (USC), Santiago, Spain, in 2017.

She is a Senior Researcher of machine learning at the InnovPlantProtect, Elvás, Portugal. Her main research area includes implementing artificial intelligence for solving agriculture problems, and thus she has applied classification and regression methods to a wide variety of applications in agriculture.

Haitham Maarouf received the Ph.D. degree (Hons.) in computer science from the University of Santiago de Compostela (USC), Santiago, Spain, in 2018.

He is a Post-Doctoral Researcher at Centro Singular de Investigación en Tecnoloxías Intelixentes da USC (CITIUS), Santiago de Compostela, Spain, and a Researcher and System Analyst at Plexus Tech, Santiago de Compostela. His research interests include artificial intelligence, data science, machine learning, knowledge representation, biomedical ontologies, and terminologies.

Senén Barro was born in As Pontes de García Rodríguez, Spain, in 1962. He received the B.S. and Ph.D. degrees in physics from the University of Santiago de Compostela (USC), Santiago, Spain, in 1985 and 1988, respectively.

He is a Full Professor of Computer Science at USC in 1995. From 2002 to 2010, he was the Rector at USC. He is currently the Scientific Director of the Centro Singular de Investigación en Tecnoloxías Intelixentes da USC (CiTIUS), Santiago de Compostela, Spain. He has authored seven books and more than 300 scientific papers in these fields.