ELLIPSOIDAL CLASSIFICATION VIA SEMIDEFINITE PROGRAMMING

ANNABELLA ASTORINO*, ANTONIO FRANGIONI†, ENRICO GORGONE‡, AND BENEDETTO MANCA§

Abstract. Separating two finite sets of points in a Euclidean space is a fundamental problem in classification. Customarily linear separation is used, but nonlinear separators such as spheres [5] have been shown to have better performances in some tasks, such as edge detection in images. We exploit the relationships between the more general version of the spherical separation, where we use general ellipsoids, and the SVM model with quadratic kernel to propose a new classification approach. The implementation basically boils down to adding a SDP constraint to the standard SVM model in order to ensure that the chosen hyperplane in the feature space represents a non-degenerate ellipsoid in the input space; albeit being somewhat more costly than the original formulation, this still allows to exploit many of the techniques developed for SVR in combination with SDP approaches. We test our approach on several classification tasks, among which the edge detection problem for gray-scale images, proving that the approach is competitive with both the spherical classification one and the quadratic-kernel SVM one without the ellipsoidal restriction.

Key words. Classification, Semidefinite Programming, Artificial intelligence

1. Introduction. The problems of separation of sets, that has traditionally been a field of mathematics, has recently garnered the interest of researchers from different areas, such as applied mathematics, optimization, statistics and computer science. This derives from the need of efficiently constructing effective separation surfaces (i.e., classifiers) to be used in practical applications of machine learning, data mining and knowledge management, such as text and web classification [1], object recognition in machine vision [12], edge detection [5], gene expression profile analysis [19], DNA and protein analysis [11], and many others.

In this work we focus on supervised binary classification [7], one of the most important tasks in machine learning and data mining. We are given a finite set of samples, each one completely captured by a n-dimensional real vector of inputs and provided with a two-valued label. The aim of the problem is to devise a methodology capable of assigning the right value of the label to any unseen sample. One of the most natural approaches to the task is that of devising a *separating surface* that partitions the space in two, with each part (hopefully) only containing samples with the same label. Since this may not be possible once the general shape of the surface is chosen, finding the separating surface is usually poised as a mathematical programming problem trying to minimize on one hand the misclassification of known inputs, and on the other hand some measure of the "complexity" of the surface itself. Indeed, it is well-known that nontrivial trade-offs exist between the complexity of the surface, its capability of separating arbitrarily complex sets, the cost of finding it (training) and the predictive power against unseen samples. Roughly speaking, "too complex"

*Istituto di Calcolo e Reti ad Alte Prestazioni–C.N.R., Rende, Italia. E-mail: astorino@icar.cnr.it

†Dipartimento di Informatica, Universitá di Pisa, Pisa, Italia. E-mail: frangio@di.unipi.it

§Dipartimento di Matematica e Informatica, Universitá di Cagliari, Cagliari, Italia. E-mail: bmanca@unica.it

‡Email: gorgone.enrico@gmail.com

separating surfaces can lead both to hard training problems, and to *overfitting*, i.e., the phenomenon whereby the separating surface works well for the given input but has little predictive power for the unseen ones. Carefully balancing these two aspects is in fact one of the most delicate tasks in practical classification.

One of the fundamental decisions in the process clearly is the shape of the separating surface. The most natural shape, and not coincidentally the most widely used, is the simplest one, i.e., a(n affine) hyperplane. This is the basis of the widely used Support Vector Machine (SVM) methodology [20, 7, 17]. However, it is self-evident that most sets arising in practice are not affinely separable. This can still be dealt with by the SVM approach via the *kernel trick*: the input space is mapped into a (typically, larger) feature space and an affine separating surface is sought for therein. The projection of the separating hyperplane on the original space can be nonlinear and therefore better able to cope with the learning of the set of inputs at hand.

Many different kernels have been developed [8, 18, 10, 17] that may be appropriate for different tasks. Like for the general case, trade-offs may exist between the complexity of a kernel and its generalisation capabilities. In this work we focus on the what is perhaps the "simplest" nonlinear kernel, i.e., the quadratic one. In this case, the projection of the separating hyperplane in the original input space is a second-order surface.

Our approach is based on the observation that by properly restricting the space of possible parameters of the separation surface we can force it to represent an *ellipsoid* in the original input space. Pattern classification by means ellipsoids has been found [4] to be promising because *(i)* the ellipsoid is the simplest nonlinear convex set that can encloses samples in a bounded region of the space, *(ii)* it is independent from invertible linear transformations of the coordinate system and *(iii)* ellipsoids are characterised by positive semidefinite (PSD) matrices, and therefore optimization problems involving them can often be casted as SemiDefinite Programs (SDP), for which there are several available off-the-shelf efficient algorithms, chiefly (but not exclusively) interior point ones. Ellipsoidal separation seems to be particularly promising for binary classification problems where a class is much smaller (in terms of number of inputs in the training set) than the other one, since it may be easy to construct an ellippsoid enclosing (most of) the samples of the smaller class and keeping outside (most of) the samples of larger one. This is, for instance, the case of the edge detection problems where the class of relevant pixels (edges) is way smaller than that of non-edge pixels in an image. Although intuitive, this notion has been proven experimentally to be correct in [5] for a very special class of ellipsoids, i.e., the spheres.

In this work we aim at combining the ellipsoidal separation idea with well-established SVM-type approaches to construct binary classification models that can be well-suited to some classes of classification problems. While the model we propose is quite close to standard SVM with quadratic kernel, the insistence on the classifier being an ellipsoid brings with it, together with nontrivial computational issues, also new opportunities for *regularization* that we show having a potentially positive impact on the generalisation capabilities of the model.

The paper is organized as follows: in Section 2 we present the model and we discuss its nuances in terms of regularization and hyperparameters, while in Section 3 we experimentally compare our model with SVM with quadratic kernel and spherical

separation on some classification tasks for which the latter has shown to be useful.

2. The model. Let $\mathcal{X} = \{x_1, \ldots, x_p\} \subset \mathbb{R}^n$ be a set of samples (or points). In the supervised learning setting we assume that for any point x_i of $\mathcal{X}$ a *label* y_i is given. The general case $y_i \in \mathbb{R}$ is known as "regression", while y_i taking values in a finite set yields a classification problem. In particular, we consider here $y_i \in \{-1, +1\}$, i.e., the binary classification setting. Alternatively, one can define $\mathcal{X}_+ = \{ i : y_i = 1 \}$ and $\mathcal{X}_- = \{ i : y_i = -1 \}$; with a small abuse of notation we will consider sets of points equivalent to sets of their indices, so as to be able to write, for instance, that $\mathcal{X}_+ \cup \mathcal{X}_- = \mathcal{X}$ and $\mathcal{X}_+ \cap \mathcal{X}_- = \emptyset$. The objective of supervised learning is to predict the label of any new sample only on the basis of the information of the labeled points (the training set $\mathcal{X}$).

The literature in supervised machine learning is extremely rich. An important role is played by the well-known SVM technique (cf. e.g. [7]), an approach exhibiting both good generalisation capabilities and high computational efficiency due to only requiring the solution of a convex problem for the training phase. The latter characteristic allows to experiment with several different variants of the basic model in order to adapt it to different setting, see for example the recent works [9, 6, 2, 3]. The main idea in the SVM technique is the introduction of the concept of "margin" in the strict separation of two sets of points by means of a hyperplane. In fact, the output of any SVM model is a hyperplane equidistant from two parallel hyperplanes, each one supporting one of the two sets. Since strict linear separability cannot be assumed for most data sets, the approach requires choosing a trade-off between maximizing the distance between the support hyperplanes and minimizing a measure of the misclassification errors.

More formally, in the SVM approach a separating hyperplane characterised by $(w, w_0) \in \mathbb{R}^{n+1}$ is constructed as the solution of the convex nondifferentiable minimization problem

$$\min_{w,w_0} \left\{ \tfrac{1}{2}\|w\|^2 + C \sum_{i \in \mathcal{X}} \max\{ 0 , 1 - y_i(w^T x_i - w_0) \} \right. \tag{2.1}$$

The second term in the objective is the *loss function* measuring the misclassification error. The first term instead corresponds to the maximisation of the margin, as the distance between two parallel hyperplanes whose normal is w can be seen to be proportional to $1/\|w\|^2$. This is also called the *regularisation term* in the objective, and corresponds to the fact that, roughly speaking, a smaller w is a "more parsimonious hypothesis" for the (approximate) separation of the two sets which can be proven, both theoretically and experimentally, to lead to better generalisation capabilities when the hyperplane is used to classify previously unseen points. Since the two terms in the objective are potentially contrasting each other, a standard scalarization technique is used with the introduction of the arbitrary *hyperparameter* $C \geq 0$, which is typically determined experimentally for the given $\mathcal{X}$ via *grid search* and *cross validation*.

Problem (2.1) is typically rewritten as the convex QP

$$\begin{array}{ll} \min\limits_{w,w_0} & \frac{1}{2}\|w\|^2 + C \sum_{i \in \mathcal{X}} \xi_i \\ \text{s.t.} & y_i(w^T x_i - w_0) \geq 1 - \xi_i \ , \ \ \xi_i \geq 0 \quad i \in \mathcal{X} \end{array} \tag{2.2}$$

which can then be tackled by means of its dual

$$\begin{array}{ll} \max\limits_{\alpha} & \sum_{i\in\mathcal{X}} \alpha_i - \frac{1}{2}\sum_{i\in\mathcal{X}}\sum_{ji\in\mathcal{X}} \alpha_i\alpha_j y_i y_j x_i^\top x_j \\ \text{s.t} & \sum_{i\in\mathcal{X}} \alpha_i y_i = 0 \\ & 0 \le \alpha_i \le C \qquad\qquad i \in \mathcal{X} \end{array} \tag{2.3}$$

This is useful for two reasons. On one hand, (2.3) can be easier to solve computationally than (2.2). Possibly more importantly, though, in (2.3) the training data only appear in the form of scalar products $x_i^T x_j$ between input vectors. This property is the basis of the "kernel trick", which allows to construct nonlinear separation surfaces in the original input space by finding a linear separation in a (typically, higher-dimensional) different *feature space*. The basic idea consists in choosing a mapping φ from $\mathbb{R}^n$ to some other (possibly infinite dimensional) Euclidean space; then, (2.3) only depends on the data through the scalar products $\varphi(x_i)^\top\varphi(x_j)$ in the new space. If there exists a *kernel function* $K : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ such that $K(x_i, x_j) = \varphi(x_i)^\top\varphi(x_j)$, the training model will only need to compute K without the need of explicitly knowing φ. It is well-known that for K to be a valid kernel function the Gram matrix $Q \in \mathbb{R}^{n\times n}$, with entries $Q_{ij} = K(x_i, x_j)$, must be symmetric and positive definite. This allows to construct many different kernel functions, such as

- polynomial kernels: $K(x_i, x_j) = (x_i^\top x_j + c)^p$ for fixed $c \in \mathbb{R}$ and integer $p > 1$;
- Radial Basis Function (RBF) kernels: $K(x_i, x_j) = e^{-\sigma\|x_i - x_j\|^2}$ for fixed σ;
- sigmoid kernels: $K(x_i, x_j) = \tanh(\mu x_i^\top x_j + \nu)$ for fixed μ and ν.

We focus on the quadratic kernel, i.e., the polynomial one with $p = 2$. While for RBF kernels the features space is infinite-dimensional, in the quadratic case the feature map $\varphi : \mathbb{R}^n \to \mathbb{R}^N$, where $N = (n+1)(n+2)/2$, can be written explicitly as

$$\varphi(x) = [\,\varphi_1(x)\,,\,\ldots\,,\,\varphi_n(x)\,,\,\sqrt{2c}x\,,\,c\,]^\top \text{ with } \varphi_i(x) = [\,x_i^2\,,\,\sqrt{2}x_i x_1 \ldots\,,\,\sqrt{2}x_i x_n\,]^\top \ .$$

The linear classifier in the feature space $\mathbb{R}^N$, i.e., the hyperplane $w^\top\varphi(x) = w_0$, corresponds in the original space $\mathbb{R}^n$ to a quadratic form $x^\top Q(w)x + q(w)x = w_0$, where the symmetric $Q(w)$ is made of the first $n(n+1)/2$ components of w (with the obvious arrangement) and $q(w)$ to the following $n+1$ ones, i.e. $Q(w)$ and $b(w)$ are the following:

$$Q(w) = \frac{1}{2}\begin{bmatrix} 2w_1 & \sqrt{2}w_2 & \cdots & \sqrt{2}w_n \\ \sqrt{2}w_2 & 2w_{n+1} & \cdots & \sqrt{2}w_{2n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \sqrt{2}w_n & \sqrt{2}w_{2n-1} & \cdots & 2w_{n(n+1)/2} \end{bmatrix},\ q(w) = \begin{bmatrix} \sqrt{2c}w_{N-n} \\ \vdots \\ \sqrt{2c}w_{N-1} \\ w_N \end{bmatrix}.$$

This defines an ellipsoid only if $Q(w)$ is positive semi-definite. Since in the SVM framework no additional conditions are given on w, the separator in the original space may be any quadratic surface. On the basis of our expertise and experience, "the right classifier" for several datasets appearing in the real world should have an ellipsoidal form, i.e., identify a compact surface (enclosing a compact region) rather than a non-compact one. To obtain this we add a SDP constraint on $Q(w)$ in the primal model

(2.2); in the input space this corresponds to finding an ellipsoid (approximately) separating $\mathcal{X}_+$ from $\mathcal{X}_-$, i.e., enclosing all points of $\mathcal{X}_+$ and no points of $\mathcal{X}_-$. This yields the SDP model

$$\begin{array}{ll} \min\limits_{w,w_0} & \frac{1}{2}\|w\|_2^2 + C_1 \sum_{i\in\mathcal{X}_+} \xi_i + C_2 \sum_{i\in\mathcal{X}_-} \xi_i + C_3 vol(Q(w)) \\ s.t. & y_i(\varphi(x_i,c)^\top w - w_0) \geq 1 - \xi_i \ \ , \ \ \xi_i \geq 0 \qquad\qquad i \in \mathcal{X} \\ & Q(w) \succeq 0 \end{array} \tag{2.4}$$

With respect to the original (2.2), the fundamental difference is the SDP constraint on $Q(w)$. Moreover, since we expect the two classes to be "rather different", we penalise differently in the objective function the misclassification of points in $\mathcal{X}_+$ and that of points in $\mathcal{X}_-$ by adding two specific hyper-parameters C_1 and C_2 (this can, of course, be done in the original SVM model as well). We also add a further regularization term, i.e., the volume of the ellipsoid; this is proportional to $\log\det(Q(w))$, which can be handled by SDP programs with standard formulation tricks [7]. This term encourages choosing "smaller" ellipsoids, which makes intuitive sense, and it is only possible when $Q(w)$ is forced to be PSD. It should also be remarked that an uncommon trade-off exists between this and the standard regularization term. In fact, if the points in $\mathcal{X}_+$ actually all belong to some lower-dimensional affine subspace of $\mathbb{R}^n$, then an almost-0-volume highly degenerate ellipsoid exists that contains them all and such that some of its axes have length very close to 0. This corresponds to the eigenvalues of $Q(w)$ for the eigenvectors of the corresponding axes, that are proportional to the square root of the inverse of the length of the axis, having very large values. One effect of the standard regularization term is to avoid this "excessive flattening" of the ellipsoid along the directions orthogonal to the subspace where $\mathcal{X}_+$ lies; besides this being intuitively advantageous for the generalization capabilities of the approach, in our experience it also reduces the significant numerical difficulties that a SDP solver could incur into in the non-stabilised case. This justifies why the volume regularization term has its own hyperparameter that need be properly tuned.

3. Numerical Experiments. As in [5] we have applied the model (2.4) to the edge detection problem, which consists in deciding whether any pixel in an image belongs to an edge or not. In our approach the pixels belonging to an edge are the pixels in the border between dark and bright zones in the image.

We have considered 27 gray-scale images taken from the repository [13], together with the binary images coming from the real world. We have considered each image I as a $m \times n$ matrix where each entry corresponds to a pixel whose value is its luminosity in the range $[0, 255]$. For the images in the training set, for each pixel we also know its edge/non-edge, i.e., the label $\{-1, 1\}$. Each entry $I_{i,j}$ in the interior of I (i.e., excluding the first and last rows and columns) is associated with a vector $z_{i,j} \in \mathbb{R}^8$ whose entries are the absolute values of the differences between $I_{i,j}$ and its neighborhood pixels (see Fig. 3):

$$z_{i,j} = \left[\begin{array}{l} |I_{i,j} - I_{i-1,j}|\,, |I_{i,j} - I_{i-1,j+1}|\,, |I_{i,j} - I_{i,j+1}|\,, |I_{i,j} - I_{i+1,j+1}|\,, \\ |I_{i,j} - I_{i+1,j}|\,, |I_{i,j} - I_{i+1,j-1}|\,, |I_{i,j} - I_{i,j-1}|\,, |I_{i,j} - I_{i-1,j-1}| \end{array} \right] . \tag{3.1}$$

The label for the element $z_{i,j}$ corresponds to that of the pixel i, j. The corresponding about 10^6 labelled vectors in $\mathbb{R}^8$ have been randomly splitted: 80% have been kept

aside as testing set, while the remaining 20% have been used for training and hyper-parameter tuning with a standard 10-fold cross validation (randomly and repeatedly splitting them into 90% for training and 10% for validation).

FIG. 3.1.

$I_{i-1,j-1}$	$I_{i-1,j}$	$I_{i-1,j+1}$
$I_{i,j-1}$	$I_{i,j}$	$I_{i,j+1}$
$I_{i+1,j-1}$	$I_{i+1,j}$	$I_{i+1,j+1}$

Hence, during training we considered points coming from images having rather different contrast than others, where the contrast of a gray-scale image I is defined as $C(I) = \max_{i,j} I_{i,j} - \min_{i,j} I_{i,j}$ [16]. To address this we have preprocessed each image so that they all have the same contrast by scaling their luminosity range in $[0, 255]$, i.e., re-scaling each of its pixel values $I_{i,j}$ as

$$255(I_{i,j} - \min_{i,j} I_{i,j})/C(I) \ .$$

We have compared the classification results obtained from the Ellipsoidal SVM model (EllSVM) 2.4, the standard SVM with quadratic kernel (SVM), and the Spherical classification model (SpherSep) described in [5]. We have implemented the EllSVM, QuadSVM, SpherSep in Python under the Skit-learn packages [15]; the semi-definite program (2.4) arising in our approach has been solved by the Mosek solver [14] under the Fusion API. The code has been ran on an Intel i5-4460 3.00 GHz 16-core with 132 GB of RAM under a i686 GNU/Linux operating system.

By means of a standard grid search and the 10-fold cross validation we have identified the best hyper-parameters for all the models independently. For the hyper-parameter c we have tested a grid of values in the interval $[0.1, 5]$ and we have observed that the results are approximately the same, thus we have set $c = 1$. For the hyper-parameters C_1 and C_2 we have tested all pairs in the grid of values 10^k for $k = -3, \ldots, 2$, obtaining the best performances for $C_1 = 10$ and $C_2 = 1$. Moreover, also the parameter C_3 does not significantly impact on the results of the edge detection problem, and therefore we have set $C_3 = 0$: this actually simplifies the SDP model eliminating the extra variables and constraints required to represent the volume term in the objective of (2.4). We expect that $C_3 \neq 0$ could improve the generalization capabilities of ELLSVM in other real applications, although possibly at the cost of making the SDP problem, that is already more difficult to be solved than the quadratic model for SVM, even more computationally expensive. The running time for the training phase of ELLSVM is two orders of magnitude larger than the one of QSVM, which in turn is about one order of magnitude larger than that of SpherSep. However, we have used off-the-shelf, general-purpose SDP solvers to run ELLSVM. It is likely that approaches exploiting the structure of the underlying problems, or even non-IP SDP approaches (e.g., using augmented Lagrangian and/or alternating direction methods [21, 22]) could significantly reduce the ELLSVM training cost.

For quadratic SVM model we have performed the grid search, as in ELLSVM, on the two different hyper-parameters C_1 and C_2 to weight the classification errors of the classes $\mathcal{X}_+$ and $\mathcal{X}_-$; however, in this case the best result is obtained for $C_1 = C_2 = 1$.

This seems to confirm that insisting that the separator is an ellipsoid (in the feature space) helps in properly differentiating the two classes of instances, thereby possibly performing a better classification. We should remark that the SVM model may in fact spontaneously select a SDP $Q(w)$, but the results show that this is not happening naturally and that adding the constraint is required.
The SpherSep model from [5] depends only on one hyper-parameters C determining the penalization of the misclassification error; by testing values in the set 10^k for $k = -3, \ldots, 3$ we have obtained the best results for $C = 1$.
We start by providing the in-sample and out-of-sample results, in the classical terms of precision and recall, in Table 3.1. The high value for the recall score of the edge-pixels indicates that ELLSVM and QSVM are able to very accurately detect them. Moreover, the fact that the models behave the same in-sample and out-of-sample confirms that they generalize well, despite the training set only being the 20%. Yet, according to these metrics ELLSVM is not better, and sometimes worse, than QSVM.

		in-sample		out-of-sample	
		Edge	Non-Edge	Edge	Non-Edge
ELLSVM	Precision	0.55	0.69	0.55	0.69
	Recall	0.88	0.28	0.88	0.27
QuadSVM	Precision	0.55	0.75	0.55	0.74
	Recall	0.92	0.24	0.92	0.23
SpherSep	Precision	0.57	0.56	0.57	0.56
	Recall	0.57	0.56	0.56	0.57

TABLE 3.1

Comparison of the precision and recall values for the pixels in the training set and in the validation set for the three models ELLSVM, QSVM and SpherSep.

We have them complemented the standard per-pixel metrics with a more comprehensive per-picture metric, i.e., Pratt's figure of merit (PFM) [16]. The PFM is a quantitative assessment for the edge detection problem defined as

$$PFM = \frac{1}{\max\{N_A, N_D\}} \sum_{k=1}^{N_D} \frac{1}{1 + \alpha d_k}, \tag{3.2}$$

where N_A and N_D are, respectively, the number of actual edge pixels and the number of the detected edge pixels. For every detected edge k, d_k is the distance, evaluated on the actual edges image E, between such a pixel and the closest edge one, while α is a scaling parameter usually taken equal to $1/9$. The PFM was introduced to analyse and balance the associated errors in edge detection process. As the value get closer to 1, it shows better detected edge values. The accuracy and PFM for every image we have considered are reported in Table 3.2. Note that the metric also considers the pixels in the training set, but the previous results show that this should not significantly change the figures (and, anyway, the training set is only 20% of the total).
The table shows that, while resulting is similar accuracies, EllSVM and QSVM are quite different when measured by the PFM; more often than not EllSVM outper-

	Accuracy			PFM		
Img	EllSVM	QSVM	SpherSep	EllSVM	QSVM	SpherSep
1	0.90	0.91	0.72	0.44	0.29	0.21
2	0.89	0.90	0.74	0.62	0.58	0.24
3	0.87	0.87	0.66	0.27	0.27	0.11
4	0.87	0.88	0.70	0.50	0.56	0.22
5	0.83	0.84	0.72	0.53	0.60	0.31
6	0.94	0.95	0.90	0.59	0.50	0.46
7	0.87	0.88	0.69	0.61	0.71	0.23
8	0.87	0.89	0.70	0.60	0.42	0.23
9	0.84	0.87	0.60	0.18	0.20	0.08
10	0.97	0.97	0.97	0.71	0.72	0.64
11	0.90	0.89	0.74	0.24	0.21	0.10
12	0.84	0.85	0.60	0.50	0.53	0.18
13	0.95	0.96	0.85	0.52	0.40	0.22
14	0.84	0.86	0.58	0.30	0.35	0.12
15	0.87	0.88	0.74	0.61	0.49	0.28
16	0.99	0.99	0.99	0.83	0.89	0.63
17	0.90	0.91	0.84	0.58	0.46	0.34
18	0.87	0.88	0.79	0.43	0.46	0.27
19	0.83	0.86	0.68	0.27	0.34	0.15
20	0.92	0.92	0.90	0.69	0.71	0.45
21	0.92	0.92	0.93	0.34	0.05	0.79
22	0.79	0.81	0.66	0.33	0.37	0.20
23	0.94	0.94	0.93	0.55	0.48	0.68
24	0.85	0.87	0.64	0.58	0.49	0.24
25	0.92	0.92	0.81	0.59	0.60	0.26
26	0.86	0.87	0.77	0.59	0.50	0.40
27	0.93	0.93	0.88	0.68	0.62	0.38

TABLE 3.2
Accuracy and PFM comparison for EllSVM, QSVM, SpherSep on the training images.

forms QSVM, sometimes by a significant margin (e.g., picture 21). Both approaches significantly outperform SpherSep.

To further verify that the results do not depend on the images chosen for the training we have also considered 14 other images not included in the training set, we predicted their edge pixels and computed the accuracy and PFM. The results of these experiments are shown in table 3.3 and fully confirm the previous ones.

Finally, Figure 3.2, 3.3, 3.4 provide some visual results that we have obtained among the testing images. We observe that the SpherSep model classifies too many pixels as edge obtaining a high number of false positive elements. On the other hand, QSVM seems to not be able to detect enough edge pixels, leaving some gaps in the contours lines. From this point of view, the model ELLSVM obtains results in between the other two, being able to detect enough edge pixels so that the contours line are almost everywhere complete, but not too many so that the binary image is not clean.

All in all, our experiments show that the newly proposed ELLSVM classification

	Accuracy			PFM		
Img	EllSVM	QSVM	SpherSep	EllSVM	QSVM	SpherSep
1	0.95	0.95	0.91	0.80	0.78	0.44
2	0.89	0.90	0.78	0.33	0.37	0.16
3	0.83	0.84	0.65	0.38	0.41	0.18
4	0.97	0.97	0.93	0.62	0.48	0.32
5	0.89	0.91	0.73	0.39	0.37	0.15
6	0.86	0.87	0.62	0.28	0.33	0.11
7	0.95	0.95	0.89	0.42	0.38	0.29
8	0.99	0.99	0.98	0.71	0.72	0.51
9	0.92	0.93	0.90	0.30	0.16	0.69
10	0.81	0.83	0.65	0.34	0.38	0.19
11	0.90	0.92	0.80	0.22	0.27	0.12
12	0.91	0.92	0.85	0.56	0.54	0.29
13	0.91	0.92	0.80	0.69	0.64	0.27
14	0.95	0.96	0.84	0.62	0.59	0.20

TABLE 3.3
Accuracy and PFM comparison for EllSVM, QSVM, SpherSep on the testing images

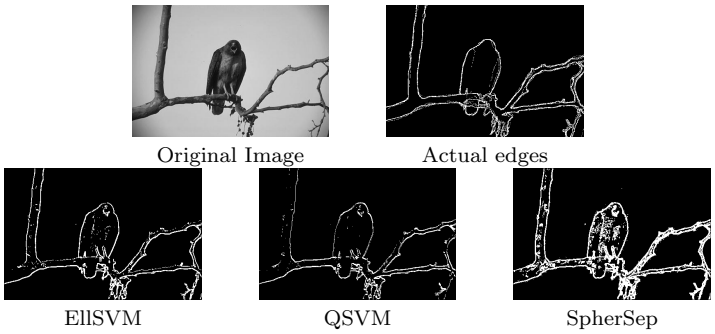

FIG. 3.2. *Testing Image1*

approach, based on the ellipsoidal separation, is comparable to QSVM—and better than SpherSep—in terms of accuracy, but it is generally better in terms of the PFM score, proving that the insertion of the SDP constraint ensuring the compactness of the separation surface can indeed help for certain classification tasks, in particular those where one class is much less numerous as the other such as the edge detection problem.

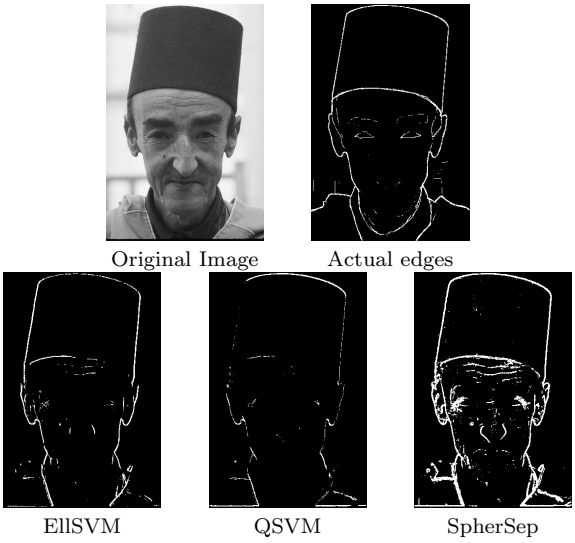

FIG. 3.3. *Testing Image4*

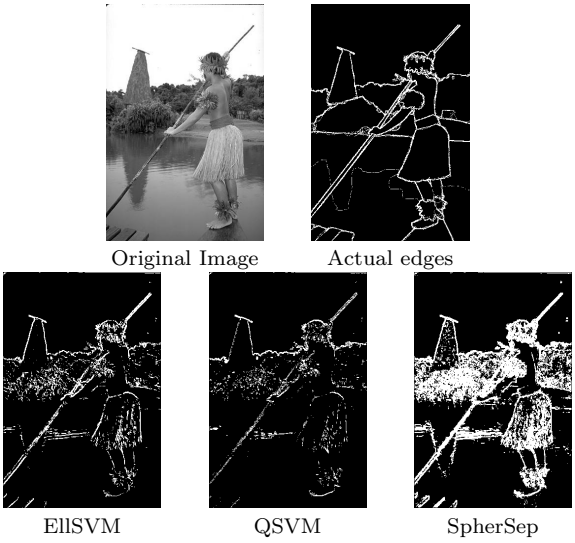

FIG. 3.4. *Testing Image13*

REFERENCES

[1] A. ASTORINO, A. CHIARELLO, M. GAUDIOSO, AND A. PICCOLO, *Malicious URL detection via spherical classification*, Neural Computing and Applications, 28 (2017), pp. 699–705.

[2] A. ASTORINO AND A. FUDULI, *The proximal trajectory algorithm in svm cross validation*, IEEE Transactions on Neural Networks and Learning Systems, 27 (2016), pp. 966–977.

[3] A. ASTORINO, A. FUDULI, G. GIALLOMBARDO, AND G. MIGLIONICO, *Svm-based multiple instance classification via dc optimization*, Algorithms, 12 (2019).

[4] A. Astorino and M. Gaudioso, *Ellipsoidal separation for classification problems*, Optimization Methods and Software, 20 (2005), pp. 261–270.

[5] A. Astorino, M. Gaudioso, and W. Khalaf, *Edge detection by spherical separation*, Computational Management Science, 11 (2014), pp. 517–530.

[6] A. Astorino, E. Gorgone, M. Gaudioso, and D. Pallaschke, *Data preprocessing in semi-supervised svm classification*, Optimization, 60 (2011), pp. 143–151.

[7] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*, Cambridge University Press, 2000.

[8] G. M. Fung, O. L. Mangasarian, and A. J. Smola, *Minimal kernel classifiers*, Journal of Machine Learning Research, 3 (2003), pp. 303–321.

[9] M. Gaudioso, E. Gorgone, M. Labbe, and A. Rodriguez-Chia, *Lagrangian relaxation for svm feature selection*, Computers and Operations Research, 87 (2017), pp. 137–145.

[10] T. Hofmann, B. Scholkopf, and A. J. Smola, *Kernel methods in machine learning*, The Annals of Statistics, 36 (2008), pp. 1171 – 1220.

[11] B. Liu, *Bioseq-analysis: A platform for dna, rna and protein sequence analysis based on machine learning approaches*, Briefings in Bioinformatics, 20 (2018), pp. 1280–1294.

[12] E. N. Malamas, E. G. M. Petrakis, M. Zervakis, L. Petit, and J. . Legat, *A survey on industrial vision systems, applications and tools*, Image and Vision Computing, 21 (2003), pp. 171–188.

[13] D. Martin, C. Fowlkes, D. Tal, and J. Malik, *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics*, Proceedings of the IEEE International Conference on Computer Vision, 2 (2001), pp. 416–423.

[14] MOSEK ApS, *MOSEK Optimizer API for Python 9.2.46*, 2019.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.

[16] W. Pratt, *Introduction to digital image processing*, CRC press, 2013.

[17] B. Schölkopf, C. Burges, and A. Smola, *Advances in kernel methods. Support vector learning*, MIT Press, Cambridge, MA, 1999.

[18] A. J. Smola and R. Kondor, *Kernels and regularization on graphs*, in Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science), vol. 2777, 2003, pp. 144–158.

[19] A. Statnikov, C. F. Aliferis, I. Tsamardinos, D. Hardin, and S. Levy, *A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis*, Bioinformatics, 21 (2005), pp. 631–643.

[20] V. Vapnik, *The nature of the statistical learning theory*, Springer Verlag, New York, 1995.

[21] Z. Wen, D. Goldfarb, and W. Yin, *Alternating direction augmented lagrangian methods for semidefinite programming*, Mathematical Programming Computation, 2 (2010), pp. 203–230.

[22] L. Yang, D. Sun, and K.-C. Toh, *SDPNAL++: a majorized semismooth newton-cg augmented lagrangian method for semidefinite programming with nonnegative constraints*, Mathematical Programming Computation, 7 (2015), pp. 331–366.