.

ELLIPSOIDAL CLASSIFICATION VIA SEMIDEFINITE PROGRAMMING

Annabella Astorino[a,*], Antonio Frangioni[b], Enrico Gorgone, Benedetto Manca[c]

[a] *Istituto di Calcolo e Reti ad Alte Prestazioni – C.N.R., Rende, Italy*
[b] *Dipartimento di Informatica, Universitá di Pisa, Pisa, Italy*
[c] *Dipartimento di Matematica, Universitá di Cagliari, Cagliari, Italy*

Abstract

We propose a classification approach exploiting relationships between ellipsoidal separation and Support-vector Machine (SVM) with quadratic kernel. By adding a (Semidefinite Programming) SDP constraint to SVM model we ensure that the chosen hyperplane in feature space represents a non-degenerate ellipsoid in input space. This allows us to exploit SDP techniques within Support-vector Regression (SVR) approaches, yielding better results in case ellipsoid-shaped separators are appropriate for classification tasks. We compare our approach with spherical separation and SVM on some classification problems.

Keywords: Classification, Semidefinite Programming, Artificial intelligence

1. Introduction

The problems of separation of sets, traditionally 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 to efficiently construct 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

*Corresponding author
Email addresses: astorino@icar.cnr.it (Annabella Astorino), frangio@di.unipi.it (Antonio Frangioni), gorgone.enrico@gmail.com (Enrico Gorgone), bmanca@unica.it (Benedetto Manca)

(Astorino et al., 2017), object recognition in machine vision (Malamas et al., 2003), edge detection (Astorino et al., 2014), gene expression profile analysis (Statnikov et al., 2005), DNA and protein analysis (Liu, 2018), and many others.

In this work, we focus on supervised binary classification (Cristianini and Shawe-Taylor, 2000), 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 binary 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 largely 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 the misclassification of known inputs, while 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" 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. Effective binary classification can then be used as the fundamental building block to develop multi-class approaches.

One of the fundamental decisions in the process is the shape of the separating surface. The most natural shape, and not coincidentally the most widely used, is the simplest one, i.e., an affine hyperplane. This is the basis of the widely used Support Vector Machine (SVM) methodology (Vapnik, 1995; Cristianini and Shawe-Taylor, 2000; Schölkopf et al., 1999). 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 (Fung et al., 2003; Smola

and Kondor, 2003; Hofmann et al., 2008; Schölkopf et al., 1999) 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 (Astorino and Gaudioso, 2005) 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 (Astorino et al., 2014) 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.

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 label information of the points in 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. (Cristianini and Shawe-Taylor, 2000)), 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 (Gaudioso et al., 2017; Astorino et al., 2011; Astorino and Fuduli, 2016; Astorino et al., 2019). 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

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

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 (1) is typically rewritten as the convex QP

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

which can then be tackled by means of its dual

$$\begin{array}{ll} \max\limits_{\alpha} & \sum_{i\in\mathcal{X}} \alpha_i - \frac{1}{2}\sum_{i\in\mathcal{X}}\sum_{ji\in\mathcal{X}} \alpha_i\alpha_j y_i y_j x_i^\top x_j \\ \text{s.t} & \sum_{i\in\mathcal{X}} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \qquad\qquad\qquad\qquad\qquad\qquad i \in \mathcal{X} \end{array} \tag{3}$$

This is useful for two reasons. On one hand, (3) can be easier to solve computationally than (2). Possibly more importantly, though, in (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, (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

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

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 $q(w)$ 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); 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

$$\begin{array}{ll} \min\limits_{w,w_0} & \frac{1}{2}\|w\|_2^2 + C_1\sum_{i\in\mathcal{X}_+}\xi_i + C_2\sum_{i\in\mathcal{X}_-}\xi_i + C_3 vol(Q(w)) \\ s.t. & y_i(\varphi(x_i,c)^\top w - w_0) \geq 1 - \xi_i \;\;,\;\; \xi_i \geq 0 \qquad\qquad i \in \mathcal{X} \\ & Q(w) \succeq 0 \end{array} \tag{4}$$

With respect to the original (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 hyper-parameter, i.e., a regularization term for the volume of the ellipsoid; this is proportional to $\log\det(Q(w))$, which can be handled by SDP programs with standard formulation tricks (Cristianini and Shawe-Taylor, 2000). 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

We have tested the model (4) on both binary and multi-class datasets taken from the LIBSVM repository (Chang and Lin, 2011). For the binary datasets we have trained the model as follows: 60% of the data have been kept aside as testing set, while the remaining 40% have been used for training and hyper-parameters tuning, using a four by four grid with values $\{10^{-2}, 10^{-1}, 1, 10^1, 10^2\}$, with a standard 5-fold cross validation (randomly and repeatedly splitting them into 90% for training and 10% for validation). We have compared the Ellipsoidal SVM model (4) (ELL_SVM) and the one with quadratic kernel SVM (QSVM). Both approaches have been implemented in Python under the Skit-learn packages (Pedregosa et al., 2011); the semi-definite program (4) arising in our approach has been solved by the Mosek solver (MOSEK ApS, 2019) under the Fusion API. The code has

been ran on an Intel i5-4460 3.00 GHz 4-core with 132 GB of RAM under a i686 GNU/Linux operating system. Table 1 shows the results in terms of accuracy.

Dataset	p	n	QSVM	ELL_SVM
Australian	690	14	0.86	0.86
Breast Cancer	683	10	0.97	0.97
Diabetes	768	8	0.76	0.75
Ionosphere	351	34	0.81	0.86
Liver Disorders	145	5	0.66	0.71

Table 1: Numerical results on binary datasets

For the multi-class datasets we have performed a standard one-vs-all approach, training the model (4) k times, in order to obtain a separating ellipsoid E_k for each one of the k class of the dataset. We have trained the model in two different ways. The first (and perhaps most natural) one, denoted with ELL_I, constructs an ellipsoid E_k which contains the points in the class k and leaves all the other points outside; in other words, class k has label -1. The second one, denoted with ELL_O, rather provides an ellipsoid E_k which contains all the points in the classes $h \neq k$ and leaves the points of the class k outside (i.e., class k has label $+1$). We observe that the quadratic expression defining the ellipsoid E_k, for each class k, can be interpreted as a signed distance function $D_k : \mathbb{R}^n \to \mathbb{R}$, which assumes negative values in the interior of E_k, vanishes in the border and assumes positive values in the exterior of E_k. Therefore, for the approach ELL_I, since the ellipsoid E_k contain the points of the class k, we assign to a point $z \in \mathbb{R}^n$ the label k if $D_k(z) = \min_h\{D_h(z)\}$, that is,

$$class(z) = \arg\min_h\{D_h\}.$$

Analogously, for the approach ELL_O, since the ellipsoid E_k does not contain the points in the class k, we assign to a point $z \in \mathbb{R}^n$ the label k if $D_k(z) = \max_h\{D_h(z)\}$, that is,

$$class(z) = \arg\max_h\{D_h\}.$$

Table 3 shows the comparison, in terms of accuracy, between ELL_I, ELL_O and QSVM, trained using a standard one-vs-all approach, for the multi-class

datasets we have tested. Both tables show, as expected, that none of the approaches strictly dominates the others (save for ELL_I being, as expected, generally better than ELL_O): while being generally comparable, for some tasks forcing the separating surface to be a compact set seems to work indeed better, while for some other it does not. Notably, QSVM could produce an ellipsoid even without being forced to: the results, however, indicated that this does not "naturally" happen unless the explicit constraint is added.

Dataset	p	n	k	QSVM	ELL_I	Ell_O
Glass	214	9	6	0.67	0.67	0.71
Iris	150	4	3	0.81	0.96	0.89
Svmguide2	391	20	3	0.83	0.82	0.80
Svmguide4	300	10	6	0.44	0.54	0.52
Vehicle	846	18	4	0.81	0.80	0.80
Vowel	528	10	11	0.84	0.85	0.81
Wine	178	13	3	0.94	0.96	0.95

Table 2: Numerical results on multi-class datasets

Moreover, as in (Astorino et al., 2014), we have applied the model (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 (Martin et al., 2001), 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:

$$z_{i,j} = \left[\begin{array}{l} |I_{i,j} - I_{i-1,j}|\,,\ |I_{i,j} - I_{i-1,j+1}|\,,\ |I_{i,j} - I_{i,j+1}|\,,\ |I_{i,j} - I_{i+1,j+1}|\,, \\ |I_{i,j} - I_{i+1,j}|\,,\ |I_{i,j} - I_{i+1,j-1}|\,,\ |I_{i,j} - I_{i,j-1}|\,,\ |I_{i,j} - I_{i-1,j-1}| \end{array} \right]. \tag{5}$$

The label for the element $z_{i,j}$ corresponds to that of the pixel i, j. The pixels corresponding to the label -1 are considered as edge (white) pixels and the pixels corresponding to the label $+1$ are considered as background (black) pixels in the binary edges image. 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 hyperparameter tuning with a standard 10-fold cross validation (randomly and repeatedly splitting them into 90% for training and 10% for validation). 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 $C(I) = \max_{i,j} I_{i,j} - \min_{i,j} I_{i,j}$ (Pratt, 2013). 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 ELL_SVM, QSVM, and the Spherical classification model (SpherSep) of (Astorino et al., 2014). We have implemented the ELL_SVM, QSVM, SpherSep in Python under the Skit-learn packages (Pedregosa et al., 2011); the semi-definite program (4) arising in our approach has been solved by the Mosek solver (MOSEK ApS, 2019) under the Fusion API. The code has been ran on an Intel i5-4460 3.00 GHz 4-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 (4). We expect that $C_3 \neq 0$ could improve the generalization capabilities of ELL_SVM 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 ELL_SVM is around 76647 seconds, the one of QSVM is around 2939 seconds and the one of SpherSep is around

4 seconds. However, we have used off-the-shelf, general-purpose SDP solvers to run ELL_SVM. 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 (Wen et al., 2010; Yang et al., 2015)) could significantly reduce the ELL_SVM training cost.
For quadratic SVM model we have performed the grid search, as in ELL_SVM, 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 (Astorino et al., 2014) 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. The high value for the recall score of the edge-pixels indicates that ELL_SVM 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 ELL_SVM is not better, and sometimes worse, than QSVM.

		in-sample		out-of-sample	
		Edge	Non-Edge	Edge	Non-Edge
ELL_SVM	precision	0.55	0.69	0.55	0.69
	recall	0.88	0.28	0.88	0.27
SVM	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: Precision and recall values for the pixels in the training and in the validation set.

We have them complemented the standard per-pixel metrics with a more

comprehensive per-picture metric, i.e., Pratt's figure of merit (PFM) (Pratt, 2013). The PFM is a quantitative assessment for the edge detection problem defined as

$$PFM = \frac{1}{\max\{N_A, N_D\}} \sum_{k=1}^{N_D} \frac{1}{1 + \alpha d_k}, \tag{6}$$

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 4. 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, ELL_SVM and QSVM are quite different when measured by the PFM; more often than not ELL_SVM outperforms 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 5 and fully confirm the previous ones.
Finally, Figure 1, 2, 3 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, ELL_SVM 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 ELL_SVM classification 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. In general, our results indicate that the insertion of the SDP constraint ensuring the compactness of the separation surface can indeed help for certain classification tasks, in particular

	Accuracy			PFM		
Img	ELL_SVM	QSVM	SpherSep	ELL_SVM	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 4: Accuracy and PFM comparison on the training images.

	Accuracy			PFM		
Img	ELL_SVM	QSVM	SpherSep	ELL_SVM	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 5: Accuracy and PFM comparison on the testing images

those where one class is much less numerous as the other such as the edge detection problem.

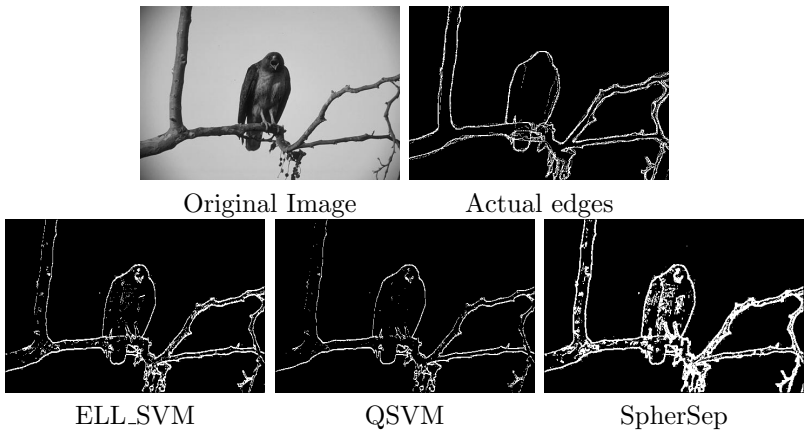

Figure 1: Testing Image1

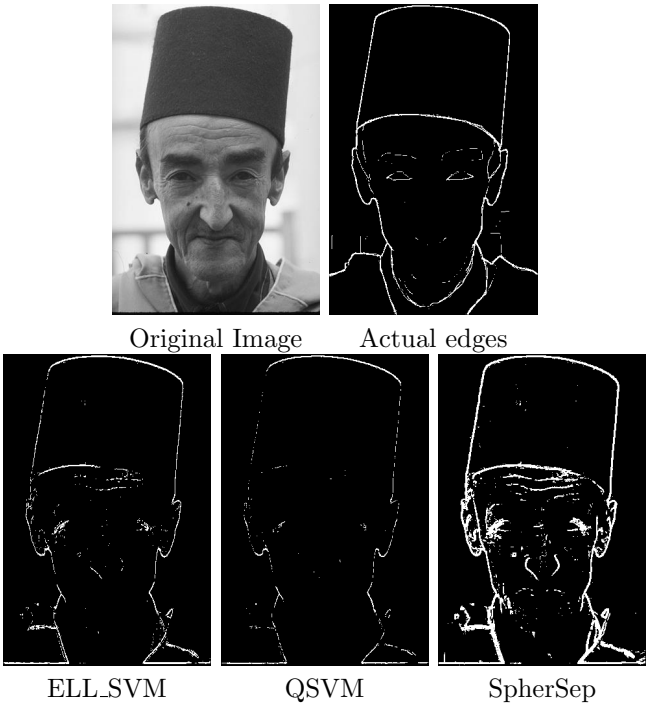

Figure 2: Testing Image4

Acknowledgements

The fourth author was supported by KASBA Funded by Regione Autonoma della Sardegna, Italy; STAGE funded by Fondazione Sardegna and by PON R&I 2014-2020

Astorino, A., Chiarello, A., Gaudioso, M., Piccolo, A., 2017. Malicious URL detection via spherical classification. Neural Computing and Applications 28, 699–705.

Astorino, A., Fuduli, A., 2016. The proximal trajectory algorithm in svm cross validation. IEEE Transactions on Neural Networks and Learning Systems 27, 966–977.

Astorino, A., Fuduli, A., Giallombardo, G., Miglionico, G., 2019. Svm-based multiple instance classification via dc optimization. Algorithms 12.

Astorino, A., Gaudioso, M., 2005. Ellipsoidal separation for classification problems. Optimization Methods and Software 20, 261–270.

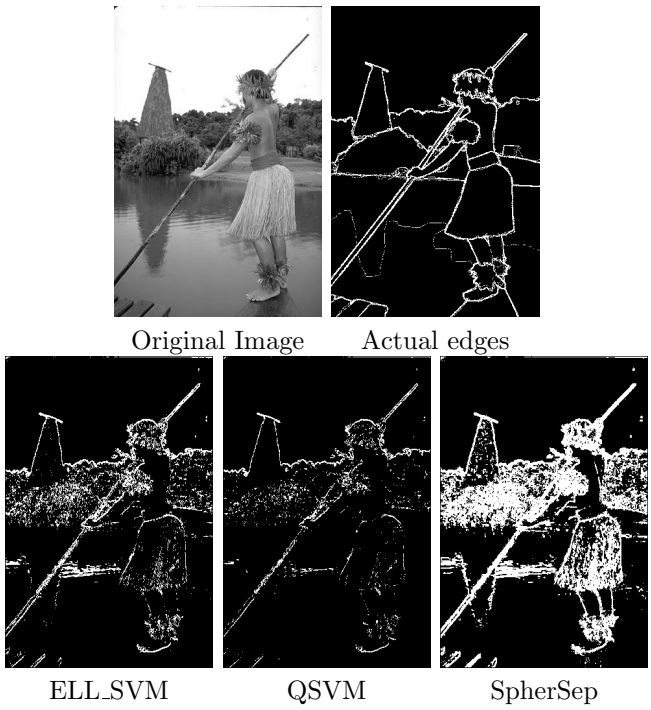

Figure 3: Testing Image13

Astorino, A., Gaudioso, M., Khalaf, W., 2014. Edge detection by spherical separation. Computational Management Science 11, 517–530.

Astorino, A., Gorgone, E., Gaudioso, M., Pallaschke, D., 2011. Data preprocessing in semi-supervised svm classification. Optimization 60, 143–151.

Chang, C.C., Lin, C.J., 2011. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27.

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

Fung, G.M., Mangasarian, O.L., Smola, A.J., 2003. Minimal kernel classifiers. Journal of Machine Learning Research 3, 303–321.

Gaudioso, M., Gorgone, E., Labbe, M., Rodriguez-Chia, A., 2017. La-

grangian relaxation for svm feature selection. Computers and Operations Research 87, 137–145.

Hofmann, T., Scholkopf, B., Smola, A.J., 2008. Kernel methods in machine learning. The Annals of Statistics 36, 1171 – 1220.

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

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

Martin, D., Fowlkes, C., Tal, D., Malik, J., 2001. 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, 416–423.

MOSEK ApS, 2019. MOSEK Optimizer API for Python 9.2.46. URL: `https://docs.mosek.com/9.2/pythonfusion/index.html`.

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

Pratt, W., 2013. Introduction to digital image processing. CRC press.

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

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

Statnikov, A., Aliferis, C.F., Tsamardinos, I., Hardin, D., Levy, S., 2005. A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. Bioinformatics 21, 631–643.

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

Wen, Z., Goldfarb, D., Yin, W., 2010. Alternating direction augmented lagrangian methods for semidefinite programming. Mathematical Programming Computation 2, 203–230.

Yang, L., Sun, D., Toh, K.C., 2015. Sdpnal+: a majorized semismooth newton-cg augmented lagrangian method for semidefinite programming with nonnegative constraints. Mathematical Programming Computation 7, 331–366.