Journal of Machine Learning Research 21 (2020) 1-39 Submitted 3/16; Revised 5/20; Published 10/20

Multiclass Anomaly Detector: the CS++ Support Vector Machine

Alistair Shilton ALISTAIR.SHILTON@DEAKIN.EDU.AU
Applied Artificial Intelligence Institute (A^2I^2)
Deakin University
Geelong, Australia

Sutharshan Rajasegarar SUTHARSHAN.RAJASEGARAR@DEAKIN.EDU.AU
School of Information Technology
Deakin University
Geelong, Australia

Marimuthu Palaniswami PALANI@UNIMELB.EDU.AU
Department of Electrical and Electronic Engineering
The University of Melbourne
Melbourne, Australia

Editor: Francis Bach

Abstract

A new support vector machine (SVM) variant, called CS++-SVM, is presented combining multiclass classification and anomaly detection in a single-step process to create a trained machine that can simultaneously classify test data belonging to classes represented in the training set and label as anomalous test data belonging to classes not represented in the training set. A theoretical analysis of the properties of the new method, showing how it combines properties inherited both from the conic-segmentation SVM (CS-SVM) and the 1-class SVM (to which the method described reduces to in the case of unlabelled training data), is given. Finally, experimental results are presented to demonstrate the effectiveness of the algorithm for both simulated and real-world data.

Keywords: Multiclass, Anomaly Detection, SVM, Kernel Machines, 1-class SVM

1. Introduction

Supervised multiclass classification involves the construction of a machine (mathematical function) capable of discriminating between multiple types of object (classes), where for each class there exists a corresponding set of training examples (the training set) that are assumed to be representative of the characteristics of the relevant class. By contrast, anomaly detection involves the construction of a machine that is capable of discriminating between objects that are in some way similar to the examples contained in the (typically unlabelled) training set and "anomalous" examples that are assessed as unlikely to belong to the set of objects represented by the training set.

Typically problems are divided into one category or the other. However in many (or even most) real-world applications what is actually required is a combination of classification

and anomaly detection. We refer to this as the combined classification/anomaly detection problem (combined problem for short), viz.:

> Given a training set containing training examples from n distinct classes, construct a machine (rule) capable of (a) correctly classifying unseen examples from each class and (b) correctly labelling examples that do not belong to any of the represented classes as anomalous.

For clarity, it is important to note that we assume that there are no anomalies present in the training data; however we must account for their presence when the trained machine is applied in-situ. Examples of this include facial recognition systems that seek to identify a small subset of people (for example known criminals) while discarding other faces; and computer network intrusion detection where a system must both identify normal traffic and known attacks (classes) and flag anomalous traffic that may represent new threats.

A related extension of the standard classification problem is the classification with a reject option problem (Bartlett and Wegkamp, 2008; Chow, 1970; Grandvalet et al., 2009; Kwok, 1999; Fumera and Roli, 2002). Classification with a reject option extends classification by applying an additional label (reject) to those examples whose label is sufficiently uncertain - that is, those examples whose the probability of lying in a given class is sufficiently close to 1/2 - while classifying other points in the usual manner. Both the combined classification/anomaly detection problem and classification with a reject option introduce an additional class (anomaly in the first case, reject in the latter) to apply to examples whose classification is in some sense uncertain. However, the motivations for the two approaches are quite different, which leads to very different methodologies. In particular, it may be noted that classification with reject option is concerned with detecting rejecting points for which the label indicates close to $50-50$ odds of lying in either class, while our method (like other anomaly detection methods) actively aims to set aside regions of input space which are sparsely populated by training vectors (not at populated at all) as anomalous regions on the basis that, lacking nearby training vectors, such regions cannot be validly said to belong to any particular class represented in the training set. We also note that, while a number of papers have considered binary SVMs classification with a reject option (Bartlett and Wegkamp, 2008; Grandvalet et al., 2009; Kwok, 1999; Fumera and Roli, 2002) none have extended this to the multiclass SVM case (Grall-Maës et al. (2006) comes closest, while for example Pillai et al. (2013) considers the multi-label case instead).

Another potential application of this multiclass anomaly detector is in the Internet of Things (IoT) domain (Gubbi et al., 2013). The realization of IoT, which enables a wide range of physical objects and environments to be monitored in fine spatial and temporal detail, presents us with the problem of dealing with extremely large and complex evolving data streams. A challenge is to automate the interpretation of such data streams, and the volume of data makes manual inspection impractical. Moreover, at any time, new or previously unseen events may emerge in the data. Classifiers, such as traditional binary and multi-class SVMs (Schölkopf and Smola, 2001) trained using known patterns are unable to identify new emerging patterns in the data and will simply mislabel such events. Hence, a new SVM algorithm is required that can both classify the multiple types of known classes in the data accurately, and simultaneously identify new emerging anomaly classes - that is, tackle the combined classification/anomaly detection problem.

The approach to solving this problem presented in this paper is called the CS++-SVM, being an amalgam of the conic-segmentation support vector machine (CS-SVM) (Shilton et al., 2012) and the 1-class SVM (Schölkopf et al., 2001; Manevitz and Yousef, 2001; Shilton et al., 2015; Erfani et al., 2016). The CS-SVM is a novel SVM based true-multiclass[1] classifier combining the standard max-margin, minimum empirical risk trade-off heuristic and the novel concept of target-space segmentation via generalised vector inequalities (Boyd and Vandenberghe, 2004). Reduction to binary (Allwein et al., 2000) (for example one-versus-all (1vsA) and voting (1vs1)) and the single-machine SVM (Weston and Watkins, 1999; Crammer and Singer, 2001) are special cases of our CS-SVM corresponding to particular choices of target space segmentation (Shilton et al., 2012). The 1-class SVM is a popular anomaly detection technique that maps the data to a higher dimensional space (using the kernel trick) and fits a smooth surface to the majority of the data (normal data), leaving the anomalous data on the other side of the smooth surface. The CS++-SVM combines these approaches by incorporating an additional (anomaly) class into the geometry of the CS-SVM and then adding a term to the primal training problem, motivated by the 1-class SVM, that "pushes the edges" of the anomaly region to fill as much space as possible without encroaching into the regions of "known" classification. A shortened summary of our CS++-SVM was previously published in Shilton et al. (2013).

It may be argued that the combined problem can be solved using a combination of the classifier and an anomaly detector (in a non-integrated way). This is referred to as the hybrid method in the present paper and is shown to be sub-optimal for two main reasons:

1. Training two separate machine learning algorithms is more computationally intensive than training a single algorithm of comparable complexity.

2. In the hybrid method, while the classifier is trained using all available information on the training set, the training data for the anomaly detector does not include class-data (that is, the class to which each training vector belongs) as this does not fit the context of the traditional anomaly detection problem. However, this class data contains valuable clues regarding potential clusters within the training set, as training vectors for individual classes are likely to form localised clusters within the overall training set. The CS++-SVM uses all available information (including class-data) to solve the combined problem in a single step. As is observed in the experimental results Section 6, this allows the CS++-SVM to outperform the hybrid method considered in all experiments.

The remainder of the paper is arranged as follows. Section 2 introduces notational conventions used in the paper. Section 3 presents relevant background material, describing both the CS-SVM and the 1-class SVM. Section 4 presents the CS++-SVM in both primal and dual forms. The theoretical properties of the CS++-SVM, both as a multiclass classifier and as an anomaly detector, are discussed in Section 5. Finally some experimental results on the CS++-SVM are presented in Section 6, and conclusions presented in Section 7.

1. In-so-far as it does not rely on combining multiple binary SVMs to obtain multiclass classification

2. Mathematical Notation

This paper follows the notational conventions of Shilton et al. (2012). Sets are written $\mathbb{A}, \mathbb{B}, \ldots$. The sets of reals, positive reals, negative reals, natural numbers (including 0), integers, positive integers and negative integers are denoted, respectively, $\mathbb{R}$, $\mathbb{R}^+$, $\mathbb{R}^-$, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Z}^+$ and $\mathbb{Z}^-$. The integers modulo $h \in \mathbb{Z}^+$ are denoted $\mathbb{Z}_h = \{0, 1, \ldots, h-1\}$.

Implicit index ranges $i, j, k, l \in \mathbb{Z}_N$, $m \in \mathbb{Z}_{d_K}$, $q, r \in \mathbb{Z}_{d_T}$ and $s, t \in \mathbb{Z}_n$ are used, where N is the size of the training set, d_K and d_T the dimension of feature space and target space, respectively, and n is the number of classes. The number of training vectors in a particular class s is denoted N_s.

Column vectors are denoted $\vec{a}, \vec{b} \in \mathbb{R}^d$ ($d \in \mathbb{Z}^+$), with elements denoted a_h, b_h, where $h \in \mathbb{Z}_d$. The zero vector is denoted $\vec{0}$, and a vector of all ones $\vec{1}$. In the special case of column vectors in target space the notation $\mathbf{a}, \mathbf{b} \in \mathbb{R}^{d_T}$ is used. Transposition is indicated by a superscript T.

A cone (Boyd and Vandenberghe, 2004) is defined to be a set $\mathbb{A} \subseteq \mathbb{R}^d$ ($d \in \mathbb{Z}^+$) for which $\lambda\vec{a} \in \mathbb{A}\ \forall \vec{a} \in \mathbb{A}, \lambda \in \mathbb{R}^+ \cup \{0\}$. A proper cone is a cone that is convex, closed, solid and pointed. The dual cone of the proper cone $\mathbb{A}$ is the proper cone $\mathbb{A}^* = \left\{ \vec{c} \in \mathbb{R}^d \middle| \vec{c}^T\vec{a} \geq 0\ \forall \vec{a} \in \mathbb{A} \right\}$ (Boyd and Vandenberghe, 2004). The (non-strict) generalised vector inequality (Boyd and Vandenberghe, 2004; Ben Tal and Nemirovski, 2001; Nemirovski, 2005) defined by the proper cone $\mathbb{A}$ is denoted $\succeq_{\mathbb{A}}$ and the strict generalised inequality $\succ_{\mathbb{A}}$, where:

$$\begin{aligned} \vec{a} \succeq_{\mathbb{A}} \vec{b} &\text{ iff } \vec{a} - \vec{b} \in \mathbb{A} \\ \vec{a} \succ_{\mathbb{A}} \vec{b} &\text{ iff } \vec{a} - \vec{b} \in \operatorname{int}(\mathbb{A}) \end{aligned} \tag{1}$$

and $\operatorname{int}(\mathbb{A})$ denotes the interior of $\mathbb{A}$.

3. Background

We first present our conic-segmentation SVM (CS-SVM) below, and then formulate our proposed CS++-SVM.

3.1 Multiclass Classification and the CS-SVM

The multiclass (n class) classification problem may be stated as follows: given a training set $\Theta = \{ (x_i, y_i) | \, i \in \mathbb{Z}_N \}$, where $x_i \in \mathbb{X}$ is the i^{th} input vector and $y_i \in \mathbb{Z}_n$ its classification, construct a classifier $h : \mathbb{X} \to \mathbb{Z}_n$ that captures the pair-wise relationship between input space and classification sampled by the training set.

This Section describes the conic-segmentation SVM (CS-SVM) originally presented in Shilton et al. (2012). The key characteristics of the CS-SVM are:

- The trained machine is selected using the principles of structural risk minimisation.
- The trained machine maps from feature space to d_T-dimensional target space rather than onto the real line.
- Target space is divided into proper-conic class regions, allowing the training constraints to be written in the form of generalised inequalities.

- The dual training problem uses the kernel trick to circumvent the curse of dimensionality.

The CS-SVM classifier has the form:

$$\mathbb{X} \underbrace{\xrightarrow{\vec{\varphi}} \mathbb{R}^{d_K} \xrightarrow{\mathbf{g}} \mathbb{R}^{d_T} \xrightarrow{\sigma}}_{h=\sigma\circ\mathbf{g}\circ\vec{\varphi}} \mathbb{Z}_n \tag{2}$$

where $\mathbb{X}$ is input space, $\vec{\varphi} : \mathbb{X} \to \mathbb{R}^{d_K}$ is the feature map, $\mathbb{R}^{d_K}$ is feature space, $\mathbf{g} : \mathbb{R}^{d_K} \to \mathbb{R}^{d_T}$ is the trained machine, $\mathbb{R}^{d_T}$ is target space, $\sigma : \mathbb{R}^{d_T} \to \mathbb{Z}_n$ is the classing function, and $\mathbb{Z}_n$ is the set of class labels. Both the feature map and the classing function are selected a-priori. The trained machine is a linear vector-valued function:

$$\mathbf{g}(\vec{a}) = \textstyle\sum_m \mathbf{w}_m a_m + \mathbf{b} \tag{3}$$

where the weight vectors $\mathbf{w}_m \in \mathbb{R}^{d_T}$ ($m \in \mathbb{Z}_{d_K}$ throughout) and bias vector $\mathbf{b} \in \mathbb{R}^{d_T}$ are selected during training. The generalised SVM multiclass decision function is therefore:

$$h(x) = \sigma\left(\textstyle\sum_m \mathbf{w}_m \varphi_m(x) + \mathbf{b}\right)$$

where $\varphi_m : \mathbb{X} \to \mathbb{R}$ is the m^{th} element of the feature map $\vec{\varphi} : \mathbb{X} \to \mathbb{R}^{d_K}$. Note that if $d_T = 1$, $n = 2$ and $\sigma = \text{sgn}$ then (2) reduces to the standard binary SVM classifier.

As usual, the feature map is defined implicitly using the usual kernel trick. To define the classing function each class s ($s, t \in \mathbb{Z}_n$ throughout) is associated with a proper conic subset $\mathbb{G}_s \subset \mathbb{R}^{d_T}$ of target space known as a *class region* such that the set of all class regions forms an almost-everywhere (Cohn, 1980) non-intersecting target-space covering (Munkres, 1999; Armstrong, 1979). The classing function labels points in target space on the basis of which class region they fall into. As the class regions are proper cones the classing function may be written in terms of generalised inequalities (Boyd and Vandenberghe, 2004) as follows:

$$\sigma(\mathbf{a}) \;=\; s \,|\, \mathbf{a} \succ_{\mathbb{G}_s} \mathbf{0} \tag{4}$$

or, equivalently, as $\sigma(\mathbf{a}) = s \,|\, \mathbf{a} \in \text{int}(\mathbb{G}_s)$.

Figure 1 shows an example of the operation of the CS-SVM classifier (2) for the case $\mathbb{X} = \mathbb{R}^2$, $d_K = d_T = 2$. Examples in input space (left) are mapped to feature space (centre) using the feature map $\vec{\varphi}$ and from there to target space (right) by the trained machine $\mathbf{g}$. The class regions are fixed in target space while their pre-images $\widetilde{\mathbb{G}}_s = \mathbf{g}^{-1}(\mathbb{G}_s)$ in feature space and $\vec{\varphi}^{-1}(\widetilde{\mathbb{G}}_s)$ in input space are defined by the trained machine $\mathbf{g}$ and the feature map $\vec{\varphi}$.

The trained machine $\mathbf{g} : \mathbb{R}^{d_K} \to \mathbb{R}^{d_T}$ is found by minimising the regularised risk function subject to the separability conditions (Shilton et al., 2012) (the primal CS-SVM training problem):

$$\begin{array}{ll} \min\limits_{\mathbf{w}_m, \mathbf{b}, \boldsymbol{\xi}_i} & R = \frac{1}{2}\sum_m \mathbf{w}_m^T \mathbf{w}_m + \frac{C}{N}\sum_i \mathbf{u}_{y_i}^T \boldsymbol{\xi}_i \\ \text{such that:} & \sum_m \mathbf{w}_m \varphi_m(x_i) + \mathbf{b} \succeq_{\mathbb{G}_{y_i}} \mathbf{u}_{y_i} - \boldsymbol{\xi}_i \;\forall i \\ & \boldsymbol{\xi}_i \succeq_{\mathbb{G}_{y_i}} \mathbf{0} \;\forall i \end{array} \tag{5}$$

where the class centres $\mathbf{u}_0, \mathbf{u}_1, \ldots$ are defined a-priori such that $\mathbf{u}_s \in \text{int}(\mathbb{G}_s \cap \mathbb{G}_s^*)$ for all s (equivalently $\mathbf{u}_s \succ_{\mathbb{G}_s} \mathbf{0}$ and $\mathbf{u}_s \succ_{\mathbb{G}_s^*} \mathbf{0}$ for all s) as described in Section 3.2. The constraints

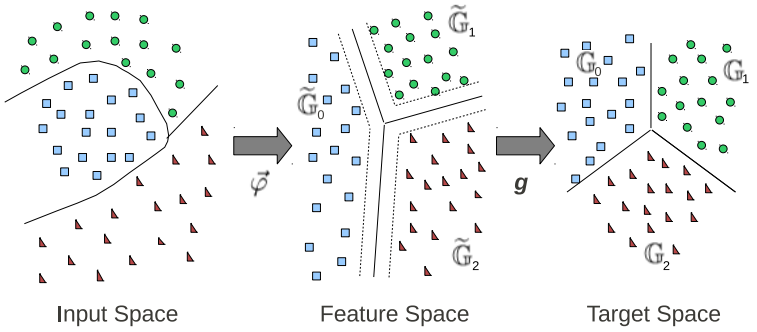

Figure 1: Input, feature and target space for the CS-SVM. Points in input space (left) are mapped to feature space (centre) by the pre-defined non-linear feature map $\vec{\varphi}$. The trained machine $\mathbf{g}$ then maps points from feature space to target space (right). The class regions $\mathbb{G}_s$ and their boundaries $\mathbb{M}_s$ are fixed in target space, while their pre-images $\widetilde{\mathbb{G}}_s = \mathbf{g}^{-1}(\mathbb{G}_s)$ and $\widetilde{\mathbb{M}}_s = \mathbf{g}^{-1}(\mathbb{M}_s)$ in feature space are defined by the trained machine $\mathbf{g}$.

in (5) ensure that $h(x_i) = y_i$ for all i for which the associated slack vector $\boldsymbol{\xi}_i$ is zero. Note that the first term in R is inversely proportional to a lower bound on the margin of separation[2] (Shilton et al., 2012), while the second term is a measure of empirical risk (training error), making R a trade-off between margin-maximisation and empirical risk minimisation (ERM) like the standard SVM primal. If C is large then ERM will tend to dominate, whereas if C is small then margin maximisation will come to the fore.

It is convenient to re-write the CS-SVM training problem in dual form, namely (Shilton et al., 2012):

$$\begin{array}{ll} \min\limits_{\boldsymbol{\alpha}_i} Q = \frac{1}{2}\sum_{i,j} K_{ij}\boldsymbol{\alpha}_i^T\boldsymbol{\alpha}_j - \sum_i \boldsymbol{\alpha}_i^T\mathbf{u}_{y_i} & \\ \text{such that:} \quad \frac{C}{N}\mathbf{u}_{y_i} \succeq_{\mathbb{G}^*_{y_i}} \boldsymbol{\alpha}_i \succeq_{\mathbb{G}^*_{y_i}} \mathbf{0}\ \forall i & \\ \sum_i \boldsymbol{\alpha}_i = \mathbf{0} & \end{array} \tag{6}$$

where $K_{ij} = K(x_i, x_j)$ and $K : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ is the Mercer kernel (Mercer, 1909; Cochran, 1972) associated with the feature map $\vec{\varphi}$. The trained machine may be written (Shilton et al., 2012):

$$\mathbf{g}(\vec{\varphi}(x)) = \sum_i K(x, x_i)\,\boldsymbol{\alpha}_i + \mathbf{b}$$

As usual, through use of the kernel trick (Herbrich, 2002; Cristianini and Shawe-Taylor, 2005; Shawe-Taylor and Cristianini, 2004; Steinwart and Christman, 2008) it is possible to train and use the CS-SVM without explicit reference to feature space or the feature map. The dual is a convex quadratic programming problem with no non-global minima. The recursive division form of the dual reduces to the standard (binary) SVM dual if $d_T = 1$ and $n = 2$ (though class labels are $0, 1$ in this case rather than $-1, +1$).

2. The margin of separation is defined as the minimum perpendicular distance between the image $\vec{\varphi}(x_i)$ of a training vector x_i in feature space and the boundary $\mathrm{bd}(\widetilde{\mathbb{G}}_{y_i})$ of the relevant class region inverse image $\widetilde{\mathbb{G}}_{y_i} = \mathbf{g}^{-1}(\mathbb{G}_{y_i})$ in feature space.

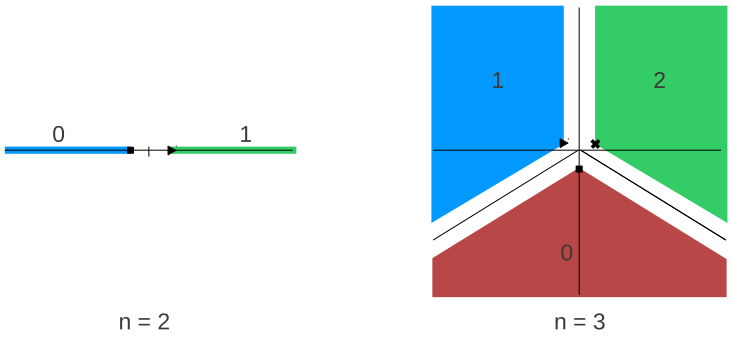

Figure 2: Binary ($n = 2$) and ternary ($n = 3$) class regions (in target space) for the recursive division class regions. In the binary case class centres $\mathbf{u}_{(2);0}$ and $\mathbf{u}_{(2);1}$ are represented as ■ and ▶, respectively, and in the ternary case $\mathbf{u}_{(3);0}$, $\mathbf{u}_{(3);1}$ and $\mathbf{u}_{(3);2}$ are shown as ■, ▶ and ×.

3.2 Class Region Selection

The CS-SVM provides a very flexible framework for multiclass classification. By appropriate selection of the class regions and centres it may be shown (Shilton et al., 2012) that both the max-wins SVM (Weston and Watkins, 1999; Crammer and Singer, 2001) and the reduction to binary SVM (Allwein et al., 2000), which encompasses 1vsA (Vapnik, 1995), 1vs1 (Hastie and Tibshirani, 1998), error-correcting output-coding ECOC (Diehl and Cauwenberghs, 2003) and minimum output coding (MOC) (Suykens et al., 2002), are derivable as special cases of the CS-SVM.

The present paper will focus exclusively on the recursive division form of the CS-SVM (Shilton et al., 2012). Under this scheme the class centres are defined to be the vertices of a regular $(n-1)$-simplex in $d_T = n-1$ dimensional target space. The class regions are derived from the class centres using the *max-projection* principle (Shilton et al., 2012), namely:

$$\mathbb{G}^{\mathrm{RD}}_{(n);s} = \left\{ \mathbf{a} \middle| \, \mathbf{u}^T_{(n);s}\mathbf{a} \geq \mathbf{u}^T_{(n);t}\mathbf{a} \; \forall t \neq s \right\} \tag{7}$$

and moreover it may be shown that the dual class regions are given by:

$$\mathbb{G}^{\mathrm{RD}*}_{(n);s} = \left\{ \mathbf{a} \middle| \, -\mathbf{u}^T_{(n);t}\mathbf{a} \geq 0 \; \forall t \neq s \right\} \tag{8}$$

where the class centres are defined recursively by (Shilton et al., 2012):

$$\begin{aligned} \mathbf{u}_{(n);0} &= \begin{bmatrix} \mathbf{0} \\ -1 \end{bmatrix} \in \mathbb{R}^{d_T} \\ \mathbf{u}_{(n);q+1} &= \tfrac{1}{n-1} \begin{bmatrix} \mathbf{u}_{(n-1);q}\sqrt{n\,(n-2)} \\ 1 \end{bmatrix} \in \mathbb{R}^{d_T} \\ \mathbf{u}_{(2);0} &= \begin{bmatrix} -1 \end{bmatrix} \quad , \quad \mathbf{u}_{(2);1} = \begin{bmatrix} +1 \end{bmatrix} \end{aligned} \tag{9}$$

($q, r \in \mathbb{Z}_{d_T}$ throughout, $d_T = n-1$ in the case of recursive division) where the subscript (n) is used to indicate the number of classes the particular class centre vector is intended for

use with. The binary and ternary class regions for the this scheme are shown in Figure 2. Note that (Shilton et al., 2012):

$$\mathbf{u}_{(n);s}^T \mathbf{u}_{(n);t} = \begin{cases} 1 & \text{if } s = t \\ -\frac{1}{n-1} & \text{otherwise} \end{cases} \tag{10}$$

For the recursive division case the dual CS-SVM training problem (6) reduces to:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}_i} Q = \tfrac{1}{2} \textstyle\sum_{i,j} K_{ij} \boldsymbol{\alpha}_i^T \boldsymbol{\alpha}_j - \sum_i \boldsymbol{\alpha}_i^T \mathbf{u}_{(n);y_i} \\ &\text{such that:} \quad -\tfrac{1}{n-1}\tfrac{C}{N} \leq \mathbf{u}_{(n);s}^T \boldsymbol{\alpha}_i \leq 0 \; \forall i, s \neq y_i \\ &\qquad\qquad \textstyle\sum_i \boldsymbol{\alpha}_i = \mathbf{0} \end{aligned} \tag{11}$$

and moreover the trained machine may be written:

$$h(x) = \operatorname*{argmax}_s \left\{ \mathbf{u}_{(n);s}^T \left(\textstyle\sum_i K(x, x_i) \boldsymbol{\alpha}_i + \mathbf{b} \right) \right\} \tag{12}$$

3.3 Anomaly Detection and the 1-class SVM

The anomaly detection problem may be stated as follows: assuming that the training vectors x_0, x_1, ... are generated from a distribution P, find a trained machine $h : \mathbb{X} \to \{-1, +1\}$ that is negative for most points generated from P and positive elsewhere.[3]

Several 1-class SVM approaches have been proposed for anomaly detection (Schölkopf et al., 2001; Tax and Duin, 2004; Laskov et al., 2004; Wang et al., 2006; Shilton et al., 2015). As with most SVM methods, the general approach of these is to first (implicitly) map the data vectors from the input space to the feature space by means of a non-linear function. A smooth surface or boundary is found in the feature space that separates the image vectors into normal and anomalous measurements. By using kernel trick, the data vectors are implicitly mapped to a higher dimensional *inner product* space without knowledge of the mapping function. The boundaries found in the implicit feature space usually yield a non-linear boundary in input space (Schölkopf and Smola, 2002).

The 1-class SVM of (Schölkopf et al., 2001; Manevitz and Yousef, 2001) formulates the problem as one of quadratic programming, specifically:[4]

$$\begin{aligned} &\min_{w_m, \rho, \xi_i} \quad R = \tfrac{1}{2} \textstyle\sum_m w_m^2 - \frac{1}{\nu N} \sum_i \xi_i + \rho \\ &\text{such that:} \quad \textstyle\sum_m w_m \varphi_m(x_i) \leq \rho - \xi_i \; \forall i \\ &\qquad\qquad \xi_i \leq 0 \; \forall i \end{aligned} \tag{13}$$

or, in dual form:

$$\begin{aligned} &\min_{\alpha_i} Q = \tfrac{1}{2} \textstyle\sum_{i,j} K_{ij} \alpha_i \alpha_j \\ &\text{such that:} \quad -\tfrac{1}{\nu N} \leq \alpha_i \leq 0 \; \forall i \\ &\qquad\qquad \textstyle\sum_i \alpha_i = -1 \end{aligned} \tag{14}$$

3. Normally the 1-class SVM uses the reverse sign convention - +1 for points generated by P, −1 for anomalies - but this will cause inconvenience later and for that reason is not used in the present paper.
4. In addition to reversing the usual sign convention and labelling anomalous vectors +1 the sign of the slack variables ξ_i is also reversed here for consistency with the CS++-SVM introduced in Section 4.

where $\nu \in (0,1)$ is a constant selected a-priori. The trained machine is then:

$$h(x) = \operatorname{sgn}\left(\sum_i \alpha_i K(x, x_i) - \rho\right)$$

where $h(x) = +1$ implies that x is an anomaly.

To understand the geometry underlying the 1-class SVM assume a radial kernel:

$$K(x, y) = \kappa(\|x - y\|)$$

This corresponds to a feature map $\vec{\varphi}$ that maps input space $\mathbb{X}$ onto a hypersphere of radius $\sqrt{\kappa(0)}$ centred at the origin in feature space (Schölkopf and Smola, 2001). The 1-class SVM separates the images of all training vectors on this hypersphere from the origin, maximising the separation between this hyperplane and the origin, thereby minimising the size of the fraction of the hypersphere labelled -1 (not anomalous). In input space $\mathbb{X}$ this may be visualised as a a set of "islands" of class -1 (centred on the support vectors) rising above an anomalous "ocean". As shown in Section 5.3, our method emulates this approach in a more general manner.

It should be noted that the variable ρ in the 1-class SVM performs a role directly analogous to the bias term b in the standard binary SVM. Indeed the primal training problem (13) may be obtained (Shilton and Palaniswami, 2008) from the standard binary SVM by:

1. Substituting $b = -\rho$ and $C = \frac{1}{\nu}$.
2. Labelling all examples as (nominally) class $y_i = -1$.
3. Removing the constant 1 in the first constraint in the primal training set, so:

   $$y_i(\textstyle\sum_m w_m \varphi_m(x_i) + b) \geq 1 - \xi_i$$

   is replaced by:

   $$\textstyle\sum_m w_m \varphi_m(x_i) \leq \rho - \xi_i$$

4. Adding a *bias forcing term* ρ to the primal, so:

   $$R = \tfrac{1}{2}\textstyle\sum_m w_m^2 - \frac{C}{N}\sum_i \xi_i$$

   becomes:

   $$R = \tfrac{1}{2}\textstyle\sum_m w_m^2 - \frac{1}{\nu N}\sum_i \xi_i + \rho$$

Considering the parameter $\nu \in (0,1)$, it may be shown that if $\rho > 0$ then, after training (Schölkopf et al., 2001):

- ν is an upper bound on the fraction of outliers (training vectors x_i for which $h(x_i) = +1$).
- ν is a lower bound for the fraction of support vectors (training vectors x_i for which $\alpha_i < 0$).
- If the distribution P does not contain discrete components then asymptotically, with probability 1, ν equals both the fraction of outliers and the fraction of support vectors.

3.3.1 Related work

Anomaly detection has become an important challenge in many applications. Surveys, such as Ruff et al. (2020); Chandola et al. (2009); Chalapathy and Chawla (2019); Pang et al. (2020); Bulusu et al. (2020); O'Rielly et al. (2014); Rajasegarar et al. (2008), provide good analysis on the latest trends in different domains. One class methods are one of the popular methods used for anomaly detection. Here we briefly review a number of alternative variations on the 1-class SVM methods proposed in the literature. Campbell and Bennett (2001) formulated a linear programming approach for the one-class SVM when used with a radial basis function (RBF) kernel. This formulation is based on attracting the hyperplane towards the average of the image data distribution, rather than minimising the maximum norm distance to the bounding hyperplane from the origin as in Schölkopf et al. (2001). Shilton et al. (2015) proposed an incremental/decremental learning based 1-class SVM.

Tax and Duin (2004) formulated the one-class SVM using a hypersphere. In this approach, a minimal radius hypersphere is fixed around the majority of the image vectors in the feature space. The data that falls outside the hypersphere are identified as anomalous. This hypersphere formulation requires quadratic programming optimisation. Wang et al. (2006) formulated the one-class SVM using hyper-ellipsoids with minimum *effective radii* around a majority of the image vectors in the feature space. This hyper-ellipsoidal formulation involves two phases. First the image vectors are partitioned into a number of distinct clusters using Ward's linkage algorithm (Ward, 1963). Second, the image vectors in each cluster are fixed with a hyper-ellipsoid that encapsulates a majority of the image vectors in that cluster. The image vectors that do not fall within any of the hyper-ellipsoids are identified as anomalous. This problem is formulated as a second order cone programming optimisation problem.

Laskov et al. (2004) have observed the one-sidedness of the data distribution in many practical applications, and exploited this property to develop a special type of SVM called a one-class quarter-sphere SVM (QSSVM). This is extended from the hypersphere-based one-class SVM approach proposed by Tax and Duin (2004). The QSSVM finds a minimal radius hypersphere centred at the origin that encapsulates a majority of the image vectors in the feature space (Laskov et al., 2004). Data vectors that fall outside the quarter sphere are classified as anomalies. This problem is formulated as a linear programming problem.

The above methods have focused on the one class detection problem, and do not incorporate the multiclass detection within the framework. Below, a multiclass anomaly detector is proposed that combines the 1-class SVM and the CS-SVM.

4. The CS++-SVM

The combined multiclass ($n-1$ class) classification and anomaly detection problem (the combined problem) is thus: given a training set $\Theta = \{(x_i, y_i) | \, i \in \mathbb{Z}_N\}$, where $x_i \in \mathbb{X}$ is the i^{th} input vector and $y_i \in \mathbb{Z}_{n-1}$ is its classification, and assuming that training samples for each class 0, 1, …, $n-2$ are generated from distributions P_0, P_1, …, P_{n-2}, respectively, construct a trained machine that can correctly label most points generated from P_0 as class 0, P_1 as class 1, …, P_{n-2} as class $n-2$, and label most other points (those points probably not generated by P_0, P_1, …, P_{n-2}) as class $n-1$, which is defined to be the *anomaly* class.

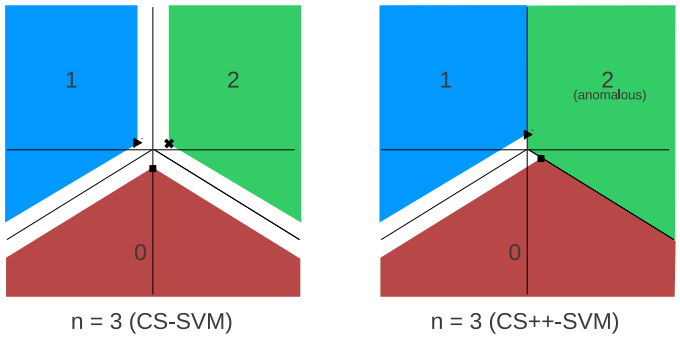

Figure 3: Comparison of class regions and centres (in target space) for the CS-SVM (left) and CS++-SVM (right) for the case $n = 3$. In this figure $\mathbf{u}_{(3);0}$, $\mathbf{u}_{(3);1}$ and $\mathbf{u}_{(3);2}$ ($\tilde{\mathbf{u}}_{(3);0}$ and $\tilde{\mathbf{u}}_{(3);1}$ only in the CS++-SVM case) are shown as ■, ▶ and ×, respectively. Note that the modified class centres in the CS++-SVM case have been shifted to lie on the boundary of the anomaly class $n-1$.

As noted in the introduction, examples that fit this problem model include face-in-a-crowd detection where only a small subset of faces are of interest,; computer network intrusion detection where a system should be able to both recognise normal traffic and known attacks and also flag anomalous traffic that may represent new, novel attacks; and emerging IoT applications where it is probable that new classes will emerge as time passes.

4.1 The CS++-SVM Primal

The basis of the CS++-SVM method is as follows: beginning with the CS-SVM formulation (5) for the recursive division case, following the heuristic basis of the 1-class SVM as explained in Section 3.3:

1. Replace the class centres $\mathbf{u}_{(n):s}$ of the CS-SVM with shifted class centres $\tilde{\mathbf{u}}_{(n);s}$ that remove any separation between class $n-1$ (the anomaly class) and all other classes $0, 1, \ldots, n-2$, as shown in Figure 3.

2. Add a bias-forcing term, $-C\mathbf{u}_{(n);n-1}^T\mathbf{b}$, analogous to the ρ term in the 1-class SVM primal.

3. Reserve the class label $n-1$ for anomalies (that is, no training vectors in class $n-1$).

4. Add class-wise scaling factors to the empirical risk to ensure that each class is equally affected by the bias-forcing term when the number of training vectors in each class differs.

The CS++-SVM primal is:

$$\begin{array}{ll}\min\limits_{\mathbf{w}_m,\mathbf{b},\boldsymbol{\xi}_i} R = \frac{1}{2}\sum\limits_m \mathbf{w}_m^T\mathbf{w}_m + C\left(\frac{1}{\nu}\sum\limits_s \frac{1}{N_s}\sum\limits_{i:y_i=s}\mathbf{u}_{(n);y_i}^T\boldsymbol{\xi}_i - \mathbf{u}_{(n);n-1}^T\mathbf{b}\right) \\ \text{such that:} \quad \sum_m \mathbf{w}_m\varphi_m(x_i) + \mathbf{b} \succeq_{\mathbb{G}^{\mathrm{RD}}_{(n);y_i}} \tilde{\mathbf{u}}_{(n);y_i} - \boldsymbol{\xi}_i \ \forall i \\ \qquad\qquad \boldsymbol{\xi}_i \succeq_{\mathbb{G}^{\mathrm{RD}}_{(n);y_i}} \mathbf{0} \ \forall i\end{array} \tag{15}$$

where:

- N_s is the number of training vectors in class $s \in \mathbb{Z}_{n-1}$.
- $\nu \in (0,1]$ and $C \in \mathbb{R}^+$ are training constants.
- Class regions $\mathbb{G}^{\mathrm{RD}}_{(n);s}$ and class centres $\mathbf{u}_{(n);s}$ are defined by recursive division (7), (9).
- For all $s \in \mathbb{Z}_{n-1}$ (using (10)):

$$\begin{aligned}\tilde{\mathbf{u}}_{(n);s} &= \frac{\mathbf{u}_{(n);s}+\mathbf{u}_{(n);n-1}}{\left\|\mathbf{u}_{(n);s}+\mathbf{u}_{(n);n-1}\right\|} \\ &= \frac{1}{\sqrt{2}}\sqrt{\frac{n-1}{n-2}}\left(\mathbf{u}_{(n);s}+\mathbf{u}_{(n);n-1}\right)\end{aligned} \tag{16}$$

  are defined to be the shifted class centres, where it may be seen that $\tilde{\mathbf{u}}_{(n);s} \succeq_{\mathbb{G}^{\mathrm{RD}}_{(n);s}} \mathbf{0}$ for all $s \in \mathbb{Z}_{n-1}$.

Note that the CS++-SVM primal training problem (15) reduces to the 1-class SVM primal training problem (13) in the case $n = 2$, $C = 1$.

The influence of the training constants ν and C are investigated in Section 5. It is shown that ν acts, independently of C, as an upper bound on the class-wise average fraction of exceptional outliers and a lower bound on the class-wise average fraction of exceptional support vectors, and is asymptotically equal to both under certain assumptions. The constant C performs the same task here as in the CS-SVM, namely controlling the trade-off between empirical risk minimisation (vis-a-vis misclassification of training examples) and regularisation. Thus ν controls the anomaly detection characteristics of the CS++-SVM while C controls the classification characteristics.

4.2 The CS++-SVM Dual

As is usually the case with SVM methods it is infeasible to solve the CS++-SVM primal training problem (15) directly due to the curse of dimensionality if d_K is large (or infinite). To overcome this the problem must be re-written in dual form.

Define Lagrange multiplier vectors $\boldsymbol{\alpha}_i, \boldsymbol{\gamma}_i \succeq_{\mathbb{G}^{\mathrm{RD}*}_{y_i}} \mathbf{0}$ for the first two constraint sets in (15), respectively. The Lagrangian may be written:

$$\begin{aligned}\mathcal{L} = \ & \tfrac{1}{2}\textstyle\sum_m \mathbf{w}_m^T\mathbf{w}_m + C\frac{1}{\nu}\sum_i \frac{1}{N_{y_i}}\mathbf{u}_{(n);y_i}^T\boldsymbol{\xi}_i - C\mathbf{u}_{(n);n-1}^T\mathbf{b} - \ldots \\ & - \textstyle\sum_{i,m}\boldsymbol{\alpha}_i^T\mathbf{w}_m\varphi_m(x_i) - \sum_i \boldsymbol{\alpha}_i^T\mathbf{b} + \sum_i \boldsymbol{\alpha}_i^T\tilde{\mathbf{u}}_{(n);y_i} - \sum_i \boldsymbol{\alpha}_i^T\boldsymbol{\xi}_i - \ldots \\ & - \textstyle\sum_i \boldsymbol{\gamma}_i^T\boldsymbol{\xi}_i\end{aligned} \tag{17}$$

Applying optimality conditions:

$$\begin{array}{rcl} \frac{\partial \mathcal{L}}{\partial \mathbf{w}_m} & = & \mathbf{w}_m - \sum_i \boldsymbol{\alpha}_i \varphi_m(x_i) = \mathbf{0} \; \forall m \\ \frac{\partial \mathcal{L}}{\partial \mathbf{b}} & = & -C\mathbf{u}_{(n);n-1} - \sum_i \boldsymbol{\alpha}_i = \mathbf{0} \\ \frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi}_i} & = & \frac{1}{\nu}\frac{C}{N_{y_i}}\mathbf{u}_{(n);y_i} - \boldsymbol{\gamma}_i - \boldsymbol{\alpha}_i = \mathbf{0} \; \forall i \end{array}$$

it may be seen that:

$$\begin{array}{l} \mathbf{w}_m = \sum_i \boldsymbol{\alpha}_i \varphi_m(x_i) \; \forall m \\ \sum_i \boldsymbol{\alpha}_i = -C\mathbf{u}_{(n);n-1} \\ \frac{1}{\nu}\frac{C}{N_{y_i}}\mathbf{u}_{(n)y_i} \succeq_{\mathbb{G}^{\mathrm{RD}*}_{(n);y_i}} \boldsymbol{\alpha}_i \; \forall i \end{array}$$

and hence the dual CS++-SVM training problem is:

$$\begin{array}{ll} \min\limits_{\boldsymbol{\alpha}_i} Q = & \frac{1}{2}\sum_{i,j} K_{ij}\boldsymbol{\alpha}_i^T\boldsymbol{\alpha}_j - \sum\limits_i \boldsymbol{\alpha}_i^T\tilde{\mathbf{u}}_{(n);y_i} \\ \text{such that:} & \frac{1}{\nu}\frac{C}{N_{y_i}}\mathbf{u}_{(n);y_i} \succeq_{\mathbb{G}^{\mathrm{RD}*}_{y_i}} \boldsymbol{\alpha}_i \succeq_{\mathbb{G}^{\mathrm{RD}*}_{y_i}} \mathbf{0} \; \forall i \\ & \sum_i \boldsymbol{\alpha}_i = -C\mathbf{u}_{(n);n-1} \end{array} \tag{18}$$

or, explicitly, using (9):

$$\begin{array}{ll} \min\limits_{\boldsymbol{\alpha}_i} Q = & \frac{1}{2}\sum_{i,j} K_{ij}\boldsymbol{\alpha}_i^T\boldsymbol{\alpha}_j - \sum\limits_i \boldsymbol{\alpha}_i^T\tilde{\mathbf{u}}_{(n);y_i} \\ \text{such that:} & -\frac{1}{\nu}\frac{1}{n-1}\frac{C}{N_{y_i}} \leq \mathbf{u}_{(n);s}^T\boldsymbol{\alpha}_i \leq 0 \; \forall i, s \neq y_i \\ & \sum_i \boldsymbol{\alpha}_i = -C\mathbf{u}_{(n);n-1} \end{array} \tag{19}$$

Note that (19) reduces to the 1-class SVM dual training problem (14) in the special case $n = 2$, $C = 1$.

As for the CS-SVM, the trained machine may be written:

$$\mathbf{g}(\vec{\varphi}(x)) = \sum_i K(x, x_i)\boldsymbol{\alpha}_i + \mathbf{b}$$

and hence using (9):

$$h(x) = \underset{s}{\operatorname{argmax}} \left\{ \mathbf{u}_{(n);s}^T \left(\sum_i K(x, x_i)\boldsymbol{\alpha}_i + \mathbf{b} \right) \right\}$$

recalling that class $n-1$ is used to label anomalies and classes 0, 1, …, $n-2$ are known classes. Note that, like the CS-SVM, the CS++-SVM may be both trained and used without reference to the primal weight vectors $\mathbf{w}_0, \mathbf{w}_1, \ldots \in \mathbb{R}^{d_K}$ or the explicit feature map $\vec{\varphi} : \mathbb{X} \to \mathbb{R}^{d_K}$, allowing the use of very high or even infinite dimensional feature maps. Note also that the dual training problem is a convex quadratic programming problem with generalised constraints and hence has no non-global minima.

5. Properties of the CS++-SVM

In this section the theoretical properties of the CS++-SVM are investigated. These may be loosely broken into properties relating to the operation of the CS++-SVM as a multiclass classifier and properties relating to its operation as an anomaly detector. Finally, some comparisons are made with the 1-class SVM.

5.1 Classification Properties

The aim of this section is to extend the concepts of separability, the margin of separation to the CS++-SVM, and the connection between regularisation and maximising the margin of separation from the standard SVM framework to the CS++-SVM. After the underlying definitions have been clarified it will be shown that the CS++-SVM maximises the margin of separation (suitably defined) on all class boundaries *except* the boundary of the anomaly class, which will in fact be minimised (in line with the 1-class SVM).

As in the previous section, let Θ be a training set containing examples from classes $0, 1, \ldots, n-2$ but not class $n-1$.

Definition 1 *A training set Θ is said to be* strictly separable *under feature map $\vec{\varphi}$ if there exists $\mathbf{w}_0$, $\mathbf{w}_1$, ... and $\mathbf{b}$ such that:*

$$\mathbf{g}(x_i) = \sum_m \mathbf{w}_m \varphi_m(x_i) + \mathbf{b} \succ_{\mathbb{G}^{\mathrm{RD}}_{(n);y_i}} \mathbf{0} \; \forall i \tag{20}$$

Moreover $\mathbf{w}_0$, $\mathbf{w}_1$, ... and $\mathbf{b}$ are said to strictly separate *a training set Θ if the above condition is met.*

Definition 2 *A training set Θ is said to be* intra-class separable *under feature map $\vec{\varphi}$ if there exists $\mathbf{w}_0$, $\mathbf{w}_1$, ... and $\mathbf{b}$ such that:*

$$\mathbf{g}(x_i) = \sum_m \mathbf{w}_m \varphi_m(x_i) + \mathbf{b} \succeq_{\mathbb{G}^{\mathrm{RD}}_{(n);y_i}} \tilde{\mathbf{u}}_{(n);y_i} \; \forall i \tag{21}$$

Moreover $\mathbf{w}_0$, $\mathbf{w}_1$, ... and $\mathbf{b}$ are said to intra-class separate *a training set Θ if the above condition is met.*

These two definitions of separability are not equivalent. Combining (16), (10) and (7) it may be seen that $\tilde{\mathbf{u}}_{(n);s} \succeq_{\mathbb{G}^{\mathrm{RD}}_{(n);s}} \mathbf{0}$ but $\tilde{\mathbf{u}}_{(n);s} \not\succ_{\mathbb{G}^{\mathrm{RD}}_{(n);s}} \mathbf{0}$ for $s \in \mathbb{Z}_{n-1}$ and hence (21) does not imply (20). Indeed, using (7), it may be seen that (using the notation of Figure 1):

- If a training set Θ is strictly separated by $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{b}$ then all images $\vec{\varphi}(x_i)$ of training vectors x_i in feature space will lie in the interior of the relevant class region pre-image $\tilde{\mathbb{G}}^{\mathrm{RD}}_{(n);y_i} = \mathbf{g}^{-1}(\mathbb{G}^{\mathrm{RD}}_{(n);y_i})$ in feature space.
- If a training set Θ is intra-class separated by $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{b}$ then all images $\vec{\varphi}(x_i)$ of training vectors x_i in feature space will lie in the interior of the relevant class region pre-image $\tilde{\mathbb{G}}^{\mathrm{RD}}_{(n);y_i} = \mathbf{g}^{-1}(\mathbb{G}^{\mathrm{RD}}_{(n);y_i})$ in feature space *or on the boundary between $\tilde{\mathbb{G}}^{\mathrm{RD}}_{(n);y_i}$ and $\tilde{\mathbb{G}}^{\mathrm{RD}}_{(n);n-1}$*.

That is, intra-class separation allows $\vec{\varphi}(x_i)$ to lie on the boundary of the anomaly class whereas strict separation does not. Note that the training constraints of the CS++-SVM primal (15) reduce to the intra-class separation constraints (21) if $\boldsymbol{\xi}_i = \mathbf{0}$ for all i. It follows that, in this case, the training set Θ is intra-class separated and the empirical risk component of the primal is zero, and the margin of separation for the CS++-SVM may be defined:

Definition 3 *The* intra-class margin of separation Δ_I *for a given* $\mathbf{w}_0$, $\mathbf{w}_1$, ..., $\mathbf{b}$ *intra-class separating the training set* Θ *with* $\boldsymbol{\xi}_i = \mathbf{0}$ *for all* i *under feature map* $\vec{\varphi}$ *is the minimum distance between the image* $\vec{\varphi}(x_i)$ *of any training vector* x_i *in feature space and any point on the boundary of the class region pre-image* $\widetilde{\mathbb{G}}^{\mathrm{RD}}_{(n);y_i} = \mathbf{g}^{-1}(\mathbb{G}^{\mathrm{RD}}_{(n);y_i})$ *in feature space excluding points on the boundary between class* y_i *and class* $n-1$.

Definition 4 *The* extra-class margin of separation Δ_E *for a given* $\mathbf{w}_0$, $\mathbf{w}_1$, ..., $\mathbf{b}$ *intra-class separating the training set* Θ *with* $\boldsymbol{\xi}_i = \mathbf{0}$ *for all* i *under feature map* $\vec{\varphi}$ *is the minimum distance between the image* $\vec{\varphi}(x_i)$ *of any training vector* x_i *in feature space and any point on the boundary between class* y_i *and class* $n-1$.

It can be shown (Shilton et al., 2012) (Appendix B) that the distance from any point $\vec{a} \in \widetilde{\mathbb{G}}^{\mathrm{RD}}_{(n);s} = \mathbf{g}^{-1}(\mathbb{G}^{\mathrm{RD}}_{(n);s})$ inside class region s to the boundary of that class region is:[5]

$$d\left(\vec{a}\right) = \min_{t|t\neq s}\left(\frac{\mathbf{v}^T_{(n);s,t}\left(\sum_m \mathbf{w}_m a_m + \mathbf{b}\right)}{\sqrt{\sum_m \mathbf{w}_m^T\left(\mathbf{v}_{(n);s,t}\mathbf{v}^T_{(n);s,t}\right)\mathbf{w}_m}}\right) \tag{22}$$

where each unit vector $\mathbf{v}_{(n);s,t}$ defines the separating hyperplane between class regions $\mathbb{G}^{\mathrm{RD}}_{(n);s}$ and $\mathbb{G}^{\mathrm{RD}}_{(n);t}$ in target space (so $\mathbf{v}^T_{(n);s,t}\mathbf{a} > 0$ for all $t \neq s, \mathbf{a} \in \mathrm{int}(\mathbb{G}^{\mathrm{RD}}_{(n);s})$). Explicitly, in the recursive division case:

$$\mathbf{v}_{(n);s,t} = \frac{\mathbf{u}_{(n);s} - \mathbf{u}_{(n);t}}{\left\|\mathbf{u}_{(n);s} - \mathbf{u}_{(n);t}\right\|_2} = \frac{1}{\sqrt{2}}\sqrt{\frac{n-1}{n}}\left(\mathbf{u}_{(n);s} - \mathbf{u}_{(n);t}\right) \tag{23}$$

It may be seen from (22) that for all $n > 2$:[6]

$$\begin{aligned}
\Delta_I &= \min_{i,s|s\neq y_i, s\neq n-1}\left(\frac{\mathbf{v}^T_{(n);y_i,s}\left(\sum_m \mathbf{w}_m\varphi_m(x_i) + \mathbf{b}\right)}{\sqrt{\sum_m \mathbf{w}_m^T\left(\mathbf{v}_{(n);y_i,s}\mathbf{v}^T_{(n);y_i,s}\right)\mathbf{w}_m}}\right) \\
\Delta_E &= \min_i\left(\frac{\mathbf{v}^T_{(n);y_i,n-1}\left(\sum_m \mathbf{w}_m\varphi_m(x_i) + \mathbf{b}\right)}{\sqrt{\sum_m \mathbf{w}_m^T\left(\mathbf{v}_{(n);y_i,n-1}\mathbf{v}^T_{(n);y_i,n-1}\right)\mathbf{w}_m}}\right)
\end{aligned}$$

Following the method of Shilton et al. (2012) let $\hat{\mathbf{w}}_m = \|\mathbf{w}_m\|_2^{-1}\mathbf{w}_m$ for all m. Using the training constraints of the primal CS++-SVM training problem (15) in the separable case $\boldsymbol{\xi}_i = \mathbf{0}$ for all i, applying Hölder's inequality (Gradshteyn and Ryzhik, 2000), recalling that $\mathbf{v}_{(n);s,t}$ are unit vectors by definition and using (23), (16) and (10):

$$\begin{aligned}
\Delta_I &\geq \frac{\min_{s,t|s\neq n-1, t\neq n-1, s\neq t}\left|\mathbf{v}^T_{(n);s,t}\tilde{\mathbf{u}}_{(n);s}\right|}{\sqrt{\sum_m(\mathbf{w}_m^T\mathbf{w}_m)}} = \frac{\frac{1}{2}\sqrt{\frac{n}{n-2}}}{\sqrt{\sum_m(\mathbf{w}_m^T\mathbf{w}_m)}} > 0 \\
\Delta_E &\geq \frac{\min_{s|s\neq n-1}\left|\mathbf{v}^T_{(n);s,n-1}\tilde{\mathbf{u}}_{(n);s}\right|}{\sqrt{\sum_m(\mathbf{w}_m^T\mathbf{w}_m)}} = 0
\end{aligned}$$

5. In Shilton et al. (2012) the notation $\mathbf{v}_{s,t;\varsigma}$ is used, where $\varsigma \in \mathbb{Z}_{d_{s,t}}$ and $1 \leq d_{s,t} = d_{t,s} \leq d_T$ for all all s, t. For the recursive division class regions, however, $d_{s,t} = 1$ for all s, t, rendering the ς index unnecessary.
6. Δ_I is ill-defined if $n = 2$.

It may be seen from this that minimising the first term $\sum_m \mathbf{w}_m^T \mathbf{w}_m$ in the primal CS++-SVM training problem (15) subject to the constraints will maximise a lower bound on the intra-class margin of separation Δ_I but not the extra-class margin of separation Δ_E. Likewise minimising the second term in (15) will tend to minimise the empirical risk (training error) associated, with the trade-off between the two factors controlled by the parameter $C \in \mathbb{R}^+$ in the usual manner. Thus the CS++-SVM, like the CS-SVM, applies a form of structural risk minimisation to the classification component of the combined problem by trading off empirical risk minimisation and margin maximisation.

5.2 Anomaly Detection Properties

The aim of this section is to investigate the influence the training parameter ν on the anomaly detection properties of the CS++-SVM.

Definition 5 *A training vector x_i is called a* support vector *(SV) if the corresponding Lagrange multiplier vector $\boldsymbol{\alpha}_i$ is non-zero. Support vectors are further divided into the following (non-exclusive) sub-categories:*

- *Intra-SV (ISV):* $\mathbf{u}_{(n);s}^T \boldsymbol{\alpha}_i < 0,\ s \neq n-1, s \neq y_i$.
- *Extra-SV (ESV):* $\mathbf{u}_{(n);n-1}^T \boldsymbol{\alpha}_i < 0$.

An error vector *(EV) is a support vector x_i for which $\mathbf{u}_{(n);s}^T \boldsymbol{\alpha}_i = -\frac{1}{\nu}\frac{1}{n-1}\frac{C}{N_{y_i}}$ for some $s \neq y_i$. Error vectors are further divided into the following (non-exclusive) sub-categories:*

- *Intra-EV (IEV):* $\mathbf{u}_{(n);s}^T \boldsymbol{\alpha}_i = -\frac{1}{\nu}\frac{1}{n-1}\frac{C}{N_{y_i}}, s \neq n-1, s \neq y_i$.
- *Extra-EV (EEV):* $\mathbf{u}_{(n);n-1}^T \boldsymbol{\alpha}_i = -\frac{1}{\nu}\frac{1}{n-1}\frac{C}{N_{y_i}}$.

Definition 6 *A training vector x_i is called an* anomaly vector *(AV) if the image $\vec{\varphi}(x_i)$ of x_i in feature space lies in the anomaly class region inverse image $\widetilde{\mathbb{G}}^{\mathrm{RD}}_{(n);n-1}$ in feature space. Anomaly vectors are further divided into:*

- *Boundary-AV (BAV):* $\vec{\varphi}(x_i) \in \mathrm{bd}(\widetilde{\mathbb{G}}^{\mathrm{RD}}_{(n);n-1})$.
- *Interior-AV (IAV):* $\vec{\varphi}(x_i) \in \mathrm{int}(\widetilde{\mathbb{G}}^{\mathrm{RD}}_{(n);n-1})$.

The number of support vectors is denoted N_{SV}, the number of anomaly vectors N_{AV} and so on for ISVs etc. Likewise, the number of support vectors in class $s \neq n-1$ is denoted $N_{\mathrm{SV};s}$ etc. The following theorem clarifies the connection between these definitions:

Theorem 7 *If x_i is an extra error vector but not an intra error vector then x_i is an anomaly vector. If x_i is an extra support vector but not an extra error vector or an intra error vector then x_i is a boundary anomaly vector.*

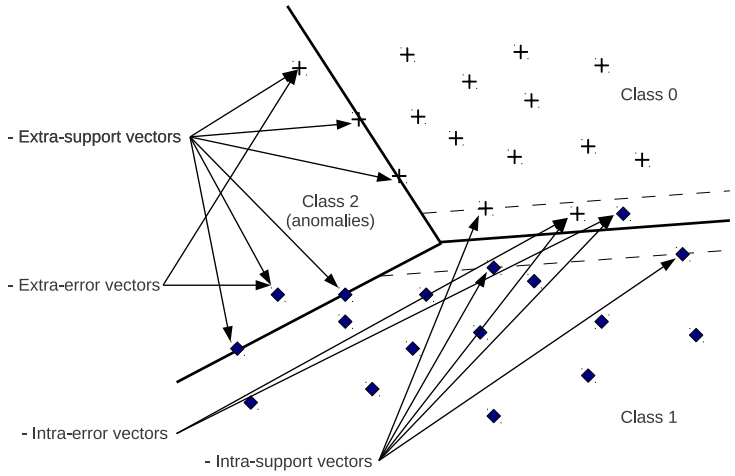

Figure 4: Placement of intra/extra support vectors and intra/extra error vectors in a 2-dimensional feature space in the case $n = 3$.

Proof See appendix A. ■

The placement of intra/extra support vectors and intra/extra error vectors is illustrated in Figure 4. The following theorems are the key to understanding the role of the parameter ν in the CS++-SVM. The first result represents an extension of the usual interpretation of ν for the 1-class SVM to the CS++-SVM context, while the second clarifies this in the CS++-SVM context:

Theorem 8 *The parameter ν in the CS++-SVM training problem (15) has the following properties:*

1. *ν is an upper bound on the class-wise average fraction of extra-error vectors.*
2. *ν is a lower bound on the class-wise average fraction of extra-support vectors.*
3. *If the distributions P_0, P_1, ..., P_{n-2} do not contain discrete components then asymptotically, with probability 1, ν equals both the class-wise average fraction of extra-error vectors and the class-wise average fraction of extra-support vectors.*

Proof Denote by $\bar{f}_{\text{EEV}} = \frac{1}{n-1}\sum_{s\in\mathbb{Z}_{n-1}} \frac{N_{\text{EEV};s}}{N_s}$ the class-wise average fraction of extra-error vectors, and likewise $\bar{f}_{\text{ESV}} = \frac{1}{n-1}\sum_{s\in\mathbb{Z}_{n-1}} \frac{N_{\text{ESV};s}}{N_s}$ the class-wise average fraction of extra-support vectors. Consider the dual CS++-SVM training problem (19). The constraint set requires that $\sum_i \mathbf{u}_{(n);n-1}^T \boldsymbol{\alpha}_i = -C$ and $\mathbf{u}_{(n);n-1}^T \boldsymbol{\alpha}_i < 0\ \forall i$. Hence:

1. If $\bar{f}_{\text{EEV}} > \nu$ then $\sum_i \mathbf{u}_{n-1}^T \boldsymbol{\alpha}_i \leq -\frac{C}{\nu}\bar{f}_{\text{EEV}} < -C$, which contradicts the constraints. Hence ν is an upper bound on the class-wise average fraction of extra-error vectors.

2. If $\bar{f}_{\text{ESV}} < \nu$ then $\sum_i \mathbf{u}_{n-1}^T \boldsymbol{\alpha}_i \geq -\frac{C}{\nu}\bar{f}_{\text{ESV}} > -C$, which contradicts the constraints. Hence ν is a lower bound on the class-wise average fraction of extra-support vectors.

The following proof of part 3 is an extension of the proof presented in Schölkopf and Smola (1998). Note (see appendix A) that the CS++-SVM classifier h may be written in max-wins form (30), where each ζ_s has the form of a standard SVM function (29). A point $\vec{a}$ in feature space that is an ESV but not an EEV must satisfy $\zeta_s(\vec{a}) - \zeta_{n-1}(\vec{a}) = 0$, where $\zeta_s - \zeta_{n-1}$ again has the form of a standard SVM function. It follows from Schölkopf and Smola (1998) (proof of theorem 1 part (iii)) that, asymptotically, under the conditions described, the probability of a satisfying this constraint almost surely converges to 0, and as the number of classes is finite it follows that the probability of a point being an ESV but not an EEV almost surely converges to 0. Combining this with results 1 and 2 shows that both the class-wise average fraction of ESVs and the class-wise average fraction of EEVs almost surely converges to ν. ■

Theorem 9 *ν is an upper bound on the class-wise average fraction of anomaly vectors. If there are no intra error vectors in the training set Θ and if the distributions P_0, P_1, ..., P_{n-2} do not contain discrete components then asymptotically, with probability 1, ν equals the class-wise average fraction of anomaly vectors.*

Proof By theorems 7 and 8, and extending the notation from the previous proof, $\bar{f}_{\text{IAV}} \leq \bar{f}_{\text{EEV}} \leq \nu$, proving the first part of the theorem. If there are no IEVs then by theorem 7 all ESVs are AVs, and the second part of the proof follows from theorem 8. ■

The implication of this is that, if all training vectors are either correctly labelled or labelled as anomalous (i.e. there are no intra-error vectors) then ν will control the class-wise average fraction of training vectors which are labelled anomalous. If ν is small then the trained machine will be better able to classify the training vectors but may not correctly detect anomalies, whereas if ν is larger then the trained machine will be better able to detect anomalies but may mistakenly classify non-anomalous vectors as anomalous. Furthermore, in this case the ν parameter acts independently to the C parameter, allowing the user to independently control both the risk minimisation/margin maximisation trade-off of the classifier and the sensitivity to anomalies.

More generally the following theorem will hold true:

Theorem 10 *ν is an upper bound on the class-wise average fraction of anomaly vectors. If there are no vectors in the training set Θ that are simultaneously intra- and extra-error vectors, and if the distributions P_0, P_1, ..., P_{n-2} do not contain discrete components, then asymptotically, with probability 1, ν equals the class-wise average fraction of anomaly vectors.*

Proof The proof follows by noting that the presence of intra-error vector will have no influence on the previous proof provided that such vectors are not simultaneously extra-error vectors. ■

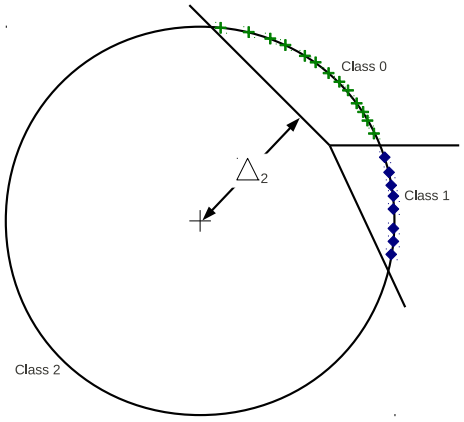

Figure 5: Illustration of the effect of regularisation on feature-space geometry for distance based kernel functions for the case $n = 3$, $d_K = 2$. Training vectors lie on the hypersphere. Minimising $\frac{1}{2}\sum_m \mathbf{w}_m^T \mathbf{w}_m$ maximises Δ_2, thereby maximising the proportion of the hypersphere labelled class 2 (anomalous).

In the most general case where we allow training vectors that are both intra- and extra-error vectors the relationship will not be exact, as extra-error vectors that are also intra-error vectors are not guaranteed to lie in class $n-1$ in feature space and hence be labelled as anomalous. However in this case ν will continue to act as an upper bound on the fraction of anomaly vectors.

5.3 Anomaly Detection Properties for Radial Kernels

This section considers the properties of the CS++-SVM for kernels of the form:

$$K(x, y) = \kappa(\|x - y\|)$$

The key feature of such kernels is that they map all input space vectors $x \in \mathbb{X}$ onto the surface of a hypersphere of radius $\sqrt{\kappa(0)}$ centred at the origin in feature space (Schölkopf and Smola, 2001). When using such kernels it is sensible to seek a trained machine which labels as much of this hypersphere class $n-1$ (anomalous) as possible.

Definition 11 *The* origin-anomaly distance Δ_{n-1} *for a trained CS++-SVM classifier given by* $\mathbf{w}_0$, $\mathbf{w}_1$, ..., $\mathbf{b}$ *is defined to be the shortest distance between the origin and the boundary of class region* $n-1$ *in feature space.*

As shown in Figure 5, maximising the origin-anomaly distance Δ_{n-1} will help to ensure that as much of the hypersphere as practical will be labelled anomalous. Moreover, it may be seen from (22) that if $\mathbf{b} \succ_{\mathbb{G}_{n-1}^{\mathrm{RD}}} \mathbf{0}$ then:

$$\Delta_{n-1} = \min_{t|t\neq n-1} \left(\frac{\mathbf{v}_{(n);n-1,t}^T \mathbf{b}}{\sqrt{\sum_m \mathbf{w}_m^T \left(\mathbf{v}_{(n);n-1,t} \mathbf{v}_{(n);n-1,t}^T \right) \mathbf{w}_m}} \right) \tag{24}$$

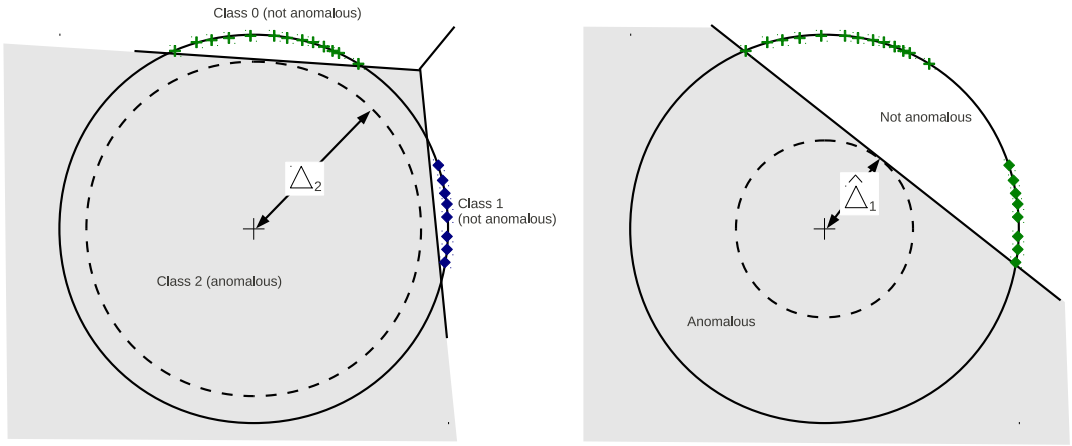

Figure 6: Comparison of the origin-anomaly distance for the CS++-SVM (left) and the 1-class SVM (right) trained on the same data set. In both cases the origin-anomaly distance is maximised. Note however that the additional class structure of the CS++-SVM allows more of the hypersphere onto which the data is mapped to be classed as anomalous (in particular the empty region between the two classes) than is possible with the 1-class SVM.

and hence:

$$\Delta_{n-1} \geq \frac{\min\limits_{s \neq n-1} \mathbf{v}^T_{(n);s,n-1}\mathbf{b}}{\sqrt{\sum\limits_m \mathbf{w}^T_m \mathbf{w}_m}} > 0 \tag{25}$$

Minimising the bias-forcing term $-C\mathbf{u}^T_{(n);n-1}\mathbf{b}$ in the primal CS++-SVM training problem (15) will help to ensure a solution for which $\mathbf{b} \succ_{\mathbb{G}^{\mathrm{RD}}_{n-1}} \mathbf{0}$ (or, equivalently, $\mathbf{u}^T_{(n);n-1}\mathbf{b} > \mathbf{u}^T_{(n);s}\mathbf{b}$ for all $s \in \mathbb{Z}_{n-1}$), while minimising the first term in the primal will maximise a lower bound on the origin-anomaly distance Δ_{n-1}.

This special case also provides a heuristic means of comparing the CS++-SVM with the 1-class SVM in terms of anomaly detection ability. Let Θ be a multiclass training set with examples from classes 0, 1, ..., $n-2$ and let $\hat{\Theta}$ be the same training class with labels removed:

$$\hat{\Theta} = \{\, x_i | \, (x_i, y_i) \in \Theta\}$$

Let Δ_{n-1} be the origin-anomaly distance for a CS++-SVM trained with Θ, and let $\hat{\Delta}_1$ be the origin-anomaly distance for a 1-class SVM trained with $\hat{\Theta}$. Suppose further that the distributions P_0, P_1, ..., P_{n-2} are distributions with little or no overlap.

Under these conditions it is not difficult to imagine cases such as that shown in Figure 6 where the additional structure of the CS++-SVM decision boundary in feature space (multiple hyperplanes) compared to the 1-class SVM (single hyperplane), and the additional cluster-identifying information available to the CS++-SVM (the class labels y_i in the training set Θ), will mean that $\Delta_{n-1} > \hat{\Delta}_1$ and moreover the CS++-SVM will label a larger portion of the hypersphere as anomalous than the 1-class SVM. Heuristically speaking, this

implies that the CS++-SVM should be able to outperform a 1-class SVM on a pure anomaly detection task when the training data is labelled. Our observations in Section 6 support this hypothesis.

6. Experimental Methodology and Results

In this section the performance of the CS++-SVM on artificial and real data is compared with several alternative schemes. In the first two experiments (artificial data and character recognition) we compare with a hybrid scheme that combines a 1-class SVM (anomaly detector) and a standard CS-SVM (multiclass classifier) as follows:

$$h_{\text{hybrid}}(x) = \begin{cases} n-1 & \text{if } h_{1-\text{class}}(x) = +1 \\ h_{\text{CS-SVM}}(x) & \text{otherwise} \end{cases}$$

where $h_{\text{CS-SVM}}$ is a CS-SVM classifier trained on the training set Θ given and $h_{1-\text{class}}$ is a 1-class SVM trained on the training set $\hat{\Theta}$, where:

$$\hat{\Theta} = \{ x_i | (x_i, y_i) \in \Theta \}$$

In our third set of experiments we compare to both this method and also the classify with reject option (Bartlett and Wegkamp, 2008; Chow, 1970; Grandvalet et al., 2009; Kwok, 1999; Fumera and Roli, 2002); and an alternative hybrid method which uses a hyperspherical SVM (Tax and Duin, 2004) instead of a 1-class SVM.

All data in training sets has been pre-processed to normalise features. Non-categorical features are normalised to give zero mean, unit variance for each feature. Categorical features have been replaced with a binary representation using 1 bit per category.[7] Missing non-categorical values have been replaced with 0 and missing categorical values with 0 for all bits in the representation. All experiments were run using SVMHeavy (Shilton, 2001–2020),[8] which is an active-set based optimisation library written in C++ (see Shilton et al. (2005) for details). Alternative optimisation libraries include Nandan et al. (2014); Claesen et al. (2014).

For these experiments we have used the radial basis kernel and polynomial kernel functions:

$$K_{\text{rbf}}(x, y) = \exp\left(-\tfrac{1}{2\gamma} \|x - y\|^2\right)$$
$$K_{\text{poly}}(x, y) = (1 + \langle x, y \rangle)^d$$

where the kernel parameters are, respectively, $10^{-2} \leq \gamma \leq 10^2$ and $d \in \{1, 2, 3, 4, 5\}$. The trade-off parameter C was selected from $10^{-2} \leq \frac{C}{N} \leq 10^2$. Parameter selection of C and d or γ (i.e. classification related training parameters) was carried out using a grid search to minimise leave-one-out error measured on the training set.

6.1 Artificial Data

This experiment investigates the performance of the CS++-SVM (and the hybrid scheme) using a 2-dimensional, 5-class data set designed for easy visualisation. The data set was

7. For example, a categorical feature with three possible values `A`, `B` or `C` will be replaced with a three-bit binary representation where `A` is encoded $1, 0, 0$; `B` is encoded $0, 1, 0$; and `C` is encoded $0, 0, 1$
8. Code available at `https://github.com/apshsh/SVMHeavy`.

	0	1	2	3	4	5 (A)
0	**96.1**%	0%	1.91%	0%	0%	*1.99%*
1	0%	**95.9**%	0.15%	0%	0%	*3.95%*
2	4.78%	0.7%	**90.41**%	0%	0.01%	*4.1%*
3	0%	0%	0%	**95.43**%	0%	*4.57%*
4	0%	0%	0%	0%	**96.58**%	*3.42%*

Table 1: Test results for the CS++-SVM. Columns indicates assigned classification, row indicates actual classification. The right column shows percentages labelled anomalous.

	0	1	2	3	4	5 (A)
0	**86.3**%	0%	2.4%	0%	0%	*11.3%*
1	0%	**94.3**%	0.5%	0.1%	0.3%	*4.8%*
2	3.9%	0.4%	**59.6**%	0%	0%	*36.1%*
3	0%	0%	0.1%	**92.6**%	0.1%	*7.2%*
4	0%	0%	0.1%	0%	**95.5**%	*4.4%*

Table 2: Test results for the hybrid scheme. Columns indicates assigned classification, row indicates actual classification. The right column shows percentages labelled anomalous.

generated from the distributions:

$$
\begin{aligned}
P_0 &= \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) \\
P_1 &= \mathcal{N}\left(\begin{bmatrix} 7 \\ -7 \end{bmatrix}, \begin{bmatrix} 1.6253 & 3.0547 \\ 3.0547 & 7.7347 \end{bmatrix}\right) \\
P_2 &= \left\{ \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix} \middle| r \in \mathcal{N}(5, 2.25), \theta \in \mathcal{U}\left(-\tfrac{\pi}{2}, \tfrac{\pi}{2}\right) \right\} \\
P_3 &= \mathcal{N}\left(\begin{bmatrix} -2 \\ -12 \end{bmatrix}, \begin{bmatrix} 1.4393 & -1.0607 \\ -1.0607 & 3.5607 \end{bmatrix}\right) \\
P_4 &= \left\{ \begin{bmatrix} r\cos\theta \\ r\sin\theta \end{bmatrix} \middle| r \in \mathcal{N}(10, 1), \theta \in \mathcal{U}\left(\tfrac{\pi}{2}, \tfrac{5\pi}{4}\right) \right\}
\end{aligned}
$$

where $\mathcal{N}$ is the normal distribution and $\mathcal{U}$ the uniform distribution. 150 training vectors were generated for each class, giving a total training set size of 750 training vectors. In addition to this a testing set of 10000 vectors (2000 from each class) was generated. For this experiment we have chosen the RBF kernel.

Representative results for the CS++-SVM and hybrid schemes are shown in Figure 7, where $\nu = 0.05$, $C = 1$ and $\gamma = 10$ (C and γ selected to minimise leave-one-out error, ν selected arbitrarily). For completeness the 1-class SVM classifier and CS-SVM classifier used to construct the hybrid result are also shown.

The classification accuracy of both the CS++-SVM and the hybrid method are summarised in tables 1 and 2, which shows classification accuracy for the 10000 testing vectors

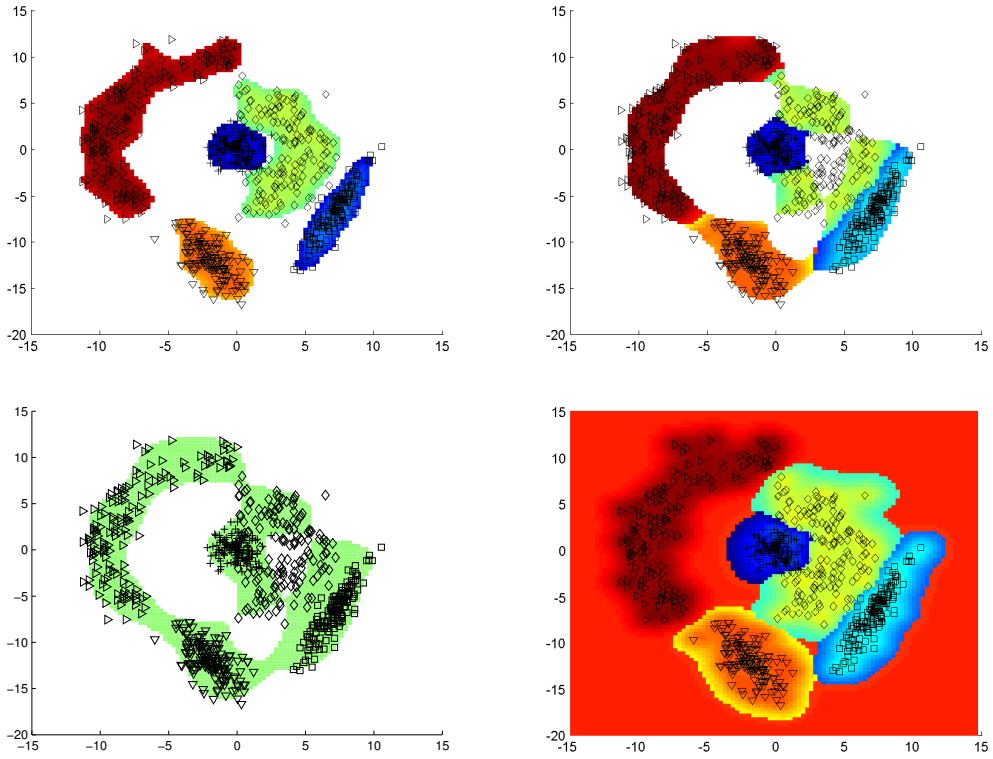

Figure 7: Results for the artificial training set. Top left shows the classification regions for the CS++-SVM, and top right for the hybrid scheme. Bottom left shows the anomaly detection regions for the 1-class SVM component of the hybrid scheme, and bottom right shows the classification regions for the CS-SVM component of the hybrid scheme. In all cases class 0 is shown in dark blue, class 1 in aqua, class 2 in green, class 3 in orange and class 4 in red. Points falling in the un-colored region are classed as anomalous.

(unseen during training). The CS++-SVM significantly outperforms the hybrid scheme for all classes. In particular, the hybrid scheme mis-identifies 36% of testing vectors from class 2, as anomalous, whereas the CS++-SVM only mis-identifies 4.1% of class 2 testing vectors as anomalous. To understand what is happening here consider Figure 7. We note that the class regions found by the CS++-SVM tend to be more distinct than those found by the hybrid scheme. In particular, under the hybrid scheme, class region 4 "blends into" class regions 2 and 3, and likewise class region 3 "blends into" class regions 1, 2 and 4. By contrast, the CS++-SVM leaves noticeable gaps between classes where there is little or no overlap between the training points.

This result supports our hypothesis in Section 5.3 that the additional structure in the decision surface allowed by the CS++-SVM, and the presence of data-labels in the CS++-SVM training set Θ that are absent from the 1-class SVM training set $\hat{\Theta}$ in the hybrid model, allow the CS++-SVM to outperform the hybrid method. This view is supported by the 1-class SVM in Figure 7, from where it may be seen that the 1-class SVM component of the hybrid scheme is unable to clearly distinguish between the clusters of data-points representing the 5 classes present in the data.[9]

With regard to training complexity, in this example both schemes took roughly the same time to train (13 seconds in each case) where each code-base shared the same quadratic optimisation routine.

Finally, while we explicitly assume that there are no anomalies in the training data in this paper, it is none-the-less interesting to consider the effect of such anomalies. Anomalies were generated from the distribution:

$$P_A = \mathcal{U}(-15, 15) \times \mathcal{U}(-20, 15)$$

10 anomalies were added to each class in the training set (giving 160 training vectors per class) and the experiment repeated. The resulting classification regions for the CS++-SVM and hybrid schemes are shown in Figure 8. It is interesting to note that the CS++-SVM is virtually unaffected by the addition of anomalies,[10] whereas the hybrid scheme differs significantly.

The reason for this becomes clear when the classification (CS-SVM) and anomaly detection (1-class SVM) components of the hybrid scheme for the original training set (Figure 7) and the training set with anomalies (Figure 8, lower figures) are compared. It may be seen that the classifiers are very nearly identical, while the anomaly detection results differ significantly. Upon reflection this is unsurprising. By design:

1. SVM classifiers are intended to be insensitive to outliers (this is the function of the "soft" margin). This is evident here in the insensitivity of the CS-SVM to the presence or absence of a small number of anomalies (outliers) in the training data.

2. 1-class SVM anomaly detectors are intended to be highly sensitive to such outliers (anomalies) and attempt to select the top νN such vectors to form a decision boundary.

9. In principle, a more advanced clustering algorithm could be used in place of the 1-class SVM to automatically detect clustering in the data and thereby improve results. This is outside the scope of the present paper. Note, however, that the impact of such increased complexity on the training time of the hybrid scheme will be of some importance when comparing it with the CS++-SVM.

10. Of the 100×100 test grid points used to generate this graph, only 5 pixels differed from the corresponding pixels in Figure 7

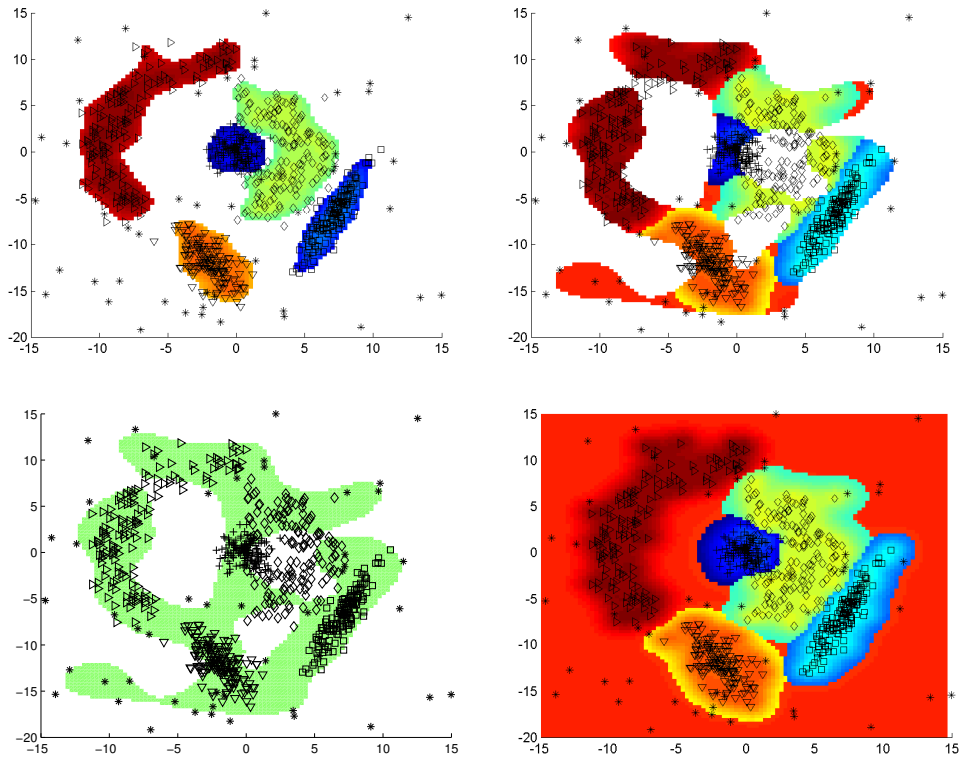

Figure 8: Results for artificial training set plus anomalies. Top left shows the classification regions for the CS++-SVM, and top right for the hybrid scheme. Bottom left shows the anomaly detection regions for the 1-class SVM component of the hybrid scheme, and bottom right shows the classification regions for the CS-SVM component of the hybrid scheme.

This is evident here in the *sensitivity* of the 1-class SVM to the presence or absence of anomalies in the training data.

It is interesting that the behaviour of the CS++-SVM with regard to added anomalies (outliers) more closely models the CS-SVM than the 1-class SVM. We surmise that this is a result of the CS++-SVMs ability to use labels to aid in the clustering of data that is not available to the 1-class SVM, as discussed in Section 5.3. We note that the fraction points in the training set labelled anomalous is roughly the same for both the hybrid and CS++-SVM methods (approximately νN), but that the hybrid method is significantly more likely to (incorrectly) apply this label to points points not generated from P_A.

6.2 Character Recognition

In this experiment the performance of the CS++-SVM and hybrid schemes were compared using the UCI (Dua and Graff, 2017) Optical Recognition of Handwritten Digits Data Set (DIG) (Kaynak, 1995; Alpaydin and Kaynak, 1998) for training and Kassel and Taskar's

OCR data set of handwritten lower-case characters (CHR) [11] (Kassel, 1995; Taskar et al., 2004) as our anomaly test set. Both data sets were resized to 8x8 pixel images with each element normalised to lie in the range $[0, 1]$. The DIG data set contains 3823 training instances (of which we used 2000 due to memory restrictions) and 1797 testing instances over 10 classes (digits 0-9); and the CHR data set contains 52152 instances over 26 classes (characters a-z). The classifier (CS++-SVM or hybrid) was trained on the DIG training set, and our aim was to compare performance both in terms of classification of the DIG testing set (all known classes) and in terms of its ability to label examples from the CHR set anomalous (not digits).

Figure 9 shows the accuracy of the comparative methods with respect to the DIG test set (non-anomalous) and CHR data set (anomalous) over a range of ν values. The first thing we note from this figure is that, while both methods have good accuracy on the DIG data set alone, the CS++-SVM is clearly better at labelling non-digits (anomalies) than the hybrid method. This supports our hypothesis in Section 5.3 regarding the CS++-SVM being able to take advantage of the label-information that is absent from the 1-class SVM training set $\hat{\Theta}$ in the hybrid model. Curiously the best overall performance for the CS++-SVM occurs when ν is quite small, with a slight decrease in performance with respect to both anomaly detection and classification accuracy when ν is large.[12]

Figures 9 shows which digits the two methods incorrectly assigned elements of the CHR data set to over a range of ν values, while table 3 shows the confusion matrix for RBF kernel with $\nu = 0.1$. We note from this that the most common error made here is the character 'a' being mislabelled as digit 4; characters 'c', 'e' and 'z' being mislabelled 8; and character 'r' being mislabelled 9. The confusion matrices for both methods show a similar pattern over a range of ν values.

Finally, Figure 10 shows the results for the linear (first order polynomial) kernel. The performance of both methods in this case is hampered by the linearity of the kernel. However it is interesting to note that, while the hybrid method is almost entirely incapable of detecting anomalies over the entire ν range in this case, the CS++-SVM is able to achieve some improvement as ν increases. We surmise that this is because the hybrid method requires the anomalies (characters) to be linearly separable from the training data (digits), whereas the CS++-SVM constructs a piecewise linear separating hyperplane for this purpose.

Once again there was no appreciable difference between the training times for both the CS++-SVM and hybrid methods.

6.3 Comparisons with Classify with Reject Option

In this section we compare our method with the classify with reject option approach (Bartlett and Wegkamp, 2008). The essence of this approach is to classify points that fall sufficiently close to the margin between classes as "reject", indicating that the point cannot be reliably classified by the classifier. To train such a classifier the empirical loss term is replaced by an alternative loss function that applies a lower penalty to reject mis-

11. Available at `http://ai.stanford.edu/~btaskar/ocr/`.

12. Investigating this particular phenomena is left as future work. However note that, even in the large ν regime, the performance of the CS++-SVM remains superior to the hybrid method.

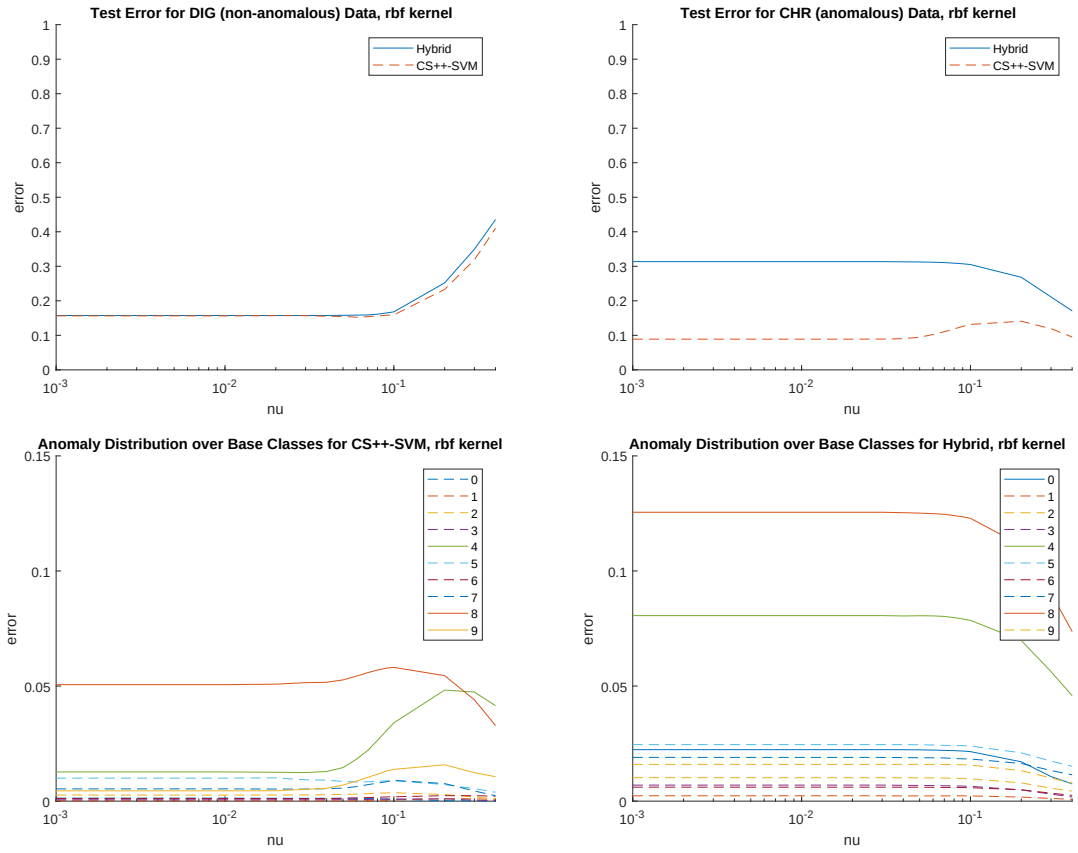

Figure 9: Character recognition results for RBF kernel. Top left shows variation of the classification error on the DIG (non-anomalous) data set as a function of ν. Top right shows variation of anomaly detection error on the CHR (anomalous) data set as a function of ν. The lower two graphs show the fraction of anomalous (CHR) samples mis-classified into the 10 digit classes over a range of ν for the CS++-SVM and hybrid scheme, respectively.

•	0	1	2	3	4	5	6	7	8	9	(A)
a	0.00%	0.07%	0.00%	0.12%	**12.99%**	1.59%	0.10%	1.19%	6.30%	1.71%	75.93%
b	0.00%	0.00%	0.00%	0.00%	0.31%	0.00%	0.23%	0.00%	0.00%	0.00%	99.45%
c	0.19%	0.00%	7.47%	0.57%	1.84%	3.17%	0.57%	0.47%	**29.90%**	0.38%	55.44%
d	0.00%	0.00%	0.00%	0.00%	0.49%	0.00%	0.07%	0.00%	0.14%	0.00%	99.31%
e	0.00%	0.12%	0.32%	0.04%	8.68%	0.59%	0.42%	0.16%	**19.92%**	0.59%	69.16%
f	0.00%	0.22%	0.00%	0.00%	0.43%	0.33%	0.00%	0.11%	2.28%	0.76%	95.87%
g	0.00%	0.00%	0.04%	0.00%	0.65%	0.08%	0.04%	0.12%	1.05%	0.04%	97.98%
h	0.00%	0.00%	0.00%	0.00%	0.12%	0.00%	0.23%	0.12%	0.12%	0.12%	99.30%
i	0.00%	0.00%	0.04%	0.04%	0.16%	0.22%	0.04%	0.02%	0.39%	0.04%	99.04%
j	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.06%	0.53%	98.41%
k	0.00%	0.00%	0.00%	0.00%	1.21%	0.11%	0.33%	0.11%	0.88%	0.00%	97.36%
l	0.00%	0.00%	0.32%	0.22%	0.13%	0.25%	0.10%	0.06%	0.41%	0.00%	98.50%
m	0.00%	0.00%	0.00%	0.00%	5.74%	1.62%	0.00%	0.69%	4.56%	1.50%	85.89%
n	0.40%	0.00%	0.00%	0.06%	4.74%	2.47%	0.06%	1.41%	1.00%	4.62%	85.25%
o	0.36%	0.00%	0.00%	0.00%	3.62%	0.03%	0.03%	0.13%	4.11%	0.08%	91.66%
p	0.00%	0.00%	0.00%	0.00%	0.07%	0.00%	0.00%	0.00%	0.00%	0.29%	99.64%
q	0.00%	0.00%	0.00%	0.00%	0.59%	0.00%	0.00%	0.00%	0.00%	0.00%	99.41%
r	0.11%	0.04%	0.07%	0.00%	2.36%	4.15%	0.04%	0.07%	9.39%	**10.40%**	73.36%
s	0.00%	0.00%	0.14%	0.57%	2.37%	0.93%	0.50%	0.43%	6.81%	1.29%	86.94%
t	0.00%	0.61%	0.00%	0.00%	0.56%	0.09%	0.09%	0.33%	4.87%	1.50%	91.95%
u	0.00%	0.00%	0.00%	0.27%	1.87%	0.04%	1.01%	9.45%	0.78%	0.00%	86.57%
v	0.00%	0.00%	0.00%	0.00%	0.60%	0.00%	0.30%	1.36%	0.75%	0.00%	96.99%
w	0.00%	0.00%	0.00%	0.38%	3.46%	0.38%	0.77%	6.35%	1.92%	0.00%	86.73%
x	0.00%	0.00%	0.00%	0.00%	6.05%	0.00%	0.00%	0.24%	5.57%	0.24%	87.89%
y	0.00%	0.00%	0.00%	0.00%	0.33%	0.00%	0.00%	0.49%	0.74%	0.08%	98.36%
z	0.00%	0.00%	0.09%	0.00%	4.11%	0.00%	0.09%	0.18%	**24.31%**	0.82%	70.38%

Table 3: Test results for the CS++-SVM on CHR data set. Columns indicates assigned classification, row indicates actual classification, rightmost column shows percentages (correctly) labelled anomalous.

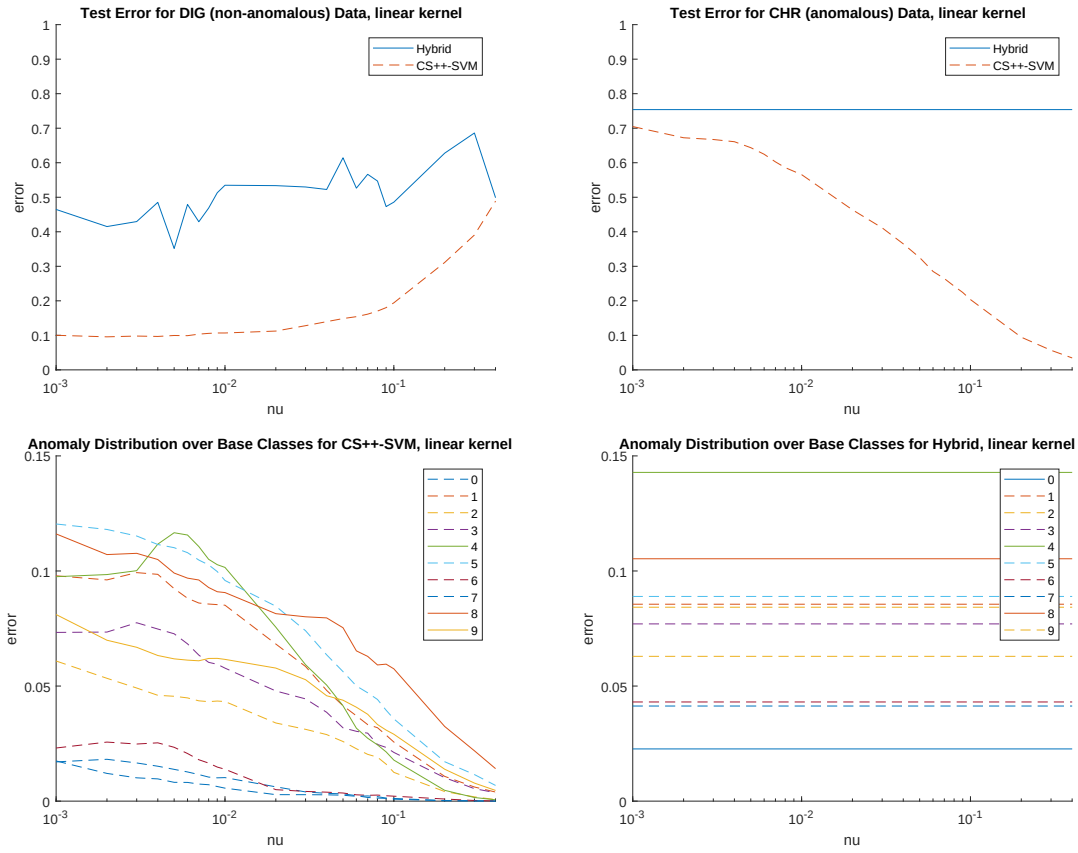

Figure 10: Character recognition results for linear kernel. Top left shows variation of the classification error on the DIG (non-anomalous) data set as a function of ν. Top right shows variation of anomaly detection error on the CHR (anomalous) data set as a function of ν. The lower two graphs show the fraction of anomalous (CHR) samples mis-classified into the 10 digit classes over a range of ν for the CS++-SVM and hybrid scheme, respectively.

classification than other (non-reject) class classification - see (Bartlett and Wegkamp, 2008) for details. While we treat the "reject" and "anomalous" labels as synonyms for the purposes of this experiment, it is important to note that they are distinct in both motivation and interpretation. We will discuss the distinction and its implications for this experiment in more details shortly.

In this experiment we compare the CS++-SVM, classify with reject, the hybrid approach described previously (Hybrid), and an alternative hybrid approach using a hyperspherical anomaly detector (Tax and Duin, 2004) instead of a 1-class SVM (alt-Hybrid). For consistency with previous experiments we have chosen $\nu = 0.05$ for our experiments. We have considered three data sets from the UCI repository (Dua and Graff, 2017) here, namely human activity recognition (HAR (Anguita et al., 2013)), which consists of accelerometry and gyroscope sensor measurements collected from 30 subjects performing activities of daily living (ADL), such as walking (upstairs, downstairs, normal), sitting, standing, and laying carrying a waist-mounted smartphone; daily and sports activities (DSA (Altun et al., 2010; Barshan and Yüksek, 2014; Altun and Barshan, 2010)), which consists of motion sensor data of 19 daily and sports activities each performed by 8 subjects in their own style for 5 minutes; and forest type mapping (forest (Johnson et al., 2012)), which contains data sourced from a remote sensing study, which mapped different forest types based on their spectral characteristics at visible-to-near infrared wavelengths, using ASTER satellite imagery.

For the HAR data set we use sitting and standing as our known classes and everything else as anomalous. For the DSA data set we treat activities 1-5 as class "1" (known), activities 6-11 as class "2" (known), and activites 12-19 as our unknown (anomaly) set; where 5000 points from class 1 and 5000 points from class 2 were randomly selected for the training set. Finally, for the forest data set we used classes 's' (Sugi forest), 'h' (Hinoki forest) and 'd' (Mixed deciduous forest) as our known training classes and class 'o' (other forest land) as our unknown (anomaly) class.

Before presenting our results, it is important to emphasise once more that classify-with-reject is not designed to detect anomalies. In fact, classify-with-reject as per (Bartlett and Wegkamp, 2008) is designed on the (until recently almost universal) implicit assuption that all points that will be labelled by the trained machine belong to one of the classes with which the machine was trained. Rather it is intended to detect and (label as) "reject" any vectors that it cannot label with confidence, specifically those lying close to the margin between classes. Other anomalies such as outliers lying well away from both the margin and the main clusters of points in each class will not be detected. Thus it is unreasonable to expect it to perform well in this scenario, so the results presented here *should not* be viewed as showing that classify-with-reject is ineffective, but rather the unsurprising fact that it is not well suited to the combined classification/anomaly detection domain *for which it was not designed.* With this caveat in mind, results on the three data sets are given in table 4. Unsurprisingly the proposed CS++-SVM framework is able to outperform classify-with-reject, and also the other baselines (in line with other experiments).

7. Conclusion

In this paper the CS++-SVM has been presented, which is a new form of SVM combining both multiclass classification and anomaly detection into a single step process. This

	CS++-SVM	Hybrid	alt-Hybrid	Classify-with-Reject
HAR	96.4%(99.5%)	90.8%(99.5%)	94.8%(1.0%)	94.8%(1.0%)
DSA	90.3%(93.8%)	69.0%(90.0%)	76.5%(92.4%)	96.7%(1.5%)
Forest	87.8%(65.2%)	1.4%(100%)	83.2%(0%)	72.8%(0%)

Table 4: Classification accuracy on test set for HAR, DSA and forest data sets. Results are shown as NA%(A%), where NA% is the test error for non-anomaly classes and A% is the test error for the anomaly class. Refer to text for further discussion.

addresses the combined classification/anomaly detection problem whereby not all samples presented to the trained machine while in use are guaranteed to belong to one of the training classes from which the machine was constructed, examples of which include facial recognition, intrusion detection and data analysis for the IoT. The CS++-SVM is motivated as a natural extension and combination of the CS-SVM (a multiclass classifier) and the 1-class SVM. Analysis of the properties of the CS++-SVM has considered the influence of the various training parameters and also an intuitive explanation of how the CS++-SVM may outperform alternative schemes. The CS++-SVM has been experimentally compared to a hybrid alternative for both artificial and real data and it has been shown that the CS++-SVM is both effective and efficient.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and constructive suggestions. This research was partially funded by the Australian Government through the Australian Research Council (ARC). Prof Venkatesh is the recipient of an ARC Australian Laureate Fellowship (FL170100006). The work is supported in part by the ARC Discovery Project grant DP200101960 and Oceania Cyber Security Centre (OCSC) proof of concept (POC) scheme.

Appendix A. A Proof of Theorem 7

In this appendix we give a full proof of theorem 7. Before proceeding some preliminary results are required:

Theorem 12 *For all $n \geq 2$, and assuming recursive division class centres, $\sum_{s\in\mathbb{Z}_n} \mathbf{u}_{(n);s} = \mathbf{0}$.*

Proof Consider the class centres $\mathbf{u}_{(n);s}$, $s \in \mathbb{Z}_n$, for the recursive division case. From (9) it may be seen that:

$$\begin{aligned} \textstyle\sum_{s\in\mathbb{Z}_2} \mathbf{u}_{(2);s} &= \mathbf{0} \\ \textstyle\sum_{s\in\mathbb{Z}_n} \mathbf{u}_{(n);s} &= \begin{bmatrix} \sum_{s\in\mathbb{Z}_{n-1}} \mathbf{u}_{(n-1);s} \\ 0 \end{bmatrix} \quad \forall n > 2 \end{aligned}$$

and the desired result follows by induction. ■

Theorem 13 *For all $n \geq 2$, $s \in \mathbb{Z}_n$, and assuming recursive division class centres, $\{\mathbf{u}_{(n);t} | t \in \mathbb{Z}_n, t \neq s\}$ spans target space $\mathbb{R}^{d_T}$ (where $d_T = n-1$).*

Proof Trivially, $\left\{\mathbf{u}_{(2);0}\right\} = \{[-1]\}$ spans $\mathbb{R}$. For $n > 2$:

$$\begin{array}{l} \left\{ \mathbf{u}_{(n);q} \middle| \, q \in \mathbb{Z}_{n-1} \right\} = \ldots \\ \left\{ \left[\begin{array}{c} \mathbf{0} \\ -1 \end{array} \right], \frac{1}{n-1} \left[\begin{array}{c} \mathbf{u}_{(n-1);q} \sqrt{n\,(n-2)} \\ 1 \end{array} \right] \middle| \, q \in \mathbb{Z}_{n-2} \right\} \end{array}$$

will span $\mathbb{R}^{n-1}$ if $\left\{ \mathbf{u}_{(n-1);q} \middle| \, q \in \mathbb{Z}_{n-2} \right\}$ spans $\mathbb{R}^{n-2}$. It follows by induction that $\{\mathbf{u}_{(n);q} | q \in \mathbb{Z}_{n-1}\}$ spans $\mathbb{R}^{d_T} = \mathbb{R}^{n-1}$ for all $n \geq 2$. The desired result follows from this by application theorem 12. ■

We now proceed to the prove the central result:

Theorem 7: *If x_i is an extra error vector but not an intra error vector then x_i is an anomaly vector. If x_i is an extra support vector but not an extra error vector or an intra error vector then x_i is a boundary anomaly vector.*

Proof We begin by re-deriving the dual CS++-SVM training problem in modified form. In Section 4.2 the Lagrangian form of the CS++-SVM training problem was derived (17):

$$\begin{array}{rl} \mathcal{L} = & \frac{1}{2} \sum_m \mathbf{w}_m^T \mathbf{w}_m + C \frac{1}{\nu} \sum_i \frac{1}{N_{y_i}} \mathbf{u}_{(n);y_i}^T \boldsymbol{\xi}_i - C \mathbf{u}_{(n);n-1}^T \mathbf{b} - \ldots \\ & - \sum_{i,m} \boldsymbol{\alpha}_i^T \mathbf{w}_m \varphi_m(x_i) - \sum_i \boldsymbol{\alpha}_i^T \mathbf{b} + \sum_i \boldsymbol{\alpha}_i^T \tilde{\mathbf{u}}_{(n);y_i} - \sum_i \boldsymbol{\alpha}_i^T \boldsymbol{\xi}_i - \ldots \\ & - \sum_i \boldsymbol{\gamma}_i^T \boldsymbol{\xi}_i \end{array}$$

from which it was shown that, optimally:

$$\begin{array}{l} \mathbf{w}_m = \sum_i \boldsymbol{\alpha}_i \varphi_m(x_i) \;\; \forall m \\ \sum_i \boldsymbol{\alpha}_i = -C \mathbf{u}_{(n);n-1} \\ \frac{1}{\nu} \frac{C}{N_{y_i}} \mathbf{u}_{(n)y_i} \succeq_{\mathbb{G}_{(n);y_i}^{\mathrm{RD}*}} \boldsymbol{\alpha}_i \; \forall i \end{array}$$

Applying the first and third optimality conditions (but not the second) to the (17) gives the semi-dual CS++-SVM training problem:

$$\begin{array}{l} \max_{\mathbf{b}} \min_{\boldsymbol{\alpha}_i} Q = \frac{1}{2} \sum_{i,j} K_{ij} \boldsymbol{\alpha}_i^T \boldsymbol{\alpha}_j - \sum_i \tilde{\mathbf{u}}_{(n);y_i}^T \boldsymbol{\alpha}_i + \sum_i \boldsymbol{\alpha}_i^T \mathbf{b} + C \mathbf{u}_{(n);n-1}^T \mathbf{b} \\ \text{such that:} \quad -\frac{1}{\nu} \frac{1}{n-1} \frac{C}{N_{y_i}} \leq \mathbf{u}_{(n);s}^T \boldsymbol{\alpha}_i \leq 0 \; \forall i, s \neq y_i \end{array} \tag{26}$$

Next, define:

$$\overline{U} = \left[\begin{array}{cccc} \mathbf{u}_{(n);0} & \mathbf{u}_{(n);1} & \cdots & \mathbf{u}_{(n);n-1} \end{array} \right] \in \mathbb{R}^{d_T \times (d_T+1)}$$

Using (10) and recalling that $n = d_T + 1$ for the recursive division case it may be seen that:

$$\overline{U}^T \overline{U} = \frac{d_T+1}{d_T} \overline{I} - \frac{1}{d_T} \vec{1} \vec{1}^T \in \mathbb{R}^{(d_T+1) \times (d_T+1)}$$

where $\overline{I}$ is the identity matrix and $\vec{1}$ a column vector whose every element is 1. Define $\vec{\tau}_0, \vec{\tau}_1, \ldots, \vec{\tau}_{N-1} \in \mathbb{R}^{d_T+1}$ and $\vec{\beta} \in \mathbb{R}^{d_T+1}$ such that $\vec{1}^T\vec{\tau}_i = 0$ for all i and $\vec{1}^T\vec{\beta} = 0$; and let:

$$\begin{aligned}\boldsymbol{\alpha}_i &= \sqrt{\tfrac{d_T}{d_T+1}}\overline{U}\vec{\tau}_i = \sqrt{\tfrac{d_T}{d_T+1}}\textstyle\sum_s \mathbf{u}_{(n);s}\tau_{i,s} \;\; \forall i \\ \mathbf{b} &= \sqrt{\tfrac{d_T}{d_T+1}}\overline{U}\vec{\beta} = \sqrt{\tfrac{d_T}{d_T+1}}\textstyle\sum_s \mathbf{u}_{(n);s}\beta_s\end{aligned}$$

where it may be seen from theorems 12 and 13 that this does not restrict the range of $\boldsymbol{\alpha}_i$ or $\mathbf{b}$. It follows that:

$$\begin{aligned}\overline{U}^T\boldsymbol{\alpha}_i &= \sqrt{\tfrac{d_T}{d_T+1}}\overline{U}^T\overline{U}\vec{\tau}_i = \sqrt{\tfrac{d_T+1}{d_T}}\vec{\tau}_i \;\; \forall i \\ \overline{U}^T\mathbf{b} &= \sqrt{\tfrac{d_T}{d_T+1}}\overline{U}^T\overline{U}\vec{\beta} = \sqrt{\tfrac{d_T+1}{d_T}}\vec{\beta}\end{aligned}$$

and hence:

$$\begin{aligned}\vec{\tau}_i &= \sqrt{\tfrac{d_T}{d_T+1}}\overline{U}^T\boldsymbol{\alpha}_i \;\; \forall i \\ \vec{\beta} &= \sqrt{\tfrac{d_T}{d_T+1}}\overline{U}^T\mathbf{b}\end{aligned} \tag{27}$$

Substituting back into semi-dual CS++-SVM training problem, (26) may be re-written:

$$\max_{\beta_s}\min_{\tau_{i,s}} Q = \tfrac{1}{2}\sum_{i,j,s} K_{ij}\tau_{i,s}\tau_{j,s} + \sum_{i,s}\tau_{i,s}\beta_s - \tfrac{1}{\sqrt{2}}\sqrt{\tfrac{d_T+1}{d_T-1}}\sum_i\left(\tau_{i,y_i} + \tau_{i,n-1}\right) + C\sqrt{\tfrac{d_T+1}{d_T}}\beta_{n-1}$$

$$\begin{aligned}\text{such that: } & -\sqrt{\tfrac{d_T}{d_T+1}}\tfrac{1}{\nu}\tfrac{1}{n-1}\tfrac{C}{N_{y_i}} \leq \tau_{i,s} \leq 0 \;\forall i, s \neq y_i \\ & \textstyle\sum_s \tau_{i,s} = 0 \;\forall i \\ & \textstyle\sum_s \beta_s = 0\end{aligned} \tag{28}$$

which is the modified semi-dual CS++-SVM training problem. For reference note that the trained machine may be re-written:

$$\mathbf{g}\left(\vec{\varphi}\left(x\right)\right) = \sqrt{\tfrac{d_T}{d_T+1}}\textstyle\sum_s \mathbf{u}_{(n);s}\zeta_s\left(x\right)$$

where:

$$\zeta_s\left(x\right) = \textstyle\sum_i K\left(x, x_i\right)\tau_{i,s} + \beta_s \tag{29}$$

and hence, noting that $\sum_s \zeta_s\left(x\right) = 0 \;\forall x$ it follows that:

$$\mathbf{u}_{(n);s}^T\mathbf{g}\left(\vec{\varphi}\left(x\right)\right) = \sqrt{\tfrac{d_T+1}{d_T}}\left(\textstyle\sum_i K\left(x, x_i\right)\tau_{i,s} + \beta_s\right)$$

from which it may be seen that:

$$h\left(x\right) = \underset{s}{\operatorname{argmax}}\,\zeta_s\left(x\right) \tag{30}$$

We now apply Lagrangian techniques to the modified semi-dual CS++-SVM training problem (28). To this end we define Lagrange multipliers μ_i and ρ for the second and third constraint sets, respectively, in (28), and construct the Lagrangian:

$$\begin{aligned}\mathcal{Q} = \tfrac{1}{2}\sum_{i,j,s} K_{ij}\tau_{i,s}\tau_{j,s} + \sum_{i,s}\tau_{i,s}\beta_s - \sqrt{\tfrac{d_T+1}{d_T-1}}\sum_i \tau_{i,y_i} - \sqrt{\tfrac{d_T+1}{d_T-1}}\sum_i \tau_{i,n-1} + C\sqrt{\tfrac{d_T+1}{d_T}}\beta_{n-1} - \ldots \\ \ldots - \sum_{i,s}\mu_i\tau_{i,s} - \sum_s \rho\beta_s\end{aligned}$$

Optimality conditions on $\tau_{i,s}$ and β_s for all i, s are:

$$\begin{array}{rl} \frac{\partial \mathcal{Q}}{\partial \tau_{i,s}} & \begin{cases} \leq 0 & \text{if } \tau_{i,s} = 0 \\ = 0 & \text{if } \tau_{i,s} \in \left(-\sqrt{\frac{d_T}{d_T+1}}\frac{1}{\nu}\frac{1}{n-1}\frac{C}{N_{y_i}}, 0\right) \\ \geq 0 & \text{if } \tau_{i,s} = -\sqrt{\frac{d_T}{d_T+1}}\frac{1}{\nu}\frac{1}{n-1}\frac{C}{N_{y_i}} \end{cases} \forall s \neq y_i \\ \frac{\partial \mathcal{Q}}{\partial \tau_{i,y_i}} & = 0 \\ \frac{\partial \mathcal{Q}}{\partial \beta_s} & = 0 \end{array}$$

where:

$$\begin{array}{rll} \frac{\partial \mathcal{Q}}{\partial \tau_{i,s}} & = \sum_j K_{ij}\tau_{j,s} + \beta_s - \sqrt{\frac{d_T+1}{d_T-1}} - \mu_i & \forall s = n-1, y_i \\ \frac{\partial \mathcal{Q}}{\partial \tau_{i,s}} & = \sum_j K_{ij}\tau_{j,s} + \beta_s - \mu_i & \forall s \neq n-1, y_i \\ \frac{\partial \mathcal{Q}}{\partial \beta_{n-1}} & = \sum_i \tau_{i,n-1} + C\sqrt{\frac{d_T+1}{d_T}} - \rho & \\ \frac{\partial \mathcal{Q}}{\partial \beta_s} & = \sum_i \tau_{i,s} - \rho & \forall s \neq 0 \end{array}$$

If x_i is an ESV but not an EEV or an IEV then, using (27) and definition 5:

$$\begin{aligned} \tau_{i,n-1} &\in \left(-\sqrt{\tfrac{d_T}{d_T+1}}\tfrac{1}{\nu}\tfrac{1}{n-1}\tfrac{C}{N_{y_i}}, 0\right) \\ \tau_{i,s} &\in \left(-\sqrt{\tfrac{d_T}{d_T+1}}\tfrac{1}{\nu}\tfrac{1}{n-1}\tfrac{C}{N_{y_i}}, 0\right] \ \forall s \neq n-1 \end{aligned}$$

It follows that for optimality:

$$\begin{aligned} \mu_i &= \textstyle\sum_j K_{ij}\tau_{j,n-1} + \beta_{n-1} - \sqrt{\frac{d_T+1}{d_T-1}} \\ \mu_i &= \textstyle\sum_j K_{ij}\tau_{j,y_i} + \beta_{y_i} - \sqrt{\frac{d_T+1}{d_T-1}} \\ \mu_i &\geq \textstyle\sum_j K_{ij}\tau_{j,s} + \beta_s \ \ \forall s \neq n-1, y_i \end{aligned}$$

and hence:

$$\begin{aligned} \mathbf{u}_{(n);y_i}^T \mathbf{g}\left(\vec{\varphi}\left(x_i\right)\right) &= \mathbf{u}_{(n);n-1}^T \mathbf{g}\left(\vec{\varphi}\left(x_i\right)\right) \\ \mathbf{u}_{(n);y_i}^T \mathbf{g}\left(\vec{\varphi}\left(x_i\right)\right) &> \mathbf{u}_{(n);s}^T \mathbf{g}\left(\vec{\varphi}\left(x_i\right)\right) \ \ \forall s \neq n-1, y_i \\ \mathbf{u}_{(n);n-1}^T \mathbf{g}\left(\vec{\varphi}\left(x_i\right)\right) &> \mathbf{u}_{(n);s}^T \mathbf{g}\left(\vec{\varphi}\left(x_i\right)\right) \ \ \forall s \neq n-1, y_i \end{aligned}$$

from which it may be seen from (7) that $\vec{\varphi}\left(x_i\right) \in \widetilde{\mathbb{G}}_{y_i} \cap \widetilde{\mathbb{G}}_{n-1}$ - i.e. x_i is a boundary anomaly vector.

If x_i is an EEV but not an IEV then, using (27) and definition 5:

$$\begin{aligned} \tau_{i,n-1} &= -\sqrt{\tfrac{d_T}{d_T+1}}\tfrac{1}{\nu}\tfrac{1}{n-1}\tfrac{C}{N_{y_i}} \\ \tau_{i,s} &\in \left(-\sqrt{\tfrac{d_T}{d_T+1}}\tfrac{1}{\nu}\tfrac{1}{n-1}\tfrac{C}{N_{y_i}}, 0\right] \ \forall s \neq n-1 \end{aligned}$$

It follows that for optimality:

$$\begin{aligned} \mu_i &\leq \textstyle\sum_j K_{ij}\tau_{j,n-1} + \beta_{n-1} - \sqrt{\frac{d_T+1}{d_T-1}} \\ \mu_i &= \textstyle\sum_j K_{ij}\tau_{j,y_i} + \beta_{y_i} - \sqrt{\frac{d_T+1}{d_T-1}} \\ \mu_i &\geq \textstyle\sum_j K_{ij}\tau_{j,s} + \beta_s \ \ \forall s \neq n-1, y_i \end{aligned}$$

and hence:

$$\begin{aligned}
\mathbf{u}_{(n);y_i}^T \mathbf{g}\left(\vec{\varphi}\left(x_i\right)\right) &\leq \mathbf{u}_{(n);n-1}^T \mathbf{g}\left(\vec{\varphi}\left(x_i\right)\right) \\
\mathbf{u}_{(n);y_i}^T \mathbf{g}\left(\vec{\varphi}\left(x_i\right)\right) &> \mathbf{u}_{(n);s}^T \mathbf{g}\left(\vec{\varphi}\left(x_i\right)\right) \quad \forall s \neq n-1, y_i \\
\mathbf{u}_{(n);n-1}^T \mathbf{g}\left(\vec{\varphi}\left(x_i\right)\right) &> \mathbf{u}_{(n);s}^T \mathbf{g}\left(\vec{\varphi}\left(x_i\right)\right) \quad \forall s \neq n-1, y_i
\end{aligned}$$

from which it may be seen from (7) that $\vec{\varphi}\left(x_i\right) \in \text{int}(\widetilde{\mathbb{G}}_{n-1}) \cup (\widetilde{\mathbb{G}}_{y_i} \cap \widetilde{\mathbb{G}}_{n-1})$ - i.e. x_i is an anomaly vector. ■

References

E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 9–16, 2000.

E. Alpaydin and C. Kaynak. Cascading classifiers. *Kybernetika*, 34(4):369–374, 1998.

K. Altun and B. Barshan. Human activity recognition using inertial/magnetic sensor units. In *Proceedings of the First International Workshop on Human Behavior Understanding (in conjunction with the* 20^{th} *Inernational Conference on Pattern Recognition)*, August 2010.

K. Altun, B. Barshan, and O. Tunçel. Comparative study on classifying human activities with miniature inertial and magnetic sensors. *Pattern Recognition*, 43(10):3605–3620, October 2010.

D. Anguita, A. Ghio, O. Luca, X. Parra, and J. L. Reyes-Oritiz. A public domain dataset for human activity recognition using smartphones. In *Proceedings of the* 21^{st} *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN2013)*, April 2013.

M. A. Armstrong. *Basic Topology.* Springer-Verlag, New York, 1979.

B. Barshan and M. C. Yüksek. Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units. *The Computer Journal*, 57 (11):1649–1667, November 2014.

P. L. Bartlett and M. H. Wegkamp. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9:1823–1840, June 2008.

A. Ben Tal and A. Nemirovski. *Lectures on Modern Convex Optimization.* MPS-SIAM Series on Optimization. SIAM, Philadelphia, 2001.

S. Boyd and L. Vandenberghe. *Convex Optimization.* Press Syndicate of the University of Cambridge, Cambridge, 2004.

S. Bulusu, B. Kailkhura, B. Li, P. K. Varshney, and D. Song. Anomalous instance detection in deep learning: A survey, 2020. URL `https://arxiv.org/abs/2003.06979`.

C. Campbell and K. P. Bennett. A linear programming approach to novelty detection. *Advances in Neural Information Processing Systems*, 13:395–401, 2001.

R. Chalapathy and S. Chawla. Deep learning for anomaly detection: A survey, 2019. URL `https://arxiv.org/abs/1901.03407`.

V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), July 2009. ISSN 0360-0300.

C. K. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, Jan 1970.

M. Claesen, F. De Smet, and J. A. K. Suykens. EnsembleSVM: A library for ensemble learning using support vector machines. *Journal of Machine Learning Research*, 15:141–145, 2014.

J. A. Cochran. *The Analysis of Linear Integral Equations*. McGraw-Hill Book Company, 1972.

D. L. Cohn. *Measure theory*. Birkhäuser, 1980.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, UK, 2005.

C. P. Diehl and G. Cauwenberghs. SVM incremental learning, adaptation and optimization. In *Proceedings of the 2003 International Joint Conference on Neural Networks*, pages 2685–2690, 2003.

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

S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognition*, 58:121–134, 2016.

G. Fumera and F. Roli. Support vector machines with embedded reject option. In S.-W. Lee and A. Verri, editors, *Proceedings of Pattern Recognition with Support Vector Machines: First International Workshop*, pages 68–82, Berlin, Heidelberg, August 2002. Springer Berlin Heidelberg.

I. S. Gradshteyn and I. M. Ryzhik. *Table of Integrals, Series, and Products*. Academic Press, London, 2000.

E. Grall-Maës, P. Beauseroy, and A. Bounsair. Quality assessment of a supervised multilabel classification rule with performance constraints. In *Proceedings of the* 14$^{\text{th}}$ *European Signal Processing Conference*, September 2006.

Y. Grandvalet, A. Rakotomamonjy, J. Keshet, and S. Canu. Support vector machines with a reject option. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 537–544. Curran Associates, Inc., 2009.

J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.*, 29(7):1645–1660, Sep 2013.

T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(2):451–471, 1998.

R. Herbrich. *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press, 2002.

B. Johnson, R. Tateishi, and Z. Xie. Using geographically weighted variables for image classification. *Remote Sensing Letters*, 3(6):491–499, November 2012.

R. H. Kassel. *A Comparison of Approaches to On-line Handwritten Character Recognition*. PhD thesis, Cambridge, MA, USA, 1995. Not available from University Microfilms Int.

C. Kaynak. Methods of combining multiple classifiers and their applications to handwritten digit recognition. Master's thesis, Institute of Graduate Studies in Science and Engineering, Bogazici University, 1995.

J. Kwok. Moderating the outputs of support vector machine classifiers. *IEEE Transactions on Neural Networks*, 16(1), September 1999.

P. Laskov, C. Schafer, and I. Kotenko. Intrusion detection in unlabeled data with quarter sphere support vector machines. In *Proceedings of the Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, Dortmund, 2004.

L. M. Manevitz and M. Yousef. One-class SVMs for document classification. *Journal of Machine Learning Research*, 2:139–154, 2001.

J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Transactions of the Royal Society of London*, 209(A), 1909.

J. Munkres. *Topology*. Prentice Hall, 2 edition, 1999.

M. Nandan, P. P. Khargonekar, and S. S. Talathi. Fast SVM training using approximate extreme points. *Journal of Machine Learning Research*, 15:59–98, 2014.

A. Nemirovski. Lectures on modern convex optmization (C.O.R.E. summer school on modern convex optimization), 2005. URL `http://www2.isye.gatech.edu/~{}nemirovs/`.

C. O'Rielly, A. Gluhak, M. A. Imran, and S. Rajasegarar. Anomaly detection in wireless sensor networks in a non-stationary environment. *IEEE Communications Surveys & Tutorials*, PP(99):1–20, 2014.

G. Pang, C. Shen, L. Cao, and Hengel A. V. Deep learning for anomaly detection: A review, 2020. URL `https://arxiv.org/abs/2007.02500`.

I. Pillai, G. Fumera, and F. Roli. Multi-label classification with a reject option. *Pattern Recognition*, 46:2256–2266, 2013.

S. Rajasegarar, C. Leckie, and M. Palaniswami. Anomaly detection in wireless sensor networks. *IEEE Wireless Communication Magazine*, 15(4):34–40, 2008.

L. Ruff, J. R Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K. R. Muller. A unifying review of deep and shallow anomaly detection, 2020. URL `https://arxiv.org/abs/2009.11732`.

B. Schölkopf and A. J. Smola. New support vector algorithms. Technical Report NeuroCOLT2 Technical Report Series, NC2-TR-1998-031, Royal Holloway College, University of London, UK, November 1998.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, Massachusetts, 2001.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

A. Shilton. SVMHeavy: SVM, machine learning and optimisation libary. `https://github.com/apshsh/SVMHeavy`, 2001–2020.

A. Shilton and M. Palaniswami. A unified approach to support vector machines. In B. Verma and M. Blumenstein, editors, *Pattern Recognition Technologies and Applications: Recent Advances*, pages 299–324. IGI Global Press, Hershey, 2008.

A. Shilton, M. Palaniswami, D. Ralph, and A. C. Tsoi. Incremental training of support vector machines. *IEEE Transactions on Neural Networks*, 16(1):114–131, January 2005.

A. Shilton, D. T. H. Lai, and M. Palaniswami. The conic-segmentation support vector machine - a target space method for multiclass classification. In *The 2012 International Joint Conference on Neural Networks (IJCNN2012)*, pages 1–8, June 2012.

A. Shilton, S. Rajasegarar, and M. Palaniswami. Combined multiclass classification and anomaly detection for large-scale wireless sensor networks. In *Proceedings of the IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 491–496, 2013.

A. Shilton, S. Rajasegarar, C. Leckie, and M. Palaniswami. DP1SVM: A dynamic planar one-class support vector machine for internet of things environment. In *2015 International Conference on Recent Advances in Internet of Things (RIoT)*, pages 1–6, April 2015.

I. Steinwart and A. Christman. *Support Vector Machines*. Springer, 2008.

J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific Publishing, New Jersey, 2002.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Advances in neural information processing systems*, pages 25–32, 2004.

D. M. J. Tax and R. P. W. Duin. Support vector data description. *Machine Learning*, 54 (1):45–66, 2004.

V. Vapnik. *Statistical Learning Theory*. Springer-Verlag, New York, 1995.

D. Wang, D. S. Yeung, and E. C. C. Tsang. Structured one-class classification. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 36(6):1283–1295, December 2006.

J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

J. Weston and C. Watkins. Support vector machines for multi-class pattern recognition. In *ESANN'1999 proceedings - European Symposium on Artificial Neural Networks*, 1999.