A classification method based on a cloud of spheres ⋆

Tiago Dias[a], Paula Amaral[a,b]

[a] *Nova SST—FCT Nova, Campus de Caparica, 2829-516 Caparica, Portugal.*
[b] *NovaMath, CMA Nova, Campus de Caparica, 2829-516 Caparica, Portugal.*

Abstract

In this article we propose a binary classification model to distinguish a specific class that corresponds to a characteristic that we intend to identify (fraud, spam, disease). The classification model is based on a cloud of spheres that circumscribes the points of the class to be identified. It is intended to build a model based on a cloud and not on a disjoint set of clouds, establishing this condition on the connectivity of a graph induced by the spheres. To solve the problem, designed by a Cloud of Connected Spheres, a quadratic model with continuous and binary variables (MINLP) is proposed with the minimization of the number of spheres. The issue of connectivity implies in many models the imposition of an exponential number of constraints. However, because of the specific conditions of the problem under study, connectivity is enforced with linear constraints that scale quadratically with K, which serves as an upper bound on the number of spheres. This classification model is effective when the structure of the class to be identified is highly non-linear and non-convex, also adapting to the case of linear separation. Unlike neural networks, the classification model is transparent, with the structure perfectly identified. No kernel functions are used and it is not necessary to use meta-parameters unless it is intended also to maximize the separation margin as it is done in SVM. Finding the global optima for large instances is quite challenging, and to address this, a heuristic is proposed. The heuristic demonstrates nice results on a set of frequently tested real problems when compared to state-of-the-art algorithms.

⋆This work is funded by national funds through the FCT - Fundação para a Ciência e a Tecnologia, I.P., under the scope of the project UIDB/00297/2020 (Center for Mathematics and Applications).

Email address: tme.dias@campus.fct.unl.pt (Tiago Dias)

Keywords: Automatic Classification, MINLP, Spherical separation, Anomaly detection.

1. Introduction

Automatic Classification, in a broad sense, consists in assigning to an object, represented by a $n \in \mathbb{N}$ dimensional vector of characteristics or features, a K-dimensional vector of class membership, $\mu(k)$, for $k \in K$. It is assumed that each individual class is defined by a certain property, and all objects having that property to some degree are elements of that class. We may define the classification as crisp if $\mu(k) = \{0, 1\}$ and $\sum_{k \in K} \mu(k) = 1$, meaning that the object belongs strictly to a class, or as soft or fuzzy ([1, 2]) if $0 \leq \mu(k) \leq 1$ and $\sum_{k \in K} \mu(k) = 1$, meaning that an object may belong to more than one class with different membership values. As examples of crisp classification, given an image of a pet we define if it is a cat or a dog, or given information about a bank transaction we aim to classify it as fraudulent or not. In fuzzy classification, for instance, a bank transaction can be fraudulent or legal to some extent.

The oracle that makes the classification can be trained based on a set of objects or points, defined as the training set, to which, besides the vector of features, the class $k \in K$ to which it belongs is known. This type of methods are known as supervised methods. Alternatively, unsupervised methods are constructed with the assumption that the class is not known. The architecture of supervised algorithms can be quite different, but generally speaking is based on a model that depends on some parameters that must be estimated (training phase) maximizing a fitting function (or minimizing an error function) that returns a numerical value representing the agreement between the observed labels and their classification (prediction) using the model. In general, the fitting or error function is not globally optimized for the training data, to avoid over-fitting, preventing poor generalizations for new data for which the class is, obviously, unknown.

In this article, we propose a binary classification model based on a cloud of spheres. The arguments in defense of this new model relate to the ability to reproduce highly non-linear and non-convex separation structures, that unlike black-box methods are transparent. In addition it is not necessary to

use meta-parameters. We propose a quadratic mixed integer formulation for this problem and report some results for small\medium size problems. Since finding the global optima is hard for large instances, a heuristic approach is proposed.

The following section presents contributions within the scope of supervised classification models and discusses the weaknesses of black-box methods. Section 3 introduces the motivation and the Cloud of Connected Spheres Problem, followed by the section where a MINLP formulation for this problem is proposed. A heuristic approach is addressed in section 5, and some numerical experiences are reported in section 6. Finally, section 7 closes this article with conclusions and future work.

2. State of the art

The list of classification methods is extensive, diverse and plural. In the specific case of supervised Machine Learning (ML) classification tasks, there are several well-established methods. We highlight the following:

- Logistic Regression (LR): One of the earliest classification methods [3, 4], it consists of a linear classifier that, instead of predicting the association of an instance with a class, predicts the probability of an instance belonging to a particular class. A threshold on this probability can be used for binary classification.

- Naive Bayes (NB): This method [5, 6] infers the probability that a new instance belongs to some class based on the assumption that all attributes are independent of each other (i.e. applying Bayes' theorem with the strict assumption of conditional independence). It is worth mention that even if this conditional independence assumption is not valid (which rarely is in practical learning cases), experiments on real-world data have shown some competitiveness regarding more sophisticated induction algorithms.

- K-nearest neighbor (KNN): more traditional but still commonly used in the scientific community [7, 8, 9]. This method assigns each observation the most common class of its k nearest neighbors, in accordance with a selected distance function.

- Decision Trees (DT): Non-parametric models, which predict the class of an instance by following a hierarchical sequence of decision rules, forming a tree like structure [10]. DT methods respect a top-down framework: departing from a root node, a data set is broken down into homogeneous subsets (*i.e.* recursively partitioning), while at the same time an associated decision tree is incrementally developed - each node represents a feature in an instance subject to the classification process, and each branch represents a value that the node can assume. The final result is a tree with decision and leaf nodes, which represent the classification [11], that can be reinforced by ensemble learning (*e.g.* random forest) [12]. One of the caveats of DT is over-fitting.

- Random Forests (RF): Also known as random decision forests, it is a class of methods for classification or regression based on the construction of a multitude of decision trees [13]. For classification purposes, the output of the RF is the class selected by most trees. RF generally outperform DT, avoiding its over-fitting caveat. Still, their accuracy is lower than gradient boosted trees, being the RF performance more exposed to the data characteristics than the latter [14].

- Neural network (ANN): An artificial neutral network is a system based on the biological neural network, such as the human brain. This architecture enables the development of complex relationships between inputs and outputs, through a succession of hidden layers of neurons/nodes [15]. ANNs are comprised of node layers, containing an input layer, one or more hidden layers, and an output layer. Each node connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network. Neural network based algorithms such as deep neural networks (DNN), convolutional neural networks (CNN), recurrent neural networks (RNN), deep belief network (DBN), hierarchical attention networks (HAN), and combination techniques have gained some popularity in recent years. [16].

Despite all of the enumerated methods have their own merits and research focus, the Support Vector Machine (SVM) method , especially kernel SVMs, are of particular interest in the Optimization realm [17], not only

for the successful applications, sometimes outperforming more sophisticated methods like ANN [18, 19], but also for its elegant mathematical model, clear definition of the structure defining the class separation and interesting generalization.

- Support Vector Machines (SVMs): Firstly introduced by Vapnik in 1995 [20], the classical SVM model separates the instances of a data set into two distinct classes, by means of a hyperplane. The parameters of the hyperplane are obtained by the maximization of the minimal distance between the points of the two different classes (*i.e.* maximization of the separation margin), whilst minimizing the occurrence of misclassification (in the case of soft margins). Since the application of traditional SVMs entails the separation of data via the use of a linear surface, the method's performance is significantly hindered in non-linearly separable data sets. This handicap has been widely addressed by leveraging the standard SVM method with the use of kernel functions, a process which enables the separation using linear surfaces by mapping the original data set into a higher dimensional space, in which such separation is possible [21].

Albeit the existence of a comprehensive literature on the application and suitability of SVM with kernel functions for classification, the use of such techniques presents drawbacks [22, 23]. First, there is no criterion for selecting an appropriate kernel function for a given data set, being the selection dependent on comparability analysis. Additionally, the performance of SVM models is highly dependent on the parameters specified in the kernel function, causing the need for substantial hyperparameter tuning.

In the past two decades, several scholars have also started to develop research on the nature of the separating surface, specifically on the definition of nonlinear surfaces, fomenting such separation at the original input space. These studies have been developed under both supervised and semi-supervised (*i.e.* a partition of the observations is unlabeled) realms, consisting in non-smooth optimization problems [24]. Some of the developed techniques make use of geometrical well-behaved structures intuition, with the likes of polyhedral, spherical, ellipsoidal and conical surfaces [25]. In the case of polyhedral separation, this concept was introduced in Megiddo

(1988) [26] and applied within the classification framework by Astorino and colleagues [27, 28, 29]. The basic idea of polyhedral separation is that if two finite disjoint set of instances A and B are not linearly separable, it is possible to define a h-polyhedral separation surface, comprised by $h > 1$ hyperplanes, such that A is in the convex polyhedron given by the intersection of h half-spaces and B lies outside such polyhedron [27]. Focusing on the particular case of spherical separation, its use was initially proposed in Tax and Duin in 1999 [30], where the main objective was set to find a minimal volume hypersphere separating two finite disjoint sets of instances A and B (i.e. a sphere enclosing all elements of A and no elements of B). Since then, further extensions of this idea have been developed and refined, highlighting the contributions of Astorino, Fudelli and Gaudioso [31, 32, 33, 34, 35], introducing techniques that tackle the minimization of error functions (i.e. penalty for misclassification), since real-life data sets are hardly separable by means of a specific surface.

2.1. A critical view over black-box machine learning methods

It is known that some ML algorithms, in particular ANN, DNN and RF have the ability of representing complex nonlinear associations in data. However, ANN and DNN are black-box algorithms, meaning that we cannot perceive what is actually learned and how it is learned. [36]. Failure of ANN and DNN has been more than ever recognized and investigated [37, 38]. The absence of a reflection on these algorithms, and an indiscriminate and uncritical application in real-world situations, can lead to very undesirable, unethical, unfair, sensitive and even dangerous results. ML models and algorithms even exhibit a propensity to amplify discrimination based on gender [39] or race [40].
Several authors have raised the awareness regarding this issue [36, 40, 41, 42, 43, 39, 38, 37]. We strongly recommend their reading, emphasizing two articles:

- in [41] where the authors "demonstrate how easy it is for modern machine-learned systems to violate common deontological ethical principles and social norms such as "favor the less fortunate, and do not penalize good attributes"";
- quoting [42] "Today, machine-learning software is used to help make decisions that affect people's lives. Some people believe that the ap-

plication of such software results in fairer decisions because, unlike humans, machine-learning software generates models that are not biased. Think again. Machine-learning software is also biased, sometimes in similar ways to humans, often in different ways."

Besides these undesirable issues, ANN and DNN are difficult to train [44]. Parameter optimization is too complex to achieve an optimum. Notwithstanding, arguments in favor of the sub-optimization of parameters have been made, alluding to the fact that it is even desirable to avoid over-fitting. In addition to the parameter's optimization, the network design itself must be evaluated. With the complexity of neural network structures, there are too many components to be evaluated (regularization, learning rate, activation functions, number of layers, hidden units, intermediate nodes). Given the impossibility of analyzing all possible combinations of structures and parameters, only a part of these combinations is evaluated. The criterion of choice is often supported by arguments that are not solid and too dependent on the training and testing data, raising questions concerning their suitability for new data.

In SVM, kernel functions are introduced to deal with non-linearity. Although Kernel SVM are more transparent than DNN, and have fewer combinations of model aspects to study and evaluate, it still exhibits some effort of a combinatorial nature in training. The right choice of the Kernel function, along with parameters and the meta-parameters in case of soft margins, can be cumbersome. Additionally, the type of linearity that can be described is limited, with strong difficulties in learning the non-convexities of the class separation curves.

3. Classification using the Connected Cloud of Spheres Problem (CCSP)

Criticism over ANN, DNN and SVM kernel methods motivated the search for more transparent and simple classification methods. Inspired by the positive aspects of SVM we looked for a model-based method, where the relation between the model and the solution could be clearly identified, with only a reasonable number of parameters to estimate, but capable of capturing complex separation structures among data.

If we consider a classification problem with two classes, finding a separation surface can be difficult, in a high dimensional space such as $\mathbb{R}^n$ for large

n, depending on the geometry of the objects or points. If the points from the two classes are linearly separable, most of the classic methods proposed in the literature will work well, such as SVM. If the points are not linearly separable, then finding a separation surface can be hard. Usually, some kind of regular behavior on the spatial distribution of the points in $\mathbb{R}^n$ is assumed (spherical, polyhedral) to justify the application of a method tailored to some assumption on the geometry of the data. But to speculate about the geometry of data in high dimensional spaces can be misleading.

When we think about the geometry of a hypercube, for instance, a four-dimensional cube (Tesseract), we intuitively try to project in our mind an extension of the idea of a square in a three-dimensional space. If we consider Fig. (1), we understand that our intuition is of little use when dealing with a n^{th} dimensional space. In classification methods we construct separation

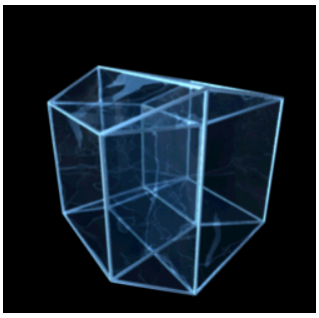


Figure 1: An image from the 3D projection of a Tesseract performing a simple rotation about a plane in 4-dimensional space.

lines, curves well defined mathematically, that, usually, generalize the concepts that we have for a two or three-dimensional space to higher dimensions. However, we may admit that better separation frontiers might have a structure and properties that have no parallel with those generalizations.

In that sense, we believe that the more flexible the separation surface is the better, allowing the definition of different and complex shapes. If we consider, for instance, the two-dimensional example in Fig.2, the separation curve defined in Fig.3 encapsulates a non-linear structure that is complex and non-convex.

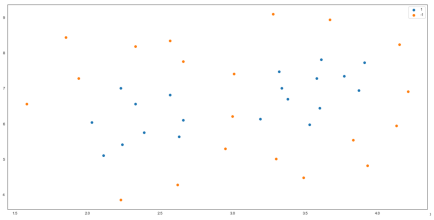


Figure 2: Two classes of points, blue and red.

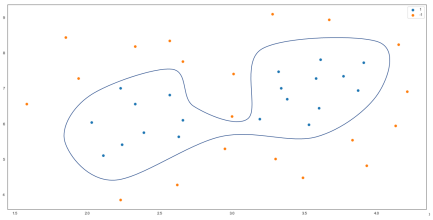


Figure 3: The same points with a separation curve in blue.

In this paper, we propose a new problem, the Connected Cloud of Spheres Problem (CCSP), to define a set of connected spheres containing only the points of one class as in Fig.4, with the purpose that the points in the boundary of the reunion of the spheres, as in Fig.5, approximate a geometric domain (the one in Fig.3) defining a separation between the two classes of points.

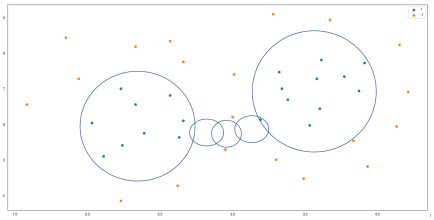


Figure 4: Connected cloud of spheres.

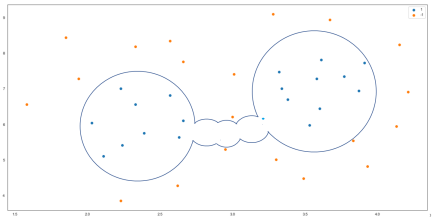


Figure 5: Boundary of the connected cloud of spheres.

The boundary that separates the two regions is completely and simply defined by the set of centers and radii. The rule for classification of a new point is a query that requires a number of comparisons equal to the size of the cloud (number of spheres). No massive meta-parameters tuning is required. As noted by one of the referees, maintaining the classification process within the original feature space instead of relying on kernel transformations, greatly enhances the transparency of the results. This plainness, simplicity and transparency are some of the advantages of this method. As in any classification problem there are a few premises:

- There is a class of objects that we want to identify in a binary scenario (e.g. fraud/legitimate, spam/non-spam, sick/healthy).

- The region separating the two classes can be highly non-convex but is connected. It is a unique region and not a collection of separated domains. This condition makes the problem harder but more general and interesting by the mathematical point of view. However, this requirement may be waived if necessary. It can be defined as a feature of the method that is optional, allowing it to be turned on or off, especially when there is suspicion of inflated false positives.
- An upper bound on the maximal necessary number of spheres is known.
- The two classes are geometrically separable, at least to a certain degree, meaning that a large subset of the data from the class to be identified is not mixed with the points from the other class.
- The classes do not need to be balanced in size, meaning that the method is suitable for anomaly detection.

In the next section we propose a Mixed Integer Nonlinear Problem (MINLP) formulation to solve the CCSP.

4. MINLP Formulation of the Connected Cloud of Spheres Problem

In this section, we consider a set of data points divided into two separable sets $X1 = \{x_{1i},\ i \in C_1\}$ and $X2 = \{x_{2i},\ i \in C_2\}$. $N_1 = \#C_1$, the size of C_1 and $N_2 = \#C_2$ are not necessarily the same. The goal is to find a connected cloud of at most K spheres with center x_{0k} and radius r_k, such that all points x_{1i} for $i \in C_1$ lay inside at least one sphere and no point x_{2i} for $i \in C_2$ belongs to any sphere. This condition can be relaxed, conducting to a soft separation problem. In addition, the set of spheres is connected. The goal is to minimize the number of spheres. Similarly to SVM, a margin can be considered, representing a positive separation distance between the two classes. In this case, we also consider the maximization of the minimal margin separation between the two classes.

4.1. Variables

We consider the following set of variables:

$$x_{0k} \in \mathbb{R}^d - \text{ center of the sphere } k \text{ for } k = 1, .., K \tag{1}$$

$$r_k \geq 0 - \text{ radius of sphere } k \text{ for } k = 1, .., K \tag{2}$$

$z_{ik} \in \{0,1\}$ – takes the value 1 if point x_i is covered by sphere k, and 0 otherwise, for $i \in C_1, k = 1, .., K$ (3)

$w_k \in \{0,1\}$ – takes the value 1 if sphere k is used, and 0 otherwise, for $k = 1, .., K$ (4)

$\delta_k \geq 0$ – separation margin of sphere k, for $k = 1, .., K$ (5)

$\delta_{min} \geq 0$ – minimal separation margin (6)

$y_{kj} \in \{0,1\}$ – takes the value 1 if sphere k and j overlap, and 0 otherwise, for $k = 1, .., K-1, j = k+1, .., K$ (7)

4.2. Objective function

The goal is to construct a cloud with as few spheres as possible.

$$\min F = \sum_{i=1}^{K} w_k \tag{8}$$

4.3. Constraints defining the set of spheres

First we consider the constraints that ensures that every points x_{1i} for $i \in C_1$ falls inside at least one sphere:

$$\sum_{k=1}^{K} z_{ik} \geq 1, \quad \text{for} \quad \forall i \in C_1 \tag{9}$$

$$||x_{0k} - x_{1i}||^2 \leq r_k^2 + (1 - z_{ik})M, \quad \text{for} \quad i \in C_1, k = 1, \ldots, K \tag{10}$$

$$z_{ik} \leq w_k, \quad \text{for} \quad i \in C_1, k = 1, \ldots, K. \tag{11}$$

where M is large enough so that if $z_{ik} = 0$ the constraint (10) is redundant. On the other hand, it is known that too large values for M can interfere with the performance of the method. A reasonable choice for M could be the maximum distance between any two points from $X1$ and $X2$.
To guarantee that no point x_i for $i \in C_2$ belongs to any existing ($w_k = 1$) sphere k we have:

$$||x_{0k} - x_{2i}||^2 \geq r_k^2 - M(1 - w_k), \quad \text{for} \quad i \in C_2, k = 1, \ldots, K. \tag{12}$$

4.4. Constraints defining a sequential numbering of the spheres

To ensure that the index of the spheres created respect a lexicographic order we also consider the constraints,

$$w_k \geq w_{k+1}, \quad \text{for} \quad k = 1, \ldots, K-1. \tag{13}$$

We can only assign a label $k+1$ to a new sphere if there is a sphere with label k. This constraint prevents the creation, for instance, with $K = 4$, of two spheres with labeling $2, 4$.

4.5. Constraints defining the connectivity of the graph induced by the cloud of the spheres

To ensure that the cloud of spheres is connected, we can resort to the representation of the problem in a graph $G = (V, E)$ where the set of vertices V is defined by the index of the spheres and an edge between spheres i and j exists if they intersect, this is if $||x_{0i} - x_{0j}||^2 < (r_i + r_j)^2$, case in which the variable y_{ij} (7) assumes the value 1. The connectivity of the cloud of spheres is equivalent to the connectivity of graph G.

Definition 4.1. *A graph $G = (V, E)$, where V and E are respectively the set of vertices and edges, is connected if there is a path between every pair of vertices i and j with $i, j \in V$.*

In Fig.6 we exemplify a disconnected graph induced by a set of three spheres.

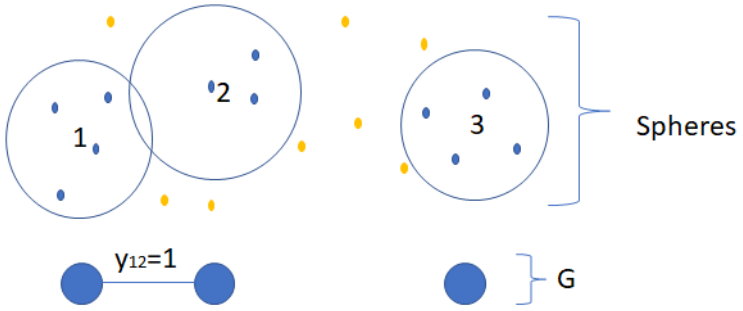


Figure 6: Graph induced by the spheres.

To ensure that the variables y_{ij} represent the intersection of spheres i and j,

as expressed in (7), the following constraints must be considered:

$$M(y_{ij}-1) \leq (r_i + r_j)^2 - ||x_{0i} - x_{0j}||^2 \leq y_{ij}M, \quad \text{for} \quad i = 1, \ldots, K-1, \; j = i+1, .., K \tag{14}$$

$$y_{ij} \leq \frac{w_i + w_j}{2} \quad \text{for} \quad i = 1, \ldots, K-1, j = i+1, .., K \tag{15}$$

Graph connectivity is a well-studied subject with many contributions in terms of characterization and detection through the application of algorithms. [45, 46]. However, these contributions are of little use here since we need necessary and sufficient conditions that can be formulated as constraints in our model. For instance, the necessary condition expressed by the inequality

$$\sum_{i=1}^{n} \rho(i) \geq \frac{1}{2}(n-1), (\text{where } \rho(i) \text{ is the degree of vertex } i)$$

for a connected graph with n vertices, could be easily formulated but it is only necessary and it is easy to check that the bound is not tight even for very small graphs. Connectivity constraints are important in many spatial optimization problems such as forest harvesting [47], network design [48], pursuit evasion problems [49], sensor networks [50, 51], among others. Some integer programming formulations to imposing connectivity have been proposed [47], but most are impractical for large scale instances. The classical constraint, from which other are derived, imposes the existence of at least one edge linking any subset of nodes to its complement set. This condition could be easily formulated within the structure of our model.

$$\sum_{i \in S} \sum_{j \notin S} y_{ji} \geq 1 \quad \forall S \subset \{1, 2, \ldots K\} \tag{16}$$

The downside of this approach is that it exponentially generates many constraints. However, we may use a major and crucial advantage in our model - the numbering (labeling) of the nodes is not predefined and can be set in any convenient way.

This simple feature of the mathematical formulation will allow writing connectivity constraints in a much simpler way. First, we have to stress that the specific way the numbering of spheres is done is irrelevant regarding the previously defined constraints, allowing complete freedom in that choice. So far, the only imposition introduced was through constraints (13), with the sole purpose of avoiding, for instance, having a cloud of three spheres labelled 3, 6, 8 instead of 1, 2 and 3. Still, what is the first, second and third sphere is irrelevant. For this reasoning, we may number the spheres in the following way:

- Label 1 is assigned to a sphere (any sphere).
- If sphere 2 exists ($w_2 = 1$) then sphere 2 must be connected to sphere 1, this is, $y_{12} \geq w_2$.
- If sphere 3 exists ($w_3 = 1$) then sphere 3 must be connected to sphere 1 or 2, this is, $y_{13} + y_{23} \geq w_2$.

Continuing this reasoning, we have the following set of constraints to ensure connectivity:

$$\sum_{i<j} y_{ij} \geq w_j \quad \forall j \in \{1, 2, \dots K\} \qquad . \tag{17}$$

If $w_j = 0$, constraints (17) become redundant. The number of constraints (17) is $K - 1$ and they are linear. It is easy to prove, by induction on the number of nodes, that using the constraints (17) the graph induced by the cloud of spheres is connected.

Theorem 4.1. *A graph $G = (V, E)$ with n vertices, such that we may label the set of nodes $i = 1, \dots, n$ in such a way that for every node i there is at least a node $j < i$ such that the edge $(j, i) \in E$, is connected.*

Proof. Let $G_n = (V(n), E(n))$ be the sub-graph of G, induced by the set of vertices $1, \dots, n$. If $n = 1$ the singleton graph is connected. Now suppose that the graph G_n is connected and such that $\forall i \in \{1, ..., n\}$ there is a $j \in \{1, ..., i-1\}$ such that a edge $(j, i) \in E(n)$. Considering the graph G_{n+1} if an edge $(j, n+1) \in E(n+1)$ exists then G_{n+1} is connected. □

Example 4.1. *Let us consider two examples (Butterfly in Fig. (7) and Separation in Fig. (8)) where the data of the Class 1 (blue stars) is surrounded*

by points of the Class 2 (red crosses) and it forms a non-convex structure. In Butterfly the points of Class 1 are more separated than in Separation. Solving the CCSP, we obtain the solutions depicted in Fig. (7) and (8) where the cloud has four and two spheres respectively.

Figure 7: Case Butterfly

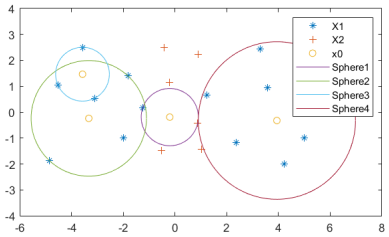
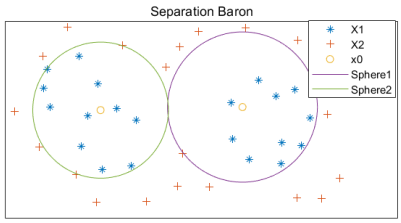


Figure 8: Case Separation



4.6. CCSP with margins

To encompass the existence of a margin in the separation of classes, as it is done in SVM, we need to modify the constraints (10) and (12) in order to incorporate this concept.

$$||x_{0k} - x_{1i}||^2 \leq (r_k^2 - \delta_k) + (1 - z_{ij})M, \quad \text{for} \quad i \in C_1, k = 1, \ldots, K \tag{18}$$

$$||x_{0k} - x_{2i}||^2 \geq (r_k^2 + \delta_k) - M(1 - w_k), \quad \text{for} \quad i \in C_2, k = 1, \ldots, K. \tag{19}$$

To maximize the minimum margin we have

$$\delta_k \geq \delta_{min} \tag{20}$$

$$\max G = \delta_{min} \tag{21}$$

To avoid considering a multi objective optimization problem we may aggregate objective functions (8) and (21) as:

$$\min G = \sum_{i=1}^{K} w_k - \theta\delta_{min}. \tag{22}$$

Example 4.2. *To visualize the solution obtained by using the CCSP with margin we present in (9), (10) and (11) for the data set Separation, the clouds for different θ values.*

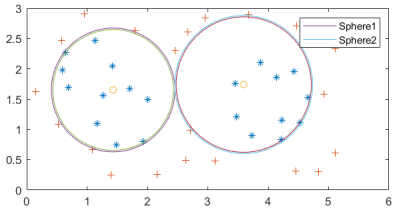


Figure 9: $\theta = 0.1$

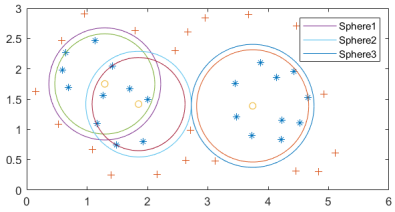


Figure 10: $\theta = 10$

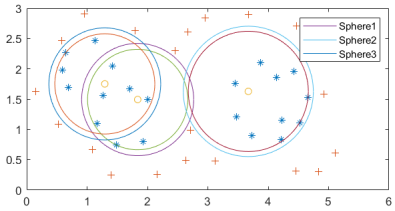


Figure 11: $\theta = 100$

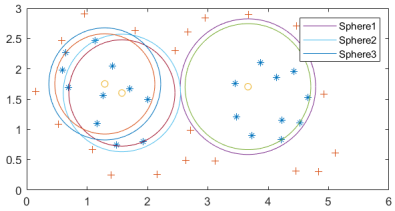


Figure 12: $\theta = 1000$

5. Heuristic approach

Given the difficulty in finding an optimal solution by an exact method, in this section, we propose an algorithmic approach. The general structure of the algorithm is defined in Algorithm 1. The objective of the heuristic is as follows: cover all points of a class (C_1) with a connected set of spheres S, while guaranteeing that no point of the other class (C_2) is covered by S. At any given iteration a new sphere S_k is added to S, updating set of spheres S and the set of covered C_1 points. The algorithm stops when all point of class C_1 are covered, returning set S. To achieve the expected result, each iteration of the algorithm is two-fold divided: first the iteration's candidate sphere S_k is defined (Algorithm 2) and then the connectivity of S_k to the already defined spheres in S is ensured (Algorithm 3).

Algorithm 1 Heuristic: Largest Coverage - General Framework

Data: Set of points of both classes
$C_1 = \{x_{1i},\ i = 1, \ldots, N_1\},\ C_2 = \{x_{2j},\ j = 1, \ldots, N_2\}$

Result: Set of spheres S

```
t ← 1                                          ▷ Iteration count
I = C_1                                        ▷ Set of uncovered C_1 points by S
while I ≠ {} do
    Define the iteration's candiate sphere S_k
    Guarantee Connectivity of S_k to the already defined spheres in S
    Update S and I
end
```

5.1. Define the iteration's candidate sphere S_k

To find an iteration's candidate sphere, the following steps are required (detailed scheme in Algorithm 2):

1. Definition of an initial candidate S_k
   (a) P is defined as the pairing set of every uncovered C_1 class point to its closest C_2 class point. The pairing of P in which the two points are furthest from each other (x_{1k}, x_{2j}) is selected.
   (b) The center and radius of S_k are derived from (x_{1k}, x_{2j}).
      - $x_{0k} = x_{1k}$ is set as the initial center of S_k.
      - The Euclidean distance between x_{0k} and x_{2j} becomes the radius r_k.
   (c) Z_{ik} is defined as the set of uncovered C_1 points covered by S_k.
2. Improvement attempts to increase the set of covered C_1 points by S_k
   (a) Using the notion of centroid, a new sphere S_c is envisioned.
      - The sphere has its center at c_i, the centroid of Z_{ik}.
      - Its radius (r_c) is defined as the Euclidean distance between c_i and its closest C_2 class point (c_j).
   (b) C_{ik} is defined as the set of uncovered C_1 points covered by S_c.
   (c) Comparison between the sets Z_{ik} and C_{ik} is made
      - If $C_{ik} \subset Z_{ik}$, the initial candidate S_k is accepted.
      - If $C_{ik} == Z_{ik}$, S_c becomes S_k. The updated S_k is accepted.
      - Otherwise, the initial candidate S_k is updated:
        – c_i, r_c and C_{ik} become the updated values of x_{0k}, r_k and Z_{ik}, respectively.
        – New values for c_i, r_c and C_{ik} are computed.

- The comparison process is repeated until $C_{ik} == Z_{ik}$.

Algorithm 2 Largest Coverage - Define iteration's candidate sphere

Data: I, C_2
Result: S_k

$P \leftarrow \{(x_{1i}, x_{2j}) \mid x_{1i} \in I,\ x_{2j} = \underset{(x_{2j'} \in C_2)}{\arg\min}\ d(x_{1i}, x_{2j'})\}$ ▷ Pairing set P

$(x_{1k}, x_{2j}) \leftarrow \underset{((x_{1i}, x_{2j}) \in P)}{\arg\max}\ d(x_{1i}, x_{2j})$ ▷ Finding the maximum distance pairing

$x_{0k} = x_{1k}$ ▷ Center of S_k

$r_k \leftarrow d(x_{1k}, x_{2j})$ ▷ Radius of S_k

$Z_{ik} \leftarrow \{x_{1i} \mid x_{1i} \in I \wedge ||x_{0k} - x_{1i}||^2 \leq (r_k)^2\}$ ▷ Set of points covered by S_k

$c_i \leftarrow Centroid(Z_{ik})$

$c_j \leftarrow \underset{(x_{2j} \in C_2)}{\arg\min}\ d(c_i, x_{2j})$ ▷ Finding the closest C_2 point to c_i

$r_c \leftarrow d(c_i, c_j)$ ▷ Radius of S_c

$C_{ik} \leftarrow \{x_{1i} \mid x_{1i} \in I \wedge \|c_i - x_{1i}\|^2 \leq (r_c)^2\}$ ▷ Set of points covered by S_c

$S_k \leftarrow \{(x_{0k}, r_k)\}$ ▷ Defining the candidate sphere

$S_c \leftarrow \{(c_i, r_c)\}$ ▷ Defining the sphere centred at c_i

$StopFlag \leftarrow 0$
while $StopFlag == 0$ **do**
  if $C_{ik} \subset Z_{ik}$ **then**
    $StopFlag = 1$ ▷ S_k cannot be improved
  else
    if $C_{ik} == Z_{ik}$ **then**
      $StopFlag = 1$ ▷ S_k cannot be further improved
      $x_{0k} = c_i$ ▷ S_k center and radius are updated
      $r_k = r_c$
    else
      $x_{0k} = c_i$ ▷ S_k center and radius are updated
      $r_k = r_c$
      $Z_{ik} := C_{ik}$ ▷ Z_{ik} is updated
      $c_i := Centroid(Z_{ik})$ ▷ Centroid of the updated Z_{ik} set
      $c_j \leftarrow \underset{(x_{2j} \in C_2)}{\arg\min}\ d(c_i, x_{2j})$
      $r_c \leftarrow d(c_i, c_j)$
      $C_{ik} := \{x_i \mid x_i \in I \wedge ||c_i - x_i||^2 \leq (r_c)^2\}$ ▷ Updated C_{ik} set
      $S_c \leftarrow \{(c_i, r_c)\}$ ▷ Updated S_c
    end
  end
end

Example 5.1. *Let us consider the example of (Fig. 8). At iteration 1 of the proposed heuristic, the initially proposed S_k suffers an updated before the algorithm advances to stage 2 (Algorithm 3), as portrayed in Fig. (13).*

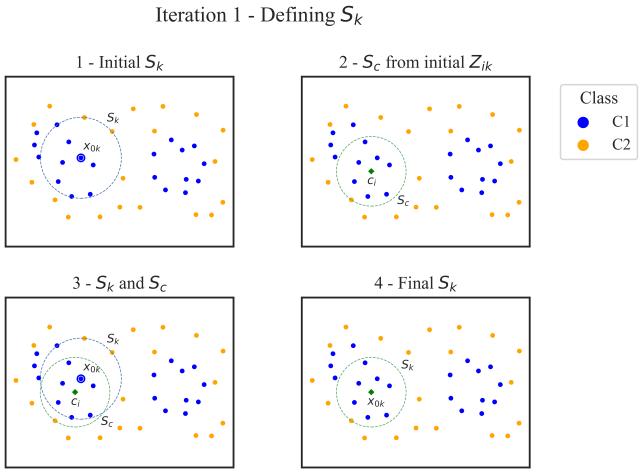


Figure 13: Generation of S_k at iteration 1 for the *Separation* data set.

5.2. Guarantee Connectivity of S_k to the already defined spheres in S

Once the iteration's candidate sphere is defined it is necessary to ensure its connectivity to S, which is not guaranteed on the previous stage. The first stage of the heuristic (Algorithm 2) does not take into account the existence of the previously defined spheres of S. Instead, an attempt to form a sphere with the largest possible radius based solely on the information about the original data set is performed, which may yield a sphere that is not connected to S. Hence, if $S \cup S_k$ does not form a connected set, before moving to the next iteration, artificial spheres will be added to S to preserve its connectivity property. These artificial spheres may not include any uncovered point.

The second stage of the heuristic (Algorithm 3) respects the following structure: if the algorithm is in the first iteration, no connectivity test is required. S_k is added to S, and the algorithm advances to the next iteration; Otherwise, it is accessed whether S_k is connected to any sphere of S.

1. To make such assessment, the Euclidean distance between the center of S_k to the centers of every sphere of S is calculated. This distance is then compared with the sum of the radius of each pairing of spheres.
    - If the distance between centers of any of the pairings is less or equal to the sum of the radius of the two spheres, then S_k is connected to at least one sphere of S. S_k is added to S, the set

of uncovered points (I) is updated and the algorithm advances to the next iteration.

- Otherwise, S_k is not connected to S. N_{S_k} is defined as the closest sphere of S to S_k. This sphere is centered at n_i, having a radius of r_n.

2. If S_k is not connected to S, an artificial sphere that connects S_k to N_{S_k} is envisioned (S_{art}). This sphere has the smallest radius possible, being centered at the midpoint of the line segment that represents the smallest distance between the spheres N_{S_k} and S_k.
   - $\overrightarrow{s}$ is set as the vector from the center of N_{S_k} to the center of S_k.
   - The closest points of N_{S_k} and S_k to the opposite sphere are defined as m_N and m_K, respectively. The midpoint between m_N and m_K is set as mid.
   - The closest C_2 class point to mid is defined as n_{mid}.

   (a) If $d(mid, m_N) \leq d(mid, n_{mid})$, then S_{art} is added to S, connecting S_k to N_{S_k}. S_k is added to S, Z_{ik} and I are updated and the algorithm advances to the next iteration.

   (b) Otherwise, it is not possible to connect S_k to N_{S_k} with a single artificial sphere of minimum radius.

3. Not being possible to connect S_k to N_{S_k} with a single artificial sphere of minimum radius, another artificial sphere is envisioned (S_{art}). This sphere increases the coverage of S towards S_k.
   - To find this sphere, first the relevant points of class C_2 (i.e. closest points to both N_{S_k} and S_k) must be identified. Once identified, by taking into account the position of these points in relation to N_{S_k}, a sphere connected to N_{S_k} that achieves the highest possible coverage will be formed.
   - The identification of the relevant C_2 points is also made by the use of spherical forms: a sphere (S_a) with center in the midpoint between the centers of N_{S_k} and S_k and radius equal to half of the distance between the two centers is envisioned.
   - For every relevant point, a sphere between that point and its closest point on the surface of N_{S_k} is envisioned. Of this group of spheres, the one which achieves the largest coverage while not covering any of the relevant points (S_{art}) is added to S.

- S and Z_{ik} are updated. The algorithm goes back to step 2.
- The process ends when S_k becomes connected with S.

Algorithm 3 Largest Coverage - Guarantee Connexity & Update S

Data:
S_k, I, C_2
Result:
S, I, t

$StopFlag = 0$
while $StopFlag == 0$ **do**
  if $t == 1$ **then**
    $StopFlag = 1$ ▷ No connectivity test is required
  else
    $(n_i, r_n) \leftarrow \underset{(\{n_i, r_n\} \in S)}{\arg\min} [d(x_{0k}, n_i) - (r_k + r_n)]$ ▷ Finding the closest sphere to S_k
    $N_{S_k} \leftarrow (n_i, r_n)$
    if $d(x_{0k}, n_i) \leq (r_k + r_n)$ **then**
      $StopFlag = 1$ ▷ S_k is connected to at least one sphere of S
    else
      $\overrightarrow{s} \leftarrow \overrightarrow{n_i\ x_{0k}}$ ▷ An artificial sphere must be introduced
      $m_N \leftarrow n_i + \frac{r_n}{||\overrightarrow{s}||} \times \overrightarrow{s}$ ▷ Closest point on the surface of N_{S_k} to S_k
      $m_K \leftarrow x_{0k} - \frac{r_k}{||\overrightarrow{s}||} \times \overrightarrow{s}$ ▷ Closest point on the surface of S_k to N_{S_k}
      $mid \leftarrow \frac{m_N + m_K}{2}$ ▷ Midpoint between m_N and m_K
      $n_{mid} \leftarrow \underset{(x_{2j} \in C_2)}{\arg\min}\ d(mid, x_{2j})$ ▷ Closest C_2 point to mid
      if $d(mid, m_N) \leq d(mid, n_{mid})$ **then**
        $StopFlag = 1$ ▷ The artificial sphere connects S_k to S
        $S_{art} \leftarrow \{(mid, d(mid, m_N))\}$ ▷ S and Z_{ik} are updated
        $S := S \cup S_{art}$
        $Z_{ik} := Z_{ik} \cup \{x_{1i} \mid x_{1i} \in I \wedge ||mid - x_{1i}||^2 \leq ||mid - n_M||^2\}$
      else
        $c_a \leftarrow (\frac{x_{0k} + n_i}{2}),\ r_a \leftarrow d(x_{0k}, n_i),\ S_a \leftarrow \{c_a, r_a\}$
        $A \leftarrow \{x_{2i} \mid x_{2i} \in C_2 \wedge ||c_a - x_{2i}||^2 \leq (r_a)^2\}$ ▷ Relevant C_2 points
        $N \leftarrow \{(c_s, r_s) \mid c_s \in\ ||c - c_s||^2 = r^2 \wedge \{c, r\} \in S,\ r_s = \underset{(x_{2j} \in A)}{\min}\ d(c_s, x_{2j})\}$
        $\{c_{art}, r_{art}\} \leftarrow \underset{(\{c,r\} \in N)}{\arg\max}\ r\ ,\ s.t.\ ||c - x_{2j}||^2 \geq r^2\ ,\ \forall x_{2j} \in A$
        $S_{art} \leftarrow \{c_{art}, r_{art}\}$ ▷ From set N, the sphere with the largest radius
        $S := S \cup S_{art}$
        $Z_{ik} := Z_{ik} \cup \{x_{1i} \mid x_{1i} \in I \wedge ||m_{art} - x_{1i}||^2 \leq ||m_{art} - n_M||^2\}$
      end
    end
  end
  $S := S \cup S_k,\ I := I \backslash Z_{ik},\ t := t + 1$
end

Example 5.2. *Returning to the example, at iteration 2 of the proposed heuristic, the suggested S_k at the end of stage 1 (Algorithm 2) is not con-*

nected with any sphere of S. The process of adding artificial spheres to S in order to connect S_k with it is illustrated in Fig. (14).

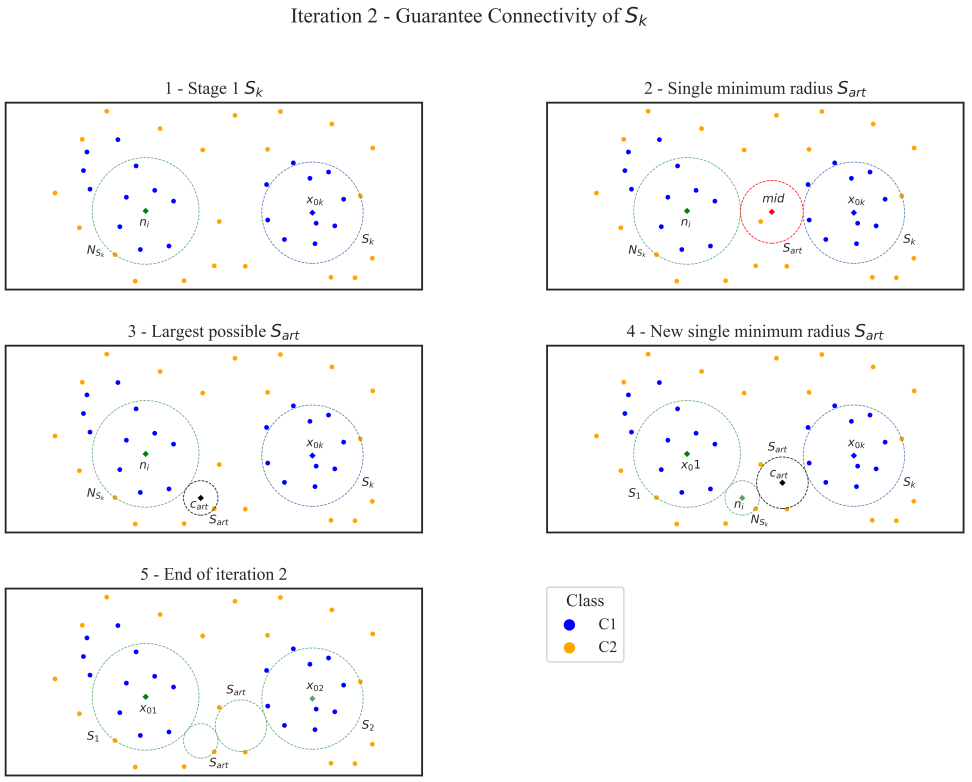


Figure 14: Generation of artificial spheres at iteration 2 for the *Separation* data set.

6. Computational Experience

In this section, we present the result of the exact method (for minimization of the number of clusters, no margin maximization) and the heuristic, for bi-dimensional problems so we may understand the complex structure of the data and observe the exact and heuristic solution. Having confirmed the expected behavior of the heuristic approach in the bi-dimensional setting, we also tested this approach on higher dimensional data sets drawn from the binary classification literature, comparing its performance against some of the classical Machine Learning classification methods.

To solve the CCSP with an exact method, we used Matlab (R2019a) and the optimization software Baron [52] version 23.1.5 (5-January-2023), on a machine with 2*CPU: AMD EPYC™ 7702 (64 Cores 256MB Cache, 2.0GHz to 3.35GHz GHz), 512GB RAM @ 3.200GHz. To run the heuristic we used Python version 3.8.16 (6-December-2022) in a Jupyter Notebook version 6.5.2 (30-October-2022) environment, on a Intel(R) Core(TM) $i7-11800H$ computer with CPU 2.30GHz.

We generated six bi-dimensional instances, where the elements of the target class has some complex, non-linear and non-convex structure as depicted in (Fig. 15).

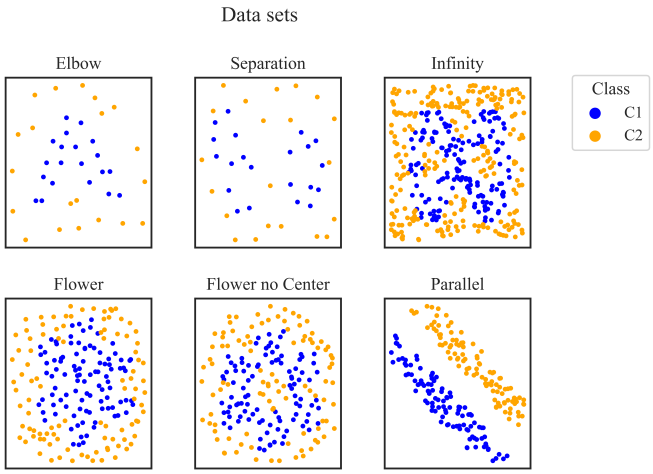


Figure 15: Example data sets to access the heuristic behavior.

More information about the bi-dimensional data sets and the solution obtained with the exact method and the heuristic approach is displayed in Table 1. The columns $N_1 + N_2$ contain the total number of points of the training set from both classes, while column $\%N_1$ represents the percentage of training points in the target class. Columns under"Heuristic" and "Baron" refer to information regarding the heuristic and the exact method implemented in Baron software. Under this designation, the running time, in seconds, of both methods ("Time (s)") and the number of spheres in the cloud ("# Spheres"), as well as the percentage of artificial spheres produced

by the heuristic ("% Art. Spheres"), are displayed.

Data set	$N_1 + N_2$	$\%N1$	Heuristic			Baron	
			Time (s)	#Spheres	%Art. Spheres	Time (s)	#Spheres
Elbow	40	50.00	0.01	4	-	57.95	3
Separation	42	50.00	0.01	7	42.86	1.82	2
Infinity	400	39.25	0.60	26	34.62	-	-
Flower	200	50.00	0.11	20	20.00	-	-
Flower no Center	200	50.00	0.15	25	40.00	-	-
Parallel	200	50.00	0.05	3	-	5.53	1

Table 1: Bi-dimensional data sets information.

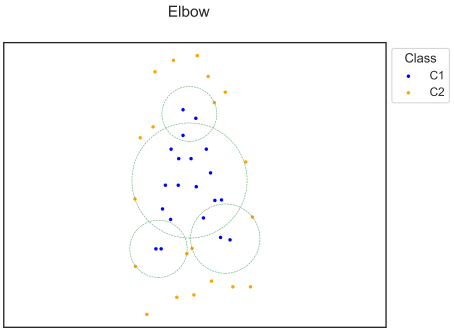


Figure 16: Cloud heuristic

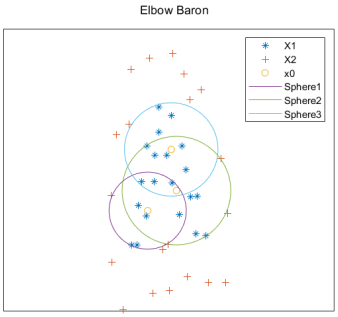


Figure 17: Cloud Baron

For Baron we made the maximum number of spheres K (Section 4) equal to the number of spheres obtained from the heuristic. We noticed that the running time of Baron is highly influenced by this upper bound on the number of spheres in the cloud. We set a maximum of 15000 iterations and 1000 seconds as the maximum time. With these limitations it was not possible to obtain a feasible solution (even less optimal) solution, for the "Infinity", "Flower" and "Flower no Center" instances. For the other sets, the optimal solution was clearly better but the running time was worse, as expected. Fig. 16 and 17 show the cloud obtained for the data set "Elbow" with the heuristic and Baron respectively. The same for data set "Separation" (Fig. 18 and 19) and "Parallels" (Fig 20 and 21).

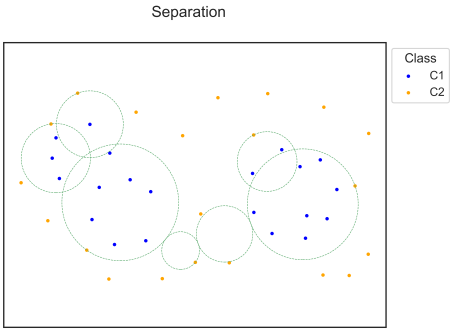


Figure 18: Cloud heuristic.

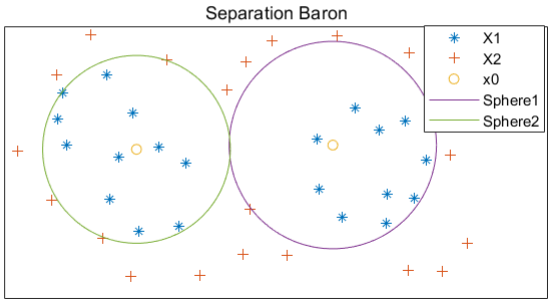


Figure 19: Cloud Baron.

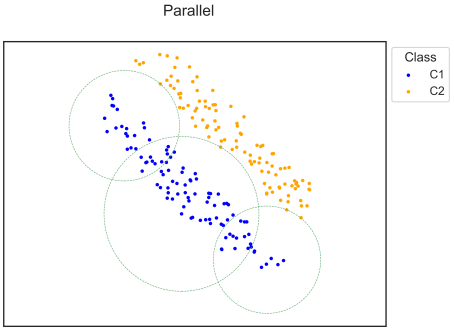


Figure 20: Cloud heuristic.

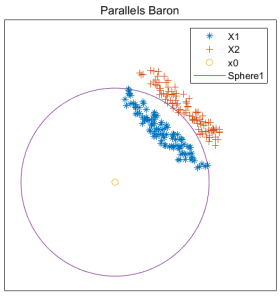


Figure 21: Cloud Baron.

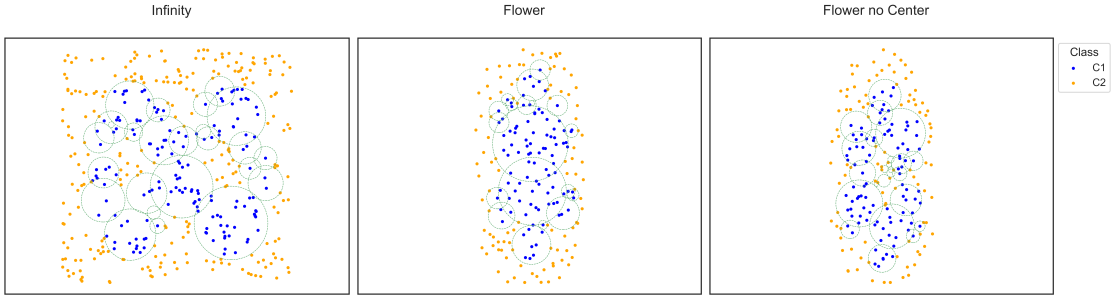


Figure 22: Cloud heuristic Figure 23: Cloud heuristic Figure 24: Cloud heuristic

To access the performance of the CCSP on real data sets, 8 commonly used data sets from the supervised classification literature were selected. All

the selected data sets were retrieved from the UCI Machine Learning Repository [53]. A brief description of these sets is provided in Table 2. For all 8 data sets we were only able to retrieve results for the heuristic approach: solving the CCSP with an exact method in Baron did not yield any feasible solution. As performance benchmarks, we compare the heuristic approach results against the algorithms Logistic Regression (LR), Linear SVM (L-SVM), Fine Gaussian SVM (FG-SVM) and Random Forest (RF), methods briefly introduced in section 2. These four methods were implemented via the Matlab (R2019a) Classification Learner.

Data set	Dimension	$N_1 + N_2$	$\%N1$	CCSP Heuristic		
				Time (sec)	#Spheres	%Art. Spheres
Banknotes	4	1372	44.46	7.08	12	16.67
BCW Original	9	683	34.99	0.02	27	-
BCW Diagnostic	30	569	37.26	0.70	18	-
Ionosphere	34	351	64.10	0.46	13	-
Mushrooms	22	8124	48.20	138.79	9	-
Pima Diabetes	8	768	34.90	22.90	151	14.57
Sonar	60	208	53.37	0.32	21	-
Voting (US)	31	435	38.62	0.99	31	-

Table 2: Real data sets information.

For all datasets, we have performed a standard 10-fold cross-validation protocol, implementing one-hot-encoding and normalization on categorical and quantitative features, respectively (such pre-processing was not implemented when running the RF algorithm). Figures presented in Table 3 consist in the average values observed across the 10 performed validations for each data set. The performance of the algorithms on each classification task was accessed with the calculation of the respective classification Accuracy (i.e. ratio of correct predictions to total predictions made), Precision (i.e. ratio of correct positive predictions), Recall (i.e. ratio of actual positives correctly predicted) and F1 score (i.e. harmonic mean of precision and recall) - the four metrics are detailed for the CCSP heuristic. The best results in terms of classification Accuracy have been underlined.

Data set	Avg. Accuracy (%)					CCSP Heuristic		
	LR	L-SVM	FG-SVM	RF	CCSP Heu.	Precision	Recall	F1 Score
Banknotes	98.91	98.40	99.93	94.97	94.66	91.87	96.65	94.16
BCW Original	96.63	96.78	97.07	97.51	93.64	91.64	90.14	90.82
BCW Diagnostic	95.25	99.65	97.89	95.96	90.18	88.09	85.85	86.69
Ionosphere	86.89	87.18	94.59	93.45	84.27	87.08	88.79	87.83
Mushrooms	100	99.90	100	100	99.12	98.38	99.81	99.09
Pima Diabetes	77.60	77.21	73.31	76.82	65.31	50.38	42.66	45.62
Sonar	75.96	77.40	83.65	84.62	62.88	68.37	57.96	62.07
Voting (US)	94.25	95.63	96.09	95.86	87.79	89.02	78.38	82.85

Table 3: Real data sets results.

The numerical results indicate that the CCSP has potential to become a feasible and interesting method for supervised Machine Learning tasks. Although the proposed heuristic underperformed compared to the established algorithms in terms of classification Accuracy, it came in general close to the best results displaying a good trade-off between variance and bias in most of the data sets.

It is also important to note that the heuristic approach to solve the CCSP still has room for further improvement.

7. Conclusion

In this paper, we propose a supervised classification method based on a cloud of spheres. This process allows the identification of non-linear and non-convex structures of the data. It is transparent, in the sense that the process of classification is based on a structure whose representation is known, well-defined and kept within the original feature space. The number of parameters to estimate is low. The classification process is simple and the method does not need balanced classes, being suitable for anomaly detection. We assume a binary classification problem, consisting in identifying elements of a target class.

A MINLP formulation is presented to find a cloud of spheres, defined by the centers and radii of the spheres. The formulation of connectivity conditions is usual heavy, requiring an exponential number of constraints. In this case, we were able to formulate the connectivity condition of the cloud imposing constraints that scale quadratically with the size of the maximal number of spheres.

We present the application to some small\medium scale examples with a highly non-linear and non-convex structure using the exact approach with the optimization software Baron. We also propose a constructive heuristic to find good feasible solutions, of particular interest for large instances.We compare the heuristic approach results against the algorithms Logistic Regression (LR), Linear SVM (L-SVM), Fine Gaussian SVM (FG-SVM) and Random Forest (RF), on 8 commonly used data sets from the supervised classification literature. The performance of the heuristic was satisfactory.
As future work we intend to explore MINLP exact approaches and metaheuristic methods. We also want to explore copositive reformulations and SDP relaxation to find tight lower bounds. We also intend to investigate the use of ellipsoids instead of spheres.

We would like to express our sincere appreciation for the invaluable contributions of the referees, which greatly contributed to the improvement of the paper.

References

[1] W. An, M. Liang, Fuzzy support vector machine based on within-class scatter for classification problems with outliers or noises, Neurocomputing 110 (2013) 101–110. URL: https://www.sciencedirect.com/science/article/pii/S0925231213000106. doi:https://doi.org/10.1016/j.neucom.2012.11.023.

[2] J. Hang, J. Zhang, M. Cheng, Application of multi-class fuzzy support vector machine classifier for fault diagnosis of wind turbine, Fuzzy Sets and Systems 297 (2016) 128–140. URL: https://www.sciencedirect.com/science/article/pii/S0165011415003176. doi:https://doi.org/10.1016/j.fss.2015.07.005, themed Section: Fuzzy Systems.

[3] D. R. Cox, The regression analysis of binary sequences, Journal of the Royal Statistical Society: Series B (Methodological) 20 (1958) 215–232. URL: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1958.tb00292.x. doi:https://doi.org/10.1111/j.2517-6161.1958.tb00292.x.

[4] M. P. LaValley, Logistic regression, Circulation 117 (2008) 2395–2399.

[5] H. Zhang, L. Jiang, L. Yu, Attribute and instance weighted naive bayes, Pattern Recognition 111 (2021) 107674.

[6] D. Soria, J. M. Garibaldi, F. Ambrogi, E. M. Biganzoli, I. O. Ellis, A 'non-parametric' version of the naive bayes classifier, Knowledge-Based Systems 24 (2011) 775–784. URL: https://www.sciencedirect.com/science/article/pii/S0950705111000414. doi:https://doi.org/10.1016/j.knosys.2011.02.014.

[7] E. Fix, J. L. Hodges Jr, Discriminatory analysis-nonparametric discrimination: Small sample performance, Technical Report, California Univ Berkeley, 1952.

[8] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE transactions on information theory 13 (1967) 21–27.

[9] P. Cunningham, S. J. Delany, K-nearest neighbour classifiers-a tutorial, ACM Computing Surveys (CSUR) 54 (2021) 1–25.

[10] E. Carrizosa, C. Molero-Rio, D. Romero Morales, Mathematical optimization in classification and regression trees, Top 29 (2021) 5–33.

[11] M. Somvanshi, P. Chavan, S. Tambade, S. Shinde, A review of machine learning techniques using decision tree and support vector machine, in: 2016 international conference on computing communication control and automation (ICCUBEA), IEEE, 2016, pp. 1–7.

[12] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.

[13] T. K. Ho, Random decision forests, in: Proceedings of 3rd international conference on document analysis and recognition, volume 1, IEEE, 1995, pp. 278–282.

[14] S. M. Piryonesi, T. E. El-Diraby, Role of data analytics in infrastructure asset management: Overcoming data size and quality problems, Journal of Transportation Engineering, Part B: Pavements 146 (2020) 04020022.

[15] M. A. El Mrabet, K. El Makkaoui, A. Faize, Supervised machine learning: a survey, in: 2021 4th International Conference on Advanced Communication Technologies and Networking (CommNet), IEEE, 2021, pp. 1–10.

[16] L. Deng, D. Yu, et al., Deep learning: methods and applications, Foundations and trends® in signal processing 7 (2014) 197–387.

[17] E. Carrizosa, D. R. Morales, Supervised classification and mathematical optimization, Computers & Operations Research 40 (2013) 150–165.

[18] D. A. Otchere, T. O. Arbi Ganat, R. Gholami, S. Ridha, Application of supervised machine learning paradigms in the prediction of petroleum reservoir properties: Comparative analysis of ann and svm models, Journal of Petroleum Science and Engineering 200 (2021) 108182. URL: https://www.sciencedirect.com/science/article/pii/S0920410520312365. doi:https://doi.org/10.1016/j.petrol.2020.108182.

[19] Y. B. Hamdan, et al., Construction of statistical svm based recognition model for handwritten character recognition, Journal of Information Technology 3 (2021) 92–107.

[20] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.

[21] C. Campbell, Kernel methods: a survey of current techniques, Neurocomputing 48 (2002) 63–84.

[22] Y. Motai, Kernel association for classification and prediction: A survey, IEEE transactions on neural networks and learning systems 26 (2014) 208–223.

[23] T. B. Trafalis, R. C. Gilbert, Robust support vector machines for classification and computational issues, Optimisation Methods and Software 22 (2007) 187–198.

[24] A. Astorino, A. Fuduli, E. Gorgone, Non-smoothness in classification problems, Optimisation Methods & Software 23 (2008) 675–688.

[25] A. Astorino, A. Fuduli, M. Gaudioso, Nonlinear programming for classification problems in machine learning, in: AIP conference proceedings, volume 1776, AIP Publishing LLC, 2016, p. 040004.

[26] N. Megiddo, On the complexity of polyhedral separability, Discrete & Computational Geometry 3 (1988) 325–337.

[27] A. Astorino, M. Gaudioso, Polyhedral separability through successive lp, Journal of Optimization theory and applications 112 (2002) 265–293.

[28] A. Astorino, A. Fuduli, Support vector machine polyhedral separability in semisupervised learning, Journal of Optimization Theory and Applications 164 (2015) 1039–1050.

[29] A. Astorino, M. D. Francesco, M. Gaudioso, E. Gorgone, B. Manca, Polyhedral separation via difference of convex (dc) programming, Soft Computing 25 (2021) 12605–12613.

[30] D. M. Tax, R. P. Duin, Support vector domain description, Pattern recognition letters 20 (1999) 1191–1199.

[31] A. Astorino, M. Gaudioso, A fixed-center spherical separation algorithm with kernel transformations for classification problems, Computational Management Science 6 (2009) 357–372.

[32] A. Astorino, A. Fuduli, M. Gaudioso, Dc models for spherical separation, Journal of Global Optimization 48 (2010) 657–669.

[33] A. Astorino, A. Fuduli, M. Gaudioso, Margin maximization in spherical separation, Computational Optimization and Applications 53 (2012) 301–322.

[34] A. Astorino, A. Fuduli, Semisupervised spherical separation, Applied Mathematical Modelling 39 (2015) 6351–6358.

[35] A. Astorino, A. Fuduli, Spherical separation with infinitely far center, Soft Computing 24 (2020) 17751–17759.

[36] M. C. de Almeida Calado, Explaining Incorrect Feature Learning in the Training of Deep Neural Networks, Master thesis, NOVA School of Science and Technology, NOVA University Lisbon, 2022.

[37] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, arXiv preprint arXiv:1312.6199 (2013).

[38] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, arXiv preprint arXiv:1412.6572 (2014).

[39] T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, A. T. Kalai, Man is to computer programmer as woman is to homemaker? debiasing word embeddings, Advances in neural information processing systems 29 (2016).

[40] J. Angwin, J. Larson, S. Mattu, L. Kirchner, Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks, 2016. URL: `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing`.

[41] S. Wang, M. Gupta, Deontological ethics by monotonicity shape constraints, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 2043–2054.

[42] R. K. Bellamy, K. Dey, M. Hind, S. C. Hoffman, S. Houde, K. Kannan, P. Lohia, S. Mehta, A. Mojsilovic, S. Nagar, et al., Think your artificial intelligence software is fair? think again, IEEE Software 36 (2019) 76–80.

[43] S. Jentzsch, P. Schramowski, C. Rothkopf, K. Kersting, Semantics derived automatically from language corpora contain human-like moral choices, in: Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society, 2019, pp. 37–44.

[44] R. Livni, S. Shalev-Shwartz, O. Shamir, On the computational efficiency of training neural networks, Advances in neural information processing systems 27 (2014).

[45] D. M. Cardoso, J. Szymanski, M. Rostami, Matemática discreta, Escolar Editora (2009).

[46] H. Nagamochi, T. Ibaraki, Algorithmic aspects of graph connectivity, Cambridge University Press, 2008.

[47] R. Carvajal, M. Constantino, M. Goycoolea, J. P. Vielma, A. Weintraub, Imposing connectivity constraints in forest planning models, Operations Research 61 (2013) 824–836.

[48] T. L. Magnanti, S. Raghavan, Strong formulations for network design problems with connectivity requirements, Networks: An International Journal 45 (2005) 61–79.

[49] J. Thunberg, P. Ögren, A mixed integer linear programming approach to pursuit evasion problems with optional connectivity constraints, Autonomous Robots 31 (2011) 333–343.

[50] P. Li, X. Huang, J. Qi, H. Wei, X. Bai, A connectivity constrained milp model for optimal transmission switching, IEEE Transactions on Power Systems 36 (2021) 4820–4823.

[51] S. Elloumi, O. Hudry, E. Marie, A. Martin, A. Plateau, S. Rovedakis, Optimization of wireless sensor networks deployment with coverage and connectivity constraints, Annals of Operations Research 298 (2021) 183–206.

[52] N. V. Sahinidis, Baron: A general purpose global optimization software package, Journal of global optimization 8 (1996) 201–205.

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