Union of Convex Separators

Amit Prasad
School of Information Technology
Indian Institute of Technology, Delhi
New Delhi, India 110016
amit.prasad@sit.iitd.ac.in

Rahul Garg
Department of Computer Science and Engineering
Indian Institute of Delhi, Delhi
New Delhi, India
rahulgarg@cse.iitd.ac.in

Yogish Sabharwal
IBM Research, India
Gurugram, India
ysabharwal@in.ibm.com

Abstract

We present a novel algorithm for learning a union of convex separators to separate a class from another using decision boundaries, each of which is a convex separator. A convex separator is a collection of hyperplanes that separate points of one class from another using an intersection of half-spaces. A union of convex separators can be thought of as ensembles of such convex separators. In this work, we propose the notion of separability using a union of convex separators previously known in earlier works as min-max separability, suggest an algorithm to learn the separators using a simple gradient-based method, and also evaluate the algorithm on some datasets of recent interests against other popular classification algorithms.

1 Introduction

Binary classification is the task of learning a classifier that distinguishes data into two separate classes. Models built for the task assume the data to follow certain assumptions. One such assumption is linear separability. Informally, two classes are said to be linearly separable in $\mathbb{R}^d$ if there exists a hyperplane in $\mathbb{R}^d$ that can separate two classes from each other. Two notable classifiers built on these assumptions are logistic regression and support vector machines (SVMs) [8]. These models have achieved good results on a wide variety of real-world datasets. SVMs with their computationally efficient kernel incorporation have achieved good results on datasets where the linear separability assumption does not hold.

Convex Separability, also defined as polyhedral separability in previous literature [19] is a weaker assumption than linear separability. Informally, a class is said to be convex separable from another if there exists a decision boundary that encloses a convex set that separates one class from the other. Since the convex separability assumption is a weaker assumption than linear separability, it is expected that a model built around this assumption will achieve a better classification score than the models built around the linear separability assumption.

Intuitively, in a binary classification setting, it seems natural that no matter how complicated a decision boundary is required to separate one class from the other, subsets of points of one class can form convex separable sets. If a classifier can learn the decision boundaries of these subsets (which can be convex separators), a union of such convex separators can be used to separate one class from the other. In this piece of work, our goal is to learn this union of convex separators from the training data.

1.1 Related work

Any dataset in general may have complicated patterns whose classification is difficult for any linear classifier (SVM or logistic regression classifier). Piecewise linear classifiers (PLCs) were studied as a way of combining many linear classifiers and coming up with a way of classifying more complex patterns. [14] introduced a way of learning a PLC wherein the points of one class on one side of a hyperplane were maximized. With the points on the other side, the process is repeated to obtain a collection of hyperplanes that separate the two classes. Another work [9] iteratively introduces a pair of hyperplanes (starting with 1 pair) till the desired level of classification is reached. The assumption is that in lieu of linear separability, there will exist 2 hyperplanes that will together enclose one class in between them. The misclassified points between them can act as a new dataset, a classification of which can be achieved by introducing two more hyperplanes. This is done till the desired level of classification is reached. Another series of works [21], [25], [26] use clustering algorithms to partition the dataset into regions. Pairs of points belonging to 2 different clusters are then selected to make ends of line segments called links. A minimal set of hyperplanes is then found which aims to cut through a maximum of these links using heuristics to yield a decision boundary made up of hyperplanes. [20] also uses a clustering procedure to first divide the entire space into separable spaces for which hyperplanes are learned. [13] uses hyperplanes as nodes of decision trees to come up with piecewise linear classifiers. Common amongst all these approaches is trying to get regions that are "easily" separable which easily means either linear or convex separability. Heuristics such as clustering are therefore used. It seems intuitive that subsets of features of a class are polyhedral/convex separable [19] from the rest of the features.
Given that all points of one class are convex separable from another, there are proposed methods in the literature that learn a convex separator for the convex separable class. Some of the recent methods proposed are [17], [18] and [12].
Informally, if there are well-defined convex separable regions of one class in a binary classification setting, this class is min-max separable [5] from the other. Bagirov in [5] has proven that two sets A and B are min-max separable if there exists a piecewise linear function separating them which makes the problem of learning a min-max classifier equivalent to learning a piecewise linear classifier. The work also learns a classifier by minimizing an error function using the discrete gradient method. This method, by the author's own admission, is computationally inefficient for large datasets. Also, it requires apriori knowledge of the number of hyperplanes required. In another work [22] this min-max separability is viewed as the union of c convex style separators in which each convex separator is learned as an intersection of some k hyperplanes by minimizing a "minimax" bound over negative class (class labeled 0) and hinge type loss over the positive class (class labeled 1). In both these methods, the number of hyperplanes must be known apriori. In [22], alongside the hyperplanes, the number of convex regions must be known apriori. To remove this limitation, [6] researchers came up with a way of inserting hyperplanes as per the requirements of the dataset. This is done by running the algorithm in 2 phases. One in which indeterminate regions are identified and hyperplanes are added accordingly and then training the added hyperplanes along with the ones already present. Their work also presents a performance evaluation of datasets and a comparison to other algorithms. While being close to other methods in performance, it doesn't seem to have a lot of performance improvement. [16] is another work that uses "conlitron" to separate out a convex region in the dataset. An ensemble of these is trained to separate out the union of convex regions known as "multiconlitron". In order to overcome the growing complexity of the model, later the same authors introduced alternating multicolitron [15] uses the concept of the maximal convexly separable set to come up with the minimal number of separators in the ensemble required. However, both these methods apparently have produced results subpar to SVM with RBF kernel and at times even subpar to SVM with linear kernel.

1.2 Our contributions

Our contributions are the following:

1. Design a machine learning algorithm for binary classification that uses a union/ensemble of convex separators to enclose convex separable subsets of one class to distinguish it from the other. We name it "Union of convex separators".
2. Design the above-said algorithm in such a way that it does not require prior knowledge of a number of hyperplanes or convex sets.

3. Present a rigorous performance analysis of the developed algorithms against other machine learning methods on a variety of tabular datasets.

We make our implementation publicly available[1] to the community for further exploration.

2 Notations and Definitions

Definition 1 ***Hyperplane****: A hyperplane in $\mathbb{R}^d$ is defined as the set $\{x : (w^T x+b) = 0, x \in \mathbb{R}^d, w \in \mathbb{R}^d, b \in \mathbb{R}\}$. It is denoted in this work using notation (w, b).*

Definition 2 ***Convex combination****: For vectors x_1, x_2,...,x_n in $\mathbb{R}^d$, y is said to be a convex combination of x_1, x_2,...,x_n if*

$$y = a_1x_1 + a_2x_2 + ... + a_nx_n$$

for $a_i \geq 0$, $\sum_i a_i = 1$ and $a_i \in \mathbb{R}$

Definition 3 ***Convex sets****: A set S is called a convex set if all convex combinations of any subset of S belong to the set S.*

Definition 4 ***Convex hull****: A convex hull of a set of points $C \subset \mathbb{R}^d$ is the set of all possible convex combinations of subsets of C. The convex hull of C is denoted by CH(C).*

Definition 5 ***Linear Separability****: Two sets $S \subset \mathbb{R}^d$ and $T \subset \mathbb{R}^d$ are linearly separable in $\mathbb{R}^d$ if there exists a hyperplane with coefficients $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that for every $s \in S$ and $t \in T$, $(w^T s + b) * (w^T t + b) < 0$. '*' stands for scalar multiplication.*

Definition 6 ***Convex Separability****: A set S is said to be convex separable from another set T if the convex hull of S does not enclose any $t \in T$.*

Definition 7 ***k-Convex Separability****: A set $S \subset \mathbb{R}^d$ is said to be k-convex separable from another set $T \subset \mathbb{R}^d$ if there exists a set H of hyperplanes in $\mathbb{R}^d$ such that $|H| = k$ and $min_{(w,b)\in H}(w^T s+b) \geq 0$ for every $s \in S$ and $min_{(w,b)\in H}(w^T t + b) < 0$ for every $t \in T$. (w, b) represents the coefficients of a hyperplane in $\mathbb{R}^d$, $w \in \mathbb{R}^d, b \in R$.*

Definition 8 ***k-Convex Separator****: For a k convex separable set S, the set of k hyperplanes is called a k-Convex Separator.*

Definition 9 ***Min-Max separability****: A set S is said to be min-max separable from another set T if there exists a collection of n hyperplanes $\mathcal{H} = \{(w_1, b_1), (w_2, b_2), ..., (w_n, b_n)\}$ that can be partitioned into finite non-empty sets $\mathcal{H} = \{H_1, H_2, ..., H_l\}$ such that $H_i \cap H_j = \phi$ and for all $s \in S$ there exists some H_k such that $min_{(w,b)\in H_k}(w^T s + b) \geq 0$ and for all $t \in T$, for all $H_p \in \mathcal{H}, min_{(w,b)\in H_p}(w^T t + b) < 0$.*

Definition 10 ***Union of convex separators****: A set of n hyperplanes $\mathcal{H} = \{(w_1, b_1), (w_2, b_2), ..., (w_n, b_n)\}$ is said to from a union of convex separators if $\mathcal{H}$ can be partitioned into finite non empty sets $\{H_1, H_2, ..., H_l\}$ such that $H_i \cap H_j = \phi$ and each H_i is a n_i-convex separator.*

3 The model

We assume that for a given example in the data, x, there exists a latent random variable z that maps to a number p if x belongs to the convex separator H_p. Let the number of separators be l.

$$\mathcal{H} = \{H_1, H_2, ..., H_l\} \tag{1}$$

[1] `https://github.com/Amit-Prasad/union_of_convex_separators`

Each convex separator H_j is assumed to be made up of n_j hyperplanes each with coefficients $w_{jk} \in \mathbb{R}^d$ and $b_{jk} \in \mathbb{R}$.

$$H_i = \{(w_{j1}, b_{j1}), (w_{j2}, b_{j2}), ..., (w_{jn_j}, b_{jn_j})\} \tag{2}$$

Now the probability of a point x being classified as class labeled 1 given that it belongs to the t^{th} separator is given by

$$p(y|z = t, x) = \hat{y_t} = \frac{1}{1 + exp(-min_p(w_{tp}^T x + b_{tp}))} \tag{3}$$

At the moment, we assume that the probability $p(z = t|x)$ is known and is given by

$$p(z = t|x) = \phi_t \tag{4}$$

We intend to find both the label as well the separator for a point x in the data. Using equations (3) and (4), the joint probability distribution is given by

$$p((y = 1, z = t)|x) = \hat{y_t}^y (1 - \hat{y_t})^{(1-y)} \phi_t \tag{5}$$

Now, given data of the form $(x_i, y_i)_{i=1}^m$, we can write the log-likelihood of the data as:

$$l(\mathcal{H}) = \sum_{i=1}^{m} log(\sum_{z_i} p(y_i, z_i|x_i)). \tag{6}$$

Even if the probabilities in equation (4) are known, the objective is not simple to optimize. Also, it must be noted here that as mentioned in section 1.2, the algorithm that needs to be developed has no prior knowledge of the number of hyperplanes or the number of convex separators required. Due to these reasons, direct optimization of the objective (equation 6) is not possible. So, we need 3 things at this point to make an algorithm that works.

1. A way to assign probabilities $p(z = t|x)$ for some point x. This has been presented in detail in section 3.1.
2. Once the probabilities are assigned, for a set of points that have a "most probable separator", we need to learn a convex separator. This is presented in detail in section 3.2.
3. Since no prior knowledge of the number of hyperplanes or the number of convex separators is assumed, we need an algorithm to grow the number of hyperplanes per separator and also a growing number of convex separators if needed. This is presented in section 3.3.

3.1 Assigning probabilities $p(z = j|x)$

Assume that at some stage during the running of the algorithm, we have some k convex separators. We assume the probability of a point x belonging to some separator H_j is given by a multinoulli over the number of separators. This implies

$$p(z = j|x) = \phi_j$$

and

$$\sum_j \phi_j = 1$$

We choose

$$p(z = q|x) = \phi_q = \mathbb{1}\{argmax_i(min_j(w_{ij}^T x + b_{ij})) = q\} \tag{7}$$

So the entire feature space is split into k sets, and each feature is assigned to only one of k separators given by equation (7).

3.2 Learning Convex Separators

Consider all points in the dataset which have been assigned to some separator H_p. Given a separator H_p and points of the form $(x_i, y_i)_{i=1}^{m_p}$ where m_p is the number of points assigned to separator H_p and given that $|H_p| = n_p$, the convex separator is learned using the minimization of the objective

$$l(H_p) = \sum_{i=1}^{m_p} y_i log(\hat{y_i}) + (1 - y_i) log(1 - \hat{y_i}) \tag{8}$$

where

$$\hat{y_i} = \frac{1}{1 + exp(-min_{(w,b)\in H_p}(w^T x_i + b))} \tag{9}$$

which comes from our assumptions in equation (3). The objective can be optimized using gradient descent. This algorithm was proposed by [17]. However, the algorithm depends heavily on the initialization of hyperplanes and a random initialization of hyperplanes does not always work. Another problem is adding another hyperplane to an existing convex separator has not been presented in their work. We address this in our work and the next section introduces how to do this efficiently.

3.3 Growing convex separators in size and number

In this section, we present the method used for growing the number of hyperplanes per separator and also increasing the number of convex separators if required by the algorithm.
Consider the algorithm at some stage where there are some l separators. For each point, x_i using equation (7) there exists an allocation of it to one of l separators. If a hyperplane is to be added at this stage it needs to be decided whether this hyperplane must be added to a convex separator to increase the number of hyperplanes of that separator or as a new hyperplane to create a new separator. To resolve this we take a point x_{fp} that is a false positive (ground truth label is 0 but labeled 1 by the algorithm). This false positive will belong to exactly one separator. Let this separator be numbered k and have m_k points assigned to it. Let the set of points assigned to this separator form the set S_k. Out of the number of points assigned to the separator numbered k, assume m_k^+ are labeled 1 and m_k^- are labeled 0. Now, if all m_k^+ points are convex separable, from m_k^- points that are labeled 0, a hyperplane needs to be added to the present separator to correctly classify x_{fp}. If the m_k^+ points are not convex separable from the rest, a new separator is required. We just have x_{fp} and S_k to decide this. So we solve the following optimization problem in order to make this decision.

$$f(x_{fp}, S_k) = max_w \sum_{s_i \in S_k} min(w^T(s_i - x_{fp}), 1) \tag{10}$$

This optimization gives some coefficient $w \in R^d$. Now, if this w gives a values $f(x_{fp}, S_k) = |S_k|$ upon optimisation, the m_k^+ points labelled 1 are convex separable from other points in S_k (using the result in theorem 6.2). So the hyperplane coefficient that we get upon optimization can be used to grow the same convex separator to which x_{fp} is assigned such that it reduces the error rate per separator. However, in case the $f(x_{fp}, S_k) < |S_k|$, it can be concluded that the set of points S_k are not convex separable (using the result in theorem 6.3) and the new hyperplane must be used to split the separator which is done by the algorithm (6). In case this hyperplane is used to grow the present separator, b is calculated by $b = max_{s\in S}(-1 * w^T s)$. In the other case where it is decided to split the separator, we find a random hyperplane through the chosen false positive that has an almost equal number of points on each side of it that belongs to the present separator. This hyperplane gives two hyperplanes with coefficients $(-w, -b)$ and (w, b) and also divides the set of points inside separator S_k into 2 sets, say S_{k1} and S_{k2}. These hyperplanes now form the first hyperplanes of two new separators. Now, each of the existing hyperplanes of this separator needs to be re-assigned to one of the two new separators. For this, each of the hyperplanes is taken and determined to be one of S_{k1} and S_{k2} as done in the algorithm (6).

4 Learning Union of Convex Separators

The algorithm adds hyperplanes one by one either splitting a set of convex separators to make 2 separators or growing a separator depending on where a false positive lies. Given the number of separators q and the n_q hyperplanes in each separator, the algorithm learns by iterative execution of 2 steps namely, assignment and training.

1. Assignment step: We assign a point x_i to separator k if $\mathbb{1}\{argmax_i(min_j(w_{ij}^T x + b_{ij})) = k$ where $i \in \{1, 2, ..., q\}$ and $j \in \{1, 2, ..., |H_i|\}$.
2. Training step: For a given assignment of points to separators, we optimize the objective in equation 8 for each separator H_p.

This is done until a further addition of hyperplanes and any re-assignment of points cannot yield an improvement in error. The complete algorithm (6) puts together everything in the previous sections and this section.

5 Experimentation and evaluation

5.1 Data preprocessing

All empty rows are removed. All columns containing irrelevant information (like serial numbers) have been removed. All attributes are divided by the maximum absolute value of the attribute.

Table 1: Rows and columns of the real datasets in numbers

Datasets	Breast cancer [27]	Churn [2]	Covtype (Subset) [7]	Diabetes [11]	Ionosphere [24]	Philippine [1]	Santander (subset) [3]	Shoppers [23]	Spambase [10]	Telco churn [4]
Number of rows (Train)	185	7000	7286	514	234	3907	7000	8630	3082	4922
Number of rows (Test)	92	3000	3123	254	116	1925	3000	3699	1519	2110
Number of columns	9	10	54	8	33	308	200	17	57	19

5.2 Training, hyperparameter tuning and inference

Each dataset (structure tabulated in Table (1)) is split into a 7:3 ratio where 30% of the data is used for inference while the other 70% is used in one of the following ways depending upon the type of model:

5.2.1 Models requiring hyperparameter tuning (Random Forest, XGBoost, Neural Networks)

In this case, 70% of the data is used for a grid search. The score that decides the best hyperparameter is chosen to be balanced accuracy. The best parameters undergo a 5-fold cross-validation to make sure the optimal hyperparameters have a low variance. This best hyperparameter set is now used to train the model on the whole 70% of the data followed by inference on the 30% of the dataset.

5.2.2 Models not requiring hyperparameter tuning or having termination criteria to stop parameter increase (Logistic regression, UCS)

In this case, the grid search step is skipped and the rest stays the same as in here section (5.2.1).

Note that the inference on 30% of the data is the one reported everywhere in this work for all algorithms.

6 Algorithms and theorems

This section presents a list of all theorems that have been employed in the design of all the algorithms in this paper.

Theorem 6.1 *If a set S is min-max separable from T, then there exists a union of convex separators to separate all $s \in S$ from $t \in T$.*

Proof: If S is min-max separable from T, a set of hyperplanes of the form mentioned in definition 9 is present. We need to prove that this partition over the hyperplane set in the form $\mathcal{H} = \{H_1, H_2, ..., H_l\}$ has H_i which forms convex separators.
Corresponding to H_i, we define set $S_i = \{s : min_{(w,b)\in H_i}(w^T s + b) \geq 0\}$. Since S is known to be min-max separable from set T, not all of S_i can be empty.
Now, for every $s \in S_i$, $min_{(w,b)\in H_i}(w^T s + b) \geq 0$ and for every $t \in ((S \cup T) \setminus S_i)$, $min_{(w,b)\in H_i}(w^T s + b) < 0$ (by definition of S_i). So each H_i is a convex separator. Also $H_i \cap H_j = \phi$ by definition of min-max separability.

Theorem 6.2 *For any point $t \in T$ which is not in CH(S), the maximum of $f_w(t, S) = |S|$ and the corresponding maxima w is such that for every $s \in S$, $w^T(s - t) \geq 1$.*

Proof: Since t is not in CH(S), t is linearly separable from S. So, there exists a hyperplane with coefficients (w, b) such that $(w^T s + b) \geq 0$ for all $s \in S$ and $(w^T t + b) < 0$. For this w, $w^T(s - t) > 0$

Algorithm 1 Find Union of Convex Separators

Input $(x_i, y_i)_{i=1}^m$ (Training data), l (maximum number of hyperplanes allowed)
Output $\mathcal{H} = \{H_1, H_2, ..., H_n\}$ (Union of convex separators)

Initialise $\mathcal{H} = \{\}$
Initialise $S[x_i] = 1 \forall x_i$
Initialise $k \leftarrow 1$
Initialise $z_i \leftarrow 1 \forall i$
while $k \leq l$ **do**
  $Q \leftarrow \{x_i : y_i = 0, z_i = 1\}\{Q$ is a set of false positives$\}$
  Pick an $x \in Q$
  $p = S[x]$
  $S_p \leftarrow \{x_i : S[x_i] = p, y_i = 1\}$
  $\mathcal{H} \leftarrow \mathcal{H} \backslash H_p$
  $w_i, b_i \leftarrow$ Find Separator(x, S_p)
  if $f_w(x, S_p) == |S_p|$ **then**
    $H_p \leftarrow H_p \cup \{(w_i, b_i)\}$
    $\mathcal{H} \leftarrow \mathcal{H} \cup H_p$
    $k \leftarrow k + 1$
  else
    $a \leftarrow |\mathcal{H}| + 1$
    $H_a, H_{a+1} \leftarrow$ Split hyperplanes(H_p, S_p)
    $\mathcal{H} \leftarrow \mathcal{H} \cup H_a$
    $\mathcal{H} \leftarrow \mathcal{H} \cup H_{a+1}$
    $k \leftarrow k + 2$
  end if
  $\mathcal{H} \leftarrow$Training$((x_i, y_i)_{i=1}^m, \mathcal{H})$
  $z_i = max(0, sign(max_{H \in \mathcal{H}} min_{(w,b) \in H}(w^T x_i + b) \geq 0))$ for all i
end while
return $\mathcal{H}$

Algorithm 2 Find Separator(t, S)

Input t (a point in the dataset), S (a set of points in the dataset)
Output w, b Coefficients of hyperplane to add to a convex separator

$w \leftarrow argmax_w \sum_{s_i \in S} min(w^T(s_i - t), 1)$
$b \leftarrow max_{s_i \in S}(-w^T s_i)$
return w, b

for all $s \in S$. The (w, b) can be scaled such that $w^T(s - t) \geq 1$. Therefore, $min(w^T(s - t), 1) = 1$ for every $s \in S$. So $max_w \sum_{s \in S} min(w^T(s - t), 1) = |S|$.

Theorem 6.3 *If at optimum w, $f_w(t, S) < |S|$, for some $t \in T$ then t lies in CH(S) and hence, S is not convex separable from T.*

Proof: From the contrapositive of Theorem 6.2, if $f_w(t, S) < |S|$ for some t, then there exists some point s such that $w(s - t) < 1$. Therefore t has to be in CH(S) which makes S not convex separable from T.

7 Results on synthetic datasets: Qualitative

For an indicator of qualitative performance of the working of the union of convex separators (algorithm 6) algorithm, we generate 2 different synthetic datasets in $\mathbb{R}^2$. The figure(1), shows two datasets along with the hyperplanes forming the convex separators which in turn form the union of convex separators. The labels have also been shown using legends. It can be seen that the algorithm seems effective in separating one class from the other. We also look into the individual separators for a thorough analysis, we also look into individual separators that form the convex separators. For the first example with two circles labeled class 1 in a rectangle labeled class 0 in figure (3), we can see

Algorithm 3 Split hyperplanes

Input H_i Hyperplanes forming convex separator, S_i points that belong to the separator H_i
Output H_p, H_q Hyperplanes split into two sets

repeat
  $w, b \leftarrow$ coefficients of random hyperplane passing through x
  $S_1 = \{x_j : x_j \in S_p \text{ and } w^T x_j + b \geq 0\}$
  $S_2 = \{x_j : x_j \in S_p \text{ and } w^T x_j + b < 0\}$
until $(|S_1| - |S_2|) > \frac{1}{4}(|S_1| + |S_2|)$
Init $H_p = \{(w, b)\}$
Init $H_q = \{(-1 * w, -1 * b)\}$
for each $(w, b) \in H_p$ **do**
  $m_1 = min_{x_j \in S_1}(w^T x_j + b)$
  $m_2 = min_{x_j \in S_2}(w^T x_j + b)$
  if $m_1 < m_2$ **then**
    $H_p \leftarrow H_p \cup \{(w, b)\}$
  else
    $H_q \leftarrow H_q \cup \{(w, b)\}$
  end if
end for
return H_p, H_q

Algorithm 4 Training

Input $(x_i, y_i)_{i=1}^m$ (Training data), $\mathcal{H}$ (convex separators forming a union)
Output $\mathcal{H} = \{H_1, H_2, ..., H_n\}$ (Hyperplanes forming a union of convex separator)

$S[x_i] \leftarrow argmax_j min_{w \in H_j}(w^T x_i)$
$\gamma_i = min_{(w,b) \in H_{S[x_i]}}(w^T x_i + b) \forall i$
$\hat{y_i} = \frac{1}{(1+exp(-\gamma_i))} \forall i$
$\mathcal{H} \leftarrow$ Gradient Ascent over Separators$((x_i, y_i)_{i=1}^m, \mathcal{H})$
return $\mathcal{H}$

that we have 6 separators enclosing each of the rectangles which are convex separable subsets of all points labeled as class 1. In a similar vein, for the second example with convex separable rectangles labeled as class 1 (refer figure 2), we can see that we have 2 separators enclosing each of the circles which are convex separable subsets of all points labeled as class 1.

Table 2: Balanced accuracy values for Union of Convex Separators with different numbers of split-grow iterations

Datasets	**Breast cancer**		**Churn**		**Covertype (Subset)**		**Diabetes**		**Ionosphere**	
Algorithms	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
UCS_5	75.14	60.78	66.73	67.16	80.03	78.06	76.26	71.99	100.00	88.73
UCS_10	82.07	57.97	71.15	70.79	80.28	78.81	80.62	**74.95**	100.00	**86.74**
UCS_20	80.70	**64.34**	73.03	**71.38**	80.59	78.36	78.23	75.24	100.00	86.74
UCS_30	83.33	63.36	73.99	70.92	83.74	**79.30**	86.50	72.29	100.00	86.74

8 Results on real datasets: Quantitative

For an indicator of quantitative performance, we consider two metrics AUROC (area under the ROC curve) and balanced accuracy. The following tables present a quantitative performance analysis for the union of convex separators. In the absence of termination criteria, we limit the number of times a split in convex separators or growing a separator happens. We alternatively use the term split-grow iterations indicating the number of times a false positive was considered for growing size or number

Table 3: Balanced accuracy values for Union of Convex Separators with different numbers of split-grow iterations

Datasets	**Philippine**		**Santander (Subset)**		**Shoppers**		**Spambase**		**Telco churn**	
Algorithms	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
UCS_5	80.73	**70.07**	68.88	61.71	76.55	75.08	94.41	92.68	72.15	70.76
UCS_10	79.69	69.15	68.68	**64.38**	77.02	74.61	94.62	92.87	73.44	**71.93**
UCS_20	80.67	69.35	70.10	61.40	77.83	74.74	95.19	**92.96**	75.07	69.83
UCS_30	81.28	68.95	71.13	62.16	78.39	**75.48**	96.24	92.87	76.99	69.68

Table 4: Balanced accuracy values for Union of Convex Separators compared to other algorithms

Datasets	**Breast cancer**		**Churn**		**Covertype (Subset)**		**Diabetes**		**Ionosphere**	
Algorithms	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
UCS	80.70	64.34	73.03	71.38	83.74	79.30	80.62	74.95	100.00	88.73
Logistic Regression	69.79	64.74	57.1	57.72	76.67	76.8	74.16	69.96	94.79	87.92
MLP2	85.24	58.78	74.83	73.12	84.67	79.79	73.05	73.01	99.38	85.87
MLP3	76.76	65.79	79.38	**75.01**	92.01	83.3	76.14	**75.5**	100	90.21
Random forest	79.51	**69.41**	75.4	71.46	98.34	**83.62**	90.57	74.1	98.52	87.55
MLP1	83.7	66.06	72.26	71.87	82.39	80.17	71.65	72.54	98.45	91.35
SPLA2	69.79	64.74	57.06	57.5	76.77	76.91	73.21	72.54	94.79	87.92
XGBoost	82.84	56.5	84.44	72.1	100	84.65	84.93	70.94	95.16	**92.06**

Table 5: Balanced accuracy values for Union of Convex Separators compared to other algorithms

Datasets	**Philippine**		**Santander (Subset)**		**Shoppers**		**Spambase**		**Telco churn**	
Algorithms	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
UCS	80.73	70.07	68.68	**64.38**	78.39	75.48	95.19	92.96	73.44	71.93
Logistic Regression	76.82	71.59	65.59	62.68	67.48	67.94	92.5	91.77	72.67	72.2
MLP2	86.66	67.19	85.3	61.88	82.62	**78.54**	98.9	92.9	62.02	62.44
MLP3	80.25	69.19	96.36	64	80.76	77.09	97.86	92.48	67.29	67.06
Random forest	93.8	**76.05**	72.4	55.69	81.8	78.08	98.09	94.52	77.36	67.97
MLP1	73.31	71.53	100	61.16	78.15	75.86	97.73	93.77	74	**74.19**
SPLA2	75.64	71.77	65.68	62.54	67.52	67.97	92.84	91.9	72.71	72.06
XGBoost	100	74.96	100	57.35	88.67	78.35	99.88	**95.03**	88.26	68.09

Table 6: AUROC values for Union of Convex Separators with different numbers of split-grow iterations

Datasets	**Breast cancer**		**Churn**		**Covertype (Subset)**		**Diabetes**		**Ionosphere**	
Algorithms	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
UCS_5	0.87	0.66	0.86	**0.85**	0.89	0.87	0.90	**0.86**	1.00	**0.96**
UCS_10	0.87	**0.69**	0.88	0.85	0.90	0.87	0.89	0.83	1.00	0.96
UCS_20	0.92	0.66	0.88	0.84	0.92	**0.88**	0.94	0.80	1.00	0.96
UCS_30	0.82	0.69	0.81	0.81	0.88	0.86	0.88	0.81	1.00	0.95

Table 7: AUROC values for Union of Convex Separators with different numbers of split-grow iterations

Datasets	**Philippine**		**Santander (Subset)**		**Shoppers**		**Spambase**		**Telco churn**	
Algorithms	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
UCS_5	0.88	0.76	0.91	**0.82**	0.92	**0.91**	0.99	**0.98**	0.85	0.84
UCS_10	0.89	0.76	0.91	0.82	0.93	0.90	0.99	0.98	0.87	0.83
UCS_20	0.90	0.76	0.91	0.81	0.93	0.91	0.99	0.98	0.89	0.83
UCS_30	0.89	**0.77**	0.90	0.81	0.92	0.91	0.99	0.98	0.85	**0.85**

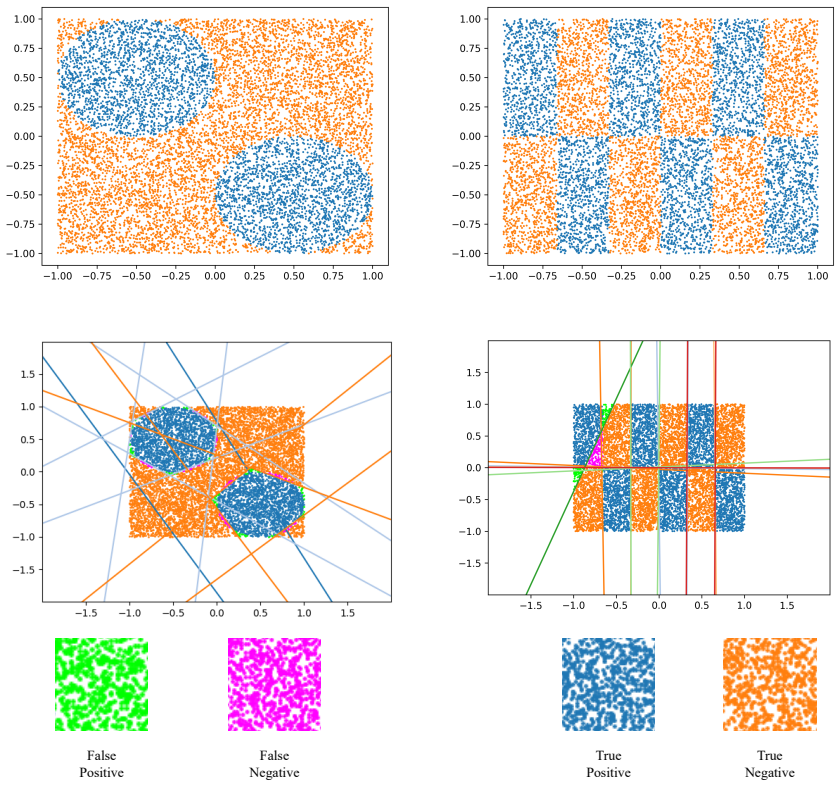

Figure 1: Results of Union of convex separators algorithm for 3 different 2D synthetic datasets.

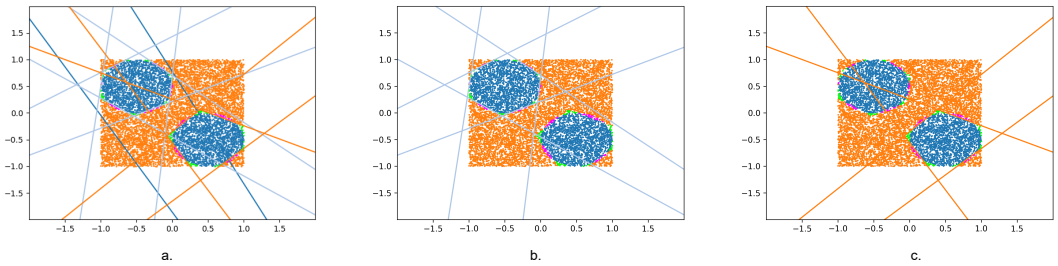

Figure 2: Union of convex separators (a.) along with its constituent separators (b.) and (c.)

of separators. Tables 2, 3 present balanced accuracy values for 5, 10, 20, and 30 split-grow iterations. In addition to balanced accuracy, we also consider AUROC scores given in tables 6, 7. We also compare the performance of the Union of Convex Separators to other algorithms using balanced accuracy values (in tables 4) and 5) and AUROC values (in tables 8) and 9). It can be seen that in most cases UCS performs better than logistic regression. The performance of UCS is also close to the best methods in most cases. For one of the datasets, Santander (subset), it performs better than other methods according to the balanced accuracy metric.

9 Ablation study

We study in detail the impact of variations in the number of convex separators and the size of each separator.

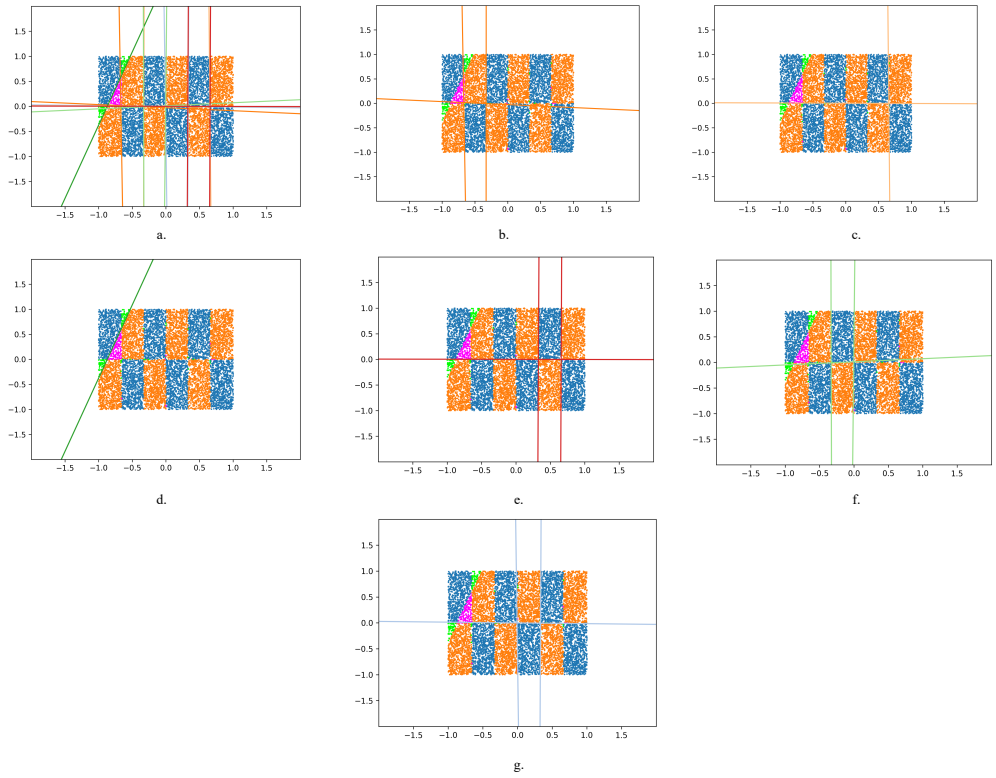

Figure 3: Union of convex separators (a.) along with its constituent separators b, c, d, e, f and g.

Table 8: AUROC values for Union of Convex Separators compared to other algorithms

Datasets	**Philippine**		**Santander (subset)**		**Shoppers**		**Spambase**		**Telco churn**	
Algorithms	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
UCS	0.89	0.77	0.91	0.82	0.92	0.91	0.99	0.98	0.85	**0.85**
Logistic Regression	0.85	0.78	0.87	**0.84**	0.89	0.89	0.98	0.97	0.85	**0.85**
MLP2	0.95	0.75	0.97	0.75	0.95	0.91	1	0.98	0.62	0.62
MLP3	0.89	0.76	0.99	0.72	0.94	0.91	1	0.98	0.81	0.82
Random forest	0.99	**0.84**	0.76	0.56	0.96	**0.92**	1	**0.99**	0.91	0.81
MLP1	0.81	0.79	1	0.73	0.93	0.92	1	0.98	0.81	0.81
SPLA2	0.84	0.8	0.87	**0.84**	0.89	0.89	0.98	0.97	0.85	**0.85**
XGBoost	1	**0.84**	1	0.8	0.98	**0.92**	1	0.99	0.96	0.8

9.1 Variation in the number of convex separators

For this, we run the UCS algorithm for 20 split-grow iterations. However, we limit the maximum number of separators by some number, say c. So for a dataset, when the algorithm is run and due to some false positive, if a split occurs that increases the number of separators beyond c, we discard this false positive permanently and do not consider this to be a part of the dataset and wait for the algorithm to pick a false positive that does not cause an increase in the number of separators. This c is increased and impart on test balanced accuracy and AUROC is observed (see Table 11 and 10). The AUROC values increase with the increase in c for some datasets (4 out of 10 datasets) while staying flat for others (6 out of 10 datasets). Same is the case with balanced accuracy.

Table 9: AUROC values for Union of Convex Separators compared to other algorithms

Datasets	**Breast cancer**		**Churn**		**Covertype (subset)**		**Diabetes**		**Ionosphere**	
Algorithms	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**	**Train**	**Test**
UCS	0.87	0.69	0.86	0.85	0.92	0.88	0.90	**0.86**	1.00	0.96
Logistic Regression	0.76	0.74	0.76	0.75	0.85	0.85	0.86	0.79	0.99	0.91
MLP2	0.95	0.69	0.89	**0.86**	0.94	0.89	0.85	0.85	1	0.94
MLP3	0.84	**0.77**	0.91	0.85	0.98	**0.91**	0.86	0.85	1	0.98
Random forest	0.95	0.71	0.9	0.82	1	**0.91**	0.98	0.85	1	0.97
MLP1	0.95	0.73	0.87	**0.86**	0.92	0.89	0.83	**0.86**	1	**0.99**
SPLA2	0.76	0.74	0.76	0.75	0.85	0.85	0.83	**0.86**	0.99	0.91
XGBoost	0.95	0.61	0.97	0.83	1	**0.93**	0.93	0.8	0.99	0.98

Table 10: Variation in AUROC values with increase in the number of separators

Datasets	**Breast cancer**	**Churn**	**Covertype (Subset)**	**Diabetes**	**Ionosphere**	**Philippine**	**Santander (Subset)**	**Shoppers**	**Spambase**	**Telco churn**
Max separators	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**
5	0.54	0.85	0.88	0.78	0.84	0.76	0.81	0.91	0.98	0.83
10	0.60	0.85	0.88	0.78	0.90	0.79	0.82	0.91	0.97	0.82
15	0.61	0.85	0.87	0.84	0.93	0.78	0.82	0.91	0.98	0.81
20	0.67	0.85	0.89	0.83	0.96	0.79	0.81	0.91	0.97	0.82

Table 11: Variation in balanced accuracy values with increase in the number of separators

Datasets	**Breast cancer**	**Churn**	**Covertype (Subset)**	**Diabetes**	**Ionosphere**	**Philippine**	**Santander (Subset)**	**Shoppers**	**Spambase**	**Telco churn**
Max separators	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**
5	53.55	70.96	79.65	69.64	84.04	69.28	62.44	75.34	92.90	68.22
10	56.32	71.36	79.31	71.83	79.15	71.56	62.83	75.25	91.78	69.35
15	58.24	71.94	78.67	78.08	89.17	69.49	62.24	75.09	93.25	68.00
20	58.77	70.61	80.15	74.65	90.98	72.14	62.17	76.40	92.87	69.22

Table 12: Variation in AUROC values with increase in the size of separators

Datasets	**Breast cancer**	**Churn**	**Covertype (Subset)**	**Diabetes**	**Ionosphere**	**Philippine**	**Santander (Subset)**	**Shoppers**	**Spambase**	**Telco churn**
Max hyperplanes per separator	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**
3	0.63	0.85	0.87	0.80	0.84	0.76	0.81	0.91	0.98	0.84
6	0.65	0.85	0.88	0.83	0.90	0.77	0.82	0.91	0.97	0.82
9	0.73	0.85	0.89	0.84	0.93	0.81	0.82	0.91	0.98	0.85
12	0.65	0.85	0.87	0.79	0.96	0.78	0.82	0.90	0.98	0.83
15	0.68	0.85	0.89	0.76	0.90	0.78	0.83	0.90	0.92	0.83
18	0.66	0.85	0.89	0.78	0.82	0.77	0.81	0.91	0.97	0.83
21	0.77	0.85	0.88	0.77	0.93	0.79	0.82	0.90	0.97	0.82

Table 13: Variation in balanced accuracy values with increase in the size of separators

Datasets	**Breast cancer**	**Churn**	**Covertype (Subset)**	**Diabetes**	**Ionosphere**	**Philippine**	**Santander (Subset)**	**Shoppers**	**Spambase**	**Telco churn**
Max hyperplanes per separator	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**	**Test**
3	58.58	69.74	78.43	69.63	84.04	69.44	63.13	74.13	92.98	69.83
6	64.40	69.09	79.46	75.21	79.15	68.63	63.17	75.93	92.44	70.46
9	67.82	70.45	80.49	74.98	89.17	73.51	64.36	75.15	92.35	71.41
12	59.69	71.41	79.61	70.20	90.98	71.49	63.71	73.96	92.46	69.77
15	57.20	70.55	80.29	67.81	86.56	70.88	62.45	73.71	82.86	68.87
18	60.16	72.50	80.41	67.29	82.59	70.76	62.22	76.44	92.35	69.79
21	67.92	70.52	79.46	69.62	86.88	70.83	62.57	73.56	92.05	68.78

9.2 Variation in the size of convex separators

xFor this we run the UCS algorithm for 20 split-grow iterations. However, we limit the maximum number of hyperplanes that can be added to any of the convex separators, say h. So for a dataset, if a false positive is picked such that it causes the hyperplane count to increase beyond h for some separator, we simply ignore this hyperplane and continue the algorithm so that it later picks some other false positive that belongs to a separator whose size (number of hyperplanes) is less than h. This h is increased and impart on test balanced accuracy and AUROC is observed (see Table 13 and 12). Both AUROC and balanced accuracy seem to increase and then decrease with increasing h, implying the presence of an optimal h (for 5 out of 10 datasets). For the other 5 datasets, it seems to stay flat.

10 Conclusion

In this work, we propose a Union of Convex Separators as a new binary classification method and give an algorithm design to learn the hyperplanes that define this union. The performance has been highlighted. This is still a work in progress requiring improved methods of stopping and a bit of engineering to extract more performance.

References

[1] AutoML - Data — automl.chalearn.org. https://automl.chalearn.org/data. [Accessed 29-09-2023].

[2] Churn Modelling — kaggle.com. https://www.kaggle.com/datasets/shrutimechlearn/churn-modelling/data. [Accessed 29-09-2023].

[3] Santander Customer Transaction Prediction | Kaggle.

[4] Telco Customer Churn — kaggle.com. https://www.kaggle.com/datasets/blastchar/telco-customer-churn/data. [Accessed 29-09-2023].

[5] Adil M. Bagirov. Max–min separability. *Optimization Methods and Software*, 20(2-3):277–296, 2005.

[6] Adil M. Bagirov, Julien Ugon, and Dean Webb. An efficient algorithm for the incremental construction of a piecewise linear classifier. *Inf. Syst.*, 36:782–790, 2011.

[7] Jock Blackard. Covertype. UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C50K5N.

[8] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20:273–297, 1995.

[9] Gabor T Herman and KT Daniel Yeung. On piecewise-linear classification. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 14(07):782–786, 1992.

[10] Reeber Erik Forman George Hopkins, Mark and Jaap Suermondt. Spambase. UCI Machine Learning Repository, 1999. DOI: https://doi.org/10.24432/C53G6X.

[11] Michael Kahn. Diabetes. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5T59G.

[12] Alex Kantchelian, Michael Carl Tschantz, Ling Huang, Peter L. Bartlett, Anthony D. Joseph, and J. Doug Tygar. Large-margin convex polytope machine. In *NIPS*, 2014.

[13] Alexander E. Kostin. A simple and fast multi-class piecewise linear pattern classifier. *Pattern Recognit.*, 39:1949–1962, 2006.

[14] T. Lee and John A. Richards. Piecewise linear classification using seniority logic committee methods, with application to remote sensing. *Pattern Recognit.*, 17:453–464, 1984.

[15] Yujian Li and Qiangkui Leng. Alternating multiconlitron: A novel framework for piecewise linear classification. *Pattern Recognition*, 48(3):968–975, 2015.

[16] Yujian Li, Bowen Liu, Xinwu Yang, Yaozong Fu, and Houjun Li. Multiconlitron: A general piecewise linear classifier. *IEEE Transactions on Neural Networks*, 22:276–289, 2011.

[17] Naresh Manwani and P. S. Sastry. Learning polyhedral classifiers using logistic function. In Masashi Sugiyama and Qiang Yang, editors, *Proceedings of 2nd Asian Conference on Machine Learning*, volume 13 of *Proceedings of Machine Learning Research*, pages 17–30, Tokyo, Japan, 08–10 Nov 2010. PMLR.

[18] Naresh Manwani and P. S. Sastry. Polyceptron: A polyhedral learning algorithm, 2014.

[19] Nimrod Megiddo. On the complexity of polyhedral separability. *Discrete & Computational Geometry*, 3:325–337, 1988.

[20] Hans Christian Palm. A new piecewise linear classifier. *[1990] Proceedings. 10th International Conference on Pattern Recognition*, i:742–744 vol.1, 1990.

[21] Youngtae Park and Jack Sklansky. Automated design of multiple-class piecewise linear classifiers. *Journal of Classification*, 6:195–222, 1989.

[22] Dolev Raviv, Tamir Hazan, and Margarita Osadchy. Hinge-minimax learner for the ensemble of hyperplanes. *Journal of Machine Learning Research*, 19(62):1–30, 2018.

[23] C. Sakar and Yomi Kastro. Online Shoppers Purchasing Intention Dataset. UCI Machine Learning Repository, 2018. DOI: https://doi.org/10.24432/C5F88Q.

[24] Wing S. Hutton L. Sigillito, V. and K. Baker. Ionosphere. UCI Machine Learning Repository, 1989. DOI: https://doi.org/10.24432/C5W01B.

[25] Jack Sklansky and Leo Michelotti. Locally trained piecewise linear classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2:101–111, 1980.

[26] HIROSHI TENMOTO, MINEICHI KUDO, and MASARU SHIMBO. Piecewise linear classifiers with an appropriate number of hyperplanes. *Pattern Recognition*, 31(11):1627–1634, 1998.

[27] Matjaz Zwitter and Milan Soklic. Breast Cancer. UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C51P4M.