Lossless Visual Analysis of Multidimensional Data in Collocated Paired Coordinates for Machine Learning

Boris Kovalerchuk[1] [0000-0002-0995-9539], Alice Williams[1,2] [0009-0001-6154-2407]

[1]Central Washington University, Ellensburg, WA, 98926, USA
[2]Western Washington University, Bellingham, WA, 98225, USA

Corresponding author: Boris Kovalerchuk

Abstract. Artificial Intelligence and Machine Learning methods rely on multidimensional (n-D) data. However, traditional n-D data visualizations use dimension reduction methods for n-D data, which are lossy and do not preserve all n-D information when converting to 2-D or 3-D points thereby warping distances, point orders, outliers, trends, and correlations. This limits the ability to discover truthful n-D patterns. Alternatively, lossless visualization methods preserve all n-D information by converting each n-D point to a mathematically reversible graph with n or n/2 nodes visualized in 2-D or 3-D space. Parallel Coordinates and Radial Coordinates are among these methods. The emerging General Line Coordinates (GLC) generalize them in various ways allowing coordinates to be curvilinear, non-orthogonal, disconnected, overlapping, collocated, shifted, specific geometric forms like, circles, ellipses, squares, n-gons, and others. One such GLC method is Collocated Paired Coordinates (CPC). This paper explores the benefits of CPC to represent and visualize sets of n-D data known as hyper-rectangles or hyperblocks (HBs). HBs are important in supervised machine learning forming interpretable classification rules. The paper presents types of hyperblocks with their lossless visualization in CPC as interpretable classification rules. The algorithms are proposed for discovering and visualizing rules for outliers of hyperblocks in CPC based on hyperblocks, mainstream and counterfactual cases, scatter plots and hierarchical features. The comparative analysis visualization of hyperblocks in CPC and Parallel Coordinates with a case study on real data confirms benefits of CPC for visualizing HBs of different types and discovering interpretable rules.

Keywords: Visual Analytics, Machine Learning, Multidimensional Data, Collocated Paired Coordinates, Outliers, Classification Rules, Hyper-rectangle.

1. Introduction

Artificial Intelligence and Machine Learning (AI/ML) methods fundamentally rely on multidimensional (n-D) data. However, traditional methods of n-D data visualization use dimension reduction of n-D data, which are lossy, such as Principal Component Analysis (PCA), Multi-Dimensional Scaling (MDS), and t-Distributed Stochastic Neighbor Embedding (t-SNE). Lossy methods do not preserve all n-D information when converting to 2-D or 3-D points thereby warping distances, point orders, outliers, trends, and correlations. This limits the ability to use such visualizations to discover truthful n-D patterns. Moreover, lossy representations are commonly static and visualized ad-hoc limiting human interaction.

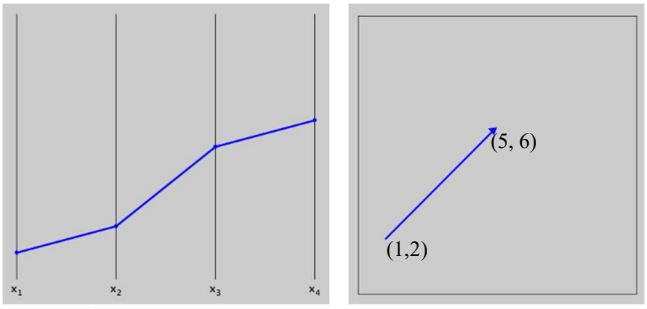
Alternatively, lossless visualization methods preserve all n-D information by converting each n-D point to a mathematically reversible graph with n or $n/2$ nodes visualized in 2-D or 3-D space. The well-known Parallel Coordinates (PC) and Radial Coordinates (RC) are among these methods. The emerging General Line Coordinates (GLC) [1] generalize these coordinates allowing coordinates to be curvilinear, non-orthogonal, disconnected, overlapping, collocated, shifted, specific geometric forms like, circles, ellipses, squares, n-gons, and others. One such GLC method is Collocated Paired Coordinates (CPC).

In this paper we explore the benefits of CPC to represent and visualize sets of n-D data known as hyper-rectangles or hyperblocks (HBs), also referred to as hyperboxes in the literature [2]. HBs are important in ML because they are interpretable representations. An important advantage of CPC over PC and RC is requiring only $n/2$ nodes in the graph of each n-D point. In contrast PC and RC require n nodes for the same n-D point visualized. This enhances human-interaction driven ML model discovery directly in lossless data visualizations. It opens an opportunity to interactively manipulate the hypothesis space using data visualization and graph algorithm approaches. These approaches have been shown to significantly facilitate the discovery of novel patterns in n-dimensional data [3].

Machine learning with hyper-rectangles/hyperblocks. Hyper-rectangles, hyperblocks, and hyperboxes refer to the same underlying concept and have inspired several distinct approaches surveyed in [2,4]. They open opportunities for building efficient models in ML with the following properties: (1) *interpretability* [5], (2) abilities to classify *mixed hyperblocks* [4], (3) *effective Visualization* [6], (4) abilities to be *base models* for *ensemble models* [7, 8], (5) abilities to be *base representation* units for models s*calable to* dynamic and streaming data [2], (6) *flexibility to* handle *binary, discrete, or continuous variables* [9,10], (7) easy *generalization* [8], and (8) abilities to find *most informative hyperblocks* [11].

Comparison of hyperblocks with other state-of-the-art approaches to interpretability. The major difference of the hyperblock approach from popular methods like LIME [12] and SHAP [13] is that the hyperblock approach creates *intrinsically interpretable ML models* from the beginning. In contrast, LIME and SHAP assume that first a non-interpretable black box model is built and then LIME and SHAP attempt to explain it. LIME explains a black box by approximating it locally by a linear classifier, which may not fully represent the underlying model and may not work for heterogeneous data.

The key idea of CPC [1]. Consider a 4-D point $(x_1, x_2, x_3, x_4) = (1, 2, 5, 6)$. As the name indicates, CPC collocates coordinates X_1 and X_3 as a single axis horizontally. Coordinates X_2 and X_4 are also collocated on a single axis vertically. This yields two pairs of collocated Cartesian coordinates (X_1, X_2) and (X_3, X_4). Then, 4-D point $\mathbf{x} = (x_1, x_2, x_3, x_4) = (1, 2, 5, 6)$ is represented by only two 2-D points $(x_1, x_2) = (1, 2)$ and $(x_3, x_4) = (5, 6)$. These points are connected by a directed edge $(1, 2) \rightarrow (5, 6)$ forming an arrow from (1, 2) to (5, 6). This arrow between two nodes losslessly represents the entire 4-D point in just 2-D space, see Fig. 1b. In contrast, PC requires four nodes to visualize the same 4-D point. See Fig. 1a. For higher dimensional points, pairs (x_1, x_2), (x_3, x_4), ..., (x_{n-1}, x_n) are formed and connected by directed edges to produce a directed graph with only $n / 2$ nodes total.



(a) 4-D point in PC. (b) 4-D point in CPC.

Fig. 1. 4-D point **x** = (x_1, x_2, x_3, x_4) = (1, 2, 5, 6) visualized in parallel coordinates (a) and Collocated Paired Coordinates (b).

The idea of hyperblock is as follows. Consider two n-D points **a** and **b**, such that $a_i \leq b_i$ for all $i = 1{:}n$, then a **hyperblock** HB is a set of all n-D points **x** such that $a_i \leq x_i \leq b_i$ for all $i = 1{:}n$. i.e., points **x** are between **a** and **b**: $\mathbf{a} \leq \boldsymbol{x} \leq \mathbf{b}$. Point **a** is called a **low edge** and point **b** is called a **top edge** of the HB. The exploration of benefits of CPC for visualizing HBs losslessly requires developing methods to visualize different types of HBs in CPC individually, and jointly. The paper presents both with demonstrations of specific benefits for specific types of HBs and datasets.

This paper is organized as follows. Section 2 describes the proposed approach that includes types of hyperblocks, methods of visualization of them in CPC, and methods of generating and visualizing classification rules with CPC illustrated with a running example of Iris flowers data. Section 3 describes a case study with the WBC9 dataset [14] for breast cancer classification. The paper concludes with the summary of results and future work.

2. Approach

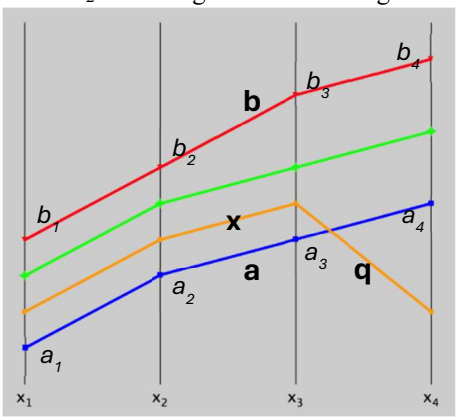
2.1. Types of hyperblocks and their visualization in CPC

The types of HBs that we consider are as follows:

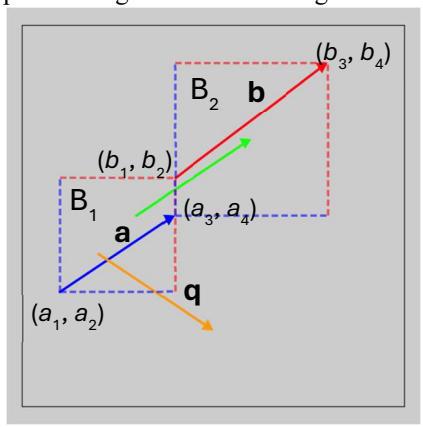
1. Both top and low edges **a** and **b** monotonically grow, $a_i \leq a_{i+1}$ and $b_i \leq b_{i+1}$ for all i.
2. Only the top edges of HBs monotonically grow, $b_i \leq b_{i+1}$ for all i.
3. Only the low edges of HBs monotonically grow, $a_i \leq a_{i+1}$ for all i.
4. No edges of HBs grow monotonically.

These 4 types of HBs are defined for a given order of attributes. If data are not time series, the order of attributes can be freely changed. This implies that HBs of type 4 can be changed to HBs of type 2 or 3 by respective reordering of the attributes. For each type of HB, we explore their visualization alternatives. Occlusion is the major challenge of visualizing multiple HBs jointly when HBs overlap in visualization. Figs. 2 and 3 visualize hyperblocks of type 1 and 2, respectively, in PC and CPC. In PC in Fig. 2a, the HB is represented by an irregular area between blue and red edges. In contrast the HB is represented by two simple rectangles in CPC in Fig. 2b. Also, in Fig. 2a each HB edge consists of 4 points and 3 linear

segments, while in Fig. 2b, the blue edge is a blue arrow, and the red edge is the red arrow. in CPC every 4-D point in the hyperblock is an arrow that starts in the box B_1 and ends in the box B_2. See the green arrow in Fig. 2b that represents a green case from Fig.2a.



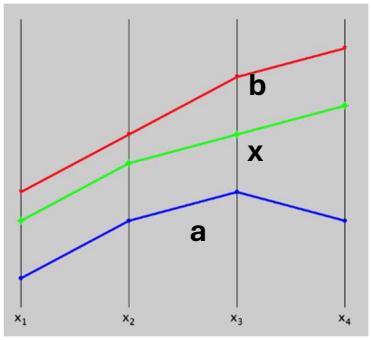
(a) 4-D HB of type 1 in PC as the area between the blue low edge and the red top edge. Both edges are monotone $x_1 < x_2 < x_3 < x_4$.



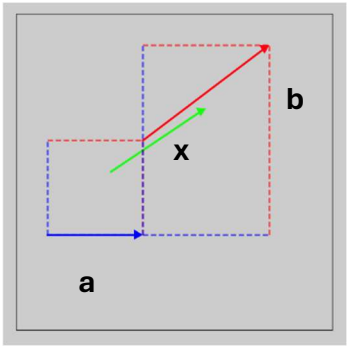
(b) The same 4-D HB in CPC as the area of the dotted rectangles. The HB low edge is the blue arrow, and the top edge is the red arrow. All cases of HB start in box B_1 and end in box B_2.

Fig. 2. HB of type 1 (monotonic) in PC and CPC. The green case is in HB, the orange case is partially in HB.

The justification of hyperblocks in CPC is as follows. First, every 4-D point **x** within HB with low edge **a** and top edge **b** must have $b_1 \geq x_1 \geq a_1$ and $b_2 \geq x_2 \geq a_2$. Point (a_1, a_2) is the left lower corner and point (b_1, b_2) is the top right corner of the dotted lower rectangle. We denote this rectangle as box B_1. It means that the starting points (x_1, x_2) of **x** must be within B_1. Second, every 4-D point **x** within HB must have $b_3 \geq x_3 \geq a_3$ and $b_4 \geq x_4 \geq a_4$. It means **x** must end in the upper dotted rectangle that has the low left corner point (a_3, a_4) and top right corner point (b_3, b_4). We denote this rectangle as box B_2. The orange 4-D point **q** is not



(a) 4-D HB PC as the area between edges.

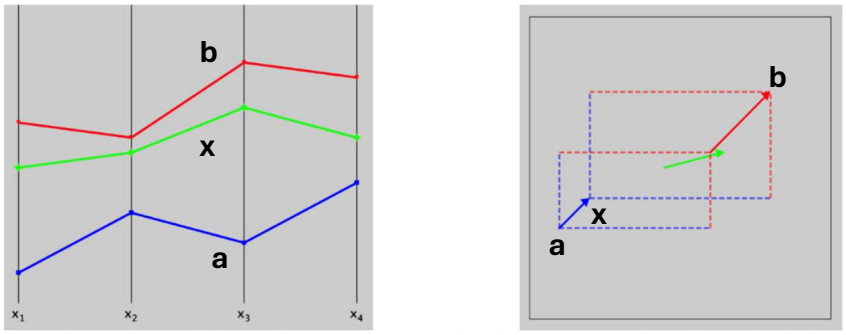


(b) The 4-D HB in CPC as the dotted rectangles.

Fig. 3. HB of type 2 with top edge monotonic in PC and CPC. The green case is in HB.

in the HB because it does not end in B_2. Boxes B_1 and B_2 differ from boxes that can be separately built for **a** and **b** 4-D points, like boxes with opposite corners (a_1, a_2), (a_3, a_4) for **a** and (b_1, b_2), (b_3, b_4) for **b.** Next, Figs. 3 nd 4 present methods of visualizing HBs of types 2 and 4, respectively, in PC and CPC.

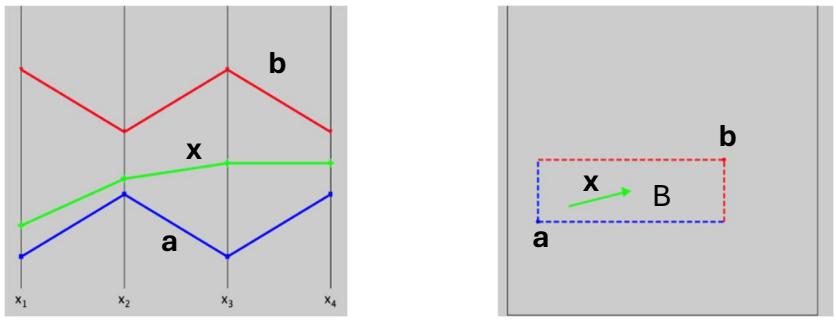
An example of a zig-zag case where the attribute values of edges repeat $a_1 = a_3$, $a_2 = a_4$ and $b_1 = b_3$, $b_2 = b_4$ is shown in Fig. 5. In CPC edges **a** and **b** collapse to a single point because $(a_1, a_2) = (a_3, a_4)$ and $(b_1, b_2) = (b_3, b_4)$. This makes zig-zag HB visualization in CPC simplified to a single box B where a 4-D green point is located, which is its advantage over parallel coordinates. While for HB type 4 we need 2 rectangles but it's still simpler than in PC. We can make one of the edges monotonic by reordering coordinates to simplify CPC Next, we describe the idea of visualizing HBs of dimensions greater than four in Figs. 6-8.



(a) 4-D HB of type 4 in Parallel Coordinates as the area between the blue low edge and the red top edge.

(b) The same 4-D HB in Collocated Paired Coordinates as the area of the dotted rectangles.

Fig. 4. HB of type 4 with non-monotonic edges in PC (a) and CPC (b). The green case is in HB.



(a) 4-D zig-zag HB in PC as the area between edges.

(b) 4-D HB in CPC as the dotted box.

Fig. 5. Visualization of a zig-zag HB of type 4 with equal values in alternating attributes leading to a single box.

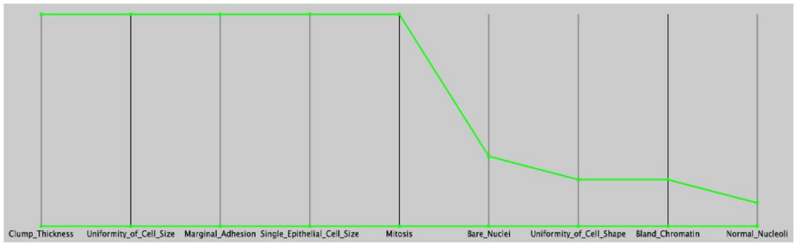


Fig. 6. 9-D HB is Parallel Coordinates.

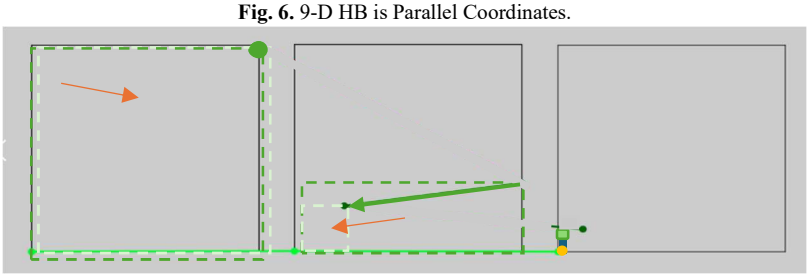


Fig. 7. 9-D HB in CPC as 4D-4D-2D CPC groups. The larger box includes smaller box (nested boxes).

They show a hyperblock from the Wisconsin Breast Cancer 9-D data (WBC9) [14] in PC, CPC and SPC. These visualizations with monotonically ordered top edge of HB take advantage of monotonicity to simplify visualization for faster visual analysis.

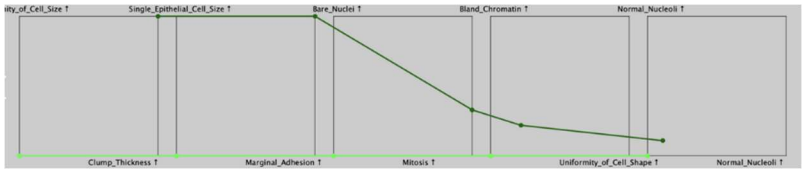
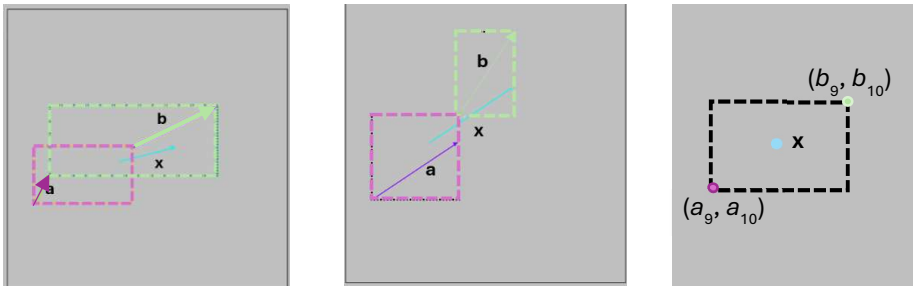


Fig. 8. 9-D hyperblock is Shifted Paired Coordinates (SPC).

Fig. 9 shows an artificial 10-D hyperblock in CPC. The coordinates are grouped by multiples of four, and the remaining attributes with less than four are kept as is or can be duplicated to have a full four attributes.



(a) HB in first 4 coordinates. (b) HB in second 4 coordinates. (c) HB in last 2coordinates.

Fig. 9. Visualized 10-D HB in CPC with grouped coordinates.

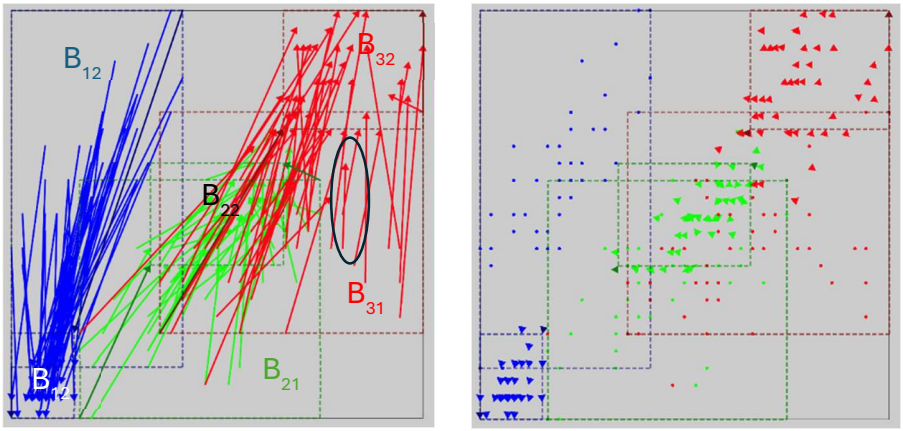
2.2. Rules for outliers of 4-D hyperblocks in Collocated Paired Coordinates

Fig. 10 shows the Fisher Iris dataset [15] with and without arrows and hyperblocks. In Fig. 10b each arrow is presented by a small arrow as the start point and a dot as the end point. It allows to decrease occlusion. These examples illustrate that in CPC each hyperblock HB_i of 4-D data consists of a pair of rectangles (blocks, boxes) – (B_{i1}, B_{i2}) described in the previous section. For Iris classes we found several sets of HBs. Fig. 10 shows one of these sets with HB_1, HB_2, HB_3 for red, blue and green classes. Some cases of other classes are present in these hyperblocks. Also, some cases of those classes are left out of their HBs. They are **outliers for HBs** like a red case in the oval in Fig. 10a.

Building rules for outliers. Below we present a method to build rules for outliers. We define binary properties, predicates for each HB_i with its pair of blocks (B_{i1}, B_{i2}) and $\mathbf{x}=(x_1,x_2,x_3,x_4)$

$$P_{i1}(\mathbf{x}) = \text{true} \Leftrightarrow (x_1, x_2) \in B_{i1}, P_{i2}(\mathbf{x}) = \text{true} \Leftrightarrow (x_3, x_4) \in B_{i2} \quad (1)$$

In Fig. 10 red and green HB_2 and HB_3 consist of 4 boxes B_{21}, B_{22}, B_{31} and B_{32} and, respectively, four binary properties P_{21}, P_{22}, P_{31}, and P_{32}. Each 4-D point $\mathbf{x}$ can be linked to a 4-D binary vector. For instance (1010) indicates that $P_{21}(\mathbf{x}) = 1$ (true), $P_{22}(\mathbf{x}) = 0$ (false), $P_{31}(\mathbf{x}) = 1$ (true), and $P_{32}(\mathbf{x}) = 0$ (false). For cases $\mathbf{x}$ in HB_2 it can be written as a logical conjunctive clause (P_{21} & $\neg P_{22}$ & P_{31} & $\neg P_{32}$) with omitting $\mathbf{x}$ in a logical rule



(a) All cases fully visualized as arrows. (b) All cases partially visualized.

Fig. 10. Iris data with hyperblocks in Collocated Paired Coordinates.

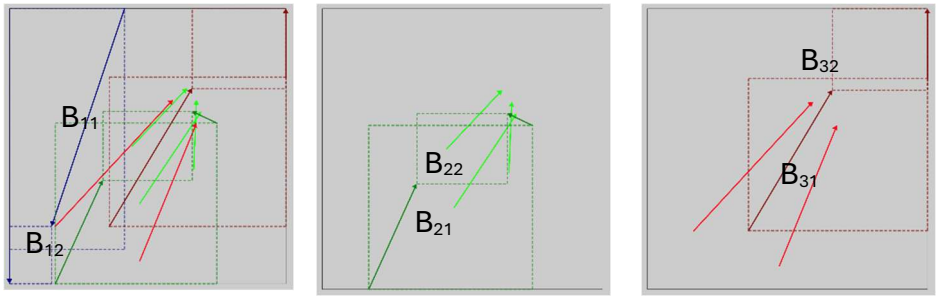
If for $\mathbf{x}$ (P_{21} & $\neg P_{22}$ & P_{31} & $\neg P_{32}$) then $\mathbf{x} \in HB_2$ (2)

When HB_2 is for class 2 then this logical rule is a classification rule for class C_2.

Now consider the red case in the oval on Fig. 10. It is not in HB_3 while it belongs to class C_3. It has a clause ($\neg P_{21}$ & $\neg P_{22}$ & P_{31} & $\neg P_{32}$) belonging only to box B_{31}. We can produce a rule for this outlier

If for **x** ($\neg P_{21}$ & $\neg P_{22}$ & P_{31} & $\neg P_{32}$) then **x** ∈ HB_3 (3)

Below we illustrate this idea for HB outliers shown on Fig. 11.



(a) HBs and 5 outliers. (b) HB of class C_2 with 3 outliers of this class. (c) HB of class C_3 with 2 outliers of this class.

Fig. 11. Three iris hyperblocks and 5 outlier cases for these hyperblocks.

The common properties of red 4-D outlier points of class C_3 are as follows. They start in B_{21}, $(x_1, x_2) \in B_{21}$ i.e., predicate $P_{21}(\mathbf{x}) = 1$. They also end outside of blue box B_{22}, $(x_3, x_4) \notin B_{22}$, i.e., predicate $P_{22}(\mathbf{x}) = 0$, $\neg P_{22}$. See Fig. 11b. It distinguishes them from other cases of class 2 that start in box B_{21}, $(x_1, x_2) \in B_{21}$ and end in B_{22}, $(x_3, x_4) \in B_{22}$. It makes these three cases outliers of HB_2. What is their unique property making them not cases of HB_3 of their class 3? Those cases must end in B_{32}, but red cases do not end in B_{32}, $(x_3, x_4) \notin B_{32}$, i.e., predicate $P_{32}(\mathbf{x}) = 0$, $\neg P_{32}$. Thus, we can write a rule that combines properties of these cases,

If $(x_1, x_2) \in B_{21}$ & $(x_3, x_4) \notin B_{22}$ & $(x_3, x_4) \notin B_{32}$ then **x** ∈ Class 3 (4)

It also can be written with predicates

If P_{21} & $\neg P_{22}$ & $\neg P_{32}$ then **x** ∈ Class 3 (5)

Now we can analyze green cases on Fig. 11b. They do not belong to HB_3 of class 3 because they do not start in B_{31}, $(x_1, x_2) \notin B_{31}$ and do not end in B_{32}, $(x_3, x_4) \notin B_{32}$. They also do not belong to HB_2 of class 2 because they do not end in B_{22}, $(x_3, x_4) \notin B_{22}$. Despite these properties they belong to class 2, being outliers of this class. Thus, we can write a rule as a combination of these listed properties of these two blue cases,

If B_{22}, $(x_3, x_4) \notin B_{22}$ & $(x_1, x_2) \notin B_{31}$ & $(x_3, x_4) \notin B_{32}$ then **x** ∈ Class 2 (6)

It also can be written in an abbreviated predicate form

If $\neg P_{22}$ & $\neg P_{31}$ & $\neg P_{32}$ then **x** ∈ Class 2 (7)

For completeness we can add to (6) and (7) that **x** does not belong to HB_1 of class 1.

$(x_1, x_2) \notin B_{11}$ & $(x_3, x_4) \notin B_{12}$ (8)

These rules have an advantage or real generalization to classify outliers correctly beyond just memorizing them that often happen when classifiers produce overfitting models. It also avoids heuristics to define the size of vicinities of memorized cases to generalize them.

These examples give us an idea of the algorithm to build rules for outliers. In includes (1) building a clause such as ($\neg P_{21}$ & $\neg P_{22}$ & P_{31} & $\neg$P$_{32}$) for an outlier of two HBs, (2) testing if all cases that satisfy the given clause are from a single class under some class purity threshold, e.g., 95%, and (3) if for the given clause the answer for testing is true, then building rules like rules (1) - (7) for this clause.

Outlier rule algorithm (OutR) for two or more hyperblocks in 4-D (OutR). The main steps of the algorithm are as follows:
Input: A set of hyperblocks **H** = {HB$_i$} with boxes (B_{i1}, B_{i2}) found on dataset S of 4-D data for 2 classes and a set U of 4-D points that do not belong to {HB$_i$} called outliers for **H.**
Step 1: For each outlier **x** from set U search and record boxes $\{B_{ij}\}$ that contain (x_1, x_2) and boxes $\{B_{km}\}$ that do not contain (x_1, x_2). Build properties $\{P_{ij}\}$ and $\{P_{km}\}$ from these boxes.
Step 2: For each outlier **x** from set U search and record boxes $\{B_{ij}\}$ that contain (x_3, x_4) and boxes $\{B_{km}\}$ that do not contain (x_3, x_4). Build properties $\{P_{ij}\}$ and $\{P_{km}\}$ from these boxes.
Step 3: Combine properties found in steps 1 and 2 into rules like (1) - (7).
Step 4: Check that no rule satisfies two or more classes. If such rules are found, then exclude them searching for alternative rules for these cases by other methods.

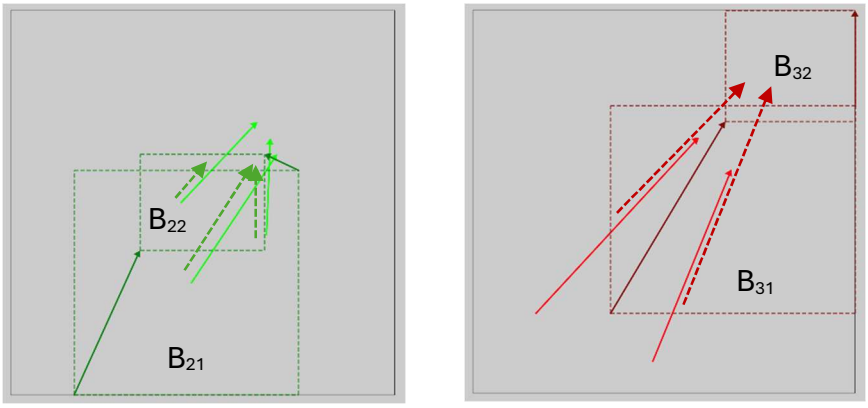
2.3. Rules for outliers with mainstream and counterfactual cases

It is possible that the algorithm above will build rules for outliers only for a subset of outliers and other approaches and methods are needed to deal with them that we present below.

Outlier substitution approach. The first idea is using visualization in CPC to analyze outliers versus **mainstream cases** that are in hyperblocks. We are looking for mainstream cases that resemble outliers. Fig. 12 shows mainstream cases as dashed arrows of the same colors as outliers. A domain expert can explore substituting outliers by dashed cases as representing the class more correctly. With a positive conclusion the training data can be updated by those mainstream cases.

Outlier generative approach. If the conclusion is negative, the domain expert can suggest adding more cases like outliers to better represent the class in training data and discover updated hyperblocks on updated data. In this situation the outliers would not be called outliers anymore. Visualization in Fig. 12 helps the domain expert to design such new cases.

Counterfactual generative approach. The next approach to deal with outliers is based on counterfactual cases. It visualizes and generates counterfactuals for any real or synthetic case in CPC. Fig. 13 shows dashed cases that are **counterfactual cases** for outliers found in CPC visualization. These cases belong to another class and respectively colored to that class. A dashed green arrow from HB$_2$ of class 2 is a counterfactual case for the red case of class 3. Respectively, red dashed arrow from HB$_3$ of class 3 is a counterfactual case for the green outlier of class 2. It is visible on Fig. 13 that the green outlier has a longer red counterfactual, and the red outlier has a shorter green counterfactual. A domain expert can explore if the produced counterfactuals can be added to the training data to represent the classes more completely and to retrain the HB model.



(a) Class 2. (b) Class 3.

Fig. 12. Dashed cases like outliers that belong to HBs of the same class as candidates to substitute outliers.

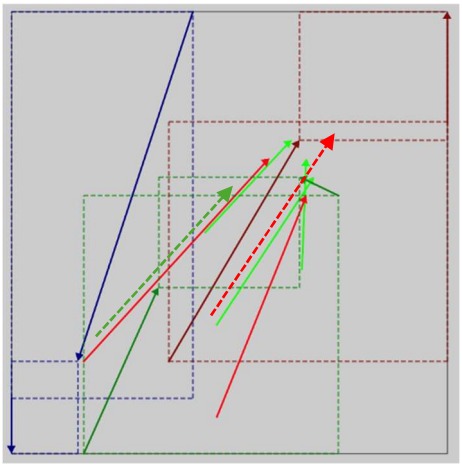
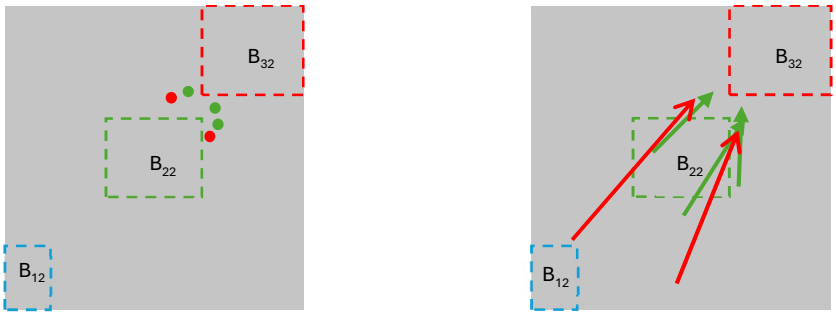


Fig. 13. Dashed counterfactual cases for outliers that belong to another class.

The important advantage of this design of counterfactuals is that produced synthetic counterfactuals are interpretable being based on hyperblocks with readily available explanation. Traditional methods either (1) search for counterfactuals among existing training cases or (2) produce synthetic counterfactuals. Both require setting up a similarity measure between cases that is typically a heuristic measure, which is difficult to justify and explain to the end user. In addition, (2) requires computing the values of the predictive model M for the synthetic case. When model M is a black-box model it is difficult to justify and explain its prediction to the end user too.

2.4. Rules for outliers from scatter plot and hierarchical feature development

Below we present a way of **hierarchical development of features and classification rules** for outliers starting from standard scatter plots combined with collocated paired coordinates at the next stage. We use Fig. 14 to present the approach. First, analysis of data in the prior figures shows that just 2-D boxes B_{12}, B_{22} and B_{32} are sufficient to represent respective Iris hyperblocks HB_1, HB_2 and HB_3 because these boxes do not overlap. See Fig. 14. All these boxes are in (x_3, x_4) coordinates. Therefore, we can visualize Iris data simply as a traditional scatter plot in these two coordinates only. Fig. 14 shows 5 outliers outside of boxes as green and red dots, where red dots are closer to B_{22} than green dots that are closer to B_{32}, i.e., not to their own classes. Thus, a common nearest neighbors' approach will misclassify these cases. We lost 4-D information that was available with (x_1, x_2). In contrast, Fig. 14b shows outliers as arrows with their full 4-D information in CPC. It allowed us to build interpretable rules for these outliers in section 2.2 by using full 4-D information. This is an opportunity of **hierarchical development of features and classification rules** when two 2-D hyperblocks are combined to build interpretable rules while only (x_3, x_4) are not sufficient to build such meaningful interpretable rules in Fig. 14 a.



(a) Outliers as dots in (x_3, x_4). (b) Outliers as arrows in (x_1, x_2) and (x_3, x_4).

Fig 14. Three non-overlapping boxes of hyperblocks of three Iris classes with outliers.

Below is a modification of OutR algorithm from section 2.2 for discovering features and producing interpretable rules from Scatter Plots and CPC (**S-CPC algorithm**) as follows:
Step 1: Build $HB_1(x_1, x_2)$ and $HB_2(x_1, x_2)$ for classes C_1 and C_2 in coordinates (x_1, x_2).
Step 2: Build $HB_1(x_3, x_4)$ and $HB_2(x_3, x_4)$ for classes C_1 and C_2 in coordinates (x_3, x_4).
Step 3: Find outliers $OT(x_1, x_2)$ and $OT(x_3, x_4)$ for HBs from steps 1 and 2
Step 4: Visualize all outliers from step 3 in CPC with coordinates (x_1, x_2) and (x_3, x_4) collocated.
Step 5: Visualize all HBs boxes from steps 1 and 2 in the CPC from step 4.
Step 6: Describe each outlier in terms of HBs from steps 1 and 2 as it is done in the outlier rule algorithm OutR.
Step 7: Build rules for outliers as it is done in the outlier rule algorithm OutR.

In contrast with outR algorithm, this algorithm does not build 4-D HBs but builds 2-D "hyperblocks" (simple 2-D boxes) that domain expert can construct interactively without a sophisticated knowledge in machine learning and extensive computations. There is a risk

that these boxes can be insufficient to build rules for outliers. If it happens, then building full 4-D HBs will be required by OutR algorithm. If this fails, then the next level of the hierarchy will be needed with the use of sets of 4-D CPC visualizations. The S-CPC algorithm uses the concept of hierarchy starting from simple 2-D boxes and then building more complex **features** as their combinations that may not be hyperblocks to form rules. In this aspect S-CPC algorithm generalizes OutR algorithm.

2.5. Rules with hierarchical feature development

The process in the section above resembles neural networks that aggregate individual input values x_i of attributes like pixels of images. The S-CPC algorithm starts from pairs (x_i,x_j), while it could be modified to start from x_i and search for 1-D intervals first like Divide and Classify algorithm in [8]. It builds simple rules like if $x_3 < T_3$ then $\mathbf{x} \in C_1$, and if $x_4 < T_4$ then $\mathbf{x} \in C_2$ for Iris data. For many other datasets such simple rules do not exist and properties like $x_3 < T_3$ do not yield a case classification but can be considered as new **engineered features** to build more complex features by combining them hierarchically as it is done in convolutional neural networks with other properties and aggregation methods. With CPC we *build interpretable features hieratically* starting from simple low-dimensional hyperblocks with pairs of attributes in 2-D, then generalize to attributes in 4-D in CPC and so on with more attributes as this section demonstrates. Below we specify the concept of interpretable feature discovering and engineering.

Types of features in the form of predicates:

Base Feature type BF1: $F(\mathbf{x}) = 1 \Leftrightarrow x_i \le T$, $F(\mathbf{x}) = 1 \Leftrightarrow x_i \ge T$. Decision trees algorithms combine only these features as their conjunctions like $F_1(\mathbf{x})$ & $F_2(\mathbf{x})$ & … & $F_k(\mathbf{x})$ for each branch of the decision tree. No feature of the higher level is created as combinations of these features. Such algorithms often are called shallow algorithms vs. deep learning algorithms.

Base Feature type BF2: $F(\mathbf{x}) = 1 \Leftrightarrow T_1 \ge x_i \ge T_2$. Hyperblock algorithms combine only this type of features as their conjunctions, like $F_1(\mathbf{x})$ & $F_2(\mathbf{x})$ & … & $F_k(\mathbf{x})$ for each hyperblock.

Categories of interpretable combinations that form new **features combined** (**FC**):

Feature Combined FC1: Conjunction $C = F_1(\mathbf{x})$ & $F_2(\mathbf{x})$ & … & $F_k(\mathbf{x})$ of features of the same type. Example, $C_1 = F_1(\mathbf{x})$ & $F_2(\mathbf{x})$ and $C_2 = F_3(\mathbf{x})$ & $F_4(\mathbf{x})$ are combined to a new feature $C = C_1$ & $C_2 = F_1(\mathbf{x})$ & $F_2(\mathbf{x})$ & $F_3(\mathbf{x})$ & $F_4(\mathbf{x})$.

Feature Combined FC2: Conjunction $C = F_1^*(\mathbf{x})$ & $F_2^*(\mathbf{x})$ & … & $F_k^*(\mathbf{x})$ of features of the same type, where * indicate that negation of the properties is allowed. Expansion of algorithms OutR and S-CPC to high dimensions use these features.

Feature Combined FC3: Conjunction $C = F_1^*(\mathbf{x})$ & $F_2^*(\mathbf{x})$ & … & $F_k^*(\mathbf{x})$ of features of types 1 and 2. Algorithms OutR and S-CPC can use features of type 1 when the dimension is odd like 9 for a single dimension left after 8 dimensions are used in two 4-D CPC.

Feature Combined FC4: Disjunction of Conjunctions $D = C_1$ v C_2 v .... v C_m. In mathematical terms it is a disjunctive normal form DNF. Any hyperblock based algorithms including algorithms OutR and S-CPC can be expanded to these features.

Feature Combined FC5: Conjunction of Disjunctions $CD = D_1^*$ & D_2^* & … & D_k^*. Also, any hyperblock based algorithms can be expanded to these features.

This process differs from the hierarchical feature development in deep learning convolutional neural network (CNN) algorithms for images. In CNN, the sliding window of pixels (e.g., 3x3) and its shift (stride) to form features are selected heuristically before exploring values in the pixels. Next in CNN, heuristic usage of only one value such as max or average from pixels of the sliding window is not based on its factual usefulness but on its expectation to be learned later. In CNN, the use of a more complex masking to get the pooled output from the pooling windows usually is weighing or selecting some elements inside each pooling region. This requires knowing which elements inside each pooling region contribute to the pooled output. Also, this is fundamentally down sampling.

In our approach base features BF1 and BF2 are set up from the beginning by discovering them on the training data in an interpretable way. They are actual features of that data. In addition, in CNN the combination of features through using weights is not clearly interpretable. In contrast our combination with the logical formulas preserves interpretability of base features. On the other side our approach needs computationally efficient procedures to be developed. CNN has relatively efficient computational procedures to learn weights. An option here is using our features with weights when producing the hierarchical structure. This is in line with the idea of Neural Decision Trees and similar ideas of differentiable and soft decision trees [16-20]. The use of CNN structure here will require numeric output, while our features are binary predicates. Weights can be applied to them, but it will not change the value for predicates when $P(\mathbf{x})=0$. Therefore, we can substitute such predicate by its negation $\neg P(\mathbf{x})=1$, which can be updated with the weights more efficiently.

2.6. Visualization of hyperblocks in CPC, SPC and PC

Visualizing hyperblocks without overlapping in visualization space is difficult when they do not overlap in n-D space. Figures above illustrate it. Only some hyperblocks with specific properties can be visualized without overlapping as shown on Fig. 15.

A simple way to avoid overlapping hyperblocks in visualization is to show HBs separately side-by-side not in a single plot. It works when the number of HBs is about 20 [5, 6]. For example, 16 HBs and 16 coordinates lead to 16 plots in 16-D parallel coordinates. In SPC and CPC, a 16-D PC plot will be changed to plots that are 2 and 4 times smaller respectively. The deficiency of this side-by-side approach is in the difficulty of seeing clearly the difference between hyperblocks. Below we explore ways to resolve this deficiency.

We use an important property of non-overlapping HBs: for each pair of HBs that do not overlap in n-D space, at least one coordinate exists where these HBs do not overlap. This property follows directly from the definition of the non-overlapping HBs that requires existence of such coordinate that we will call a **separating coordinate**.

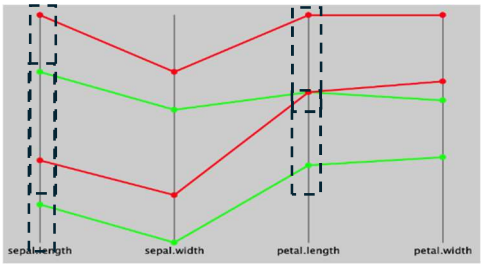
Below we show how this property benefits **HB visualization in SPC** in the proposed **HBVisS algorithm:**

Input: A set of {HB} in coordinates $X_1, X_2, \ldots, X_n$.

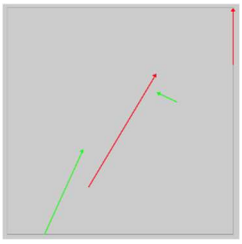
Step 1: Find all Separating Coordinates for each pair of HBs: (HB_i, HB_j).

Step 2: Select a pair (HB_i, HB_j) and one of its separating coordinates X_k. If several separating coordinates X_k exist for (HB_i, HB_j) prefer X_k that is not a separating coordinate for other HBs that differ from (HB_i, HB_j) to use those coordinates to separate other pairs.

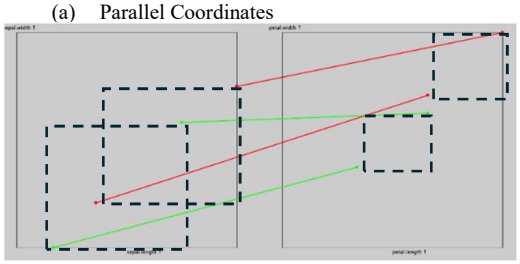
Step 3: Put coordinate X_k as a horizontal coordinate X.
Step 4: Select another coordinate X_m for Y coordinate to form pair of coordinates (X_k, X_m). If several separating coordinates X_m exist for (HB_1, HB_j) prefer X_m that is not a separating coordinate for other HBs that differ from (HB_1, HB_j) to be able to use those separating coordinates for other pairs.
Step 5: Project all HBs to (X_k, X_m) as rectangles (boxes).
Step 6: Highlight HB_i and HB_j that X_k separates and grey out or make semi-transparent all other HBs in (X_k, X_m).
Step 7. Repeat steps 2-6 for other pairs of HBs until all HBs will be selected.



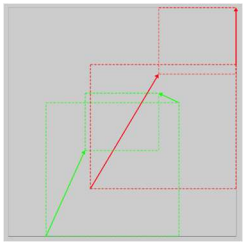
(a) Parallel Coordinates



(b) Collocated Paired Coordinates without boxes



(c) Shifted Paired Coordinates

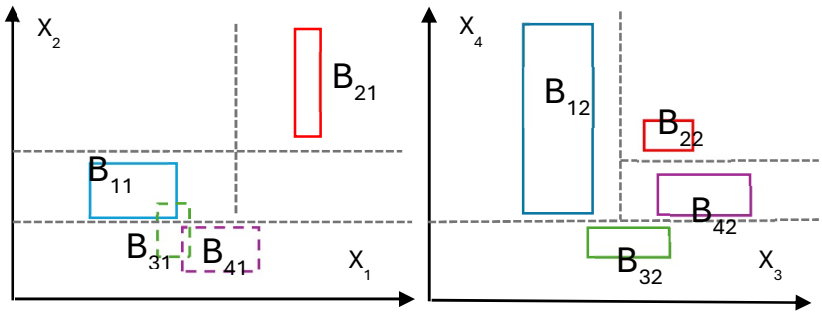


(d) Collocated Paired Coordinates with boxes

Fig. 15. Two hyperblocks in Parallel Coordinates and Shifted Paired Coordinates (a) and CPC.

HBVisS algorithm is applied on Fig. 16 to 4 HBs. In (X_1, X_2), HB_1 and HB_2 are separated by both X_1 and X_2 for their B_{11} and B_{21} boxes by grey dotted lines. In contrast HB_3 and HB_4 are not separated in (X_1, X_2), while another grey horizontal line separates HB_4 from HB_1 and HB_2. In (X_3, X_4), HB_3 and HB_4 are separated by a grey horizontal dotted line based on X_4 and another horizontal dotted grey line separates HB_2 and HB_4. Next, a vertical dotted line based on X_3 separates HB_1 from HB_2 and HB_4. In Fig. 16, we simplified visualization by using colored labeled boxes and omitting connecting lines between boxes of the same HB.

This algorithm will visualize without overlapping n hyperblocks, where n is the space dimension (two HBs per pair of coordinates for n / 2 pairs of coordinates). For n = 10 and 10 HBs all 10 HBs will be visualized in 5 pairs of coordinates plots but not all pairs of HB will be visualized with their separation lines by this algorithm in general. However, when some line separates several HBs, this algorithm can require fewer plots. For example, on Fig. 16, four HBs have six possible pairs (HB_1, HB_2), (HB_1, HB_3), (HB_1, HB_4) (HB_2, HB_3), (HB_2, HB_4), ($HB3_3$, HB_4) and all these pairs are separated in Fig. 16 with only two plots.



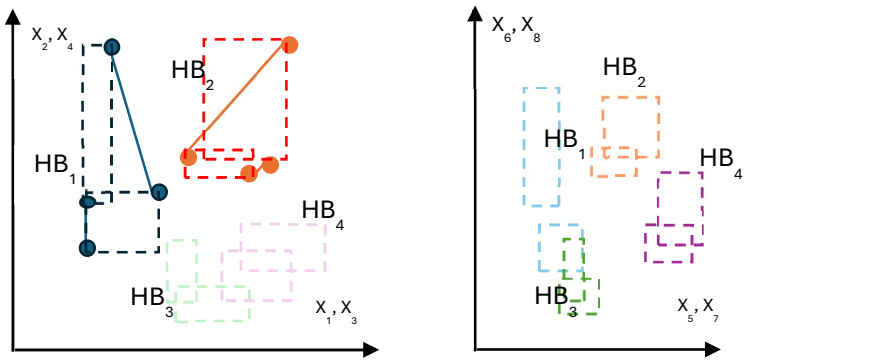
(a) (X_1,X_2) pair. (b) (X_3,X_4) pair.

Fig. 16. Example of applying algorithm HBVisS.

This algorithm modified for **HB visualization in CPC** denoted as **HBVisC** is as follows.
Input: Set of {HB} in coordinates $X_1, X_2, \ldots, X_n$.
Step 1: Find all separating coordinates for each pair of HBs: (HB_i, HB_j).
Step 2: Select a pair (HB_i, HB_j) and its separating coordinate X_k.
Step 3: Put two coordinates X_k and X_q collocated as a horizontal coordinate X.
Step 4: Select two other coordinates X_m and X_h collocated as Y coordinate to form two collocated pairs of coordinates (X_k, X_m) and (X_q, X_h). Give preference to coordinates that separate only current (HB_i, HB_j) to apply other coordinates to separate other HBs later.
Step 5: Project all HBs to (X_k, X_m) and (X_q, X_h) as pairs of rectangles (boxes).
Step 6: Highlight HB_i and HB_j that X_k separates and grey all other HBs in (X_k, X_m).

HBVisC algorithm is used on Fig. 17. Four HBs have 6 possible pairs separated on Fig. 17 with HB_3 and HB_4 separated for B_{32} and B_{42} showing that full separation is not necessary. This algorithm visualizes without overlapping *n* HBs, where *n* is the space dimension (2 HBs per two pairs of collocated coordinates for n / 2 pairs of collocated pairs of coordinates). For $n = 10$ and 10 HBs all 10 HBs are visualized in 5 plots. We can cut the number of plots if the vertical coordinate Y separates additional hyperblocks in the same plot, see Fig. 16.

Limitations. These algorithms visualize all HBs, but not all pairs of HBs. For 10 HBs we have 45 pairs of 10 HBs and need 45 SPC or CPC pairs. Separate visualization of all 45 pairs of HBs in the single screen is not feasible. We cannot visualize *m* hyperblocks in *n*-D space in SPC, CPC and PC in the screen with $p \times k$ pixels for some combinations of *m*, *n* and $p \times k$ due to insufficient number of screen pixels. For some HBs this limitation can be decreased. Having 10 attributes and 45 pairs of non-overlapping HBs, each attribute will separate 4-5 pairs of HBs on average. Thus, potentially a single CPC plot can show several pairs of separable HBs or use animation to show separable pairs of HBs sequentially in a CPC plot.



(a) Collocated (X_1, X_2) and (X_3, X_4) pairs. (b) Collocated (X_5, X_6) and (X_7, X_8) pairs.

Fig. 17. example of applying algorithm HBVis-CPC.

3. Case study

3.1. Visualization of single 9-D hyperblock in PC and CPC

Fig. 18 shows hyperblock HB1 from WBC9 dataset with attributes in the **descending order** of their values of the top edge. Wisconsin Breast Cancer (WBC9) dataset [14] is a 9-D dataset that contains 683 cases with 444 benign cases and 239 malignant cases. For this dataset hyperblocks were generated by the IMHyper program [6] using all 9 attributes of this dataset. Each HB is defined by its top and bottom edge cases that are not real cases but rather built from the bounds of real cases. Using cases that are real could be a topic of future work.

The **descending order** of the values of the top edge simplifies visualizations of this HB in PC and CPC while CPC visualization is more compact than PC visualization with the same 9-D information. Also, relations between boxes in CPC became simpler to analyze because they are nested after ordering attributes. In Fig.19, the black arrows show how cases can traverse between boxes in CPC. In Fig. 19a we omitted uninformative attributes x_1-x_4 to simplify it.

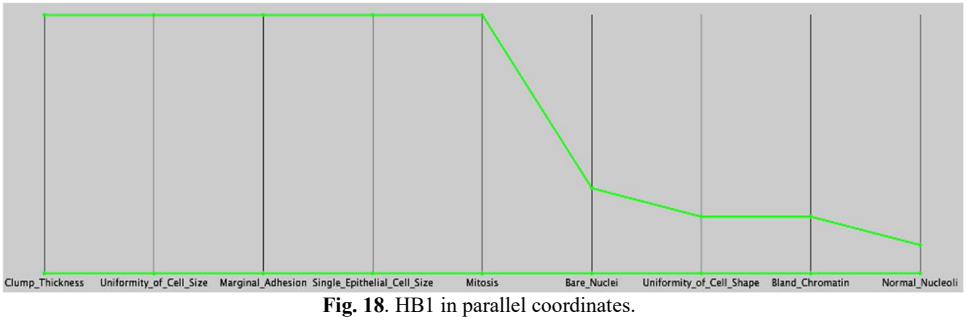
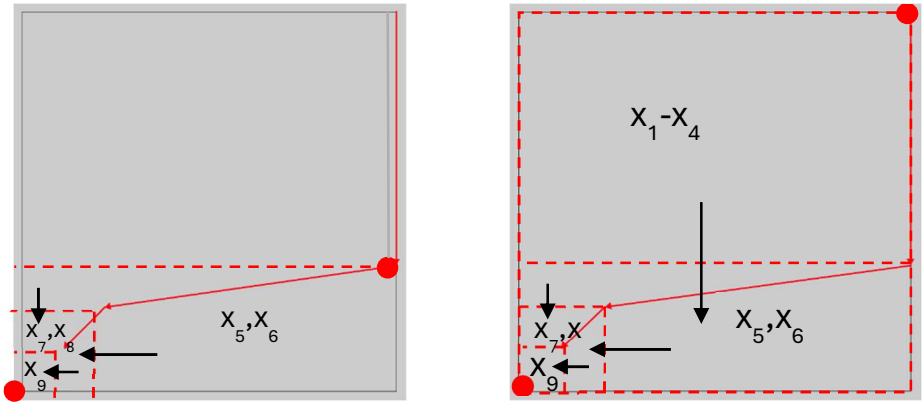


Fig. 18. HB1 in parallel coordinates.



(a) HB1 in CPC with omitted uninformative x_1-x_4. (b) HB in CPC with all coordinates x_1-x_9.

Fig. 19. Hyperblock HB1 of WBC9 dataset in collocated paired coordinates.

3.2. Visualization of two separable hyperblocks in PC

Fig. 20 and Table 1 show that hyperblocks HB8 and HB6 of WBC9 dataset are separated only in the attribute normal nucleoli with a gap between top of HB6 (0.56) and bottom of HB8 (0.89). The separation of these HBs is visible, while they overlap in PC visualization being non-overlapping in 9-D space.

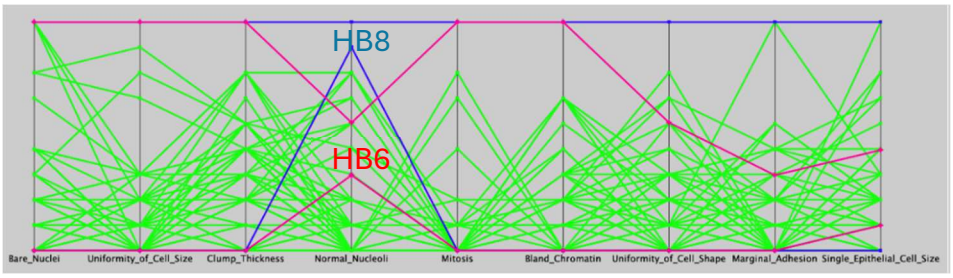


Fig. 20. HB6_and HB8 in parallel coordinates with benign cases shown. All attributes are sorted by Linear Discriminant Analysis (LDA). Attribute Normal nucleoli is a separating attribute for HB6 and HB8.

Table 1. HB6 and HB8 with order of attributes as in Fig. 20 relative to original order of attributes.

Case	1	2	3	4	5	6	7	8	9
HB6 top	1.0	1.0	1.0	**1.0**	1.0	1.0	1.0	1.0	1.0
HB6 bot	0.0	0.0	0.0	**0.89**	0.0	0.0	0.0	0.0	0.0
HB8 top	1.0	1.0	1.0	**0.56**	1.0	1.0	0.56	0.33	0.44
HB8 bot	0.0	0.0	0.0	**0.33**	0.0	0.0	0.0	0.22	0.11

3.3. Visualization of two overlapping hyperblocks

Below we analyze two large pure hyperblocks shown on Fig. 21. The pure benign HB4 contains 359 benign cases without malignant cases, and pure malignant HB10 contains 96 malignant cases without benign cases. These two HBs share four attributes with the same

maximal ranges (Clump Thickness, Uniformity of Cell Shape, Single Epithelial Cell Size, and Mitosis). The remaining five attributes have smaller ranges. These attributes are Uniformity of Cell Size, Bland Chromatin, Marginal Adhesion, Bare Nuclei, and Normal Nucleoli.

Moreover, in HB4 Bare Nuclei and Normal Nucleoli have the same interval of values. Similarly, in HB10 Bare Nuclei and Normal Nucleoli have also the same interval of values but different from HB4 as Fig. 21 shows. This duplication of attribute values allowed us to remove one of these attributes. We removed Normal Nucleoli attribute. This leaves 4 attributes to visualize the separation of these HBs instead of 9 attributes. This allows for detailed data exploration since the combinatorial space is much smaller.

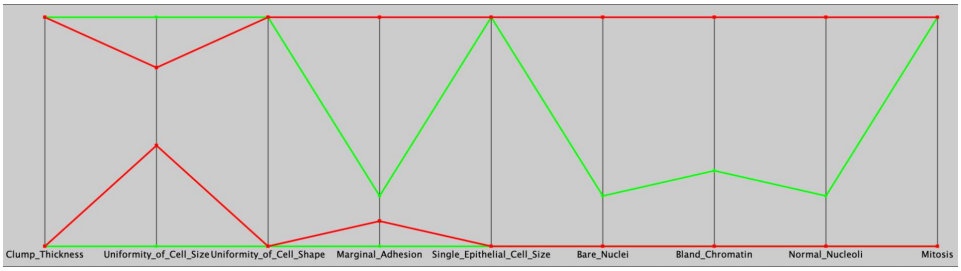


Fig. 21. Pure hyperblocks HB4 and HB10 in Parallel Coordinates.

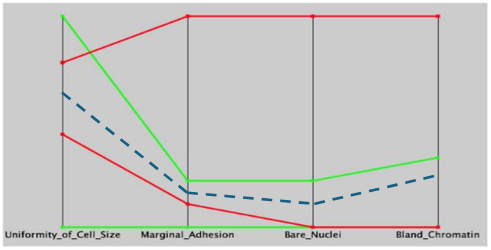
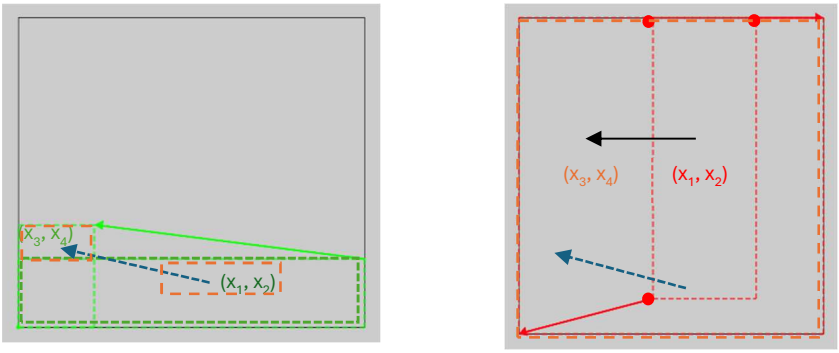


Fig. 22. HB4 and HB10 in parallel coordinates.

Fig. 22 visualizes these HBs in Parallel Coordinates using the default attribute order. This simplified visualization allowed us to discover an important property that HB4 and HB10 not only visually overlap but actually overlap in 4-D and 9-D spaces. This is in contrast with HB6 and HB8 on Fig. 20. In Fig. 22, an artificial dotted blue line belongs to both HB4 and HB10. It is informative because HB4 and HB10 are pure HBs in WBC9 data, that do not contain cases like the blue artificial case. Fig. 22 shows that these HBs do not have an attribute that fully separates them. It clearly identifies the overlap area of HB4 and HB10.

This is an opportunity for cancer domain experts to evaluate how realistic the blue case is and other cases in the overlap area. If they are realistic then the WBC9 dataset is insufficient, and adding new training cases like the blue case is needed. Fig. 23 shows HB4 and HB10 in CPC, where the overlapping areas as orange dotted boxes is shown in Fig 23a with the green case from the overlap area as a dotter blue arrow. The black arrows indicate directions of possible moves from boxes for (x_1, x_2) to boxes for (x_3, x_4).



(a) HB4 of benign class. (b) HB10 of malignant class.

Fig. 23. HB4 and HB10 in CPC.

4. Conclusion

The work has shown abilities of interpretable and lossless visualization of n-D data and hyperblocks in simplified visual forms with Collocated Paired Coordinates. Using CPC allows for full lossless visualization without needing to keep a unique axis for each attribute like in Parallel Coordinates. This is achieved while still being fully lossless General Line Coordinates. Therefore, this visualization is fully reversible.

It enables the use of this approach for machine learning and AI tasks. This includes discovering rules for hyperblock outliers, visualization of hyperblocks and relations between hyperblocks. Furthermore, it allows the visualization of higher dimensional cases in a more compact form. This opens an opportunity for working with higher dimensionality of data as is commonly needed for AI tasks. This work can be expanded to other static and dynamic General Line Coordinates such as In-line Coordinates [21], Circular [22] and Elliptic coordinates [23] to support interpretability of models.

References

1. Kovalerchuk B., Visual Knowledge Discovery and Machine Learning, Springer, 2018
2. Khuat TT, Ruta D, Gabrys B. Hyperbox-based machine learning algorithms: a comprehensive survey. Soft Computing. 2021 (2):1325-63.
3. Kovalerchuk B, Nazemi K, Andonie R, Datia N, Banissi E, (Eds). Artificial Intelligence and Visualization: Advancing Visual Knowledge Discovery, Springer, 2024
4. Kovalerchuk B, Huber L. Multilayer machine learning with visual knowledge discovery. Information Visualization. 2025 Oct;24(4):351-79.
5. Hayes D., Kovalerchuk B. Parallel coordinates for discovery of interpretable machine learning models. In: Kovalerchuk B, Nazemi K, Andonie R, Datia N, Banissi E, Eds. Artificial intelligence and visualization: advancing visual knowledge discovery, 2024, pp.125–158. Springer.

6. Huber L, Kovalerchuk B and Recaido C. Visual knowledge discovery with general line coordinates. In: Kovalerchuk B, Nazemi K, Andonie R, Datia N, Banissi E, Eds. Artificial intelligence and visualization: advancing visual knowledge discovery, 2024, pp.159–202. Springer.
7.. Konstantinov AV and Utkin LV. Interpretable ensembles of hyper-rectangles as base models. Neural Comput Appl 2023; 35(29): 21771–21795.
8. Williams A, Kovalerchuk B. Boosting of Classification Models with Human-in-the-Loop Computational Visual Knowledge Discovery. Inter. HCI Conference. Springer LNAI. vol. 15822, pp. 391-412. 2025.
9. Chen L. Nested hyper-rectangle learning model for remote sensing. Photogr. Eng Rem. Sen. 2005;71(3):333–340.
10. Salzberg S. A nearest hyperrectangle learning method. Machine Learning 1991;6: 251–276.
11. Liu J, Ma Y, Zhang H, et al. A modified fuzzy min–max neural network for data clustering and its application on pipeline internal inspection data. Neurocomputing 2017; 238: 56–66.
12. Ribeiro MT, Singh S and Guestrin C. Why should I trust you? Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD Intern. conf., 2016, pp.1135–1144.
13. Lundberg S, Lee S. A unified approach to interpreting model predictions. Adv Neural Inf Pr. Syst 2017; 30.
14. Zwitter, M. & Soklic, M. (1988). Breast Cancer Dataset. UCI Machine Learning Repository.
15. Fisher, R. (1936). Iris Dataset. UCI Machine Learning Repository.
16. Balestriero R. Neural decision trees. arXiv preprint arXiv:1702.07360. 2017 Feb 23.
17. Fuhl W, Kasneci G, Rosenstiel W, Kasneci E. Training decision trees as replacement for convolution layers. In: Proceedings of the AAAI Conference on Artificial Intelligence 2020, Vol. 34, No. 04, pp. 3882-3889.
18. Frosst N, Hinton G. Distilling a neural network into a soft decision tree. arXiv:1711.09784. 2017.
19. Rodriguez DM, Cuellar MP, Morales DP. On the fusion of soft-decision-trees and concept-based models. Applied Soft Computing. 2024 Jul 1;160:111632.
20. Gokhale G, Karimi Madahi SS, Claessens B, Develder C. Distill2Explain: Differentiable decision trees for explainable reinforcement learning in energy application controllers. In: Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems 2024, pp. 55-64.
21. Kovalerchuk B, Phan H. Full High-Dimensional Intelligible Learning in 2-D Lossless Visualization Space. In: Artificial Intelligence and Visualization: Advancing Visual Knowledge Discovery 2024, pp. 41-72. Springer.
22. Williams A, Kovalerchuk B. Synthetic data generation and automated multidimensional data labeling for AI/ML in general and circular coordinates. 28th Intern. Conf. Information Visualisation, 2024, pp. 272-279. IEEE.
23. McDonald R, Kovalerchuk B. Non-linear visual knowledge discovery with elliptic paired coordinates. Integrating Artificial Intelligence and Visualization for Visual Knowledge Discovery 2022, pp. 141-172. Springer