Explainable Machine Learning and Visual Knowledge Discovery

Boris Kovalerchuk

1 Introduction

Visual reasoning and discovery have a long history [30]. Chinese and Indians knew a visual proof of the Pythagorean Theorem in 600 B.C. Many scientists such as Bohr, Boltzmann, Einstein, Faraday, Feynman, Heisenberg, Helmholtz, Herschel, Kekule, Maxwell, Poincare, Tesla, Watson, and Watt have declared the fundamental role that *images* played in their *most creative thinking*.

1.1 Motivation

The major challenge for visual creative thinking and discovering, in multidimensional data (*n*-D data) used in ML, is that we *cannot see multidimensional data with a naked eye*. We need visual analytics tools ("*n-D glasses*") for this. The challenge starts at 4-D. Since only 2-D and 3-D data can be directly visualized in the physical 3-D world, visualization of *n*-D data becomes more difficult with higher dimensions as there is greater loss of information, occlusion, and clutter. Often, we use *non-reversible, lossy Dimension Reduction (DR)* methods such as Principal Component Analysis (PCA) that convert, say, every 10-D point into a 2-D point in visualization. While such reduction of 10 numbers to 2 numbers is valuable, in general, it is **lossy**, and producing **non-interpretable** features [49]. Thus, it can *remove key interpretable multidimensional information* before starting to learn complex *n*-D patterns. Alternative **lossless reversible methods** include General Line Coordinates

B. Kovalerchuk (✉)
Department of Computer Science, Central Washington University, Washington, DC, USA
e-mail: borisk@cwu.edu

L. Rokach et al. (eds.), *Machine Learning for Data Science Handbook*,
https://doi.org/10.1007/978-3-031-24628-9_39

(GLC) [22] that visualize multidimensional data losslessly without dimension reduction or loss of information during the visualization process. GLC projects the n-D points onto 2-D graphs preserving n-D information, but suffers more from occlusion [22] than lossy methods. Thus, there is great interest in the **hybrid methods**, which join the benefits of reversible and non-reversible methods for ML.

1.2 Analytical Versus Visual ML

Below we use the following terms. **Analytical ML models (AML)** are models built by *computational ML methods without using visual graphical tools.* **Visual ML models (VML)** are models built by using *visual graphical tools*. These models differ in many aspects, including generalization of ML training data, as we illustrate below with Iris data in Figs. 1 and 2. Here, Visual ML models generalize more conservatively and intuitively justified more than AML, which rather *overgeneralize.*

The **Human Visual and Verbal Learning** (**HVVL**) process from images shown in Fig. 1 leads to the following verbal description of Iris petal classes:

- Setosa – small length and small width of petal.
- Versicolor – medium length and medium width of petal.
- Virginica – large length and medium to large width of petal.

The Logistic Regression, Random Forest, Naïve Bayes, and Support Vector Machine (SVM) violate these meaningful descriptions (see Fig. 2). All of them allow long and short Versicolor Iris petals that violate the "medium" petal length and width of Versicolor (class 1 in Fig. 2). Next, all of them allow short *Virginica* Iris petal length that violates the "large" length of petal" of Virginica (class 2 in Fig. 2). The Logistic regression and Random forest allow extremely long Setosa Iris petals that violates the "short" length of Setosa (class 0 in Fig. 2).

All these ML methods dramatically overgeneralize unseen data, being quite correct on training data. This is an example of a *very typical overgeneralization* by *common ML methods*. It is a source of predictions errors for many unseen cases. Such errors are not a result of insufficient training data, but *overgeneralized task*

Fig. 1 Examples of Setosa, Versicolor, and Virginica Iris petals

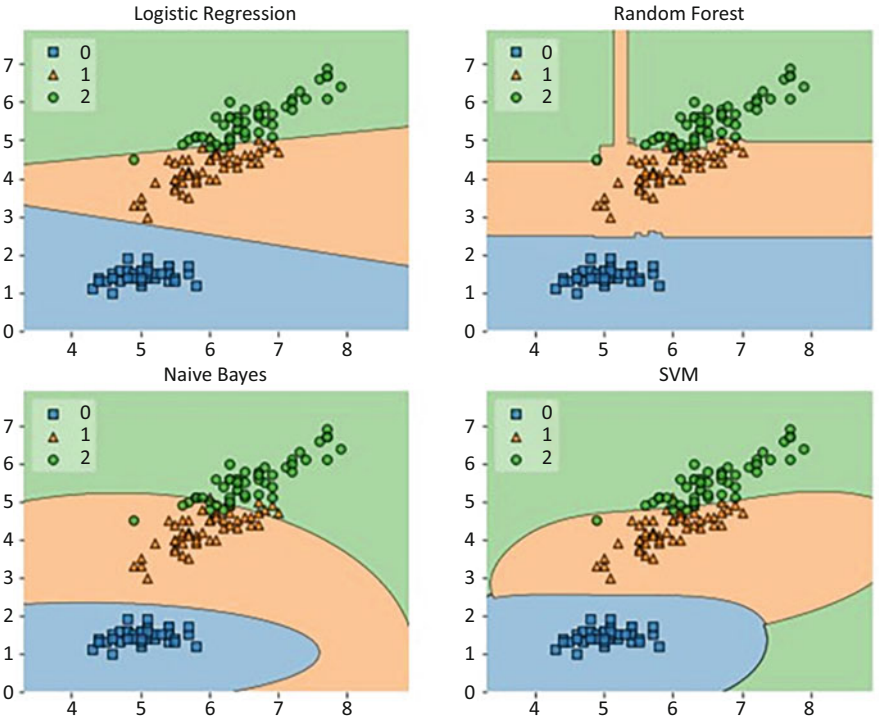


Fig. 2 Example of logistic regression classification of Iris classes. (Reproduced from ML software system [3])

formulation – discovering discrimination functions that classify *every point in the space* by Logistic Regression, Linear Discriminant Analysis, SVM, Naïve Bayes, Random Forest, Decision Trees (DT), and many other ML methods.

Figure 2 is reproduced from [3] that is a software ML package MLxtend with tools for plotting decision regions. It is surprising that not only MLxtend but many other software tools that produce such "decision regions" do not mention and do not point out how severe such "decision regions" overgeneralize classes. A more detailed analysis of these issues can be found in [23].

In contrast, the HVVL does not start with a *predefined set of models* – linear or non-linear functions, which discriminate all points in the space to three classes. It starts with *observation of the training data without any predefined model set*. It generalizes the training data more conservatively because of this observation [23].

This leads to *HVVL-inspired analytical ML models*, which use the envelopes around the training data of each class, e.g., reflecting the small length and width of petal for Iris Setosa. Such algorithms *refuse* to classify points outside of the envelopes. How can we know that we need to use an envelope? We need tools to visualize *n*-D data in 2-D without loss of *n*-D information (**lossless visualization**) allowing to see *n*-D data as we see 2-D data. In the petal example, we have a

linguistic term "*small petal*" and two corresponding 2-D visuals: a picture of the small petal in Fig. 1, and visualization of two of its attributes: length and width that form a "*small blob*" in Fig. 2. These verbal and visual representations are synergetic describing the same physical object in **intelligible** and **well-understood** ways, while both are quite uncertain and need formalization. Subjective probabilistic and fuzzy logic concepts are commonly used to formalize "small" and to "compute" and reason with words. This synergy is going further and deeper to "computing" with images [27]. The Iris petal example illustrates an important property of the **explainable ML** – it can be in *uncertain, but understandable linguistic and visual terms*, not necessarily in the formal mathematical terms.

Black Box Versus Glass Box The *Black Box models* include Deep Learning Neural Networks (DNN), Boosted Trees, Random Forests, and others. It is important to evaluate the interpretability of such methods, by comparing with human explanations, in accuracy [19]. Methods for explaining these models are surveyed in [13].

The *Glass Box models* include single Decision Trees (DT), Naive-Bayes, Bayesian Networks, Propositional and First Order Logic (FOL) rules [31], and others. Often these models are viewed as less accurate, but more intelligible, interpretable, and human understandable than black-box models. This leads to the problem of choosing between accuracy and interpretability. It is a major obstacle to the wider adoption of ML in areas with high cost of error, such as cancer diagnostics, and other domains where it is necessary to understand, validate, and trust decisions. As this chapter shows visual knowledge discovery helps getting both model accuracy and its explanation instead of choosing between them.

1.3 Approaches

Visual Analytics for Machine Learning The goals of visual analytics in ML include: aiding users in *developing* models, *understanding* how complex models work and perform, visually *debugging* models' outputs, analyzing *production*-level models, generating a *workflow* from training to production, analyzing *datasets* and model's *results*, exploring and subdividing large datasets, and comparing the *accuracy* of data groups [4, 10, 14, 18, 32, 39]. These goals belong to the stages of *developing, understanding*, *evaluating, and improving* models. The visual techniques used at these stages overlap. Heatmap is one of them. At the development stage, it visualizes *n*-D input data to get insight for selecting a class of ML models. At the model understanding stage, it highlights the salient elements identified by the model within input data, helping to understand features that the model discovered and used. At the model evaluating and improving stages, it allows seeing inconsistent salient elements to be changed.

The complementary approaches with the different *roles* of analytical, visual, black box, and glass box ML models are (1) *visualization of analytical models*, (2) *discovering analytical models* aided by *visual* methods, (3) *discovering visual mod-*

els aided by *analytical* methods, and (4) *visual explanation* of analytical models. In (1) visual methods are not involved in model creation, but help in understanding, evaluating, and improving the models. In (2) creation of the analytical ML models is guided by visual methods. In (3) analytical methods guide creation of the visual ML models. In (4) visual methods explain fully or partially the analytical models.

This chapter is organized as follows. Section 2 is devoted to visualization of trained ML models. Section 3 covers discovering *analytical* ML models aided by visual methods with case studies in Sect. 4. Section 5 is on methods to scale the process of discovering visual ML models for big data, and Sect. 6 is on methods of visual explanation of analytical models.

2 Visualization of Analytical ML Models

The goals of ML model visualizations include understanding misclassified samples, diagnosis of the model performance, model refinement, and others [34]. **Input-based** and **structure-based** are the two major approaches here. Often in the former, the model is overlaid on the data space with input data visualized. Here, **point–to-point** or **point-to-graph methods** are used to visualize data, which we discuss later.

In the structure-based approach, the model structure is visualized often along with dataflow traced in it. These visualizations are model specific, e.g., visualization of the decision tree differs from visualization of DNN [5], association rules, and the Convolutional Neural Network (CNN) for images [34]. In [34] clustering combines layers, neurons, edges, and only representative ones are visualized to decrease occlusion in visualizing CNN with thousands of neurons and millions of connections.

2.1 Matrix and Parallel Sets Visualization for Association Rules

Below we illustrate the input-based and structure-based ML model visualizations for the association rules (AR). A general AR form is $A \Rightarrow B$, where A and B are statements. Commonly A consists of several other statements, $A = P_1$ & P_2 & $\ldots$ P_k, e.g., *If customers buy both tomato (T) and cucumbers (Cu), they likely buy carrots (Ca).* Here $A = T$ & Cu and $B = Ca$.

The qualities of the AR rule are measured by *support* and *confidence* that express, respectively, *a frequency* of the itemset A in the dataset, and a portion of transactions with A and B relative to frequency of A. ARs are *interpretable* being a class of propositional rules expressed in the original domain terms.

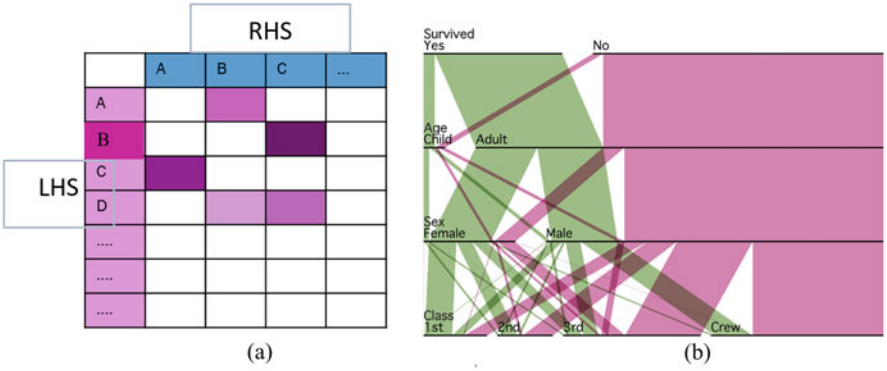


Fig. 3 Matrix and Parallel Sets visualizations of association rules: (**a**) Structure-based visualization of association rules h a matrix and heatmap. (**b**) The input-based model visualization for a set of association rules [46]

The typical questions about ARs are as follows. What are the rules with the highest support/confidence? What are outliers of the rule and their cause? Why is the rule confidence low? What is the cause of rule? What rules are non-interesting? Visualization allows answering some of these questions directly. Figure 3a shows structure-based visualization of association rules with a matrix and heatmap [46]. Here left-hand sides (LHS) of the rules are on the right and right-hand sides (RHS) of the rules are on the top. Respectively, each row shows a possible rule LHS itemset of each rule, and each column shows a possible RHS itemset. The violet cells indicate discovered rules $A \Rightarrow B$ with respective LHS and RHS. The darker color of the rule cell shows a greater rule confidence. Similarly, the darker LHS shows the larger rule support. The major challenges here are scalability and readability for a large number of LHS and RHS [46].

Figure 3b shows the input-based model visualization for a set of ARs. It uses Parallel Sets. Parallel Sets display dimensions as adjacent **parallel axes** and their values (categories) as **segments** over the axes. Connections between categories in the parallel axes form **ribbons.** The segments are similar to points and ribbons are similar to lines in Parallel Coordinates [16]. The ribbon crossings cause clutter that can be minimized by reordering coordinates and other methods [46]. Both model visualizations shown in Fig. 3 are valid for other rules-based ML models too.

2.2 Dataflow Visualization in ML Models

One of the structure-based model visualizations is model dataflow visualization, which is important for complex models such as DNN with difficulties to optimize them and understand how they work. Graph Visualizer, TensorBoard, TensorFlow's dashboard Olah's interactive essays, ConvNetJS, TensorFlow Playground, and

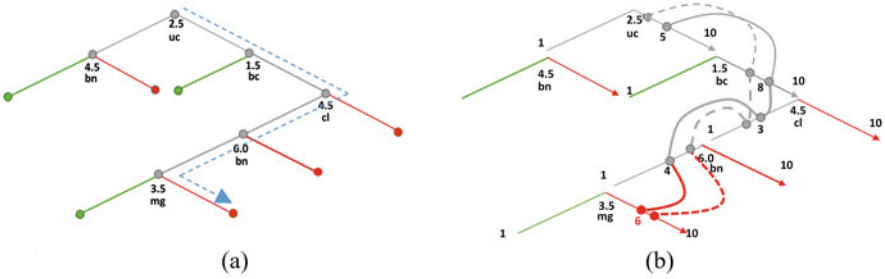


Fig. 4 DT dataflow tracing visualizations for WBC data. (**a**) Traditional visualization of WBC data decision tree. Green edges and nodes indicate the benign class and red edges and nodes indicate the malignant class. (**b**) DT with edges as Folded Coordinates in disproportional scales. The curved lines are cases that reach the DT malignant edge with different certainties due to the different distances from the threshold node

Keras are current tools for DNN dataflow visualization [45]. They allow observing scalar values, distribution of tensors, images, audio, and others to optimize and understand models by visualizing model structure at different levels of detail.

While all these tools are very useful, the major issue is that dataflow visualization itself *does not explain or optimize* the DNN model. An experienced data scientist should guide data flow visualization for this. In contrast, the dataflow for explainable models can *bring explanation* itself, as we show below for Decision Trees (DTs). Tracing the movement of a given n-D point in the DT shows all interpretable decisions made to classify this point. For instance, consider a result of tracing 4-D point $\mathbf{x} = (7, 2, 4, 1)$ in the DT through a sequence of nodes for attributes x_3, x_2, x_4, x_1 with the following thresholds: $x_3 < 5$, $x_2 > 0$, $x_4 < 5$, $x_1 > 6$ to a terminal node of class 1. The point $\mathbf{x}$ satisfies all these directly interpretable inequalities.

Figure 4a shows a traditional DT visualization for 9-D Wisconsin Breast Cancer (WBC) data from UCI Machine Learning repository [48]. It clearly presents the structure of the DT model, but without explicitly tracing individual cases. The trace is added with a dotted polyline in this figure. Figure 4b shows two 5-D points $\mathbf{a} = (2.8, 5, 2.5, 5.5, 6.5)$ and $\mathbf{b} = (5, 8, 3, 4, 6)$. Both points reach the terminal malignant edge of the DT, but with different certainty. The first point reaches it with lower certainty, having its values closer to the thresholds of uc and bn coordinates.

In this visualization, called **Folded Coordinate Decision Tree (FC-DT)** visualization, the edges of the DT not only connect decision nodes, but also serve as *Folded Coordinates* in disproportional scales for WBC data. Here, each coordinate is folded at the node threshold point with different lengths of the sides. For instance, with threshold $T = 2.5$ on the coordinate uc with the interval of values [1, 10], the left interval is [1, 2.5) and the right interval is [2.5, 10]. In Fig. 4b, these two unequal intervals are visualized with equal lengths, i.e., forming a *disproportional scale*.

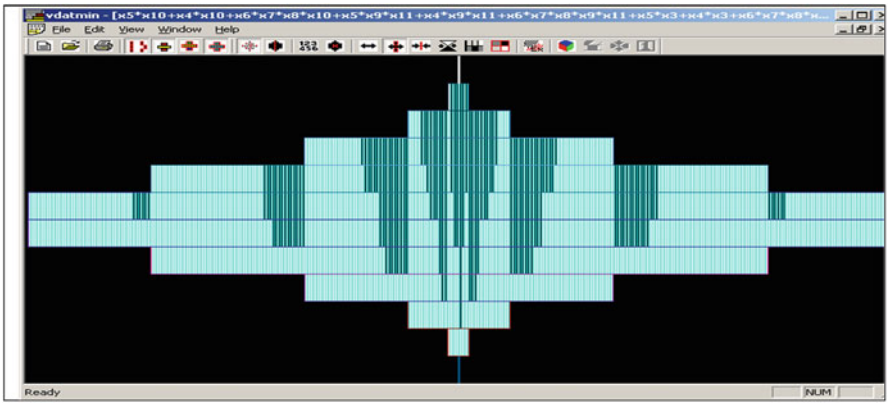


Fig. 5 Full 11-D Boolean space visualized in centered Hansel chain order with discovered classes – malignant (dark) and benign (light) breast cancer cases

2.3 Input-Based ML Model Visualization for Binary Data

This section presents the **MDF algorithm** for input-based ML model visualization for binary input n-D data [28–30]. Such data are common in medical applications with binary symptoms, where parallel line coordinates often produce heavily overlapped visualization of classes due to only two values on each binary coordinate [28, 30]. The method relies on the *monotone structural relations* between Boolean points in the n-D binary cube, E^n, and visualizes them in 2-D as chains of Boolean points located as bars in the multiple disk form (MDF). All 2048 11-D points of the E^{11} Boolean space are shown in MDF in Fig. 5, where each 11-D point is visualized as a dark bar (class 1, malignant) and light bar (class 2, benign).

To construct MDF the algorithm computes the Boolean norms of the n-D vectors – the number of 1s in each point. Then points are grouped according to these norms from 0 to n forming $n + 1$ groups. Each group occupies a strip (disk). Disks are located one on the top of another one. Then, alternative procedures assign the horizontal position to each of the n-D point on the disk.

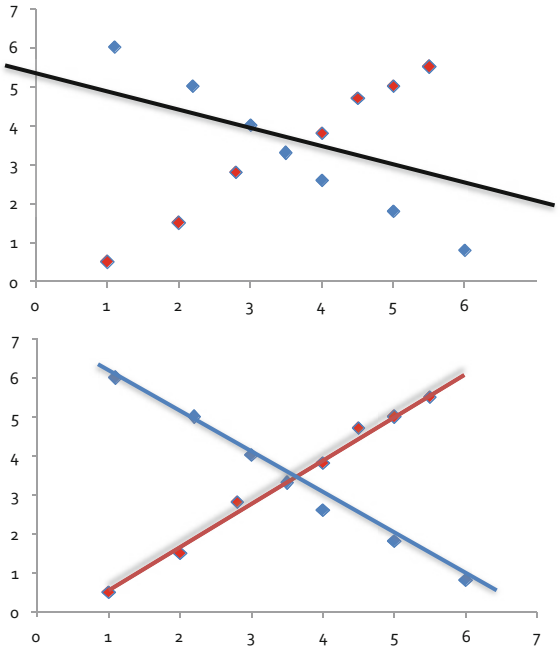
The procedure P_1 orders points according to their numerical decimal value allowing visual comparison of multiple Boolean functions. The procedure P_2 orders points according to monotone Hansel chains [29] allowing, in addition, visualizing the structure of the data along Hansel chains. The procedure P_3 orders the Hansel chains, in addition to P_2, unveiling the border between the two classes of elements, The procedure, P_4 expands Hansel chains from P_3 up and down to points absent in training data generalizing the border between classes using monotone expansion, which is first tested on the training data. Figure 5 shows the resulting directly interpretable visualization of benign and malignant classes in the original cancer features [28].

3 Discovery of Analytical ML Models Aided by Visual Methods

There is a growing trend to move from *visualization of solution* to *discovering the solution visually* [22]. The methods for discovering ML models by visual means rely on lossless and lossy methods for visualizing *n*-D data based on Parallel and Radial Coordinates, RadVis, Manifolds, t-SNE General Line Coordinates, Shifted Paired Coordinates, Collocated Paired Coordinates, and others. Methods of interactive visual model discovery leverage human *perceptual abilities* and human abilities to *adjust tasks on the fly*. Analytical ML methods enhance visual knowledge discovery by *computational means* making visual discovery easier.

As was pointed out above, the main challenge in visual knowledge discovery is that we cannot see patterns in multidimensional data by a naked eye in contrast with 2-D and 3-D spaces. Figure 6 shows an example of visual knowledge discovery in 2-D for the data in the table on the left. Here a single black line cannot discriminate these two "crossing" classes. In addition, the visualization clearly shows that any single line cannot discriminate these classes.

However, a common ML modeling practice (without visualizing the data) starts with a simplest model, which is a linear discrimination function (black line in Fig. 6) to separate the blue and red points. It will fail. In contrast, visualization immediately



x	y	class
1	0.5	1
1.1	6	2
2	1.5	1
2.2	5	2
2.8	2.8	1
3	4	2
3.5	3.3	1
4	3.8	1
4	2.6	2
4.5	4.7	1
5	1.8	2
5	5	1
5.5	5.5	1
6	0.8	2

Fig. 6 "Crossing" classes that cannot be discriminated by a single straight line

gives an insight of a correct model class of "crossing" two linear functions, with one line going over blue points, and another one going over the red points. The question is – how to reproduce such success in 2-D for *n*-D data, where we cannot see the data with a naked eye? The next section presents methods for *lossless and interpretable visualization* of *n*-D data in 2-D for this.

3.1 Approaches to Visualize Multidimensional Data

Often multidimensional data are visualized by lossy dimension reduction (e.g., Principal Component Analysis (PCA), Multidimensional Scaling (MDS), t-SNE) or by splitting *n*-D data to a set of low dimensional data (pairwise correlation plots). While splitting is useful it *destroys integrity* of *n*-D data, and leads to a shallow understanding complex *n*-D data. To mitigate splitting difficulty an additional and difficult perceptual task of assembling 2-dimensional visualized pieces to the whole *n*-D record must be solved. An alternative way for deeper understanding of *n*-D data is developing visual representations of *n*-D data in low dimensions without such data splitting and loss of information as **graphs** not 2-D points.

The examples of such visualization methods for *n*-D data are Parallel Line Coordinates (PLC) where all coordinates are parallel, Radial Line Coordinates (RLC) where all coordinate are radial and other General Line Coordinates (GLC) [22]. Figure 7 shows 6-D point $\mathbf{x} = (4, 3, 0, 5, 4, 10)$ in PLC and in RLC. The major challenges of methods like PCA, MDS, and t-SNE are in the loss of *n*-D information [35] and difficulties to interpret new coordinates, while the major challenge for GLCs is occlusion [22].

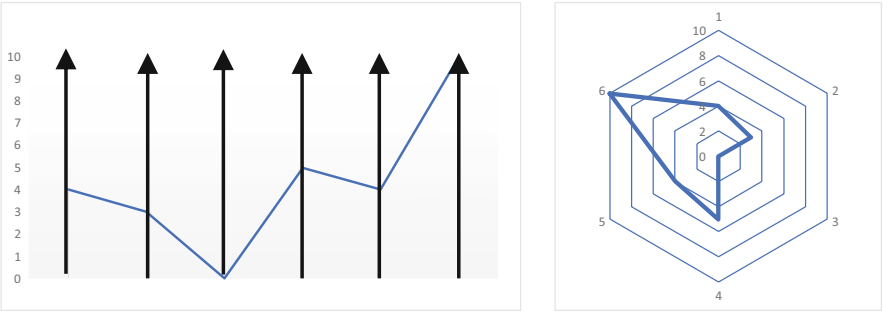


Fig. 7 6-D point $x = (4, 3, 0, 5, 4, 10)$ in PLC and 7-D point in RLC

3.2 Theoretical Limits to Preserve n-D Distances in Lower Dimensions: Johnson-Lindenstrauss Lemma

The source of the loss of information in dimension reduction from n-D to a smaller k-D is in the smaller neighborhoods in k-D. The 2-D/3-D visualization space does not have enough neighbors to represent the same n-D distances in 2-D. For instance, the 3-D binary cube has 2^3 nodes, but the 10-D hypercube has 2^{10} nodes. Mapping 2^{10} 10-D points to 2^3 3-D points leads to the distortion of n-D distances, because the variability of distances between 3-D points is much smaller than between 10-D points. It leads to the significant **corruption** of n-D distances in 2-D visualization. The Johnson-Lindenstrauss lemma states these differences explicitly. It implies that only *a small number* of *arbitrary* n-D points can be mapped to k-D points of a smaller dimension k that *preserves n-D distances with relatively small deviations.*

Johnson-Lindenstrauss Lemma [17] Given $0 < \varepsilon < 1$, a set X of m points in R^n, and a number $k > 8 \ln (m)/\varepsilon^2$, there is a linear map f: $R^n \rightarrow R^k$ such that for all $u, v \in X$.

$$(1 - \varepsilon) \parallel u - v\parallel^2 \leq \parallel f(u) - f(v)\parallel^2 \leq (1 + \varepsilon) \parallel u - v\parallel^2.$$

In other words, this lemma sets up a relation between n, k, and m when the distance can be preserved with some allowable error ε. A version of the lemma [6] defines the possible dimensions $k < n$, such that for any set of m points in R^n there is a mapping *f:* $R^n \rightarrow R^k$ with "similar" distances in R^n *and* R^k between mapped points. This similarity is expressed in terms of error $0 < \varepsilon < 1$.

For $\varepsilon = 1$ the distances in R^k are less or equal to $\sqrt{2}\,S$, where S is the distance in R^n. This means that the distance s in R^k will be in the interval [0, 1.42S]. In other words, the distances will not be more than 142% of the original distance, i.e., it will not be much exaggerated. However, it can dramatically diminish to 0. The lemma and this theorem allow to derive three formulas, to estimate the number of dimensions (sufficient and insufficient) to support the given distance errors. See Table 1. For details, see [22]. These formulas and table show that to keep distance errors within about 30%, for just 10 arbitrary high-dimensional points, the number of dimensions k needs to be over 1900 dimensions, and over 4500 dimensions for 300 arbitrary points. The point-to-point visualization methods do not meet these requirements for arbitrary datasets. Thus, the Johnson-Lindenstrauss Lemma sets up the theoretical limits to preserve n-D distances in 2-D.

3.3 Visual Knowledge Discovery Approaches

Visual knowledge discovery approaches include discovering:

Table 1 Dimensions required supporting errors within ±31%

Number of arbitrary points in *n*-D space	Sufficient dimension by formula 1	Sufficient dimension by formula 2	Insufficient dimension by formula 3
10	1974	2145	1842
20	2568	2791	2397
30	2915	3168	2721
40	3162	3436	2951
50	3353	3644	3130
60	3509	3813	3275
70	3642	3957	3399
80	3756	4081	3506
90	3857	4191	3600
100	3947	4289	3684
200	4541	4934	4239
300	4889	5312	4563

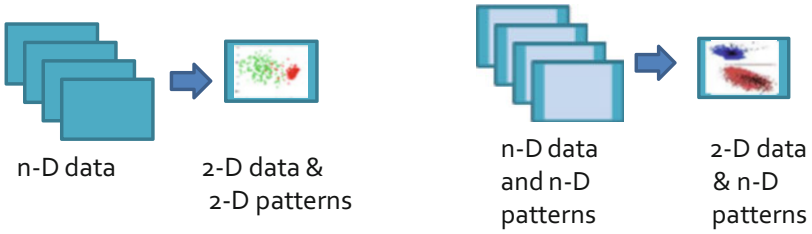


Fig. 8 Visual knowledge discovery approaches

1. **Patterns in 2-D**: *converting* *n*-D data to 2-D data and then *discovering* 2-D patterns in this visualization.
2. **Patterns in *n*-D**: *discovering* *n*-D patterns in *n*-D data then *visualizing* *n*-D patterns in 2-D as *graphs*.
3. **Patterns in 2-D and *n*-D**: some patterns are discovered in (1) with controlled errors and some are discovered in (2).

Figure 8 illustrates the first two approaches.

The patterns in 2-D approach have lossy and lossless versions. The *lossy version* includes three stages:

- **Point-to-point** lossy Dimension Reduction (DR) – converting each *n-D point to a 2-D point*.
- *Visualization* of 2-D points.
- *Interactive discovery* of 2-D patterns in *point-to-point visualization.*

The use of Principal Component Analysis (PCA) as DR with visualization of the first two principal components is an example of this approach. In general, there is no way to restore a given *n*-D point from these two principal components.

The *lossless approach* also has three stages:

- **Point to graph** lossless DR – converting each *n*-D point to a graph in 2-D (*n*-D data fully restorable from the graph).
- *Visualization* of the graph in 2-D.
- Interactive discovery of 2-D patterns on graphs in visualization.

3.4 Point-to-Point Projections

Multiple embedding techniques are in use for input-based ML model visualization, such as Principal Component Analysis, and t-Distributed Stochastic One Neighbor Embedding (t-SNE) [44]. These *point-to-point* projection methods convert *n*-D points to 2-D or 3-D points for visualization. PCA can *distort local neighborhoods* [9]. Often t-SNE attempts preserving local data neighborhoods, at the expense of *distorting global structure* [9]. Below we summarize and analyze the challenges and warnings with t-SNE highlighted in [9, 44]. One of them is that t-SNE may not help to find outliers or assign meaning to point densities in clusters. Thus, outlier and dense areas visible in t-SNE may not be them in the original *n*-D space. Despite this warning, we can see the statements that users can easily identify outliers in t-SNE [5]. It will be valid only after showing that the *n*-D metrics is not distorted in 2-D for the given data. In general, t-SNE similarity in 2-D differs from similarly in *n*-D similarly as shown in the Johnson-Lindenstrauss lemma above.

The AtSNE algorithm [11] is to resolve the difficulties of t-SNE algorithm for capturing the global *n*-D data structure by generating 2-D anchor points (2-D skeleton) from the original *n*-D data with a hierarchical optimization. This algorithm is only applicable to the cases when the global structure of the *n*-D dataset can be captured by a planar structure using point-to-point mapping (*n*-D point to 2-D point). In fact, 2-D skeleton can corrupt the *n*-D structure (see the Johnson-Lindenstrauss lemma above). Moreover, the meaningful similarity between *n*-D points can be *non-metric*. Thus, t-SNE may not *reflect n-D structures in a low-dimensional map* [44]. This is the *most fundamental deficiency of all point-to-point methods*. Therefore, we focus on **point-to-graph** GLC methods, which open a new opportunity to address this challenge.

3.5 General Line Coordinates to Convert n-D Points to Graphs

General Line Coordinates (GLC) [22, 26] break a 400-year-old tradition of using orthogonal Cartesian coordinates, which fit well to modeling the 3-D physical world, but are limited, for lossless visual representation, of the diverse and abstract high-dimensional data, which we deal with in machine learning. GLC *relaxes the requirement of orthogonality*. In GLC, the points on coordinates form **graphs**, where coordinates can overlap, collocate, be connected or disconnected, straight

Table 2 General Line Coordinates (GLC): 3-D visualization

Type	Characteristics
3-D General Line Coordinates (GLC)	Drawing n coordinate axes in 3-D in a variety of ways: curved, parallel, unparalleled, collocated, disconnected, etc.
Collocated Tripled Coordinates (CTC)	Splitting n coordinates into triples and representing each triple as 3-D point in the same three axes; and linking these points to form a directed graph. If n mod 3 is not 0 then repeat the last coordinate X_n one or two times to make it 0.
Basic Shifted Tripled Coordinates (STC)	Drawing each next triple in the shifted coordinate system by adding (1, 1, 1) to the second triple, (2, 2, 2) to the third triple $(i-1, i-1, i-1)$ to the i-th triple, and so on. More generally, shifts can be a function of some parameters.
Anchored Tripled Coordinates (ATC) in 3-D	Drawing each next triple in the shifted coordinate system, i.e., coordinates shifted to the location of the given triple of (anchor), e.g., the first triple of a given n-D point. Triple are shown relative to the anchor easing the comparison with it.
3-D Partially Collocated Coordinates (PCC)	Drawing some coordinate axes in 3-D collocated and some coordinates not collocated.
3-D In-Line Coordinates (ILC)	Drawing all coordinate axes in 3D located one after another on a single straight line.
In-Plane Coordinates (IPC)	Drawing all coordinate axes in 3D located on a single plane (2-D GLC embedded to 3-D).
Spherical and polyhedron coordinates	Drawing all coordinate axes in 3D located on a sphere or a polyhedron.
Ellipsoidal coordinates	Drawing all coordinate axes in 3Dlocated on ellipsoids.
GLC for linear functions (GLC-L)	Drawing all coordinates in 3D dynamically depending on coefficients of the linear function and value of n attributes.
Paired Crown Coordinates (PWC)	Drawing odd coordinates collocated on the closed convex hull in 3-D and even coordinates orthogonal to them as a function of the odd coordinate value.

or curvy and go to any direction. Table 2 describes 3-D GLC types. Figure 9 shows examples of several 2-D GLC types. The case studies below show the benefits of GLC for ML.

4 Case Studies on Visual Discovery

Below we show several examples of case studies with the lossless/reversible GLC approach, for ML model discovery, based on *point-to-graph methodology*.

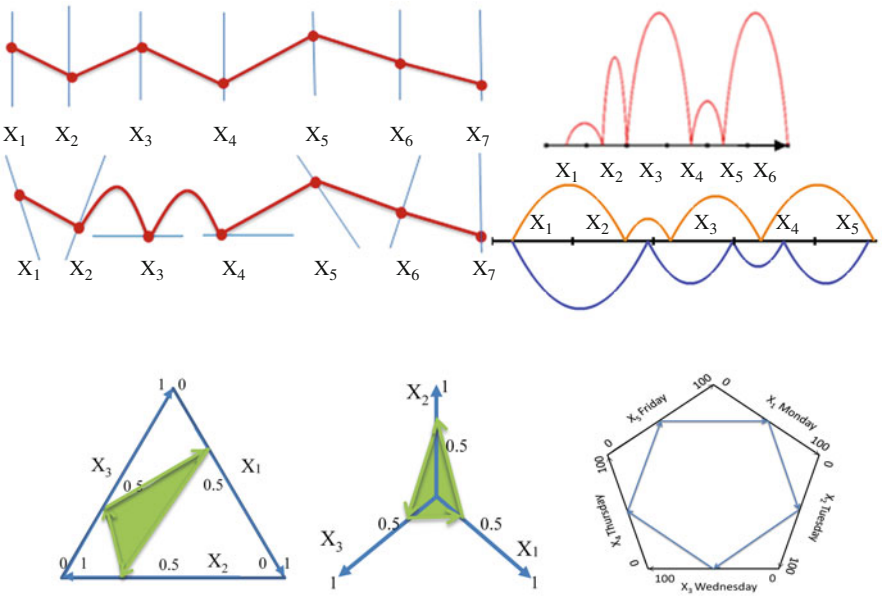


Fig. 9 Examples of different GLCs: Parallel, Non-parallel, Curved, In-line Coordinates Pentagon, Triangular, and Radial Coordinates

4.1 Case Study with GLC-L Algorithm

The first example is on the linear discrimination of 4-D data, by applying GLC-L algorithm [22, 25]. This algorithm allows both visualization of an existing trained ML model, and visual discovery of a new ML model. We start from presenting *visualization of an existing model*. GLC-L can visualize any two-class *linear or additive non-linear discrimination function*,

$$F(\mathbf{x}) = a_1 f_1(\mathbf{x}) + a_2 f_2(\mathbf{x}) + \cdots + a_n f_n(\mathbf{x}),$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. Here if $F(\mathbf{x}) > T$ then $\mathbf{x}$ belongs to class 1, else $\mathbf{x}$ belongs to class 2. In a linear case, each $f_i(\mathbf{x}) = x_i$.

Figure 10 demonstrates the GLC-L for a linear model with four vectors X_1–X_4 in black. Each of them is a coordinate that form a *non-parallel unconnected coordinate system*. This figure also contains four vectors $\mathbf{x}_1$–$\mathbf{x}_4$ (in blue) located on coordinates X_1–X_4 with lengths $|\mathbf{x}_i| = x_i$. These vectors represent a 4-D point $\mathbf{x} = (x_1, x_2, x_3, x_4) = (1, 0.8, 1, 1.2)$. Thus, each blue vector $\mathbf{x}_i$ shows one of x_i from $\mathbf{x}$ and together $\mathbf{x}_1$–$\mathbf{x}_4$ represent losslessly $\mathbf{x}$.

The next step is drawing vectors $\mathbf{x}_1$–$\mathbf{x}_4$ *one after another*, to form a directed graph (see Fig. 10 on the left). Then the last point of this graph $B = A + \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3 + \mathbf{x}_4$ is projected onto the horizontal axis U (see a blue dotted line in Fig. 10). To simplify,

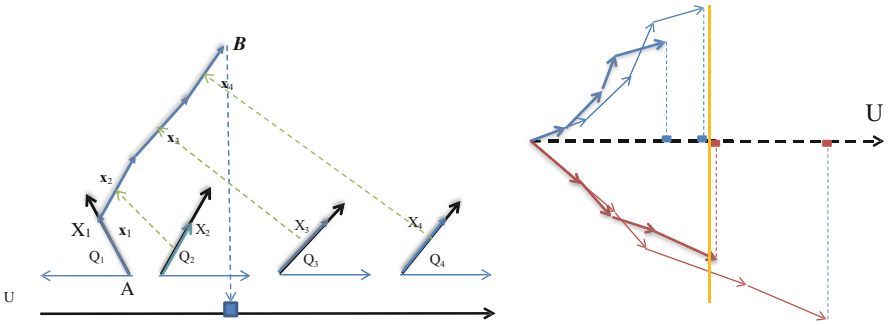


Fig. 10 GLC-L concept

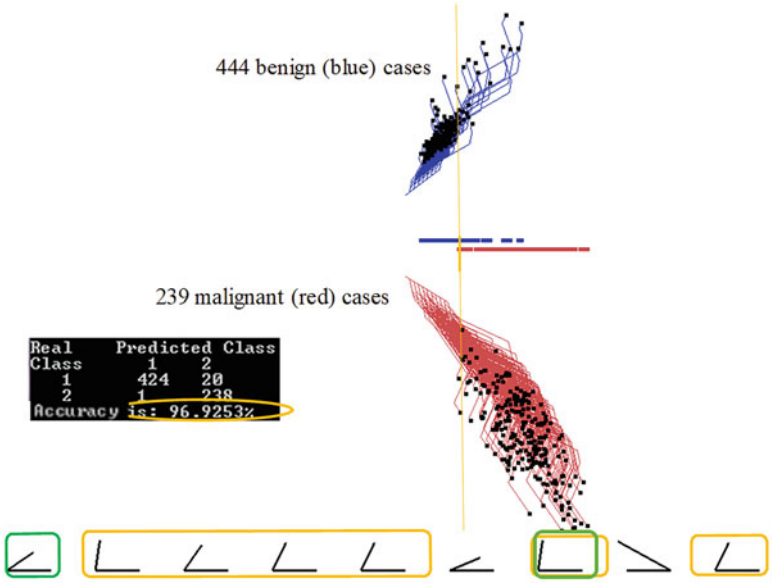


Fig. 11 GLC-L WBC data example

visualization axis U can be collocated with the horizontal lines that define the angles Q_i as shown on the left in Fig. 10. The length of the projection on coordinate U (black horizontal line) from the origin of U is a linear function $F(\mathbf{x})$.

Figure 10 on the right shows graphs of two 4-D points (in blue) and two 4-D points (in red), which are drawn mirrored relative to the axis U. The blue 4-D points belong to class 1 and red ones belong to class 2. The vertical yellow line is a threshold line, which discriminates the two classes. Figure 11 shows the application of the GLC-L algorithm for visual linear discrimination of 9-D Wisconsin Breast Cancer (WBC). The blue polylines (graphs) represent 444 benign cases and red ones 239 malignant cases with vertical yellow line discriminating these two classes.

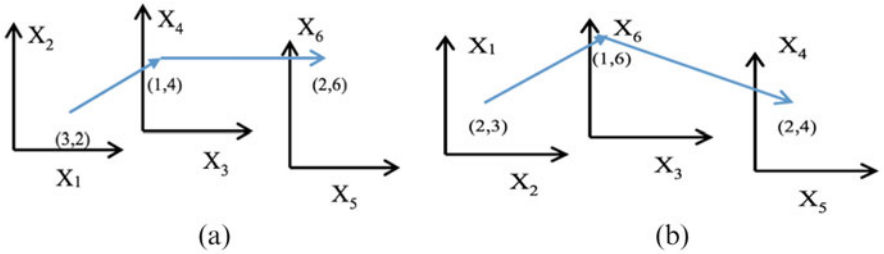


Fig. 12 6-D point **a** = (3, 2, 1, 4, 2, 6) in Shifted Paired Coordinates. (**a**) Point **a** in (X_1, X_2), (X_3, X_4), (X_5, X_6) as a sequence of pairs (3, 2), (1, 4) and (2, 6). (**b**) Point **a** in (X_2, X_1), (X_3, X_6), (X_5, X_4) as a sequence of pairs (2, 3), (1, 6) and (2, 4)

The confusion matrix shows that it misclassified 1 malignant case and 20 benign cases with the total accuracy of 96.92% [22, 25]. To get a 100% accurate classification of cancer cases, the threshold line can be moved to the left interactively. It will misclassify more benign cases, which is a less of a problem, than missing a cancer case. The angles at the bottom of Fig. 11 show the contribution of each attribute to the discrimination function $F(\mathbf{x})$. The angles with green outlines contribute the most (have larger positive coefficients in $F(\mathbf{x})$). A user can interactively adjust angles (and respective importance and contribution of attributes) to get a more meaningful and interpretable classification model. This example shows the idea of using the GLC-L algorithm beyond just visualizing the already existing ML classification model, but using GLC-L to find a better model by modifying the existing one. Alternatively, a GLC-L classification model can be built from scratch by assigning coefficients (via angles) and sliding thresholds interactively. The advantage of GLC-L is its explanatory power. It provides a *full and interpretable visual representation* of the function $F(\mathbf{x})$ without any loss of n-D information.

4.2 Case Study with the FSP Algorithm

This case study deals with the same 9-D WBC data by using the FSP algorithm [24] and Shifted Paired Coordinates (SPC) [26] for a graph representation of n-D points. The idea of SPC is presented in Fig. 12. The **Shifted Paired Coordinates** (**SPC**) visualization of the n-D data requires the splitting of n coordinates X_1–X_n into pairs producing the $n/2$ non-overlapping pairs (X_i, X_j), such as (X_1, X_2), (X_3, X_4), (X_5, X_6), . . . ,(X_{n-1}, X_n). In SPC, a pair (X_i, X_j) is represented as separate orthogonal Cartesian Coordinates (X, Y), where X_i is X and X_j is Y, respectively.

In SPC, each coordinate pair (X_i, X_j) is *shifted* relative to other pairs to avoid their overlap. This creates $n/2$ scatter plots. Next, for each n-D point $x = (x_1, x_2, \ldots, x_n)$, the point ($x_1$, x_2) in (X_1, X_2) is connected to the point (x_3, x_4) in (X_3, X_4) and so on until point (x_{n-2}, x_{n-1}) in (X_{n-2}, X_{n-1}) is connected to the point (x_{n-1},

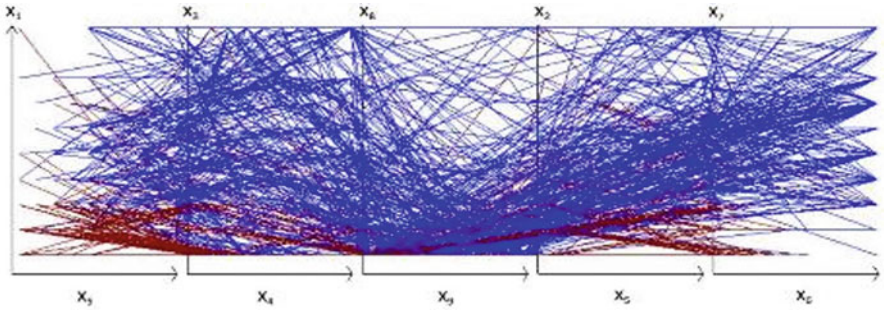


Fig. 13 Benign and malignant WBC data visualized in SPC as 2-D graphs of 10-D points

x_n) in (X_{n-1}, X_n) to form a directed graph x^*. Figure 12 shows the same 6-D point visualized in SPC in two different ways due to different pairing of coordinates.

The FSP algorithm has three major steps: *Filtering* out the less efficient visualizations from the multiple SPC visualizations, *Searching* for sequences of paired coordinates that are more efficient for classification model discovery, and *Presenting* the model discovered with a best SPC sequence, to the analyst [26]. The results of FSP applied to CPC graphs of WBC data are shown in Figs. 13, 14, and 15. Figure 13 illustrates the motivation for filtering and searching in FSP. It shows WBC data in SPC, where graphs occlude each other making it difficult to discover the pattern visually. Figure 14 presents the results of automatic filtering and searching by FSP algorithm. It shows only cases that are located outside of a small violet rectangle at the bottom in the middle, and go inside of two larger rectangles on the left. These cases are dominantly cases of the blue class. Figure 15 presents the remaining cases, which go through the small rectangle and do not go to the larger ones. The most of these cases are from the red class. Together these properties provide a rule:

$$\text{If } (x_8, x_9) \in R_1 \,\&\, (x_6, x_7) \notin R_2 \,\&\, (x_6, x_7) \notin R_3 \text{ then } \mathbf{x} \in \text{class Red else } \mathbf{x} \in \text{class Blue},$$

where R_1 and R_2 and R_3 are three rectangles described above. This rule has accuracy of 93.60% on all WBC data [26]. This fully interpretable rule is visual and intelligible by domain experts, because it uses only original domain features and relations.

This case study shows the benefits of combining analytical and visual means for interpretable knowledge discovery. The analytical FSP algorithm works on the multiple visual lossless SPC representations of n-D data, to find the interpretable patterns. While occlusion blocks discovering these properties by visual means, the analytical FSP algorithm discovers them in the SPC simplifying the pattern discovery, providing the explainable visual rules, and decreasing the cognitive load.

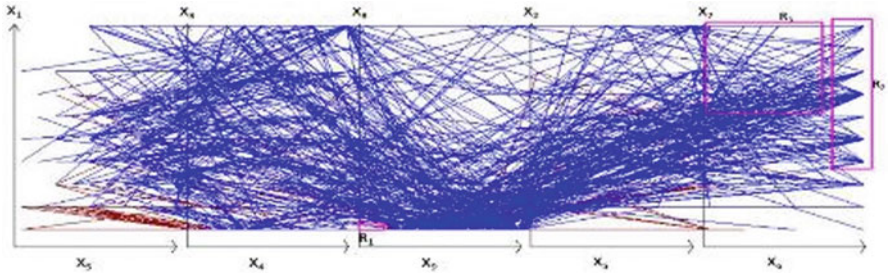


Fig. 14 SPC visualization of WBC data with areas dominated by blue class

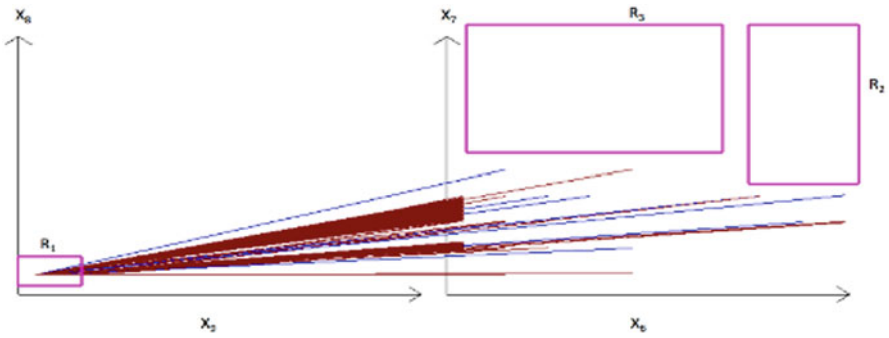


Fig. 15 WBC data in 4-D SPC as graphs in coordinates (X_9, X_8) and (X_6, X_7) that go through rectangle R_1 and not go to rectangles R_2 and R_3 in these coordinates

4.3 Case Study with GLC-L+CNN Algorithm

This case study uses the same WBC data as above. The occlusion was a major challenge there, which was dealt by FSP algorithm. Below for this, we use a combination of GLC-L algorithm, described in Sect. 4.1, and a Convolutional Neural Network (CNN) algorithm. The first step is converting non-image WBC data to images by GLC-L and the second one is discovering a classification model on these images by CNN. Each image represents a single WBC data case as a single polyline (graph) completely avoiding the occlusion. Figure 16 illustrates this design. It resulted in 97.22% accuracy on tenfold cross-validation [7].

4.4 Case Study with CPC-R+CNN Algorithm

This case study uses the same WBC data as the case studies above, but with CPC-R algorithm [21] for converting non-image data to images, and CNN algorithms for

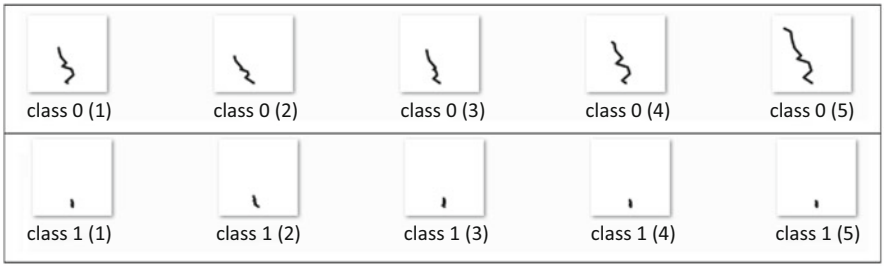


Fig. 16 WBC data samples visualized in GLC-L for CNN model

discovering the classification model in these images. Each image represents a single WBC data case, as a set of squares with a different level of intensities and colors.

The CPC-R algorithm is a modification of Collocated Paired Coordinates (CPC) algorithm [22]. The CPC algorithm first splits attributes of an n-D point $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ to consecutive pairs $(x_1, x_2), (x_3, x_4), \ldots (x_{n-1}, x_n)$. If n is an odd number, then the last attribute is repeated to get $n + 1$ attributes. Then all pairs are shown as 2-D points in the same 2-D Cartesian coordinates and connected by arrows to form a directed graph $\mathbf{x}^*$: $(x_1, x_2) \rightarrow (x_3, x_4) \rightarrow \ldots \rightarrow (x_{n-1}, x_n)$. This graph is equivalent to the n-D point $\mathbf{x}$ and it can be *fully* restored from the graph.

The CPC-R algorithm, instead of connecting pairs (x_1, x_2) by arrows, uses the grayscale intensity from black for (x_1, x_2) and very light gray for (x_{n-1}, x_n) for cells. Alternatively, intensity of a color is used. This order of intensities allows full restoration of the order of the pairs from the image. The size of the cells can be varied from a single pixel to dozens of pixels. For instance, if each attribute has 10 different values then a small image with 10×10 pixels can represent 10-D point by locating five gray scale pixels in this image. This visualization is lossless when values of all pairs (x_i, x_{i+1}) are different and do not repeat. An algorithm for treatment of colliding pairs is presented in [21].

Figure 17a shows the basic CPC-R image design and Fig. 17b shows a more complex design of images, where a colored CPC-R visualization of a case is superimposed with mean images of the two classes, which are put side by side, creating double images. The experiments with such images produce accuracy between 97.36% and 97.80% in tenfold cross-validation for different CNN architectures on benchmark datasets [21]. The advantage of CPC-R is in lossless visualization of n-D cases, and the ability to overlay them using heatmap with salient points discovered by the CNN model, for model explanation. See Sect. 6 for more details on heatmap explanations.

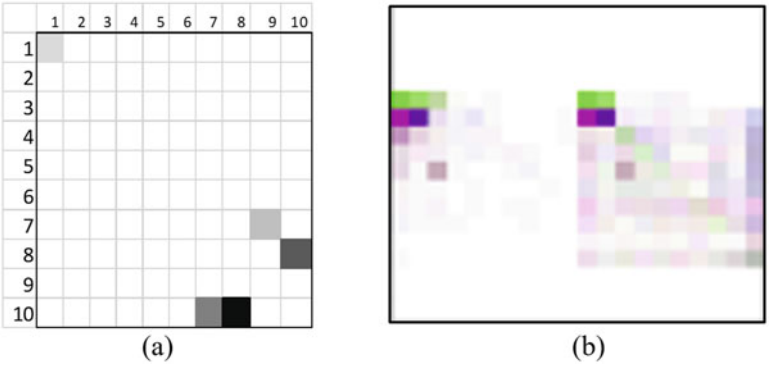


Fig. 17 CPC-R visualization of non-image 10-D points. (**a**) 10-D point (8, 10, 10, 8, 7, 10, 9, 7, 1, 1) in CPC-R. (**b**) Visualization in colored CPC-R of a case superimposed with mean images of two classes put side by side

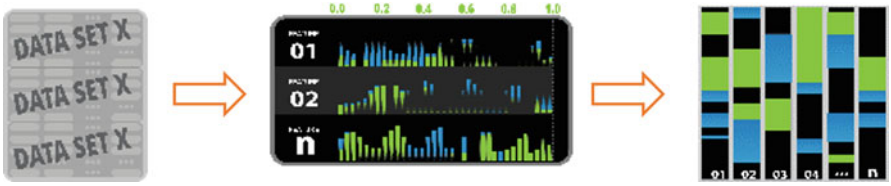


Fig. 18 DCP algorithm process

4.5 Case Study with RPPR Algorithm

This case study continues using the same WBC data as case studies above, but using Reverse Prediction Pattern Recognition (RPPR) algorithm [37]. This algorithm incorporates the prior DCP algorithm [20]. The goal is reaching both interpretability and high accuracy with of RPPR at the level or above it for non-interpretable algorithms on the same WBC data. The DCP algorithm starts with producing the dominance classifier structure $S = \langle\{V_i, h_{1i}, h_{2i}, \ldots, h_{ki},\}\rangle$, essentially a table containing intervals $\{V_i\}$ and the number of cases $h_{1i}, h_{2i}, \ldots, h_{ki}$, of each class on the training data within the respective interval on each predictor attribute X_i.

The next steps are combining dominance intervals in the voting methods, learning parameters of dominance intervals and voting methods for prediction, visualizing the dominance structure, and explaining the prediction. Figure 18 illustrates this process of construction of dominance intervals. The RPPR algorithm boosts DCP by discovering pair relations between attributes, then learning from said relations to override inaccurate DCP predictions. Figure 19 shows the result of the DCP algorithm on WBC data, with a benign case on the left and malignant case on the right, where each column represents an attribute, with colored intervals being the dominant intervals for red and blue classes, respectively.

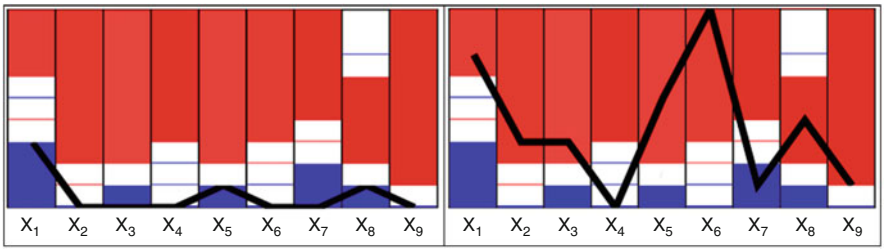


Fig. 19 Visualization of WBC data dominant intervals with a benign (on the left) and a malignant case (on the right)

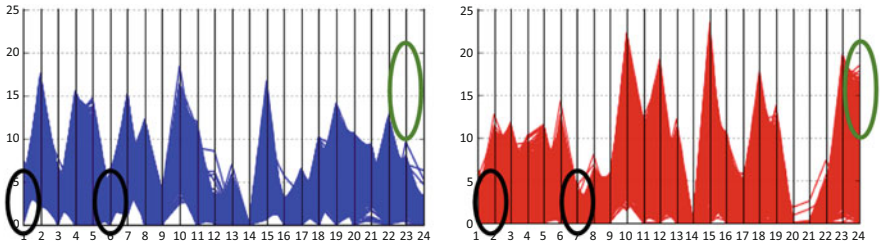
The steps of RPPR algorithm are: (1) *Encoding* elements of DCP algorithm as Boolean vectors, (2) *Finding* training cases *misclassified* by the DCP, (3) *Discovering* all the *unique* pairs for DCP False-Negative (FN) and DCP False-Positive (FP) *n*-D points on training data, (4) *Finding* FN and FP *n*-D points in the *validation/test*ing dataset with these unique pairs, and (5) *Reversing* prediction for these *n*-D points. Boosting DCP with RPPR allowed achieving accuracy over 99%, reaching the accuracy of non-interpretable algorithms and beyond using tenfold cross-validation on this WBC data. For more details and other experiments, see [37]. As far as we know, for this benchmark dataset, for the first time it is possible to have both accuracy and interpretability, rather than having to choose one at the expense of the other.

5 Scaling of Visual Discovery of ML Models

This section presents methods for scaling GLC-based methods of visual knowledge discovery and machine learning to large structured and unstructured data that is important for many application domains.

5.1 GLC and Embeddings to Cut Dimensions

The combining methods of Visual Knowledge Discovery (VKD) with Dimension Reduction (DR) methods expands applicability of VKD to large high-dimensional data. The approaches include combining General Line Coordinates (GLC) with different embeddings. Embedding [9] as mapping from discrete objects of very different nature, such as words, images, and graphs to vectors of real numbers have been very productive in many ML and DNN applications. However, usually new dimensions lack interpretability and special efforts are needed to interpret them. The combination consists of two steps. The first step is applying embedding to get



(a) (b)

Fig. 20 Comparing encoded digit 0 and digit 1 on the parallel coordinates using 24 dimensions found by the Autoencoder among 484 dimensions. Each vertical line is one of the 24 dimensions. (**a**) Digit "1". (**b**) Digit "0"

data of the lower dimension k from original n-D data and the second step is using lossless GLC-based algorithms, like used in cases studies above, on k-D data for ML model discovery. Below we illustrate this approach on classification of MNIST images of handwritten digits "0" and "1" [25].

At the *first step*, the Neural Network auto-encoder converts each 22×22 image (484-D point) to a vector of 24 features (24-D point). At the *second step*, these 24-D points are visualized in Parallel Line Coordinates. See Fig. 20 for digits "1" and "0". The *third step* is searching for discriminating features in these visualizations for the given digits. The direct observation of visualizations allows seeing the differences in these 24 coordinates that are shown by black and green circles. For instance, in Fig. 20, x_2 and x_7 are above the zero for all "1", but can be zero for "0". More specifically, $x_2 > 3$ and $x_7 > 2$ for all "1". Also, $x_{24} < 7$ for all "1", but it can be greater than 7 for "0". The *fourth step* is designing rules from discriminating features, directly from visualization without any computation, and explaining them. For instance, the observed properties of x_2, x_7 and x_{24} allow generating rules R_1–R_3:

$$R_1 : \text{if } x_2 < 3 \text{ then "0"}; \; R_2 : \text{if } x_2 < 2 \text{ then "0"}; \text{ if } x_{24} > 7 \text{ then "0"}.$$

These rules can be converted to a more conservative single rule:

$$R_4 : \text{if } (x_2 < 3) \,\&\, (x_2 < 2) \,\&\, (x_{24} > 7) \text{ then "0"}.$$

The *fifth step* is removing the cases, which satisfy these rules, and then searching for rules from the remaining cases, in the same visual way as in the above steps, or assisted by analytical ML methods. The *sixth* step is tracing these 24 features in the images of "0" and "1" to find their origin for interpreting them and to make the classification rules/models intelligible. See Sect. 6 such for methods. Many other DR methods, not only auto-encoders, can be used in this approach. For instance, the first k principal components of Principal Component Analysis (PCA) can be

used for DR as embedding, instead of the Auto-encoder. Similarly, it is not required to use the Parallel Line Coordinates, to visualize the reduced set of coordinates losslessly. Many other GLC- based methods can be used in this approach. The important advantage of this approach is the ability to *control the loss of original n-D information in the DR process*. For PCA, it can be done by selecting the first k principal components that cover, say, 90% of the total variance.

5.2 Dimension Reduction and Visual PCA Interpretation with GLC

While PCA is a most known and commonly used method for Dimension Reduction (DR) and n-D data visualization, it suffers from two major deficiencies: (1) difficulties to explain principal components and (2) loss of information when only the first two principal components are used to visualize n-D data.

The GLC-PCA algorithm presented below provides a solution for both challenges. It combines a visual explanation of each of the principal components using the GLC-Linear (GLC-L) visualization algorithm [22], and the one described above, which visualizes the n-D linear functions, in 2-D and Shifted Paired Coordinates [22], also described above, to visualize all the n principal components in 2-D, without the loss of information.

Figure 21 illustrates the GLC-PCA visualization algorithm, which allows representing all PCA principal components, without loss of information, using two types of General Line Coordinates (GLC) – Shifted Paired Coordinates (SPC) and GLC-L. In Fig. 21, the value of *explained variance (EV) ratio* for each principal component w_i is shown next to its name w_i. All principal components are ordered according to their EV value. The black polyline (graph) shows losslessly all 8 Principal Components of 8-D data, as a sequence of pairs in SPC: $(w_1, w_2) \rightarrow (w_3, w_4) \rightarrow (w_5, w_6) \rightarrow (w_7, w_8)$.

The red and blue polylines show, in GLC-L coordinates, how each Principal Component w_i is formed as a linear combination of the original attributes x_1–x_8 of the 8-D point x. The length of each red and blue segment shows the value of the respective original attribute x_j. The angle q_{ij} of the segment to the respective

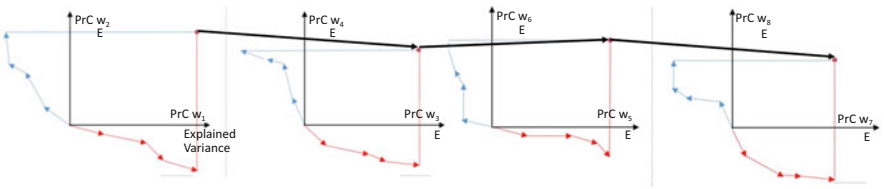


Fig. 21 All PCA principal components w_1–w_8, visualized in Shifted Paired Coordinates, and GLC-Linear coordinates, for an 8-D data point $x = (x_1, x_2, \ldots, x_8)$, without loss of information

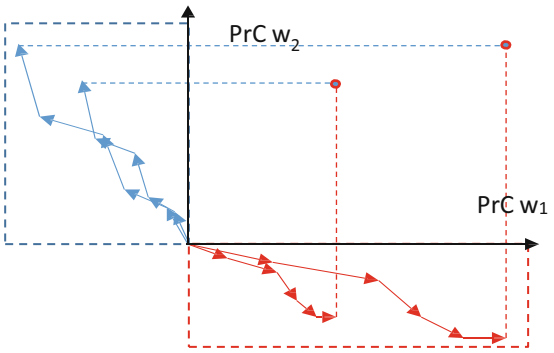


Fig. 22 Two 4-D points in the first two principal components in GLC-PCA

coordinate represents the contribution of the original attribute x_i to the principal component w_i. The original attributes x_j with smaller angles q_{ij} are most contributing to the principal component w_i. Figure 22 shows two 4-D points in the first two principal components in GLC-PCA.

The idea of GLC-PCA algorithm is visualizing not only the first two components y_1 and y_2 of each n-D point y, as it is done in visualization using PCA, but other y_i components of y too. In addition, the GLC-PCA algorithm shows visually how each y_i from y is formed as a linear combination of x_i of n-D point x. GLC-PCA allows the visualization of dimension reduction naturally. For instance, let the first 4 principal components out of 8 principal components, cover 85% of total variance and it is considered that these 85% are sufficient for the task then only these 4 will be visualized in GLC-PCA and used later for the model discovery.

5.3 Cutting the Number of Points and Dealing with Complex Data

Another aspect of visual knowledge discovery at scale is dealing with a large number of n-D points not only with large dimensions. A common approach to move from big data to smaller data is selecting a data subset, which captures properties of big data. The most popular idea here is data clustering with selecting representative cases from each cluster for further model discovery. Such clustering often is conducted by solving the optimization problems to maximize different similarity measures between n-D points. The review of these approaches can be found in [8]. This general idea is specified for different data types such as sequential and relational data, which is the area of active research [8, 34].

6 Visual Explanation of Analytical ML Models

Often visual ML model explanation relies on the same techniques as ML model visualization, while the goal of explanation is more specific. The ML model visualization typically does not produce an explanation itself but can create a basis for deriving the explanation. In this section, we focus on conceptual differences between the different methods of *visual explanations*.

6.1 Types of Explanations

While multiple definitions of terms "understanding", "interpreting", and "explaining" exist [1, 33] we favor one that requires describing the trained ML model in terms of domain ontology without using terms that are foreign to the domain where the ML task must be solved. Below we focus on externally and internally interpreted ML models.

The **externally interpreted models** are trained ML model explained in terms of interpretable input data and variables, but without interpreting the model structure. In [36] it is called as *post-hoc interpretability, functional understanding,* and *decision understanding*. This type of explanation is common for trained "black-box" Deep Neural Networks (DNN). Visualization of the space of input and output data of the trained ML model is an attractive approach for visual explanation of the trained ML model. It allows seeing the borders between classes produced by the model, e.g., see Figs. 2 and 5 in this chapter. While it is often challenging for the high-dimensional data, lossless or hybrid visualization methods make it feasible for many datasets.

The **internally interpreted models** are trained ML models explained in terms of interpreted elements of their structure not only inputs, e.g., trained decision trees, which are both internally and externally explainable ML models. Visualization of the model structure, in addition to visualization of the input data space and outputs of the model, is an attractive approach for this type of visual explanation.

Another aspect of the interpretability is its *coverage*: **entire model explaining** vs. **explaining individual model predictions**. Often individual predictions are explained using what we call *tabular pro-con explanation*. For a given new case **c** to be predicted, it shows pro cases (similar cases) and con cases (dissimilar cases) from classes. This is also an external explanation that does not go inside the model structure. This is a common explanation idea in *k*-nearest neighbors and case-based reasoning algorithms [33]. Respectively, there are two types of visual explanations based on (1) visualization of the entire model and entire input data space and (2) visualization of the given case and its nearest neighbors.

Two other explanation types are **explicit** and **implicit explanations**. Decision trees exemplify the former and heatmap-based explanations for deep neural network exemplify the latter. The next section is devoted to implicit explanations.

6.2 Heatmap Pixel-Based Implicit Explanations

Classification of an image A by deep neural network (DNN) can be **explained implicitly** by showing another image B, as an explanation. The idea is that if two inputs and predictions are similar then the explanation should be similar. If the similar property is not identified, then it is an *implicit explanation* that depends on the person who see the image B and needs to discover this similar property.

Often a similar image B contains **dominant (salient)** pixels of image A that are presented as an explanation of A. For example, DNN classifies the image A as a boat vs. a car and a truck [36]. The dominant pixels of image B represent the mast as a distinct feature of the boat relative to a car and a truck. Commonly such dominant pixels are colored according to their positive and negative scores for classes forming a **heatmap**.

While this is an acceptable *implicit explanation*, it assumes a human who recognizes a mast in these pixels. Thus, it is **incomplete explanation**, a more *complete explanation names a group of pixels*, e.g., mast. Next, this explanation is not applicable to another boat in the same image A, because that boat has no mast and requires its own explanation with another concept represented by its named group of pixels, e.g., people. Such conceptual explanations cannot be derived from the given trained DNN model, which recognized the boat, because it was not trained to recognize a mast or people explicitly. It is easy for a human to recognize such meaningful feature as a mast in a picture of the natural scene. In contrast, in medical imaging, if a radiologist cannot match DNN dominant pixels with the domain concepts such as tumor, these pixels will not serve as an explanation for the radiologist.

One of the methods to generate similar images B with dominant pixels is the *activation maximization*. The objective function of the optimization model for this maximization includes two components: one for maximization of activation and another one (regularizer) to ensure similarity of images [36]. The major challenges in activation maximization methods are (1) getting an image B with clear features, (2) selecting between competing images B (and dominant pixels) [38], and (3) interpreting dominant pixels as meaningful features.

Some alternative methods to find dominant (salient) pixels in DNN include: (i) *sensitivity analysis* by using partial derivatives of the activation function to find the max of its gradient, (ii) *Taylor decomposition* of the activation functions by using its first-order components to find scores for the pixels, (iii) *Layer-wise relevance propagation* (LPR) by mapping the activation value to the prior layers, and (iv) *blocking* (*occluding, perturbing*) sets of pixels and finding sets, which cause the largest change of activation value that can be accompanies by the class change of the image [36]. More approaches are reviewed and compared in [2, 5, 12, 40, 42, 47].

Two criteria are commonly used to find the sets of dominant (salient) pixels: (i) max contribution to output activation, and (ii) max of the changes in the output (class change). These criteria can contradict each other and, in general, *a difficult*

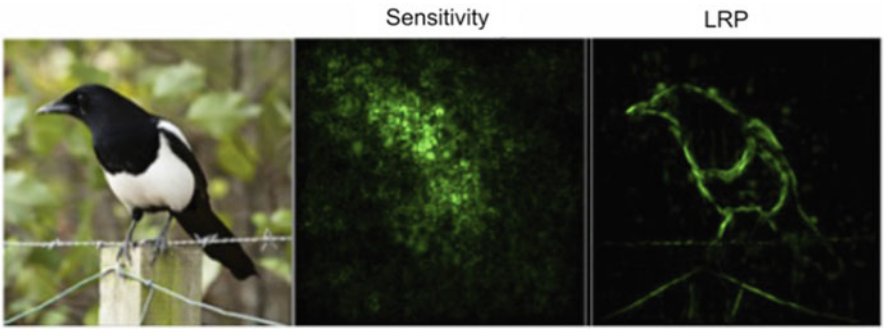


Fig. 23 Example of different heatmap methods [41]

multi-objective optimization problem must be formulated and solved. In practice, different linear combinations of them are used.

Guidotti et al. [13] review methods for explaining black box models based on Saliency Masks (maps), SM, visualization serving as meta-predictors that predicts the response of a black box to certain inputs. The sources of the SM are network activations. These approaches are known as Class Activation Mapping (CAM). They can visualize the linear combination of a late layer's activations, label-specific weights, or gradients by invoking back propagation and/or activation. Often these results are aesthetically pleasing providing heuristic explanations of image saliency [13]. For texts, it can provide a rationale (local "explanator) – a few words sufficient for the prediction of the original text.

Figure 23 illustrates the difference between the two heatmap methods. It shows that LRP explains the bird better, but this explanation is still implicit. The formal measure based on destroying some pixels and checking the change of the activation function captured this difference [41]. A human can understand that Fig. 23 shows a bird, due to the nose n, the tail t, and their relation $R(n, b, t)$ with the body b, which is between them. This *relation R* is not a part of the *heatmap implicit explanation*. A human derives it from the image, knowing that the body must be between the nose and the tail to be a bird. This is a common situation for all *heatmap implicit explanation* – they do not identify explicitly the *relations* between the features that they represent. This example shows the need to go beyond heatmaps.

7 Conclusion and Future Work

This chapter surveyed explainable machine learning approaches boosted by visual means. It includes motivation and comparison of analytical and visual ML methodologies, the input-based and structure-based types of methods of visualization of analytical ML. The chapter demonstrated that the approaches for discovering analytical ML models aided by visual methods are diverse and are growing. The

theoretical limits to preserve *n*-D distanced in lower dimensions are presented based on the Johnson-Lindenstrauss Lemma for point-to-point approaches. Further studies beyond the arbitrary points explored in this lemma are needed for the point-to-point approaches. In contrast, point-to-graph GLC approaches do not suffer from the limitations established in this lemma. Several real-world case studies, based on multiple GLC-based algorithms, had shown their advantages while multiple enhancements will be beneficial. The dimension reduction and clustering methods are outlined to support scalability and interpretability of the reviewed methods, including the visual PCA interpretation with GLC, and clustering for cutting the number of points.

Heatmap methods belong to the growing *sensitivity* [19] and the *attribution* [43] approaches that identify the elements of the input, which most affect the output. It is likely that heatmap implicit visual explanations will continue to be the focus of further studies, while this chapter has shown the need to go beyond heatmaps. Next, both the local and global explanation approaches need to be developed deeper. Below we list just a few future directions in interpretable visual knowledge discovery and ML among many others that can be developed. The first one is explaining ML models in human-relatable features in the Concept Activation Vectors (CAV) [19]. The next direction is combining neural networks with logic rules [15], which opens an opportunity to visual ML models, beyond the pixels and heatmaps, in line with the association rule visualization shown in Sect. 2.1. The GLC-based methods can contribute significantly to developing future interpretable ML models, due to their advantages such as lossless (reversible) visualization of *n*-D data, as graphs, and the interpretability of GLC-based predictive ML models.

References

1. Ahmad, M., Eckert, C, Teredesai, A., McKelvey, G. Interpretable Machine Learning in Healthcare, IEEE Intelligent Informatics Bulletin August 2018 Vol. 19, No. 1, 1–7.
2. Ancona M., Ceolini E., Oztireli A., Gross M., A unified view of gradient-based attribution methods for deep neural networks, *CoRR*, vol. abs/1711.06104, 2017. http://arxiv.org/abs/1711.06104.
3. Raschka S., MLxtend: Plotting Decision Regions, Journal of Open Source Software, 2018, https://doi.org/10.21105/joss.00638, https://rasbt.github.io/mlxtend/user_guide/plotting/plot_decision_regions/.
4. Cabrera A., Epperson WS., Hohman F., Kahng M., Morgenstern J., Chau D., FairVis: Visual Analytics for Discovering Intersectional Bias in Machine Learning, 2019, arXiv:1904.05419.
5. Choo J, Liu S. Visual analytics for explainable deep learning. IEEE computer graphics and applications. 2018 July 3;38(4):84–92.
6. Dasgupta, S.; Gupta, A., An elementary proof of a theorem of Johnson and Lindenstrauss, Random Structures & Algorithms, 22 (1): 60–65, 2003.
7. Dovhalets D., Kovalerchuk B., Vajda S., Andonie R., Deep Learning of 2-D Images Representing n-D Data in General Line Coordinates, Intern, Symp. on Affective Science and Engineering, pp. 1–6, 2018, https://doi.org/10.5057/isase.2018-C000025.
8. Elhamifar E., Recent Advances in Visual Data Summarization, CVPR 2019 Tutorial, https://rpand002.github.io/cvpr19_sumt.html.

9. Embeddings, Tensorflow guide, 2019, https://www.tensorflow.org/guide/embedding.
10. Facets visualizations for ML datasets. 2017. https://pair-code.github.io/what-if-tool/, 2018.
11. Fu C, Zhang Y, Cai D, Ren X. AtSNE: Efficient and Robust Visualization on GPU through Hierarchical Optimization. In: Proc. 25th ACM SIGKDD, 2019, 176–186, ACM.
12. Gilpin LH, Bau D, Yuan BZ, Bajwa A, Specter M, Kagal L. Explaining explanations: An overview of interpretability of machine learning. In: 2018 IEEE 5th Intern. Conf. on data science and advanced analytics (DSAA) 2018, 80–89, IEEE.
13. Guidotti R., Monreale A., Turini F., Pedreschi D., Giannotti F., A survey of methods for explaining black box models," *arXiv preprint arXiv:1802.01933*, 2018.
14. Hohman F, Kahng M., Pienta R., Chau D., Visual analytics in deep learning: An interrogative survey for the next frontiers. IEEE Vis. and Comp. Graphics, 25(8):2674–2693, 2019.
15. Hu Z, Ma X, Liu Z, Hovy E, Xing E. Harnessing deep neural networks with logic rules. arXiv preprint arXiv:1603.06318. 2016 Mar 21.
16. Inselberg, A., Parallel Coordinates, Springer, 2009.
17. Johnson, W., Lindenstrauss, J. Extensions of Lipschitz mappings into a Hilbert space. In Beals, et al. (eds) Conference in modern analysis and probability 1982. Contemporary Mathematics. 26. Providence, RI: AMS, 189–206, 1986.
18. Kahng M., Andrews P., Kalro A., Chau D. ActiVis: Visual exploration of industry-scale deep neural network models. IEEE Trans. on Vis. and Comp. Graphics, 24(1):88–97, 2018.
19. Kim B., Introduction to Interpretable Machine Learning Tutorial, CVPR 2018, http://deeplearning.csail.mit.edu/slide_cvpr2018/been_cvpr18tutorial.pdf.
20. Kovalerchuk B., Neuhaus, N. Toward Efficient Automation of Interpretable Machine Learning. In: Intern. Conf. on Big Data, 4933–4940, 978-1-5386-5035-6/18, 2018 IEEE.
21. Kovalerchuk B, Kalla DC, Agarwal B.: Deep Learning Image Recognition for Non-images. In: Integrating Artificial Intelligence and Visualization for Visual Knowledge Discovery 2022, 63–100, Springer, Cham.
22. Kovalerchuk, B. Visual Knowledge Discovery and Machine learning, 2018, Springer.
23. Kovalerchuk B, Grishin V. Reversible Data Visualization to Support Machine Learning. In: Intern. Conf. on Human Interface and the Management of Information 2018, 45–59. Springer.
24. Kovalerchuk B., Gharawi A., Decreasing Occlusion and Increasing Explanation in Interactive Visual Knowledge Discovery, In: Human Interface and the Management of Information. Interaction, Visualization, and Analytics, 505–526, 2018, Springer.
25. Kovalerchuk, B., Dovhalets, D., Constructing Interactive Visual Classification, Clustering and Dimension Reduction Models for n-D Data, Informatics, 4(23), 2017, http://www.mdpi.com/2227-9709/4/3/23.
26. Kovalerchuk, B. Visualization of multidimensional data with collocated paired coordinates and general line coordinates. *Proc. SPIE* 2014, *9017*, https://doi.org/10.1117/12.2042427.
27. Kovalerchuk B. Quest for rigorous intelligent tutoring systems under uncertainty: Computing with Words and Images. In: IFSA/NAFIPS, 2013, 685–690, IEEE.
28. Kovalerchuk, B.; Delizy, F.; Riggs, L.; E. Vityaev, Visual Data Mining and Discovery with Binarized Vectors, in: Data Mining: Foundations and Intelligent Paradigms, 24: 135–156, 2012, Springer.
29. Kovalerchuk, B., Balinsky, A., Visual Data Mining and Discovery in Multivariate Data using Monotone n-D Structure, In: Knowledge Processing and Data Analysis, Wolff, K.E et al., (Eds.), 297–313. Springer.
30. Kovalerchuk B., Schwing J., (Eds). Visual and spatial analysis: advances in data mining, reasoning, and problem solving. 2005, Springer.
31. Kovalerchuk B., Vityaev, E., Data Mining in Finance: Advances in Relational and Hybrid Methods, 2000, Kluwer/Springer.
32. Krause J., Perer A., Bertini E., A user study on the effect of aggregating explanations for interpreting machine-learning models. ACM KDD Workshop on Interactive Data Exploration and Analytics, 2018.
33. Lipton Z. The Mythos of Model Interpretability, Commun. of the ACM, 2018, 61, 36–43.

34. Liu S, Jampani V., Wang X, Batra D., Gupta A, Kautz J., Yang M-H, CVPR 2019 Tutorial on Learning Representations via Graph-structured Networks https://xiaolonw.github.io/graphnn/.
35. Maszczyk A., W. Duch, Support Vector Machines for visualization and dimensionality reduction, LNCS, Vol. 5163, 346–356, 2008, Springer.
36. Montavon G, Samek W, Müller KR. Methods for interpreting and understanding deep neural networks. Digital Signal Processing. 2018 Feb 1;73:1–5.
37. Neuhaus, N., Kovalerchuk, B., Interpretable Machine Learning with Boosting by Boolean Algorithm, Joint 2019 Intern. Conf. ICIEV/IVPR, Spokane, WA, 2019, 307–311. IEEE.
38. Nguyen A., Yosinski, J. Clune J, Multifaceted feature visualization: uncovering the different types of features learned by each neuron in deep neural networks, CoRR, arXiv:1602.03616, 2016.
39. Patel K., Bancroft N., Drucker S., Fogarty J., Ko A., Landay J., Gestalt: integrated support for implementation and analysis in machine learning. In Proceedings of the 23nd annual ACM symposium on User interface software and technology, 37–46. ACM, 2010.
40. Ribeiro M., Singh S., Guestrin C., Why Should I Trust You?: Explaining the Predictions of Any Classifier, Proc. the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 1135–1144, 2016.
41. Samek W., Montavon G., Müller K-R., Interpreting and Explaining Deep Models in Computer Vision CVPR 2018 Tutorial, http://interpretable-ml.org/cvpr2018tutorial/.
42. Samek W, Wiegand T, Müller KR. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. arXiv preprint arXiv:1708.08296. 2017.
43. Sundararajan M, Taly A, Yan Q. Axiomatic attribution for deep networks. In: Proceedings of the 34th International Conference on Machine Learning-Vol. 70 2017, 3319–3328.
44. van der Maaten L. Dos and Dont's of using t-SNE to Understand Vision Models, CVPR 2018 Tutorial on Interpretable Machine Learning for Computer Vision, http://deeplearning.csail.mit.edu/slide_cvpr2018/laurens_cvpr18tutorial.pdf.
45. Wongsuphasawat K, Smilkov D, et al., Visualizing dataflow graphs of deep learning models in tensorflow. IEEE trans. on visualization and computer graphics. 2018: 24(1):1–2.
46. Zhang C. et al. Association rule based approach to reducing visual clutter in parallel sets, Visual Informatics 3, 2019, 48–57.
47. Zhang Q.-S., Zhu S.-C., Visual interpretability for deep learning: a survey, Frontiers of Information Technology & Electronic Engineering, vol. 19, no. 1, 27–39, 2018.
48. Dua, D. and Graff, C. Machine Learning Repository: Wisconsin Breast Cancer Dataset, Irvine, CA: University of California, https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original), 2019.
49. Kovalerchuk, B., Ahmad, M.A., Teredesai A., Survey of Explainable Machine Learning with Visual and Granular Methods beyond Quasi-explanations, In: Interpretable Artificial Intelligence: A Perspective of Granular Computing (Eds. W. Pedrycz, S.M.Chen), Springer, 2021, 217–267.