SN Computer Science (2023) 4:226
https://doi.org/10.1007/s42979-022-01662-4

ORIGINAL RESEARCH

Stability Analysis of Supervised Decision Boundary Maps

Artur A. A. M. Oliveira[1] · **Mateus Espadoto**[1] · **Roberto Hirata Jr.**[1] · **Alexandru C. Telea**[2]

Received: 20 June 2022 / Accepted: 30 December 2022 / Published online: 21 February 2023

Abstract

Understanding how a machine learning classifier works is an important task in machine learning engineering. However, doing this is for any classifier in general difficult. We propose to leverage visualization methods for this task. For this, we extend a recent technique called Decision Boundary Map (DBM) which graphically depicts how a classifier partitions its input data space into decision zones separated by decision boundaries. We use a supervised, GPU-accelerated technique that computes bidirectional mappings between the data and projection spaces to solve several shortcomings of DBM, such as accuracy and speed. We present several experiments that show that SDBM generates results which are easier to interpret, far less prone to noise, and compute significantly faster than DBM, while maintaining the genericity and ease of use of DBM for any type of single-output classifier. We also show, in addition to earlier work, that SDBM is stable with respect to various types and amounts of changes of the training set used to construct the visualized classifiers. This property was, to our knowledge, not investigated for any comparable method for visualizing classifier decision maps, and is essential for the deployment of such visualization methods in analyzing real-world classification models.

Keywords Machine learning · Dimensionality reduction · Dense maps

Introduction

As machine learning (ML) techniques develop and address increasingly many application domains, so does their complexity and difficulty of understanding their working. This poses problems for their adoption in contexts where transparency and accountability of inference is required [1]. Such issues are especially important for deep learning (DL) models which handle very high dimensional datasets and operate essentially as black boxes having millions of hidden parameters [2].

To explain ML *classifier* models, several approaches have been proposed, using variable importance [3], locally interpretable models [1], and surrogate models [4]. Visualization techniques complement such approaches by mapping the model's predictions or internal states to various visual representations [5, 6]. A recent survey of visualization techniques for the explanation of DL models was proposed by Garcia et al. [2].

In the above family, *Decision Boundary Maps* (DBMs) [7] are a particular visualization technique for ML classifiers. Given a multidimensional projection [8] that shows how a classifier handles some input dataset, DBMs literally fill the whitespace with classification results (colormapped labels)

Artur A. A. M. Oliveira, Mateus Espadoto, Roberto Hirata Jr., Alexandru C. Telea have contributed equally to this work.

This article is part of the topical collection "Advances on Computer Vision, Imaging and Computer Graphics Theory and Applications" guest edited by Kadi Bouatouch, Augusto Sousa, Mounia Ziat and Helen Purchase.

✉ Mateus Espadoto
mespadot@ime.usp.br

Artur A. A. M. Oliveira
arturao@ime.usp.br

Roberto Hirata Jr.
hirata@ime.usp.br

Alexandru C. Telea
a.c.telea@uu.nl

[1] Institute of Mathematics and Statistics, University of São Paulo, Rua do Matão, 1010, São Paulo 05508-090, Brazil

[2] Department of Information and Computing Sciences, Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands

for data points that would project at those locations. The result is a dense image that shows how the visual projection space is partitioned in per-class decision zones. Decision zone boundaries—where two or more label colors adjoin—depict locations where the classifier changes its output. DBMs offer a simple to interpret and visually scalable way to depict the working of any classification model.

In recent work [9], we proposed Supervised Decision Boundary Maps (SDBM), which extends the DBM method [7] to address several of the latter method's shortcomings, as follows:

Quality (C1) SDBM produces decision maps that create a clearer, and far less noise-prone, visual separation of a higher number of decision zones from real-world, complex, datasets, than DBM;

Scalability (C2) SDBM has a complexity linear in the number of samples and dimensions and runs on the GPU; this allows creating megapixel maps in a few seconds on commodity hardware in contrast to the minutes needed by DBM;

Ease of use (C3) SDBM produces good results with minimal or no parameter tuning;

Genericity (C4) Like DBM, SDBM can construct decision boundaries for *any* single-value classifier.

Despite these attractive points, the interpretation of the maps produced by both DBM and SDBM relies fundamentally on a *stability* assumption: Indeed, users examine such maps to determine, for instance, the size and adjacency of decision zones, to e.g. decide whether a classifier is well trained and/or where to add more training samples to improve it [7]. If the maps—and in particular, the borders where decision zones meet—are unstable to small changes in the training data, their interpretation can easily go wrong. Such effects were already found in earlier work on DBMs [10, 11] in terms of noise-like 'islands' that appear in DBMs constructed for complex classifier models. SDBM successfully removes such small-scale artifacts. Yet, it is still unknown how stable, thus trustworthy, are the *large scale* patterns (decision zones, decision boundaries) that SDBM creates. If these patterns are not stable, then the overall interpretation of the SDBM maps is of limited value.

In this work, we address the above open question by performing a multi-faceted stability analysis on SDBM. For this, we train three classifiers on several perturbed versions of three real-world datasets, and compute and visualize the resulting decision maps as well as their changes. We also propose two novel visualizations to summarize the stability of SDBMs in presence of several training-set changes. Our analysis shows that SDBM has an additional desirable property, namely

Stability (C5) SDBM constructs decision maps which are stable with respect to change. The amount of visual change—in terms of positions and sizes of the decision zones—is following the amount of change present in the input data. In particular, small data changes only yield small visual changes which do not adversely affect the interpretation of the computed decision maps.

We structure this paper as follows: "Background" section discusses related work on visual explanation of classification models. "Methods" section details the SDBM method. "Results" section presents results that support our contributions C1-C4 outlined above, as well as our new stability analysis and novel visualizations designed to explore it (C5). "Discussion" section discusses SDBM. Finally, "Conclusion" section concludes the paper.

Background

We next introduce the notations used in further this paper. Let $\mathbf{x} = (x^1, \ldots, x^n)$, $x^i \in \mathbb{R}, 1 \leq i \leq n$ be an n-dimensional (nD) real-valued sample. Let $D = \{\mathbf{x}_j\}$, $1 \leq j \leq N$ be a dataset of N such samples, e.g., a table with N rows (samples) and n columns (dimensions). As we focus on classification models, we assume D is labeled by K label values in $C = \{c_k\}$, $1 \leq k \leq K$. Specifically, let $\mathbf{y} = \{y_j | y_j \in C\}$, $1 \leq j \leq N$ be the labels of D where sample $\mathbf{x}_j$ has label y_j. A classification model is a function

$$f : \mathbb{R}^n \rightarrow C \tag{1}$$

that maps between data samples and label values. The model f is typically obtained using a training algorithm over the dataset D, such as Logistic Regression [12], SVM [13], Random Forests [14], or Neural Networks, to name a few.

A Dimensionality Reduction (DR), or projection, technique is a function

$$P : \mathbb{R}^n \rightarrow \mathbb{R}^q \tag{2}$$

that maps a sample $\mathbf{x} \in \mathbb{R}^n$ to a point $\mathbf{p} = P(\mathbf{x})$, $\mathbf{p} \in \mathbb{R}^q$ (typically, $q = 2$). Projecting a dataset D yields a qD scatterplot denoted next as $P(D)$. The inverse of P, denoted $P^{-1}(\mathbf{p})$, maps, or backprojects, a qD point $\mathbf{p}$ to the high-dimensional space $\mathbb{R}^n$.

Decision Boundary Maps A Decision Boundary Map (DBM) is an image that depicts how a given model f partitions the projection space $\mathbb{R}^2$ into decision zones. A *decision zone* is a set of points $\mathbf{p} \in \mathbb{R}^2$ for which $f(P^{-1}(\mathbf{p})) = c_k$, i.e., back-project to data points classified by f to the same label c_k, and is colored by the label c_k. Decision zones are separated by *decision boundaries*, which are pixels $\mathbf{p}$ whose labels (colors) differ from those of at least one 8-neighbor pixel. A DBM shows, among other things, how f partitions the high-dimensional space into decision zones, how large these zones are, how they are adjacent to each other, and how smooth the decision boundaries between classes

are [10]. This gives insights on whether the model f has overfitted the training data, and how well separated the data is, i.e., how difficult is the classification task. DBMs are a step forward from the key observation of Rauber et al. [5] who showed how projections aid deciding whether a high-dimensional dataset is easily classifiable or not. Simply put, DBMs support the same tasks but provide more information by 'filling in' the white gaps between the points of a 2D scatterplot $P(D)$ by extrapolating the classifier f.

The DBM technique of Rodrigues et al. [10] relies heavily on direct and inverse projections. The direct mapping is used to create a 2D scatterplot $P(D)$ from a dataset D. The inverse mapping P^{-1} creates synthetic data points from all pixels $\mathbf{p}$ in the 2D bounding box of $P(D)$. These data points $P^{-1}(\mathbf{p})$ are then classified by f, and colored by the assigned labels $f(P^{-1}(\mathbf{p}))$. DBM has two main issues: (1) The inverse projection technique P^{-1} used, iLAMP [15], scales poorly to the hundreds of thousands of pixels a DBM has. This was addressed in [10] using low-resolution DBMs. To increase accuracy, several points were sampled over a pixel in these maps and the pixel class (color) was set by majority-voting on the labels assigned by f to the back-projections of these points. This scheme however creates artifacts visible as highly jagged decision boundaries. (2) Since DBM uses an unsupervised projection P, outliers in a dataset D can generate spurious 'islands' of pixels having a different label (color) than their neighbors, thus appearing as spurious decision zones that confuse the user.

Improved Decision Boundary Maps Several improvements were proposed to address the above-mentioned issues of DBMs. Rodrigues et al. [11] examined the DBMs generated for four classifiers and using 28 projection techniques P and found that suitably parameterized t-SNE [16] and UMAP [17] projections limit the spurious islands in the decision maps. Next, the same authors proposed a simple filtering technique to eliminate poorly projected points from $P(D)$ and use only the remaining ones to construct the inverse mapping P^{-1} [7]. They also increased the accuracy and speed of computing the DBMs by using a deep learning technique [18] to construct P^{-1} from a given direct mapping $P(D)$. Finally, they proposed ways to visualize the distance-to-closest-boundary of all points inside a decision zone to highlight areas prone to misclassification.

Related Dense Maps Besides DBMs used to visualize the working of a classifier model, dense maps have been used to analyze high-dimensional data in other contexts. Closer to our application, OptMap [19] uses dense maps to explore the optimization process of generic regressors $r : \mathbb{R}^2 \rightarrow \mathbb{R}$. StrategyAtlas [20] projects high-dimensional datasets to 2D using UMAP and creates dense maps showing the value of a user-selected dimension over the projection space using Shepard interpolation around the projected points [21]. Similar interpolation is used to construct dense maps showing errors at the projected points [22, 23] or dimensions that explain how neighbor projection points are related [24]. Among these techniques, only OptMap actually uses an inverse projection to map from the image space to the data space—all other techniques only interpolate data values at the sample points in the image space. As explained in [7], such image-space interpolation can be misleading since distances in the projection space usually do not directly reflect distances in the data space.

Dimensionality Reduction in DBMs As explained above, DBMs rely heavily on Dimensionality Reduction (DR) or projection techniques. For the DBM context, such a technique should ideally

1. Work generically for any type of high-dimensional dataset D;
2. Be computationally fast, ideally linear in the number of samples and dimensions of D;
3. Provide both the direct (P) and inverse (P^{-1}) projection;
4. Be simple to parameterize (for an easy usage in practice);
5. Provide a high-accuracy projection;
6. Be stable and have out-of-sample (OOS) capability.

The first four requirements above are, we believe, evident. Requirement 5 (accuracy) means that $P(D)$ can successfully preserve the structure of the data (clusters, neighbors, outliers) present in D. If this is not the case, any (visual) inference done on $P(D)$—such as reasoning about the sizes, shapes, and relative positions of decision zones—may be misleading. Accuracy is typically gauged by measuring several so-called projection quality metrics [22, 25–27]. Such quality metrics have been used to filter poorly projected points to improve the DBM quality [7], as outlined earlier.

Requirement 6 also deserves separate explanation: A projection technique P is called *stable* if small changes in its input dataset D cause only small changes in the created scatterplot $P(D)$. As a special case, a projection is called *deterministic* if it outputs the same $P(D)$ for the same input D. Stable and deterministic projection techniques are preferred for many visualization applications as they simplify the user's task—one e.g. can exactly reproduce the output of a projection for the same dataset D; and small-scale noise or inaccuracies in the input D do not massively affect the obtained visualization. Separately, a projection P is called to have *out of sample* (OOS) ability if it can project new, unseen, samples along those earlier provided in some dataset D, without modifying the projection $P(D)$. OOS is desirable when one needs to project a sequence of related datasets [8, 28, 29]. OOS projections are typically also stable, though the converse is not necessarily true. For DBM construction, we ideally want both P and P^{-1} to be stable and deterministic. If not, one could obtain radically different decision maps from

the *same* classifier, e.g. when trained with slightly different data D. In turn, this would make the visual interpretation of the respective classifier via the DBMs very challenging if not hardly possible.

Measuring projection *stability* is a relatively new and little explored topic, as most quantitative studies on DR focused so far on ensuring high projection quality (see the survey in [28]). Key difficulties for stability measurement are defining the 'allowable' change in the data D and in the projection $P(D)$. Vernier et al. [29] present, to our knowledge, the first attempt to quantify stability for dynamic projection techniques by measuring the correlation of changes in the 2D distances between points in $P(D)$ and their nD distances in D. Data changes are implicitly given by the application domain as D is a time-dependent dataset. Bredius et al. [30] gauge the stability of a specific projection method [31] by explicitly synthesizing noise-like changes of D and depicting the changes in $P(D)$. However, they perform no quantitative stability measurements. Espadoto et al. [32] use similar noise-like changes to train a projection method to behave less sensitively (thus, be more stable) in their presence. However, they do not explicitly measure or reason about projection stability. For inverse projections P^{-1} or DBMs, we are not aware of any stability study. Our work here is, to our knowledge, the first study that explicitly measures the stability of a DBM pipeline involving both direct and inverse projections.

Many DR techniques have been proposed over the years, as reviewed in various surveys [8, 28, 33–39]. Below we describe a few representative ones from DBM computation perspective and outline how these fare with respect to requirements 1–5 mentioned above.

Principal Component Analysis [40] (PCA) is one of the most popular DR techniques for many decades, and complies well with all requirements except 5 (accuracy), especially for data of high intrinsic dimensionality. PCA was used to compute both P and P^{-1} by the OptMap visualization method for regressor analysis [19]. However, the authors noted that higher quality results could be obtained using a more accurate projection technique.

The Manifold Learning family of methods contains techniques such as MDS [41], Isomap [42], and LLE [43], which aim to capture nonlinear data structure by mapping to 2D the high-dimensional manifold on which data is located. These methods generally yield better results than PCA (5), but do not scale well computationally (2), and also yield poor results when the intrinsic data dimensionality is higher than two. Also, many such methods require careful parameter tuning (4) to obtain suitable results.

The SNE (Stochastic Neighborhood Embedding) family of methods, of which the most popular member is t-SNE [16], are best known for the high quality of the projections they produce (5). Yet, they can be hard to tune [44], and typically have no OOS capability and/or stability (6). Parametric t-SNE [45] adds OOS and stability at the expense of a significantly slower and more complex implementation. Several refinements of t-SNE improve speed (2), such as tree-accelerated t-SNE [46], hierarchical SNE [47], and approximated t-SNE [48], and various GPU accelerations of t-SNE [49, 50]. Uniform Manifold Approximation and Projection (UMAP) [17], while not part of the SNE family, generates projections with comparable quality to t-SNE (5), but much faster (2), and with OOS capability (6).

All above projection techniques work in an *unsupervised* way—they use only distance information between points in D to compute $P(D)$. Recently, Espadoto et al. [31] proposed Neural Network Projection (NNP) to learn the projection $P(D)$, computed by any user-selected technique P, from a small subset $D' \subset D$, using a deep learning regressor. While slightly less accurate than the original P (5), NNP is computationally linear in the size and dimensionality of D (2), has OOS ability (5) and it simple to implement and parameter-free (4). A recent study [30] showed that NNP is very stable to a wide range of perturbations of its input data D. NNP was further refined [51] to use neighborhood information between samples in D and further increase the projection accuracy (5). A related idea to NNP was used by NNInv [18] to learn the inverse mapping P^{-1}. NNP and NNInv were next extended by Self-Supervised Network Projection (SSNP) [52], which can be used either in a *self-supervised* fashion, by computing pseudo-labels by a generic clustering algorithm on D, or in a *supervised* fashion (similar to NNP), using ground-truth labels $\mathbf{y}$ coming with D. We choose SSNP to create our proposed SDBM as it complies well with our earlier stated six requirements for a projection method in the DBM context:

1. SSNP works generically for any high-dimensional dataset;
2. SSNP is GPU-accelerated, which makes it one to two magnitude orders faster than DBM (see next "Computational Scalability" section);
3. SSNP provides both the direct and inverse mappings (P and P^{-1}) needed by the DBM method;
4. SSNP is parameter-free (after its training phase has completed);
5. SSNP provides good cluster separation by partitioning the data space D as a classifier would do, which is closely related to the original goal of DBM;
6. SSNP is parametric, thus has OOS and, as we show next in "Stability Analysis" section, leads to a stable DBM computation with respect of changes in the input dataset D.

Methods

We next describe our proposed SDBM technique and how it is different from its predecessor, DBM. Our technique has five steps as illustrated by the pipeline in Fig. 1. Below, we detail all these steps.

0. Input data SDBM needs only two inputs—a high-dimensional dataset D and its label vector $\mathbf{y}$. The definitions of these are given in "Background" section. As stated earlier, no restrictions exist on the data dimensionality n, data nature, or number of labels K used in $\mathbf{y}$. Simply put, any labeled dataset $(D, \mathbf{y})$ that can be used to build a classification model for some problem is acceptable as input for SDBM. Therefore, SDBM is applicable to visualize the decision boundaries and zones of any classifier.

1. Create mappings We train SSNP to create the direct and inverse projections P and P^{-1} based on D and $\mathbf{y}$. This step is fundamentally different from DBM. In detail: DBM requires the user to supply a projection technique P to map D to a 2D scatterplot $P(D)$. Next, DBM uses $P(D)$ to learn the inverse mapping, or inverse projection P^{-1}. For this, DBM uses various inverse projection techniques such as NNInv [18] or iLAMP [15] (see also "Background" section). The problem with this is that, depending on the direct projection P chosen by the user, these inverse projection techniques may have difficulties in computing an accurate inverse projection P^{-1}. That is, for several points $\mathbf{x}$ in the input domain of the classifier, $P^{-1}(P(\mathbf{x})) \neq \mathbf{x}$, i.e., P^{-1} is not the exact inverse of P. In practice, this leads to jagged decision boundaries and noise-like small islands scattered all over the dense maps created by DBM (see examples in Fig. 5 later on). SDBM does not have this problem as it uses the SSNP method to *jointly* compute both P and P^{-1}, as mentioned in "Background" section. As shown by our results in "Results" section, this joint computation of P and P^{-1} used by SDBM significantly reduces the above-mentioned artifacts in the decision maps.

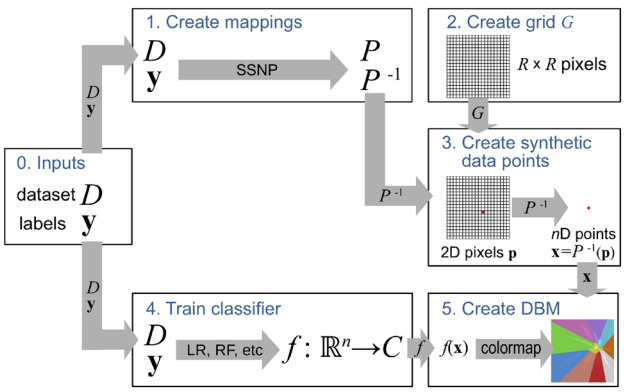

Fig. 1 SDBM pipeline (see "Methods" section)

2. Create 2D grid: Create an image $G \subset \mathbb{R}^2$ with a resolution of R pixels, where R is chosen by the user. Higher R values capture more details in the decision maps but take longer to compute—more precisely, the computation time is linear in the number of pixels of the map. This is different from DBM. In detail, DBM uses the full resolution of G to compute the direct projection $P(D)$, but then evaluates P^{-1} on a subsampled version of G of a lower resolution than R to reduce computation time (see "Background" section). In contrast, SDBM uses the full user-specified resolution R to compute both P and P^{-1} (for all experiments in this paper, we set this to $R = 300^2$ pixels). SDBM does not need to use subsampling since its underlying direct-and-inverse projection technique, SSNP, is fast enough to treat the full resolution specified by the user.

3. Create synthetic data points: Use the trained P^{-1} (delivered by SSNP in step 1) to map each pixel $\mathbf{p} \in G$ to a high-dimensional data point $\mathbf{x} \in \mathbb{R}^n$. This is similar to DBM, except the use of a dense pixel grid and the jointly-trained P and P^{-1} mappings delivered by SSNP (see step 1).

4. Train classifier: Train the classifier f to be visualized using the dataset D and its labels $\mathbf{y}$, as in an usual machine learning setting. This step is identical to DBM. Any single-class-output classifier $f : \mathbb{R}^n \rightarrow C$ can be used generically, e.g., Logistic Regression (LR), Random Forests (RF), Support Vector Machines (SVM), or neural networks. Moreover, no restrictions are placed on the design or architecture of f. Also, note that the classifier training occurs *after* the construction of the mappings P and P^{-1} in step 2. That is, these mappings have no knowledge of the class labels. Hence, we can reuse these mappings computed in step 2 to next construct decision maps to visualize any classifier to be trained on the given inputs $(D, \mathbf{y})$. Simply put, once step 2 is executed, we can next quickly construct decision maps to compare how several classifiers perform on a given $(D, \mathbf{y})$. We illustrate this further in our results ("Results" section).

5. Create DBM: Color all pixels $\mathbf{p} \in G$ by the values of $f(P^{-1}(\mathbf{p}))$, i.e., the inferred classes of their corresponding (synthetic) data points, using a categorical color map. In this paper we use the 'tab20' color map [53]. This is the same as DBM.

5b. Encode classifier confidence (optional part of step 5): For classifiers f that provide the probability of a sample $\mathbf{x}$ belonging to a class c_k, we encode this probability in the brightness of the pixel $\mathbf{p}$ that back-projects to $\mathbf{x}$. The lower the confidence of the classifier is, the darker the pixel appears in the map. This informs the user of the confidence of the decision zone in that area—dark areas in the map, typically close to decision boundaries, indicate regions in

the data space where the classifier is less confident. This is the same as DBM.

5c. Draw scatterplot (optional part of step 5): If desired, one can visualize the projection $P(D)$ of the training set D by drawing it as a scatterplot atop of the DBM. Note that this is the only place where SDBM uses the projection P. The added value of showing this scatterplot is showing to users where, in the decision map, are the actual data points from which the decision map was extrapolated. If users do not wish to see this scatterplot, we only use the inverse projection P^{-1} computed in step 3 above. Since we use SSNP to jointly compute P and P^{-1}, we obtain P for free, so there is no additional cost to drawing this scatterplot.

Summarizing the above pipeline in simple words, SDBM creates an image G, takes every pixel of this image and back-projects it to the data space $\mathbb{R}^n$ to obtain a data point, computes a label for this point using the classifier we wish to explore, and finally colors the pixel to show the class label and, optionally, the classifier's confidence, at that location. The end result is a dense colored map where same-color regions indicate regions in the data space where the classifier yields the same output (label), and color boundaries between adjacent regions indicate decision boundaries of the classifier.

Results

We next evaluate SDBM against the desirable criteria C1-C5 introduced in "Introduction" section. Specifically, we analyze quality (C1) by first using SDBM with synthetic data in a controlled setting, as we know what the 'ground truth' shapes of the decision zones are for a given synthetic dataset and given classifier ("Quality on Synthetic Datasets" section). We next assess quality for more complex real-world datasets and additional classifiers ("Quality on Real-World Datasets" section) and also compare SDBM with DBM. This shows also that SDBM is generic (C4) and that it increases quality as compared to DBM. Next, and in addition to [9], we present several experiments that measure SDBM's stability in the presence of different amounts and types of data change to support our stability claims (C5). Finally, we show how SDBM compares to DBM speed-wise and thereby justify our scalability claims (C2, "Computational Scalability" section). We end this section by providing full implementation details for SDBM ("Implementation Details" section).

Quality on Synthetic Datasets

To assess how SDBM performs in a controlled situation, we consider several synthetic Gaussian blobs with 5000 samples, with varied dimensionality (100 and 700), and varied number of classes (2 and 10). All points in a blob have the same class label. These are, thus, easily classifiable datasets, for which we expect the decision zones to 'surround' the respective blobs. We construct decision maps for four classifiers, namely Logistic Regression [12], SVM [13] (with a RBF kernel), Random Forests [14] (200 estimators), and a Neural Network (multi-layer perceptron with 3 layers of 200 units each). All these classifiers are able to handle the synthetic datasets with 100% accuracy. Consequently, as said above, we expect to see clearly-separated decision zones surrounding the data blobs in the projection.

Figure 2 shows the SDBM maps for all the dataset *vs* classifier combinations, with decision zones colored by class labels. Projected samples in $P(D)$ are drawn colored by their class too, but slightly brighter than the maps so they are visible around their respective decision zones. We first see that the projections (bright 'spots' in the figure) indicate clearly well separated blobs, which confirms the easy structure of these datasets. We also see that all decision zones are compact and with smooth boundaries, as expected for such simple datasets, and enclose the Gaussian blobs with the same respective labels. For example, the red and blue zones for the 2-class, 100-dimensional dataset (Fig. 2, top row), contain two clusters of light red, respectively light blue, projected points. The maps for Logistic Regression show almost

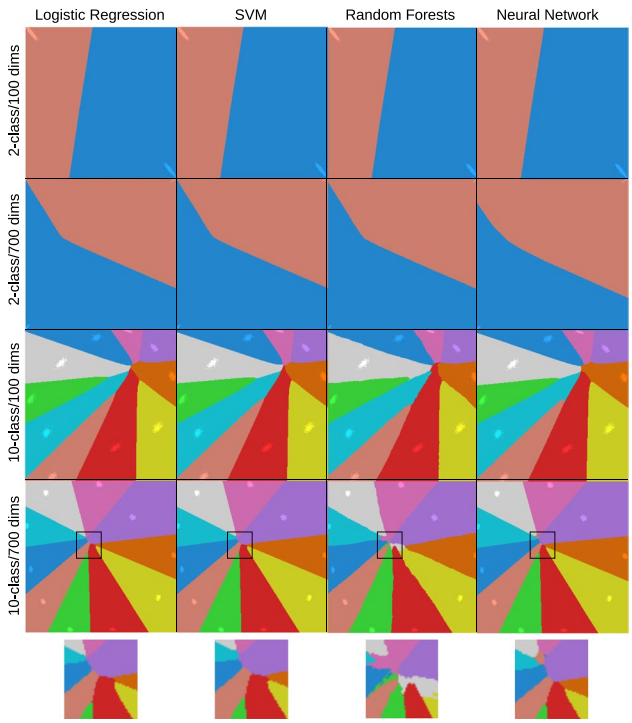

Fig. 2 Decision Boundary Maps (DBMs) created with SDBM for several classifiers (columns) and synthetic datasets (rows). Lighter pixels represent training samples from the datasets D. Insets show decision map details around the region where all decision zones meet for the 10-class, 700-dimensional dataset

perfectly straight boundaries, which is a known fact for this classifier. In contrast, the more sophisticated classifiers, Random Forests and Neural Networks, create boundaries that are slightly more complex than Logistic Regression and SVM for the most complex dataset. The differences are best visible for Random Forests and Neural Networks in the small wiggles of the decision zone shapes in the center of the maps in Fig. 2, bottom row (see also insets).

Quality on Real-World Datasets

We next show how SDBM performs on three real-world datasets. To select such datasets, we look at candidates which are (a) challenging for classification problems; (b) quite diverse in terms of data provenance, dimensionality (hundreds of dimensions), and size (thousands of samples); (c) well known, and openly accessible, to the machine learning community, for comparison and replication purposes. This quickly leads us to selecting datasets used in many ML evaluation benchmarks. Additionally, we note that using such datasets makes sense in our context since we want to evaluate a technique (SDBM) which is designed to visualize the behavior of classifier models.

With the above requirements, we selected the following datasets for evaluating SDBM:

FashionMNIST [54] $10K$ samples of $K = 10$ types of clothing images, rendered as 28×28-pixel gray scale images, flattened to 784-element vectors. We also use a subset of this dataset containing only two classes, namely *Ankle Boot* and *T-Shirt*, to show an example where classes are more easily separable.

Human Activity Recognition (HAR) [55] 10,299 samples from 30 subjects performing $K = 6$ daily activities, and used for human activity recognition. The samples have 561 dimensions that encode in the time and frequency domains 3-axial linear acceleration and 3-axial angular velocity measured on the subjects.

MNIST [56] $70K$ samples of $K = 10$ handwritten digits from 0 to 9, rendered as 28×28-pixel gray scale images, flattened to 784-element vectors. This dataset was downsampled to $10K$ observations for all uses in this paper.

Reuters Newswire Dataset [57] 8432 samples of news report documents, from which 5000 attributes were extracted using the standard TF-IDF [58] text processing method. From the full dataset, we use only the $K = 6$ most frequent classes.

Figure 3 shows the SDBM maps for these datasets for the same classifiers used in "Quality on Synthetic Datasets" section. These datasets are considerably more complex than the synthetic ones ("Quality on Synthetic Datasets" section), also seen by the varying accuracies they achieve for the different classifiers. Still, for all combinations, the classifiers' decision zones are clearly visible in Fig. 3.

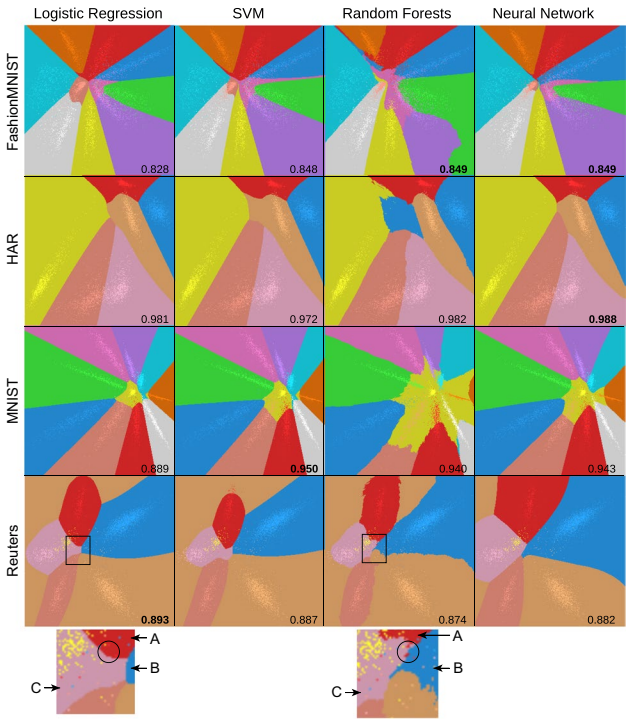

Fig. 3 Decision Boundary Maps (DBMs) created with SDBM for several classifiers (columns) and real-world datasets (rows). Numbers inside each map indicate test accuracy obtained by each classifier, bold indicating top performers. Lighter pixels represent training samples from the datasets D. Insets show details in the decision zones for Logistic Regression and Random Forests for the Reuters dataset

Also, we see—like for the synthetic datasets—how these decision zones surround the blobs of training-set samples, depicted as lighter-colored points in Fig. 3. As for the synthetic datasets, simpler classifiers (Logistic Regression and SVM) show decision zones that are more contiguous and have smoother, simpler, boundaries. More complex classifiers (Random Forests and Neural Networks) show more complex shapes and topologies of the decision zones. The maps for the Random Forest classifiers show very jagged boundaries. This can be a result of having an ensemble of classifiers working together. An interesting insight can be obtained when comparing the maps for different classifiers trained on the same dataset. Consider, e.g., for the Reuters dataset (bottom row), the best classifier (Logistic Regression (LR), accuracy 0.893) and the poorest classifier (Random Forests (RF), accuracy 0.874). The projected points are the *same* for the two maps, since we use the same training set. Still, we see different shapes and sizes of the decision zones. Consider now a small area in the two maps (see insets at the bottom of Fig. 3). As described earlier, both training-set points and decision zones are colored by label values. Hence, misclassified points will have different colors from their surrounding zones, while correctly classified points have the same (slightly lighter) colors as the surrounding

zones. Comparing the map details for LR and RF, we see that the red decision zone (A) is smaller for RF than for LR, while the blue zone (B) is comparatively larger. In the RF inset, we see that several red points in the black circle fall in the blue (B) and pink (C) decision zones, indicating misclassifications. These red points fall under the large red decision zone for the LR map. Hence, we conclude that the shapes of the LR decision zones, in this region, are more correctly following the training data than those of RF. Similar reasoning can be done to compare other decision map areas.

Encoding Classifier Confidence Fig. 4 shows SDBM maps with classifier *confidence* encoded as brightness, as described in "Methods" section. We see the added value of depicting confidence if we compare the first-*vs*-second (HAR), respectively third-*vs*-fourth (Reuters), rows in Fig. 4. The confidence maps show a brightness gradient, dark close to the decision boundaries (where colors change in the maps) and bright deep in the decision zones. This shows that confidence increases as we go deeper into the decision zones, i.e., closer to the training samples. For the HAR dataset, these dark bands are quite thin for Logistic Regression and SVM, thicker for Random Forests, and extremely and uniformly thin for Neural Networks. This tells us that Neural Networks have an overall very high confidence everywhere (except very close to the decision boundaries); Logistic Regression and SVM are less confident close to the boundaries; and Random Forests have a higher variation of confidence over the data space. Note

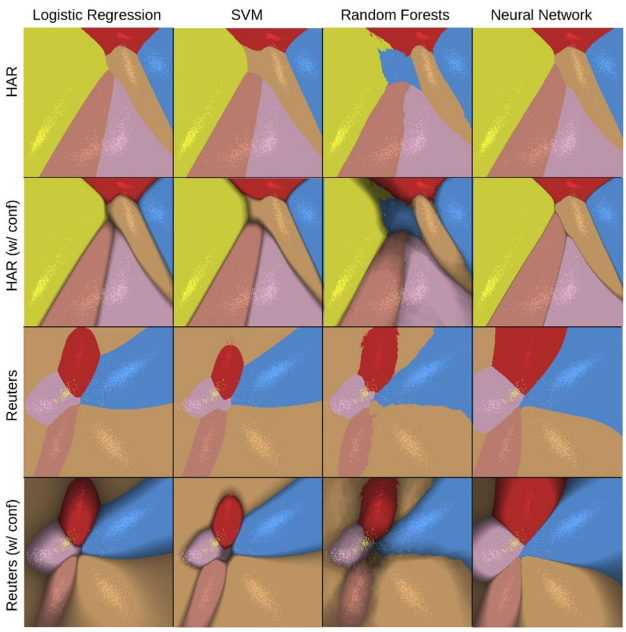

Fig. 4 Decision Boundary Maps created with SDBM for several classifiers, HAR and Reuters datasets. Columns show different classifiers. Rows show different datasets, with and without confidence encoded into brightness

how these findings match the classification accuracy values (Fig. 3). For Random Forests, the darkest region covers the central blue decision zone and the top-right of the left yellow zone. These are exactly the areas where the map for Random Forests significantly differs from those of all the other three classifiers. Hence, we can infer that the island-like blue decision zone that Random Forests created is likely wrong, as it is low confidence *and* different from what the other three classifiers created in that area. For the Reuters dataset (Fig. 4 bottom row), all classifiers produced a beige region at the top left corner. Brightness shows us that all classifiers except SVM treat this region as a low confidence one. This can be explained by the total absence of training samples in that region. This also tells us that the behavior of SVM in this region is likely wrong.

Confidence visualization also helps to quickly assess the *overall difficulty* of classifying a dataset. Consider e.g. the Reuters dataset (Fig. 4 bottom row). Compared to HAR (Fig. 4, second row), the decision maps for this dataset are darker for all four classifiers. This shows that it is harder to *extrapolate* (during inference) from a model trained on Reuters than one trained on HAR. Note that this is not the same as the usual testing-after-training in ML. Indeed, for testing, one needs to 'reserve' a set of labeled samples which cannot be used during training. In contrast, SDBM does not need to do this as it synthesizes 'testing' samples on the fly via the inverse projection P^{-1}. Also, classical ML testing only gives a global or per-class accuracy. In contrast, SDBM gives a per-region-of-the-data-space confidence, encoded by brightness.

Comparison with DBM Fig. 5 shows the SDBM maps side-by-side with maps created by the original DBM technique for Logistic Regression, Random Forest, and k-NN classifiers and three real-world datasets. For DBM, we used UMAP [17] for the direct projection and iLAMP [15] for the inverse projection. Several observations can be made, as follows.

First, we see that the SDBM and DBM projections $P(D)$ of the same datasets are not the same—compare the bright-colored dots in the corresponding figures. This is expected, since DBM employs a user-chosen projection technique P (UMAP in our case) while SDBM *learns* P from the label-based clustering of the data using the SSNP method (see "Methods" section). Since the DBM and SDBM projections $P(D)$ differ, it is expected that the overall shapes of the ensuing decision maps will also differ—see e.g. the nearly horizontal decision boundary between the blue and red zones for Random Forests with DBM for FashionMNIST (2-class) *vs* the angled boundary between the same zones for the same classifier, same dataset, with SDBM (Fig. 5, middle row, two leftmost images). For the relatively simple classification problem that FashionMNIST (2-class) is, this is not a problem. Both DBM and SDBM produce useful and usable

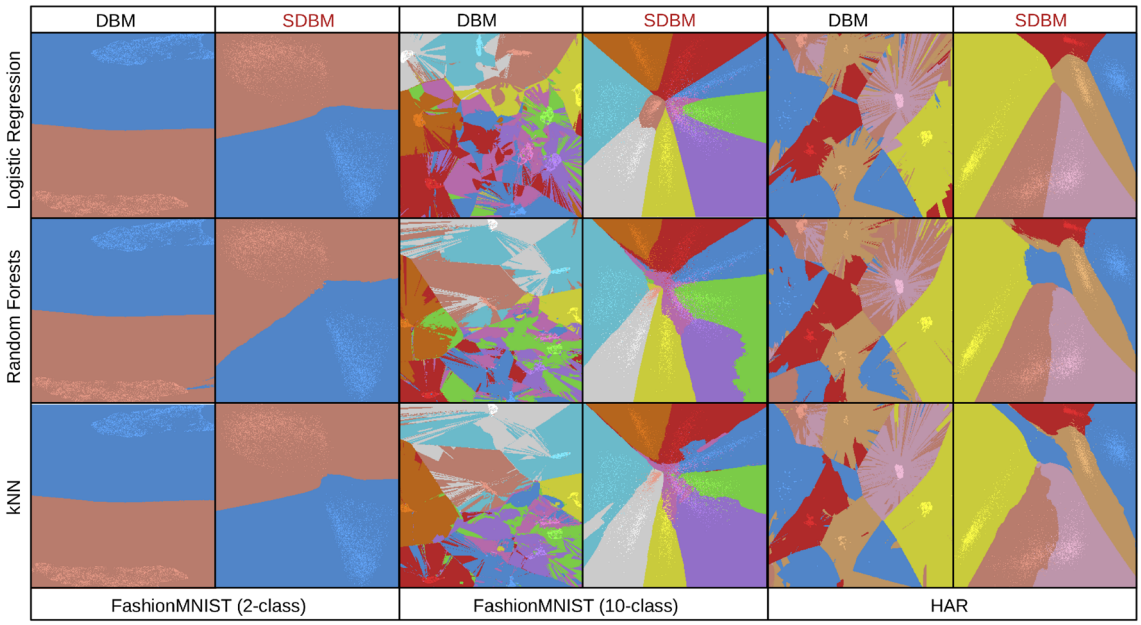

Fig. 5 Comparison between SDBM and DBM using three different datasets and three classifiers

renditions of the two resulting decision zones, showing that this classification problem succeeded with no issues.

For more difficult datasets (FashionMNIST 10-class or HAR), the situation is very different: DBM shows *highly* noisy pictures, in which it is very hard to say where and which are the actual decision zones. If these images were correct, this would mean that none of the three tested classifiers could correctly handle these two datasets. Indeed, such noise-like rapid changes as the DBM images show would mean that the classifiers would change decisions extremely rapidly and randomly as points only slightly change over the data space. This is *known* not to be the case for these classifiers. In more detail: Logistic Regression has built-in limitations of how quickly its decision boundaries can change [7]. k-NN essentially constructs a Voronoi diagram around the same-class samples in the *n*D space, partitioning that space into cells whose boundaries are smooth manifolds. DBM does not show any such behavior (Fig. 5, third and fifth columns). In contrast, SDBM shows a far lower noise level and far smoother, contiguous, decision zones and boundaries. Even though we do not have formal ground truth on how the zones and boundaries of these dataset-classifier combinations actually look, SDBM matches better the prior knowledge we have on these problems than DBM.

However, DBM's results depend on the choice of the direct projection P and inverse projection P^{-1} it uses (see "Methods" section). To compare SDBM with DBM under these degrees of freedom, we ran DBM for the three classifiers shown in Fig. 5 on the FashionMNIST 10-class dataset, but used four different projection methods P (Metric MDS [59], PLMP [60], Projection by Clustering (PBC) [61], and t-SNE [16]), and used NNInv [18] instead of iLAMP for the inverse projection P^{-1}. Figure 6 shows the decision maps created by DBM for these configurations. We see that these are practically as noisy as the DBM results shown in Fig. 5 (column 3). In contrast, the SDBM results (Fig. 5, column 4) show better separated, less noisy, smoother-boundary decision zones. This strengthens our claim that SDBM produces higher-quality maps than DBM.

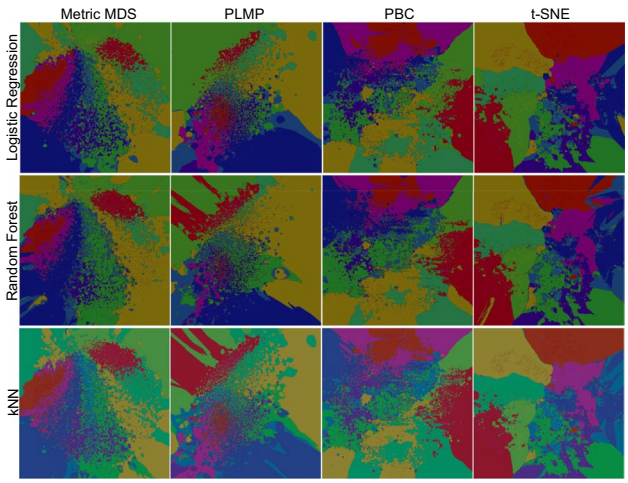

Fig. 6 DBM images using more direct and inverse projection methods, FashionMNIST (10 class). Compare these images with SDBM for the same classifiers and dataset in Fig. 5 (column 4)

Stability Analysis

We now turn to the stability desirable criterion (C5, "Introduction" section). As explained there and also in "Background" section, a stable decision map algorithm is one which shows only small changes in the output decision map when its input, i.e., the labeled dataset $(D, \mathbf{y})$ it was constructed to show, change little. By extension, when this input does not change, the output map should also not change. If this is not the case, then the decision map may show patterns which are highly influenced by irrelevant (small) changes in the input data or algorithm parameters which, in turn, can be highly misleading.

Figures 5 and 6 show two reasons why the original DBM algorithm is *unstable*. First, we see that DBM creates very noisy, discontinuous, decision maps. However, as we explained in "Quality on Real-World Datasets" section, the visualized classifiers are known to change their decision *slowly* as their inputs change. The DBM images in Figs. 5 and 6 show a different picture, suggesting that the classifiers rapidly change outputs as inputs only slightly change. Thus, DBM itself introduces instabilities in the computation of the decision maps which are not genuinely there in the visualized classifiers. In contrast, SDBM shows far smoother, less noisy, decision maps, for the *same* classifiers and datasets. Secondly, Fig. 6 shows that DBM creates very different (and still noisy and discontinuous) decision maps when we change its two hyperparameters, namely the direct projection P and inverse projection P^{-1}, for the *same* dataset-classifier combination. This is by definition an unstable algorithm.

While SDBM shows far smoother, more continuous, decision maps than DBM, we would like more evidence to claim that SDBM is stable. In this section, we address this by explicitly measuring SDBM's stability, as follows (see also Fig. 7; compare to Fig. 1 that shows the baseline SDBM method):

- let $D \in \mathbb{R}^n$ be a training dataset with labels $\mathbf{y}$ (Fig. 7 step 0);
- let $f(D)$ be a classification model trained on D and $\mathbf{y}$;
- let $S(f)$ be the decision map computed by SDBM on $f(D)$;
- let $\pi : \mathbb{R}^n \times [0, 1] \rightarrow \mathbb{R}^n$ be a change, or perturbation function. That is, $\pi(D, \sigma)$ is the dataset D changed by π with a change intensity $\sigma \in [0, 1]$. Larger σ values change D more, and $\pi(D, 0) = D$, i.e. a value $\sigma = 0$ means no change. We record this change intensity σ by a set of samples, or change amounts, σ_i (Fig. 7 step 1);
- compute $D_i = \pi(D, \sigma_i)$, a set variations of the dataset D, changed by perturbation π, with change intensities σ_i (Fig. 7 step 2);
- train the models $f_i = f(D_i)$. Labels $\mathbf{y}$ of D_i stay the same as those of D, only the sample values change (Fig. 7 step 3). Note that we apply the changes on the *training* sets of the visualized classifiers, not the test sets, since changing a test set will not change the decision map of a trained classifier;
- construct the decision maps $S_i = S(f_i)$ (Fig. 7 step 4);
- visualize the maps S_i to interpret the stability of SDBM (Fig. 7 step 5).

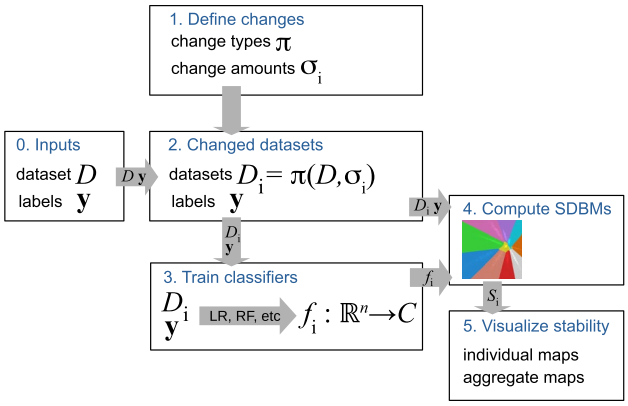

Fig. 7 Pipeline for assessing SDBM stability

The intuition of the above procedure is simple: Let S_0 be the decision map computed by SDBM for a dataset D and some classifier f. Let S_i be the decision maps computed by SDBM for the same classifier but for increasingly perturbed versions D_i of the dataset. If SDBM is a stable method, then it should produce decision maps S_i which are similar to S_0 for low i values and increasingly different from S_0 as i increases. Note that a similar definition of stability—small input data changes should lead to small output visualization changes—was used to assess other visualization techniques for high-dimensional data such as projections [29, 30] and treemaps [62, 63].

We apply this procedure to three types of data changes π defined as follows:

- *Add constant:* π adds a fixed bias value σ to all dimensions of D. For the image datasets (MNIST, FashionMNIST), we used $\sigma_i \in \{0.07, 0.15, 0.3\}$, which correspond to a 'brightening' of the images with up to 30%. For the Reuters text dataset, we used $\sigma_i \in \{0.05, 0.08, 0.2\}$;
- *Drop dimensions:* π sets to zero a given number of randomly chosen dimensions from the n ones of D. We used here $\sigma_i \in \{0.1n, 0.2n, 0.3n\}$, which means that π has an effect similar to removing up to 30% of D's dimensions;
- *Random noise:* π adds to all D's dimensions random noise sampled from a normal distribution with mean 0 and standard deviation $\sigma_i \in \{0.01, 0.05, 0.1\}$.

These changes are related to the ones used in [30] to test the stability of the NNP projection—not the same as our

classifier decision maps, but related in spirit. For additional rationale referring to the purposefulness of these changes, we refer to [30].

Visualizing Stability We next apply these three change types π, each sampled for three change intensities σ_i, for the SDBM maps constructed for Logistic Regression and Neural Networks trained with the MNIST, FashionMNIST, and Reuters datasets—thus, we compute and evaluate a total of $3 \times 3 \times 2 \times 3 = 54$ decision map images. Figures 8, 9, and 10 show these images for the three datasets with confidence encoded into brightness (see "Quality on Real-World Datasets" section). In each figure, the leftmost column (labeled 'without noise') shows the decision map of the original, unperturbed, dataset. The rightmost three columns show, per row, the decision maps for the respective (classifier, dataset, noise-type) combination, for increasing amounts of noise amounts. As explained earlier, if the decision maps slowly and increasingly change with respect to the noise-free

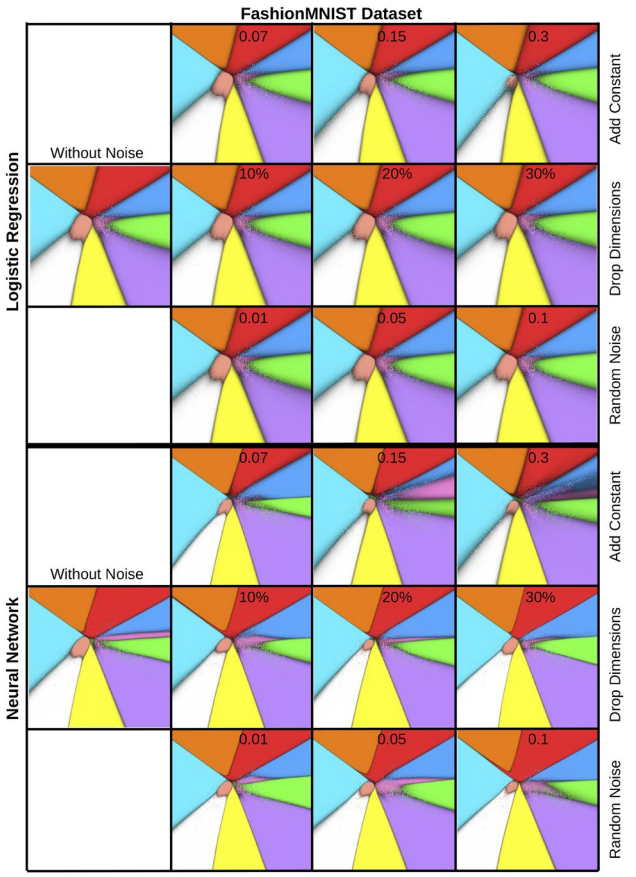

Fig. 8 SDBM decision maps for two classifiers trained with varying types and amounts of noise, FashionMNIST dataset. Confidence is encoded into brightness. Leftmost column shows the maps for the original, un-noised, datasets. As more noise is added (columns 2 to 4, noise amounts marked inside images), the decision maps start progressively diverging from the original, leftmost, map

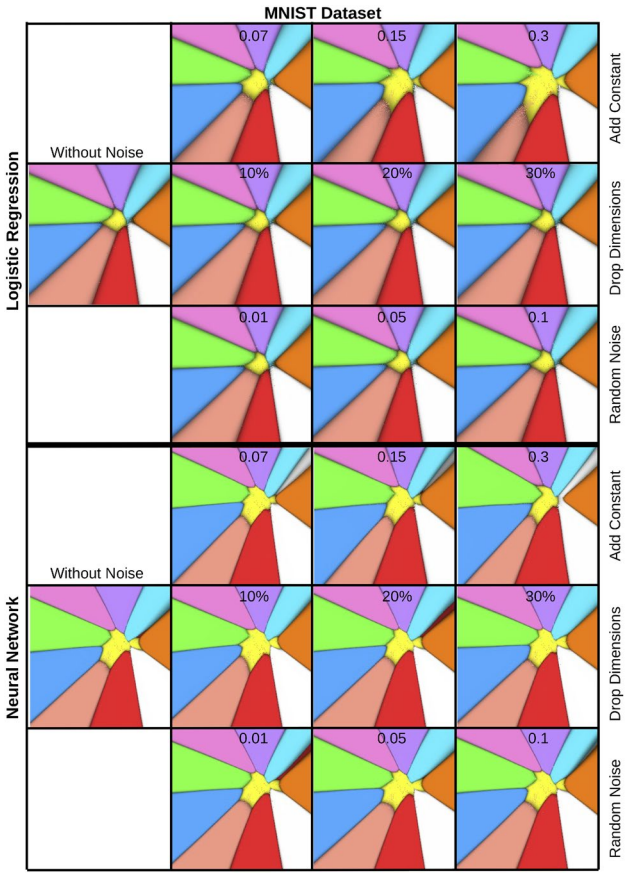

Fig. 9 SDBM decision maps for two classifiers trained with varying types and amounts of noise, MNIST dataset. Confidence is encoded into brightness. Leftmost column shows the maps for the original, un-noised, datasets. As more noise is added (columns 2 to 4, noise amounts marked inside images), the decision maps start progressively diverging from the original, leftmost, map

map as the noise level increases, this means that the SDBM method is stable.

Figures 8, 9, and 10 show indeed this stability. If we scan each row left-to-right, we see how the images become progressively more different from the leftmost image (decision map for the unchanged dataset). Different rows for a classifier show the effect of the three different change types. Interestingly, in terms of overall amount of visual change, these effects are quite similar. Take, for example, Logistic Regression trained with FashionMNIST (Fig. 8, top three rows): The nine images to the right are quite similar among themselves and also similar with the decision map of the unchanged dataset (shown in the left column). Also, we see that the changes of the maps do not seem to differ—in terms of amount—for the three datasets. The fact that SDBM appears to be quite stable for different change types *and* for different datasets is a quite unexpected result, as the nature of the three change types and the three tested datasets is quite different. Related work [30] has shown that, when

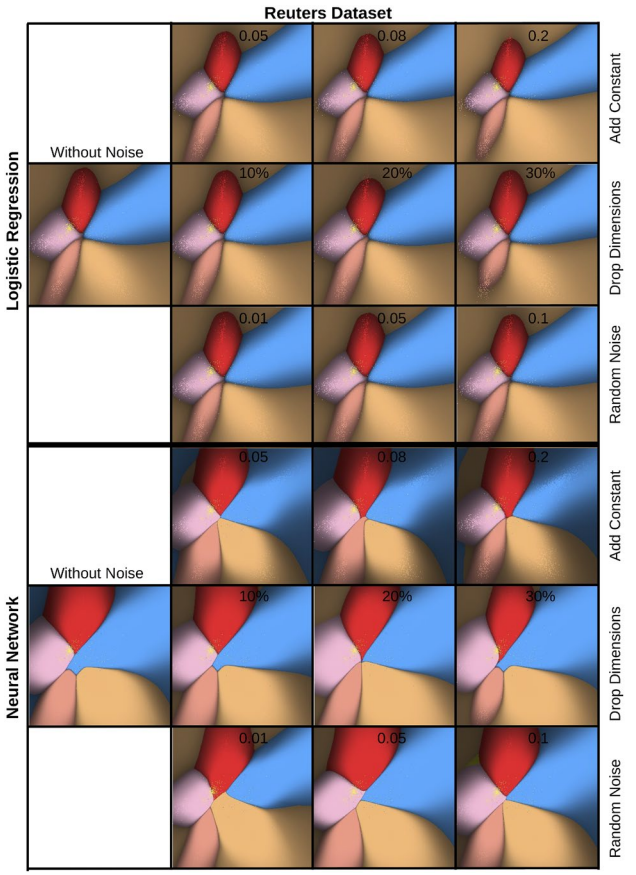

Fig. 10 SDBM decision maps for two classifiers trained with varying types and amounts of noise, Reuters dataset. Confidence is encoded into brightness. Leftmost column shows the maps for the original, un-noised, datasets. As more noise is added (columns 2 to 4, noise amounts marked inside images), the decision maps start progressively diverging from the original, leftmost, map

testing the stability of the NNP deep-learned projection [31], different change types (similar to ours) have quite different effects and also that the effects differ strongly as a function of the dataset. In other words, SDBM appears to be a more stable method than NNP. There can be many factors that make SDBM and NNP different, including the supervised nature of NNP *vs* self-supervised one of SDBM and the fact that SDBM learns and next applies both a direct and inverse projection, whereas NNP only learns a direct projection.

Figures 8, 9, and 10 also outline two other important aspects of SDBM. First, we see that not just the shapes and sizes, but also the relative *positions* in the image of the various decision zones are quite stable over change. This is important for practical SDBM usage. Indeed, if the decision zones would maintain similar sizes and shapes but wildly change positions, interpreting the maps would be hard. Moreover, such position changes would confuse the user as they would imply instability of the underlying classification model. Secondly, we see that the *confidence* of the maps changes only very little as with the dataset changes. This is a desirable result that confirms indirectly SDBM's stability, as follows: Small changes of the confidence indicate that the trained classifiers for the various changed datasets $\pi(D, \sigma_i)$ behave *similarly* to the classifier trained on the unchanged dataset. Since these classifiers are similar, their decision maps also should be similar—and this is what we observe in the above-mentioned images.

Aggregated Change Maps Visualizing individual SDBM maps for increasing amounts of change can be difficult as each such image needs to be compared with the original map (for the unchanged dataset). This becomes even harder to do when one wants to consider more than a few samples values of the change amount—which is actually useful when one wants to discern a clearer *trend* in terms of visual (map) change *vs* data change. To address this, we propose two ways to *aggregate* multiple SDBM maps, as follows. Consider all maps S_i computed multiple values $\sigma_i, 1 \leq i \leq N$, for a single change type π. We compute a single aggregated map by analyzing, at each pixel location $\mathbf{p}$, the label values f_i of the N images S_i at location $\mathbf{p}$, using a 'hard voting' procedure. The color assigned to the aggregated map at $\mathbf{p}$ will map the label appearing most frequently in the set $\{f_1, \ldots, f_n\}$ at that location. We also set the luminance of the aggregated map at $\mathbf{p}$ to the fraction of the N maps that have 'voted' for this value.

Figure 11 shows SDBM maps for the same classifiers and datasets as discussed above, aggregated using hard voting for each change type. We do not aggregate different change types together as we believe this would be confusing to interpret. For each change type, we use $N = 10$ different change values σ_i, sampled uniformly between the minimum and maximum values used earlier to generate Figs. 8, 9, and 10. We interpret the images in this figure as follows: Bright colored areas indicate *stable* decision zones which do not change as the classifier's training set is perturbed. Darker areas indicate decision zones which change as the training set is perturbed—the darker the area, the more that decision zone changes during the applied data changes. The images exhibit a 'color banded' structure, which is due to the fact that there are maximally $K = 10$ possible brightness levels, where K is the number of classes of the problem. These range from fully bright, indicating 100% agreement of all decision zones for all N trained classifiers, to a luminance of $1/K$, which indicates that the N classifiers are uniformly split into K groups each voting for a different label value. The brightness inside a decision zone has a similar gradient with that shown earlier in Figs. 8, 9, and 10—that is, points close to a decision boundary appear darkest while points deep inside a decision zone are brightest. However, the meaning of the brightness is different: In the earlier image, brightness encoded *confidence* of classification at a given map location; in the aggregated map, brightness encodes *stability* of the decision maps at a given location. Confidence and stability

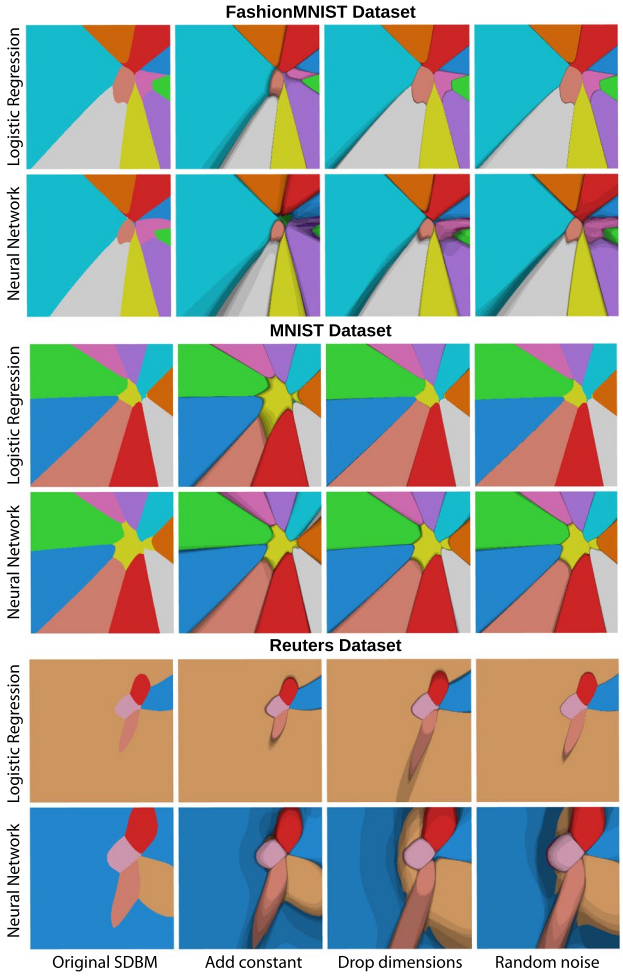

Fig. 11 Aggregated decision maps computed by hard voting for two classifiers trained with three change types, ten change amounts, on three datasets. Leftmost column shows the decision maps of the unchanged datasets

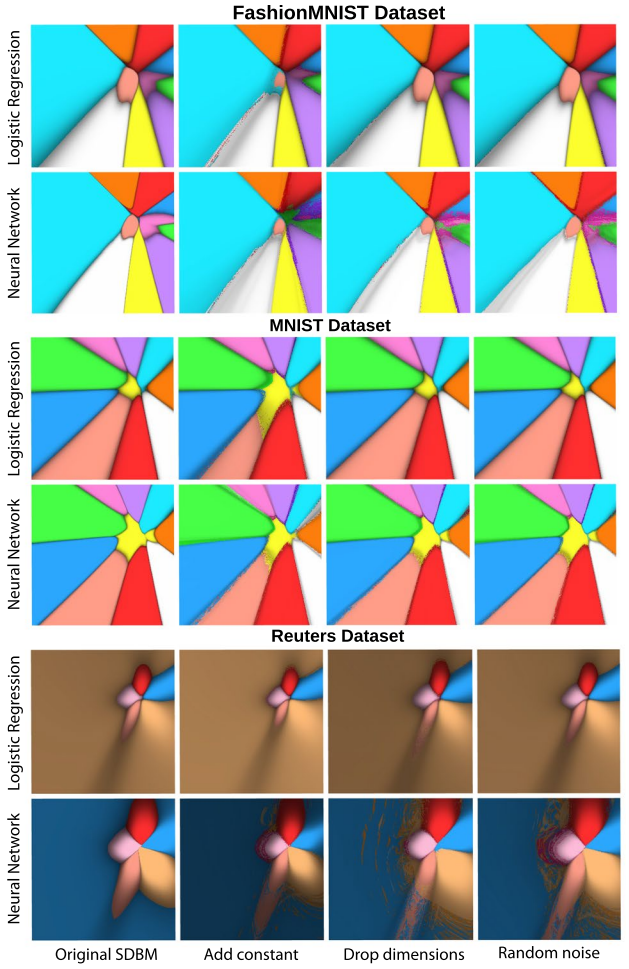

Fig. 12 Aggregated decision maps computed by soft voting for two classifiers trained with three change types, ten change amounts, on three datasets. Leftmost column shows the decision maps of the unchanged datasets

are positively correlated—a map changes the least in areas where a classifier is very confident of its inference and conversely. However, the two visualizations convey different types of insights, as explained above.

The above discussion suggests that it would be useful to create an aggregated map that visualizes *both* confidence and stability. We do this using, by analogy with the technique presented above, a 'soft voting' procedure, as follows. The color of each pixel in this soft-voting map is determined identically to the hard-voting map as the most frequent label in the set $\{f_1, \ldots, f_n\}$ at that location. However, we now set the brightness to depict the average value, at that location, of the confidences of the N classifiers whose SDBM maps we want to aggregate. This way, the soft-voting map depicts the overall confidence of the aggregated maps across all applied change levels σ_i.

Figure 12 shows the SDBMs for the same classifiers and datasets as in Fig. 11 aggregated with soft voting. The interpretation of the aggregated images in Fig. 12 is different from those in Fig. 11—darker regions indicate now areas where the aggregated decision maps have overall low confidence. We see that such areas follow the decision zone borders in the aggregated maps, just as in the original, unperturbed, maps (shown in the leftmost column in Fig. 12). We also see that the dark areas in Fig. 12 correlate well with the dark areas in Fig. 11. This tells us that, for the studied classifiers and datasets, the decision maps are less stable in areas where the classifiers are low confidence, and conversely, decision maps are *stable in areas of high classifier confidence*. The latter answers our question from "Introduction" section positively.

Computational Scalability

We next study the scalability of SDBM and compare it to the original DBM method. For this, we created maps using

synthetic Gaussian blobs datasets with 5 clusters, varying the dimensionality from 10 to 500, and varying the map size from 25^2 to 300^2 pixels. We did not use larger maps since the speed-trends were already clear from these sizes, with DBM getting considerably slower than SDBM. We used this synthetic data approach, rather than real-world datasets, as it allowed us to control the data dimensionality in a fine-grained way, which is the key factor influencing computing speed for both methods. Note that the number of samples does not heavily influence DBM and DBM's computing times. Both methods only need to project a dataset *once* after which they need to apply the inverse projection P^{-1} for each map *pixel*. For typical situations, the pixel count is far larger than the sample count, which makes the former dominate the map computation cost.

Figure 13 shows the running times of both methods as a function of both the grid size (horizontal axis) and dataset dimensionality (different-color lines). We see that DBM's runtime increases quickly with dimensionality, taking about 5 min to create a 300^2 map for the 500-dimensional dataset. In contrast, SDBM is over an order of magnitude faster, taking roughly 7 s to run for the same dataset. Also, we see that SDBM's speed depends far less on the data dimensionality, whereas this is a major slowdown factor for DBM. All in all, this shows that SDBM is significantly more scalable than DBM. This can be explained by the fact that SSNP, which underlies SDBM, *jointly* trains both the direct and inverse projections by deep learning. SDBM is GPU-accelerated, linear in the sample and dimension counts both for training and inference, and does not need to use different resolutions and sampling tricks for accelerating the 2D to *n*D mapping (see "Methods" section). In contrast, DBM uses UMAP and iLAMP for the direct, respectively, inverse projections (as mentioned earlier). None of these techniques is GPU-accelerated. Also, while UMAP is close to linear in the sample count and dimensionality, iLAMP is superlinear in dimensionality and sample count. Together, these aspects make DBM significantly slower than SDBM.

Implementation Details

All experiments presented above were run on a dual 8-core Intel Xeon Silver 4110 with 256 GB RAM and an NVidia GeForce RTX 2070 GPU with 8 GB VRAM. Table 1 lists all open-source software libraries used to build SDBM and the other tested techniques. Our implementation and all datasets used in this work are publicly available at [64].

Discussion

We discuss how our technique performs with respect to the criteria C1-C5 introduced in "Introduction" section.

Quality (C1) SDBM is able to create maps that show classifier decision boundaries very clearly, and, most importantly, much clearer than the maps created with the original DBM. For the same dataset-classifier combinations, SDBM's maps show significantly less noise, more compact decision zones, and smoother decision boundaries, than DBM. These results are in line with what we expect for dataset-classifier combinations for which we have ground-truth knowledge about their decision zones and boundaries (see Fig. 5 and related text). As such, we conclude that SDBM captures the actual decision zones better than DBM can do.

Table 1 Software packages used in the evaluation

Technique	Software used publicly available at	References
SSNP	http://keras.io (TensorFlow backend)	[65]
UMAP	http://github.com/lmcinnes/umap	[17]

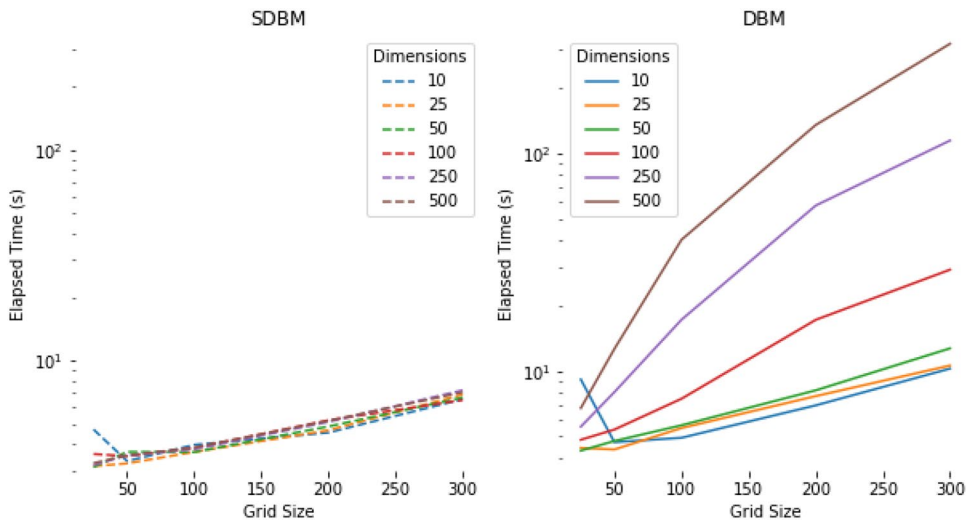

Fig. 13 Computation time to create decision maps of increasing size by SDBM and DBM using synthetic datasets of varying dimensionality. Vertical axis is in logarithmic scale

Scalability (C2) SDBM is an order of magnitude faster than DBM. Since SDBM scales linearly in the number of observations during inference/drawing, and it is end-to-end GPU-accelerated, it is able to generate maps having hundreds of thousands of pixels in a few seconds, which makes it practical for handling large datasets and rendering highly detailed decision maps.

Ease of use (C3) SDBM produces good results with practically no need to for hyperparameter tuning. In more detail, there are only two such hyperparameters. First, there is the number of epochs used to train SSNP to construct the direct and inverse projections. Following [52], we set this to a default value of 10. Secondly, there is the resolution R of the output decision map. Note, however, that this parameter does not influence the *stability* of the method, but only the level-of-detail of the produced decision maps. As Fig. 13 shows, the computation time is linear with the output resolution—which is expected, since SDBM needs to execute an inverse projection and classifier model inference per output pixel, and both these operations have a constant cost. We also note that, compared to SDBM, the parameter-setting of DBM is far more complex. Briefly put, DBM is very slow and as such uses a low resolution. However, this implies a sparse sampling of the input data space. To counter this, DBM creates multiple randomly-distributed 2D sample points in each grid cell, backprojects these, classifies the backprojections, and aggregates the resulting labels to compute the final pixel color. To obtain good results, DBM requires a careful tuning of the number of such sample points inside every pixel (for more details, we refer to [7]). SDBM does not have any of these problems as it can directly construct high-resolution images.

Genericity (C4) As for the original DBM method, SDBM is agnostic to the nature and dimensionality of the input data, and to the classifier being visualized. We show that SDBM achieves high quality on datasets of different natures and coming from a wide range of application domains, and with classifiers based on quite different algorithms. As such, SDBM does not trade any flexibility that DBM already offered, but increases quality, scalability, and ease of use, as explained above.

Stability (C5) We have described a range of experiments that show that SDBM is stable with respect to changes in the training dataset of the classifiers it visualizes. For quite significant changes amounting to additive bias up to 30% of the data range, dropping up to 30% dimensions, and adding noise up to a 0.1 standard deviation, SDBM creates decision maps which differ visually little from the ones for the unperturbed datasets. Additionally, we showed that the type of perturbation does not significantly influence the amount of change in the produced decision maps. The maps are also stable in the sense that decision zones are plotted in (roughly) the same areas of the map regardless of the perturbation. Moreover, the overall visual appearance of the decision maps, e.g. in terms of decision boundary smoothness and island-like small-scale disconnected regions, is not influenced by perturbing the training set. Interestingly, the stability of SDBM is far higher in the presence of similar input perturbations than that of a related NNP technique that also uses deep learning for projecting high-dimensional data [30]. Interpreting SDBM's stability needs, potentially, a few extra words: What we showed, is that classifiers trained by changed training-data produce similar decision maps to classifiers trained by unperturbed data. We argue that this makes sense in the formal definition of stability of a function (SDBM in our case) since we change the input of that function which, in our case, is the trained classifier f (see "Background" section). In turn, f's behavior depends solely on its training set. One could argue that a trained classifier also depends on the test set and that such a test set should be varied as well to assess the classifier. While this is true, changing a test set does not change the formal decision zones or boundaries of a trained classifier model, hence it does not change its SDBM visualization. As such, the main variable we can change to assess SDBM's stability is the classifier's training set. Importantly, we also showed that SDBM is more stable than its predecessor, DBM—which can be ascribed to the deterministic nature of the deep-learned direct and inverse projections in SSNP as opposed to the direct and inverse projections (UMAP, respectively iLAMP) used by DBM.

Limitations SDBM shares a few limitations with DBM. First and foremost, it is hard to *formally* assess the quality of the decision maps it produces for dataset-classifier combinations for which we do not have clear ground-truth on the shape and position of their decision zones and boundaries. Our work showed that SDBM produces results fully in line with known ground truth for such simple situations. However, this does not formally guarantee that the same is true for more complex datasets and any classifiers. Finding ways to assess this is an open problem to be studied in future work. Secondly, the interpretation of the SDBM maps can be enhanced. Examples shown in this paper outlined how such maps can help finding out whether a trained classifier can generalize well, and how far, from its training set, and how different classifier-dataset combinations can be compared by such maps. Yet, such evidence is qualitative. A more formal study showing how users actually *interpret* such maps to extract quantitative information on the visualized classification problems is needed. Finally, while our stability study outlined that SDBM is surprisingly stable to significant variations of a classifier's training set, a full understanding of such a stability concept needs further work. For instance, one would like to test SDBM stability in presence of varying the classifier's training hyperparameters. Also, in such a stability study, one would arguably want to use more

data-domain-dependent change types than the generic ones that we explored in "Stability Analysis" section.

Applications Decision maps are not and end by themselves but a *tool* that allows ML engineers to study a given classification model and, in the case the model performs poorly, obtain insights on how to improve it. The current paper has shown that SDBM can produce high-quality decision maps that have all the requirements needed for their application in practice (as discussed above). As such, SDBM is now ready to be actually deployed in concrete scenarios. In this respect, we believe that *imaging* applications are one of the domains where SDBM would best fit. Examples of ML applications in this domain include transfer learning for image classification [66], understanding important features for classification of histopathology images [67], analysis of misclassification results in cell image classification [68], microorganism image segmentation [69], and data augmentation for cancer image classification [70].

What all these applications have in common—from a technical perspective—is the (a) usage of complex multi-stage, deep learning, models to (b) analyze image data. As such, fine-tuning the respective models is a complex task, for which projections are typically used. We believe that using decision maps can significantly augment the insights shown by projections as one can effectively see how decision zones and decision boundaries relate to the training-set and test-set points. Moreover, since the targeted data are *images*, one can effectively display such images e.g. as users move a tooltip over the decision map image. This can show not only which existing images fall in specific decision zones (or close to decision boundaries), but also synthesize *new* images, via backprojection, that fall in the 'empty' spaces between existing samples. These synthesized images can next help understand and improve how the trained classification models work, e.g., by user-supervised data augmentation.

Conclusion

In this paper, we have presented and explored the behavior of SDBM, a new method for producing classifier Decision Boundary Maps. Compared to the only other similar technique we are aware of—DBM—our method has several desirable characteristics. First, it is able to create decision maps which are far smoother and less noisy than those created by DBM and also match the known ground-truth of the visualized classification problems far better than DBM, therefore allowing users to interpret the studied classifiers with less confusion. Secondly, SDBM is about an order of magnitude faster than DBM due to its joint computation of direct and inverse projections on a fixed-resolution image by deep learning. Thirdly, SDBM has virtually no parameters to tune (apart from the resolution of the desired final image) which makes it easier to use than DBM. Finally, in addition to our earlier work [9], we have presented a comprehensive study of SDBM's stability in presence of several types and amounts of changes of the examined classifiers' training sets. Our study shows that SDBM is quite stable for a wide range of such changes, irrespective of the classifier used or the nature of the training set. We have also presented new methods to compactly visualize SDBM's stability by means of aggregated maps which summarize the changes in several SDBM maps.

Future work can target several directions. A very relevant direction is the generation of maps for multi-output classifiers, i.e., classifiers that can output more than a single class for a sample. Secondly, we consider organizing more quantitative studies to actually gauge which are the interpretation errors that SDBM maps generate when users consider them to assess and/or compare the behavior of different classifiers, which is the core use-case that decision maps have been proposed for. Thirdly, proposing new methods to measure and visualize SDBM's stability can not only help increasing trust in this method but also help understanding stability of other regressors for changing high-dimensional data [19, 29, 30]. In this sense, proposing formal metrics to characterize SDBM's stability, akin to measuring directional derivatives of a multivariable function, or sensitivity analysis [71], is a key direction to follow. Last but not least, using SDBM to understand and improve existing complex classification models, especially for image data, is an important direction we aim to pursue.

Funding This study was financed in part by FAPESP grants 2015/22308-2, 2017/25835-9 and 2020/13275-1, and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brasil (CAPES)—Finance Code 001.

Data Availability Not applicable.

Code Availability Our implementation, plus all code used in our experiments, are publicly available at github.com/mespadoto/sdbm.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

1. Ribeiro MT, Singh S, Guestrin C. Why should i trust you?: Explaining the predictions of any classifier. In: Proc. ACM SIGMOD KDD. 2016. p. 1135–1144.

2. Garcia R, Telea A, da Silva B, Torresen J, Comba J. A task-and-technique centered survey on visual analytics for deep learning model engineering. Comput Gr. 2018;77:30–49.
3. Lundberg S.M, Lee S.-I. A unified approach to interpreting model predictions. In: Proc. NIPS. 2017. p. 4768–4777.
4. Nóbrega C, Marinho L. Towards explaining recommendations through local surrogate models. In: Proc. ACM/SIGAPP symp. on applied computing. 2019. p. 1671–1678.
5. Rauber PE, Falcao AX, Telea AC. Projections as visual aids for classification system design. Inf Vis. 2017;17(4):282–305.
6. Rauber PE, Fadel SG, Falcao AX, Telea AC. Visualizing the hidden activity of artificial neural networks. IEEE TVCG. 2017;23(1):101–10.
7. Rodrigues F, Espadoto M, Hirata R, Telea AC. Constructing and visualizing high-quality classifier decision boundary maps. Information. 2019;10(9):280.
8. Nonato L, Aupetit M. Multidimensional projection for visual analytics: linking techniques with distortions, tasks, and layout enrichment. IEEE TVCG. 2018. https://doi.org/10.1109/TVCG.2018.2846735.
9. Oliveira A.A.M, Espadoto M, Hirata R, Telea A. SDBM: supervised decision boundary maps for machine learning classifiers. In: Proc. IVAPP. 2022. p. 77–87.
10. Rodrigues FCM, Hirata R, Telea AC. Image-based visualization of classifier decision boundaries. In: Proc. IEEE conf. on graphics, patterns and images (SIBGRAPI). 2018. p. 353–360.
11. Espadoto M, Rodrigues FCM, Telea AC. Visual analytics of multidimensional projections for constructing classifier decision boundary maps. In: Proc. IVAPP. SCITEPRESS. 2019. p. 132–144.
12. Cox DR. The regression analysis of binary sequences. J R Stat Soc Ser B (Methodological). 1958;20(2):215–32.
13. Cortes C, Vapnik V. Support-vector networks. Mach Learn. 1995;20(3):273–97.
14. Breiman L. Random forests. Mach Learn. 2001;45(1):5–32.
15. Amorim E, Brazil EV, Daniels J, Joia P, Nonato L.G, Sousa MC. iLAMP: exploring high-dimensional spacing through backward multidimensional projection. In: Proc. IEEE VAST. 2012. p. 53–62.
16. Maaten LVD, Hinton G. Visualizing data using t-SNE. JMLR. 2008;9:2579–605.
17. McInnes L, Healy J. UMAP: uniform manifold approximation and projection for dimension reduction. 2018. arXiv:1802.03426 v1 [stat.ML].
18. Espadoto M, Rodrigues FCM, Hirata NST, Hirata Jr. R, Telea AC. Deep learning inverse multidimensional projections. In: Proc. EuroVA. Eurographics. 2019.
19. Espadoto M, Rodrigues FCM, Hirata N, Telea A. OptMap: using dense maps for visualizing multidimensional optimization problems. In: Proc. IVAPP. SciTePress. 2021.
20. Collaris D, van Wijk JJ. StrategyAtlas: strategy analysis for machine learning interpretability. IEEE TVCG. 2022. https://doi.org/10.1109/TVCG.2022.3146806.
21. Shepard D. A two-dimensional interpolation function for irregularly-spaced data. In: Proc. ACM national conference. 1968. p. 517–524.
22. Aupetit M. Visualizing distortions and recovering topology in continuous projection techniques. Neurocomputing. 2007;10(7):1304–30.
23. Martins R, Coimbra D, Minghim R, Telea A. Visual analysis of dimensionality reduction quality for parameterized projections. Comput Gr. 2014;41:26–42.
24. Tian Z, Zhai X, van Driel D, van Steenpaal G, Espadoto M, Telea A. Using multiple attribute-based explanations of multidimensional projections to explore high-dimensional data. Comput Gr. 2021;98:93–104.
25. Venna J, Kaski S. Visualizing gene interaction graphs with local multidimensional scaling. In: Proc. ESANN. 2006. p. 557–562.
26. Seifert C, Sabol V, Kienreich W. Stress maps: analysing local phenomena in dimensionality reduction based visualisations. In: Proc. IEEE VAST. 2010.
27. Joia P, Coimbra D, Cuminato JA, Paulovich FV, Nonato LG. Local affine multidimensional projection. IEEE TVCG. 2011;17(12):2563–71.
28. Espadoto M, Martins RM, Kerren A, Hirata NS, Telea AC. Toward a quantitative survey of dimension reduction techniques. IEEE TVCG. 2019;27(3):2153–73.
29. Vernier E, Garcia R, Silva I.d, Comba J, Telea A. Quantitative evaluation of time-dependent multidimensional projection techniques. In: Proc. EuroVis. 2020.
30. Bredius C, Tian Z, Telea A. Visual exploration of neural network projection stability. In: Proc. MLVis. Eurographics. 2022.
31. Espadoto M, Hirata NST, Telea AC. Deep learning multidimensional projections. Inf Vis. 2020;19(3):247–69.
32. Espadoto M, Falcao A, Hirata N, Telea A. Improving neural network-based multidimensional projections. In: Proc. IVAPP. 2020.
33. Hoffman P, Grinstein G. A survey of visualizations for high-dimensional data mining. Inf Vis Data Min Knowl Discov. 2002;104:47–82.
34. Maaten LVD, Postma E. Dimensionality reduction: a comparative review. Technical report, Tilburg University, Netherlands (2009)
35. Engel D, Hattenberger L, Hamann B. A survey of dimension reduction methods for high-dimensional data analysis and visualization. In: Proc. IRTG Workshop, vol. 27. Schloss Dagstuhl. 2012. p. 135–149.
36. Sorzano C, Vargas J, Pascual-Montano A. A survey of dimensionality reduction techniques. 2014. arXiv:1403.2877 [stat.ML].
37. Liu S, Maljovec D, Wang B, Bremer P-T, Pascucci V. Visualizing high-dimensional data: advances in the past decade. IEEE TVCG. 2015;23(3):1249–68.
38. Cunningham J, Ghahramani Z. Linear dimensionality reduction: survey, insights, and generalizations. JMLR. 2015;16:2859–900.
39. Xie H, Li J, Xue H. A survey of dimensionality reduction techniques based on random projection. 2017. arXiv:1706.04371 [cs.LG].
40. Jolliffe IT. Principal component analysis and factor analysis. In: Principal component analysis. Springer. 1986. p. 115–128.
41. Torgerson WS. Theory and methods of scaling. Oxford: Wiley; 1958.
42. Tenenbaum JB, Silva VD, Langford JC. A global geometric framework for nonlinear dimensionality reduction. Science. 2000;290(5500):2319–23.
43. Roweis ST, Saul LLK. Nonlinear dimensionality reduction by locally linear embedding. Science. 2000;290(5500):2323–6.
44. Wattenberg M. How to use t-SNE effectively. https://distill.pub/2016/misread-tsne. 2016.
45. Maaten LVD. Learning a parametric embedding by preserving local structure. In: Proc. AI-STATS. 2009.
46. Maaten LVD. Accelerating t-SNE using tree-based algorithms. JMLR. 2014;15:3221–45.
47. Pezzotti N, Höllt T, Lelieveldt B, Eisemann E, Vilanova A. Hierarchical stochastic neighbor embedding. Comput Gr Forum. 2016;35(3):21–30.
48. Pezzotti N, Lelieveldt B, Maaten LVD, Höllt T, Eisemann E, Vilanova A. Approximated and user steerable t-SNE for progressive visual analytics. IEEE TVCG. 2017;23:1739–52.
49. Pezzotti N, Thijssen J, Mordvintsev A, Hollt T, Lew BV, Lelieveldt B, Eisemann E, Vilanova A. GPGPU linear complexity t-SNE optimization. IEEE TVCG. 2020;26(1):1172–81.
50. Chan D, Rao R, Huang F, Canny J. T-SNE-CUDA: GPU-accelerated t-SNE and its applications to modern data. In: Proc. SBAC-PAD. 2018. p. 330–338.

51. Modrakowski TS, Espadoto M, Falcão AX, Hirata NST, Telea A. Improving deep learning projections by neighborhood analysis. Berlin: Springer; 2020.
52. Espadoto M, Hirata NS, Telea AC. Self-supervised dimensionality reduction with neural networks and pseudo-labeling. In: Proc. IVAPP. SCITEPRESS. 2021. p. 27–37.
53. Hunter JD. Matplotlib: a 2d graphics environment. Comput Sci Eng. 2007;9(3):90–5.
54. Xiao H, Rasul K, Vollgraf R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747. 2017.
55. Anguita D, Ghio A, Oneto L, Parra X, Reyes-Ortiz J.L. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In: Proc. intl. workshop on ambient assisted living. Springer. 2012. p. 216–223.
56. LeCun Y, Cortes C. MNIST handwritten digits dataset. 2010. http://yann.lecun.com/exdb/mnist.
57. Thoma M. The reuters dataset. 2017. https://martin-thoma.com/nlp-reuters.
58. Salton G, McGill MJ. Introduction to modern information retrieval. New York: McGraw-Hill; 1986.
59. Kruskal JB. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika. 1964;29(1):1–27.
60. Paulovich FV, Silva CT, Nonato LG. Two-phase mapping for projecting massive datasets. IEEE TVCG. 2010;16(6):1281–90.
61. Paulovich FV, Minghim R. Text map explorer: a tool to create and explore document maps. In: Proc. intl. conference on information visualisation (IV). IEEE. 2006. p. 245–251.
62. Vernier EF, Comba J, Telea A. Quantitative comparison of dynamic treemaps for software evolution visualization. In: Proc. IEEE VISSOFT. 2018.
63. Vernier E, Sondag M, Comba J, Speckmann B, Telea A, Verbeek K. Quantitative comparison of time-dependent treemaps. Comput Gr Forum. 2020;39(3):393–404.
64. The Authors: SDBM Implementation. 2021. https://github.com/mespadoto/sdbm.
65. Chollet F. Keras. 2015. https://keras.io
66. Rahaman M, Li C, Yao Y, Kulwa F, Rahman MA, Wang Q, Qi S, Kong F, Zhu X, Zhao X. Identification of COVID-19 samples from chest X-ray images using deep learning: a comparison of transfer learning approaches. J X-Ray Sci Technol. 2020;28(5):821–39.
67. Chen H, Li C, Wang G, Li X, Rahaman M, Sun H, Hu W, Li Y, Liu W, Sun C, Ai S, Grzegorzek M. GasHis-transformer: a multi-scale visual transformer approach for gastric histopathological image detection. Pattern Recogn. 2022;130: 108827.
68. Liu W, Li C, Xu N, Jiang T, Rahaman M, Sun H, Wu X, Hu W, Chen H, Sun C, Yao Y, Grzegorzek M. CVM-Cervix: a hybrid cervical Pap-smear image classification framework using CNN, visual transformer and multilayer perceptron. Pattern Recogn. 2022;130: 108829.
69. Zhang J, Li C, Kosov S, Grzegorzek M, Shirahamad K, Jiang T, Sun C, Li Z, Li H. LCU-Net: a novel low-cost U-Net for environmental microorganism image segmentation. Pattern Recogn. 2021;115: 107885.
70. Rahaman M, Li C, Yao Y, Kulwa F, Wu X, Li X, Wang Q. DeepCervix: a deep learning-based framework for the classification of cervical cells using hybrid deep feature fusion techniques. Comput Biol Med. 2021;136: 104649.
71. Saltelli A, Ratto M, Andres T, Campolongo F, Cariboni J, Gatelli D, Saisana M, Tarantola S. Global sensitivity analysis: the primer. New York: Wiley; 2008.