Machine Learning (2023) 112:1–43
https://doi.org/10.1007/s10994-022-06261-1

SLISEMAP: supervised dimensionality reduction through local explanations

Anton Björklund[1] Jarmo Mäkelä[1] Kai Puolamäki[1]

Received: 15 December 2021 / Revised: 13 July 2022 / Accepted: 29 September 2022 /
Published online: 23 November 2022

Abstract
Existing methods for explaining black box learning models often focus on building local explanations of the models' behaviour for particular data items. It is possible to create global explanations for all data items, but these explanations generally have low fidelity for complex black box models. We propose a new supervised manifold visualisation method, SLISEMAP, that simultaneously finds local explanations for all data items and builds a (typically) two-dimensional global visualisation of the black box model such that data items with similar local explanations are projected nearby. We provide a mathematical derivation of our problem and an open source implementation implemented using the GPU-optimised PyTorch library. We compare SLISEMAP to multiple popular dimensionality reduction methods and find that SLISEMAP is able to utilise labelled data to create embeddings with consistent local white box models. We also compare SLISEMAP to other model-agnostic local explanation methods and show that SLISEMAP provides comparable explanations and that the visualisations can give a broader understanding of black box regression and classification models.

Keywords Manifold visualisation · Explainable AI · Local approximation

1 Introduction

In the past 20 years, manifold visualisation methods are a major development in unsupervised learning. The trend that started from ISOMAP in 2000 (Tenenbaum et al., 2000) has resulted in hundreds of methods to be developed, popular examples of which include

Editors: Krzysztof Dembczynski and Emilie Devijver.

✉ Anton Björklund
anton.bjorklund@helsinki.fi

Jarmo Mäkelä
jarmo.makela@helsinki.fi

Kai Puolamäki
kai.puolamaki@helsinki.fi

[1] University of Helsinki, Helsinki, Finland

methods such as t-SNE (van der Maaten & Hinton, 2008) and UMAP (McInnes et al., 2020). Manifold visualisation methods can be used to embed data into typically two or three dimensions while preserving some of the relevant features of the data. These methods have proven to be invaluable and central to exploring and understanding complex datasets in fields from genetics (Kobak & Berens, 2019; Diaz-Papkovich et al., 2021) to astronomy (Anders et al., 2018) and linguistics (Levine et al., 2020).

Another recent development is explainable artificial intelligence (XAI), where the objective is to understand and explore *black box* supervised learning algorithms; see Guidotti et al. (2019) for a recent survey. The explanation methods can roughly be divided into *global* and *local* methods. Global methods try to explain the global behaviour of a supervised learning method by constructing a global understandable (*white box*) surrogate model that approximates the complex black box model. The drawback of the global approach is that for a sufficiently complex model, there is no simple surrogate model that would replicate the whole model with reasonable fidelity.

The alternative is local explanations focusing on how individual data items are classified or regressed. The advantage is that it is often possible to give high-fidelity, interpretable local explanations. The obvious disadvantage is that an explanation that is good for one data item may be useless for the other data items. A common model-agnostic approach for local explanations is to locally approximate the black box model with an interpretable white box model. These white box models are used to better understand the decision process by, e.g., showing which variables affect the outcome and how to achieve a different outcome.

In this paper, we combine the above two developments, namely, manifold visualisations and local explanations, to obtain *global* supervised manifold visualisations of the space of *local* explanations by using outputs of various black box supervised learning algorithms. We call the algorithm SLISEMAP.

The idea of SLISEMAP is straightforward: we want to find an embedding of data points into a (typically) two-dimensional plane such that the same interpretable model explains the supervised learning model of the data points nearby in the embedding. The embedding of the data points and the local models associated with each point in the embedding form a global explanation of the supervised learning model as a combination of the local explanations. At the same time, our method produces a visualisation of the data where the data points that are being classified (or regressed) with the same rules are shown nearby.

Example 1 First, consider a toy regression example where we have 99 data points composed of 4-dimensional covariates represented by rows of matrix $\mathbf{X} \in \mathbb{R}^{99\times 4}$ and a pretrained black box regression model given by function $f : \mathbb{R}^4 \to \mathbb{R}$, which we want to study. The response vector $\mathbf{y} \in \mathbb{R}^{99}$ is given by the regression estimates as $\mathbf{y}_i = f(\mathbf{X}_{i\cdot})$, where $\mathbf{X}_{i\cdot}$ denotes the ith row of the matrix $\mathbf{X}$. Unknown to the user, the elements of matrix $\mathbf{X}$ have been sampled at random from a normal distribution with zero mean and unit variance, and the regression function f is given by $f(\mathbf{x}) = \max \mathbf{x}_{1:3} = \max_{j\in\{1,2,3\}} \mathbf{x}_j$, where $\mathbf{x} \in \mathbb{R}^4$. In other words, the regression utilises the first three attributes in a nonlinear manner while ignoring the fourth attribute altogether.

Now, assume the user wishes to study the black box regression function and the dataset by embedding this 4-dimensional toy dataset into two dimensions. Any dimensionality

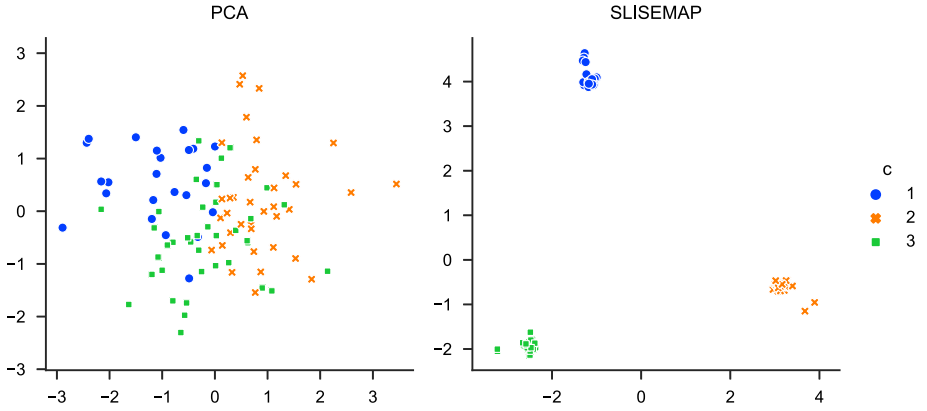

Fig. 1 PCA (left) and SLISEMAP (right) embeddings of a toy dataset described in the text. The toy data matrix consists of 4-dimensional Gaussian noise in $\mathbf{X} \in \mathbb{R}^{99\times 4}$, and the response vector $\mathbf{y} \in \mathbb{R}^{99}$ comes from a black box model $f(\mathbf{x}) = \max \mathbf{x}_{1:3}$. The legend in the plots corresponds to the value of $\mathbf{c}_i = \arg\max_{j\in\{1,2,3\}} \mathbf{X}_{ij}$. We have added some jitter to the SLISEMAP embeddings to make the points in the clusters stand out

reduction method that only take the covariate matrix $\mathbf{X}$ into account, and ignore the response variables in $\mathbf{y}$, would see only Gaussian noise; resulting in a limited insight about the data and the regression function, as shown in the PCA visualisation of Fig. 1 (left).

Then, consider a variant of SLISEMAP, where ordinary least squares linear regression is used as an interpretable white box model. SLISEMAP will produce an embedding where the data are split into three clusters indexed by $\mathbf{c}_i = \arg\max_{j\in\{1,2,3\}} \mathbf{X}_{ij}$, as shown in Fig. 1 (right). Each of the clusters corresponds to a different white box model denoted by $g_i : \mathbb{R}^4 \to \mathbb{R}$ for all $i \in \{1, \ldots, 99\}$ and are in this example simply given by $g_i(\mathbf{x}) = \mathbf{x}_{\mathbf{c}_i}$.

For these toy data, SLISEMAP can, therefore, partition the data into three clusters, each modelled locally to good accuracy by a separate linear white box model. The SLISEMAP embedding, together with the white box models, produces a global explanation of the black box model. The SLISEMAP embedding could help an analyst find "functional groups" of data points, with each group modelled by a simple linear model. This kind of "reverse-engineering" of the black box model can be instrumental in understanding how the black box model works and where it does not.

Uninformative directions in the data space, such as the 4th attribute in this example, are automatically ignored; SLISEMAP follows the possibly nonlinear manifold relevant for the supervised learning task. Note that in each of the three clusters in the SLISEMAP embedding, the value of the response variable $\mathbf{y}_i$ obeys an identical distribution: nearby points in the SLISEMAP embedding have similar white box models, not necessarily similar values of the response variables!

Example 2 A property of local explanations is that there may be several explanations, with roughly equally good fidelity, for any given data point. Consider the toy dataset described above, but let us assume that the user wants to add a new point $\mathbf{x}'$ where some of the maximal variables are identical, $\mathbf{x}'_i = \mathbf{x}'_j = \max \mathbf{x}'_{1:3}$, where $i,j \in \{1,2,3\}$ and $i \neq j$. This new point would fit equally well into both clusters i and j and, hence, has two potential local explanations. As shown later in the experiments, this also occurs with real datasets, and

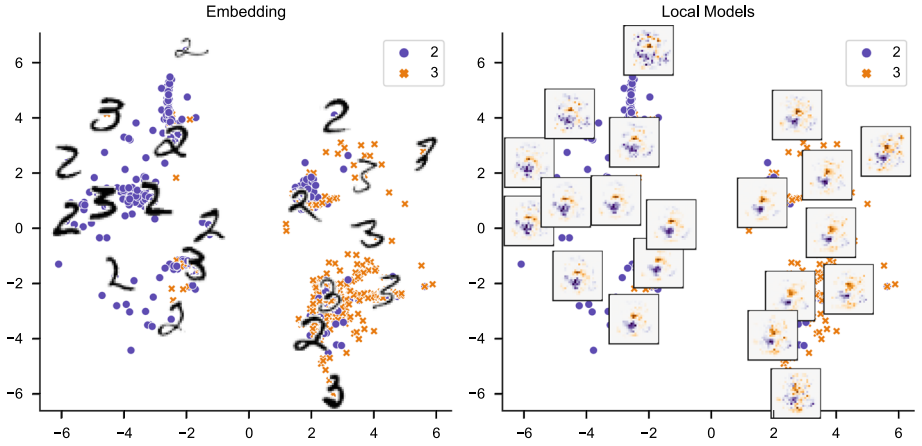

Fig. 2 SLISEMAP visualisation of 2:s and 3:s in the MNIST dataset with a black box deep learning classifier that tries to classify the digits into 2:s and 3:s. The left shows the embedding of the digits in two dimensions, with a random selection of digits shown as images. The white box models are logistic regressions that use the image pixels as attributes. The right shows the same embedding, but the images show the regression coefficients associated with each pixel for the same selection of digits. The colour intensity indicates the magnitude of the coefficient. We can see from the right image that nearby digits are described by similar white box models

SLISEMAP can be used to reveal this ambiguity, unlike more traditional local explanation methods that output only one white box model.

Example 3 As a more realistic and complex example, Fig. 2 shows the visualisation of a black box model that classifies 2 versus 3 in the classic MNIST (Lecun et al., 1998) dataset of hand-written digits. Here, the black box model is a convolutional neural network, and the white box model is a logistic regression classifier that takes the flattened image pixels as an input vector. The images are projected onto a two-dimensional plane such that the black box classifier for nearby images can, with reasonable fidelity, be approximated by the same logistic regression model. The digits are split into four visually separable clusters, with digits in each cluster classified by different sets of pixels. The logistic regression coefficients for different image pixels are shown in Fig. 2 (right). For example, the classifier separates the 2 s and 3 s at the bottom right mainly by identifying the lower curve in 3 s, while in the images on the left, the classifier is looking for black pixels below the centre. Seeing this visualisation enables us to find similar points in terms of the supervised learning problem and understand how the model classifies the data items.

The benefits of SLISEMAP compared to prior manifold visualisation or explanation methods include the following: (i) SLISEMAP finds visual patterns, like *clusters*, such that all data items within the same cluster are explained by the same simple model. For example, in Fig. 1 SLISEMAP reveals three clusters, while Fig. 2 shows roughly four clusters of digits that can be separated by a given subset of pixels. (ii) Unlike existing local explanation methods, SLISEMAP provides both *global* and *local explanations* of the data. For example, Fig. 2 compactly shows the explanations for all digits, in addition to the fact that roughly four linear models are sufficient to explain the classification of all digits to a reasonable fidelity. (iii)

SLISEMAP can be used to discover a nonlinear structure in a dataset, as shown in Fig. 2 and later in Sect. 4.4.

1.1 Contributions

The contributions of this paper are as follows: (i) We define a criterion for a supervised manifold embedding that shows local explanations and give an efficient algorithm to find such embeddings. (ii) We show experimentally that our method results in informative and useful visualisations, and local white box models can be used to explain and understand supervised learning models. (iii) We compare our contribution to manifold visualisation methods and local explanation methods.

2 Related work

This section briefly reviews the explainable AI and dimensionality reduction methods.

2.1 Explainable artificial intelligence

Explainable AI aims to provide insights into how black box (machine learning) models operate through various kinds of explanations. The explanations are used to analyse the black box models (Lapuschkin et al., 2019), when checking the results, e.g., looking for bias and systematic misclassifications (Selvaraju et al., 2020), and to discover issues and deficiencies in the data (Ribeiro et al., 2016). Furthermore, explanations can sometimes be a legal requirement (Goodman & Flaxman, 2017) or be used as a tool for facilitating AI and human collaboration (Samek et al., 2019).

The explanations of black box models can be generally divided into the exploration of global aspects, i.e., the entire model (Baehrens et al., 2010; Henelius et al., 2014, 2017; Adler et al., 2018; Datta et al., 2016), or inspection of local attributes, i.e., individual decisions (Ribeiro et al., 2016, 2018; Fong & Vedaldi, 2017; Lundberg & Lee, 2017); See Guidotti et al. (2019) for a recent survey and references. On a global level, the scope of the explanations is on understanding *how* the model has produced predictions, where the *why* is usually beyond human comprehension due to model complexity. On this level, we can examine which features affect the predictions most (Fisher et al., 2019) and what interactions there are between features (Goldstein et al., 2015; Henelius et al., 2014, 2017).

However, we are interested in local explanation methods, specifically those that can be used for any type of model (model-agnostic) and do not require any model modifications (post hoc). A common approach in this niche is to locally approximate the black box model with a simpler white box model. One of the first such methods, LIME (Ribeiro et al., 2016), generates interpretations for user-defined areas of interest by perturbing the data and training a linear model based on the predictions. Another similar method is SHAP (Lundberg & Lee, 2017), which finds the weights based on Shapley value estimation (Shapley, 1951). Nonlinear white box models, such as decision rules (Guidotti et al., 2018; Ribeiro et al., 2018), can also be used.

Many of these methods generate local explanations based on perturbed data, but designing a good data generation process is nontrivial (Guidotti et al., 2019; Laugel et al., 2018; Molnar, 2019), e.g., replacing pixels in an image with random noise seldom results in natural-looking images. One method that only utilises existing data is called SLISE (Björklund et al., 2019, 2022), which finds the largest subset of data items that can be

approximated (up to a given accuracy) by a sparse linear model. The work presented here can be seen as a global extension of SLISE.

2.2 Dimensionality reduction

Another way of assessing high-dimensional data is to reduce the number of covariates by, e.g., removing uninformative and redundant features or combining multiple features into single elements, thus making the data more interpretable. There are advantages of utilising dimensional reduction, as it removes correlated features in the data and allows for easier visualisation, e.g., in two dimensions. Still, combined features can also become less interpretable, and some information will inevitably be lost. The most straightforward dimensional reduction techniques are methods operating on the whole dataset by keeping the most dominant features with, e.g., backward elimination and forward selection, or by finding a combination of new features.

These methods include principal component analysis (PCA) and other linear methods (Cunningham & Ghahramani, 2015). Other approaches include locally linear embedding (LLE, MLLE) (Roweis & Saul, 2000; Zhang & Wang, 2006), spectral embedding (Belkin & Niyogi, 2003) and multidimensional scaling (MDS) (Kruskal, 1964), global-distance preserving MDS (Mead, 1992), ISOMAP (Tenenbaum et al., 2000), t-SNE (van der Maaten & Hinton, 2008), and UMAP (McInnes et al., 2020). Recently, some supervised methods have also become available, based on t-SNE (Kang et al., 2021; Hajderanj et al., 2019) and UMAP (McInnes et al., 2018).

There are some recent developments toward combining dimensionality reduction with explainable AI. Anbtawi (2019) presents an interactive tool which embeds data with standard t-SNE, and the user can examine the explanations of individual data items created by LIME. However, there are no interactions between t-SNE and LIME. Meanwhile, Bibal et al. (2020) use LIME to explain the t-SNE embedding, with no supervised learning method involved.

2.3 Local linear models

Local linear models, such as Nelles et al. (2000), estimate a response variable by fitting linear models to neighbourhoods of data items. Cheng and Wu (2013) improves the computational efficiency by using dimensionality reduction, after which they apply local linear models on the embedding. These methods use local models, similar to SLISEMAP. However, they are regression methods and do not produce visualisations or explanations.

3 Definitions and algorithms

3.1 Problem definition

A dataset consists of n data points $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$, where $\mathbf{x}_i \in \mathcal{X}$ are the *covariates* and $\mathbf{y}_i \in \mathcal{Y}$ are *responses* for one data point and $i \in [n] = \{1, \ldots, n\}$. $\mathcal{X}$ and $\mathcal{Y}$ are the domains of the covariates and responses, respectively. In this paper and in our software implementation, we restrict ourselves to real spaces, $\mathcal{X} = \mathbb{R}^m$ and $\mathcal{Y} = \mathbb{R}^p$, but the derivations in this subsection are general and would be valid, for example, for categorical variables as well.

The goal is to find a local *white box* model $g_i : \mathcal{X} \to \mathcal{Y}$ for every data point $(\mathbf{x}_i, \mathbf{y}_i)$, where $i \in [n]$. We use $\tilde{\mathbf{y}}_{ij} = g_i(\mathbf{x}_j)$ to denote the estimate of $\mathbf{y}_j$ obtained by a white box

model associated with data point $\mathbf{x}_i$. Again, while the derivation is general, in this paper, we focus on cases where the white box model, g_i, is either a linear projection (for regression problems) or multinomial logistic regression (for classification problems), as defined later in Sect. 3.3.

If we have access to a trained *black box* supervised learning algorithm $f : \mathcal{X} \to \mathcal{Y}$, then we can use estimates given by the model $\hat{\mathbf{y}}_i = f(\mathbf{x}_i)$ instead of $\mathbf{y}_i$. This will make the local models g_i local approximations of the black box model. These approximations can then also be used to explain the predictions of the black box model as in Björklund et al. (2019).

Additionally, we want to find a lower-dimensional embedding $\mathbf{Z}_{i\cdot}$ for every data point $i \in [n]$, where $\mathbf{Z}_{i\cdot}$ denotes the ith row of matrix $\mathbf{Z} \in \mathbb{R}^{n \times d}$. Our objective is that neighbouring data items in the embedding space have similar local models g_i. Since we focus on visualisation, in our examples, $\mathbf{Z}_{i\cdot}$ is typically 2-dimensional ($d = 2$).

We denote by $\mathbf{D}_{ij}$ the Euclidean distance between the points $\mathbf{Z}_{i\cdot}$ and $\mathbf{Z}_{j\cdot}$ in the embedding, where

$$\mathbf{D}_{ij} = \left(\sum\nolimits_{k=1}^{d} \left(\mathbf{Z}_{ik} - \mathbf{Z}_{jk} \right)^2 \right)^{1/2}. \tag{1}$$

We define the *soft neighbourhood* by using a softmax function as follows:

$$\mathbf{W}_{ij} = \frac{e^{-\mathbf{D}_{ij}}}{\sum_{k=1}^{n} e^{-\mathbf{D}_{ik}}}. \tag{2}$$

We define the *radius* of the d-dimensional embedding to be the square root of the variance of the embedding or

$$\text{radius}(\mathbf{Z}) = \left(\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{d} \mathbf{Z}_{ik}^2 \right)^{1/2}. \tag{3}$$

We further define a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ for the white box models. Here, we use the shorthand notation

$$\mathbf{L}_{ij} = l(\tilde{\mathbf{y}}_{ij}, \mathbf{y}_j) = l(g_i(\mathbf{x}_j), \mathbf{y}_j). \tag{4}$$

In this work, we use quadratic losses (for regression) and Hellinger distances between multinomial distributions (for classification), which we define later in Sect. 3.3.

The local white box model g_i can optionally have a regularisation term, which we denote by G_i. Since SLISEMAP consists of local models, regularisation can be important to handle small neighbourhoods. In this paper, we will use Lasso regularisation (Tibshirani, 1996) to be later defined in Eqs. (9) and (12) in Sect. 3.3.

Recall that the goal is that all points in the (soft) neighbourhood of point $\mathbf{Z}_{i\cdot}$ to be modelled well by the local white box model g_i. Mathematically, this can be formalised as minimising the following weighted loss:

$$\mathcal{L}_i = \sum\nolimits_{j=1}^{n} \mathbf{W}_{ij} \mathbf{L}_{ij} + G_i, \tag{5}$$

Each local model g_i has its own set of weights $\mathbf{W}_{i\cdot}$, of which $\mathbf{W}_{ii}$ is the largest (due to $\mathbf{D}_{ii} = 0$). This is what makes the models *local*. If the embedding, and therefore $\mathbf{W}$, is fixed, we can obtain the local models simply by minimising the loss of Eq. (5).

Our final loss function is obtained by summing all losses given by Eq. (5). We summarise everything in the main problem definition:

Problem 1 SLISEMAP Given dataset $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$, white box functions g_i and regularisation terms G_i for $i \in [n]$, loss function l, and the desired radius of the embedding $z_{radius} > 0$, find the parameters for $g_1, \ldots, g_n$ and embedding $\mathbf{Z} \in \mathbb{R}^{n \times d}$ that minimise the loss given by

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{W}_{ij} \mathbf{L}_{ij} + \sum_{i=1}^{n} G_i. \tag{6}$$

where $\mathbf{L}_{ij} = l(g_i(\mathbf{x}_j), \mathbf{y}_j)$, $\mathbf{W}_{ij} = e^{-\mathbf{D}_{ij}} / \sum_{k=1}^{n} e^{-\mathbf{D}_{ik}}$, and $\mathbf{D}_{ij} = (\sum_{k=1}^{d} (\mathbf{Z}_{ik} - \mathbf{Z}_{jk})^2)^{1/2}$, with the constraint that $\text{radius}(\mathbf{Z}) = z_{radius}$.

The loss function is invariant with respect to the rotation, which means that the embedding is invariant under rotation. The z_{radius} parameter essentially fixes the sizes of the neighbourhoods. At the limit of small z_{radius}, all points will be compressed close to the origin, and hence, all points will be described by the same local model. On the other hand, if z_{radius} is very large, the points are far away from each other, and the neighbourhood of each of the points consists only of the point itself.

3.2 Adding new data points to an existing solution

Often, it is useful to add new data points to an existing embedding without recomputing the whole embedding. Here, we define an auxiliary problem to this end.

Assume that we have a new data point denoted by $(\mathbf{x}_{n+1}, \mathbf{y}_{n+1})$. Define parameters for a new local model g_{n+1} and a new embedding matrix by $\mathbf{Z}' \in \mathbb{R}^{(n+1) \times d}$, such that the first n rows are the solution to Problem 1. We formulate the problem of adding a new point to an existing SLISEMAP solution as follows:

Problem 2 SLISEMAP-NEW Given the definitions above and a new data point $(\mathbf{x}_{n+1}, \mathbf{y}_{n+1})$, find the parameters for g_{n+1} and $\mathbf{Z}'_{n+1,\cdot} \in \mathbb{R}^d$ such that the loss of Eq. (6) is minimised; when g_{n+1} is added to the set of local models and $\mathbf{Z}$ is replaced by $\mathbf{Z}'$.

Solving Problem 2 is much easier than solving the full Problem 1 because in Problem 2, only the parameters for the new point need to be found, as opposed to the parameters for the n points in the full Problem 1. As a drawback, solving the full problem should result in slightly smaller loss. However, the difference should asymptotically vanish at the limit of large n. We study this difference experimentally in Sect. 4.6.

3.3 Slisemap for regression and classification

While the definitions in Sect. 3.1 were general, in this paper, we focus on regression and classification problems where the covariates are given by m-dimensional real vectors, or $\mathcal{X} = \mathbb{R}^m$. We denote the data matrix by $\mathbf{X} \in \mathbb{R}^{n \times m}$, where the rows correspond to the covariates or $\mathbf{X}_{i\cdot} = \mathbf{x}_i$. If necessary, we include in the data matrix a column of ones to account for the intercept terms.

Regression In regression problems, we use linear regression as the white box model. More specifically, we assume that the dependent variables are real numbers or $\mathcal{Y} = \mathbb{R}$. The white box regression model is given by a linear function

$$g_R(\mathbf{x}, \mathbf{b}) = \mathbf{x}^T \mathbf{b}, \tag{7}$$

where $\mathbf{b} \in \mathbb{R}^m$, and the loss is quadratic,

$$l_R(\tilde{\mathbf{y}}, \mathbf{y}) = (\tilde{\mathbf{y}} - \mathbf{y})^2. \tag{8}$$

The linear regression model g_R is parametrised by the vector $\mathbf{b} \in \mathbb{R}^m$. If we gather the parameter vectors from all the local models in Problem 1 into one matrix $\mathbf{B} \in \mathbb{R}^{n \times m}$ such that the row $\mathbf{B}_{i\cdot}$ gives the parameter vector of the local model g_i, then the parameters being optimised in Problem 1 are $\mathbf{B}$ and $\mathbf{Z}$.

We use Lasso regularisation, see Eq. (5), for any $i \in [n]$ given by

$$G_i^R = \lambda \times \sum_{j=1}^{m} \left|\mathbf{B}_{ij}\right|, \tag{9}$$

where λ is a parameter setting the strength of the regularisation. We can then write Eq. (6) to be optimised explicitly as $\mathcal{L}_R(\mathbf{X}, \mathbf{y}, \mathbf{B}, \mathbf{Z})$ with $\mathbf{L}_{ij} = \left((\mathbf{X}\mathbf{B}^T)_{ij} - \mathbf{y}_j\right)^2$.

Classification In classification problems, we assume that the black box classifier outputs class probabilities for p classes. We use multinomial logistic regression as the white box model. The dependent variables are multinomial probabilities in p-dimensional simplex or $\mathcal{Y} = \{\mathbf{y} \in \mathbb{R}^p_{\geq 0} \mid \sum_{i=1}^p \mathbf{y}_i = 1\}$. Multinomial logistic regression can be parametrised by $\mathbf{b} \in \mathbb{R}^{(p-1)m}$. The white box classification model is that of the multinomial logistic regression (Hastie et al., 2009),

$$\tilde{\mathbf{y}}_i = g_C(\mathbf{x}, \mathbf{b})_i = \begin{cases} \dfrac{\exp\left(\mathbf{x}^T \mathbf{b}_{((i-1)m+1):(im)}\right)}{1 + \sum_{j=1}^{p-1} \exp\left(\mathbf{x}^T \mathbf{b}_{((j-1)m+1):(jm)}\right)} & \text{if } i < p \\ \dfrac{1}{1 + \sum_{j=1}^{p-1} \exp\left(\mathbf{x}^T \mathbf{b}_{((j-1)m+1):(jm)}\right)} & \text{if } i = p \end{cases}, \tag{10}$$

We used $\mathbf{b}_{a:b}$ to denote an $(b - a + 1)$-dimensional vector $(\mathbf{b}_a, \mathbf{b}_{a+1}, \ldots, \mathbf{b}_b)^T$. When using g_C as the white box model in Problem 1, we can express the parameters for all the local models using a matrix $\mathbf{B} \in \mathbb{R}^{n \times (p-1)m}$, where the ith row $\mathbf{B}_{i\cdot}$ corresponds to the parameter vector of the ith data point.

The loss function could be any distance measure between multinomial probabilities, such as Kullback–Leibler (KL) divergence. Here, however, we choose the more numerically stable squared Hellinger distance (Ali & Silvey, 1966; Liese & Vajda, 2006),

$$l_C(\tilde{\mathbf{y}}, \mathbf{y}) = \frac{1}{2} \sum_{i=1}^{p} \left(\sqrt{\tilde{\mathbf{y}}_i} - \sqrt{\mathbf{y}_i}\right)^2 = 1 - \sum_{i=1}^{p} \sqrt{\tilde{\mathbf{y}}_i \mathbf{y}_i}. \tag{11}$$

The squared Hellinger distance is symmetric and bounded in interval [0, 1], unlike the KL, which is not symmetric or upper bounded. The squared Hellinger distance has convenient information-theoretic properties; for example, it is proportional to a tight lower bound for the KL divergence.

Note that when there are only two classes ($p = 2$), the multinomial logistic regression reduces to the standard logistic regression.

As in the regression formulation, we use Lasso regularisation for $i \in [n]$ given by

$$G_i^C = \lambda \times \sum_{j=1}^{(p-1)m} \left|\mathbf{B}_{ij}\right|. \tag{12}$$

where λ is a parameter setting the strength of the regularisation.

We can then write Eq. (6) to be explicitly optimised as $\mathcal{L}_C(\mathbf{X}, \mathbf{y}, \mathbf{B}, \mathbf{Z})$ with $\mathbf{L}_{ij} = l_C(g_{Ci}(\mathbf{x}_j), \mathbf{y}_j)$ expressed by using the Hellinger loss l_C of Eq. (11) and multinomial logistic regression g_C of Eq. (10).

Alternative formulation for binary classification In case the targets are given by a black box model, we can also use an alternative formulation for binary classification ($p = 2$). Here, we simply transform the probability $\hat{y}_1$ with a logit function, $\hat{y}'_1 = \log(\hat{y}_1/(1-\hat{y}_1))$, from the interval [0, 1] to the interval $[-\infty, \infty]$ and then run SLISEMAP for regression with quadratic loss, as above. Using a logit transformation followed by a linear model matches the behaviour of SHAP (Lundberg & Lee, 2017) and SLISE (Björklund et al., 2019).

3.4 Algorithm

Pseudocode for SLISEMAP is given in Algorithm 1. As the initial values for the embedding $\mathbf{Z}$, we use the principal component projection of the data (PCA). Then, we optimise the values of $\mathbf{B}$ and $\mathbf{Z}$ by minimising the loss given by Eq. (6).

Function `Slisemap`($\mathbf{X}$, $\mathbf{y}$, z_{radius}, d)
  `/* Use the` d `first principal components as the initial embedding */`
  $\mathbf{Z} \leftarrow \text{PCA}(\mathbf{X})_{\cdot,1:d}$
  $\mathbf{Z} \leftarrow \mathbf{Z}/\text{radius}(\mathbf{Z})$ `/* Normalise Z using Equation (3) */`
  `/* Optimise B (only) as the initial parameter matrix */`
  $\mathbf{B} \leftarrow \arg\min_{\mathbf{B}} [\mathscr{L}(\mathbf{X}, \mathbf{y}, \mathbf{B}, \mathbf{Z} \times z_{\text{radius}})]$
  do
    `/* Shuffle points to escape local minima */`
    $\mathbf{B}'_{i\cdot}, \mathbf{Z}'_{i\cdot} \leftarrow$ `Escape`$(\mathbf{X}_{i\cdot}, \mathbf{y}_i, \mathbf{B}, \mathbf{Z} \times z_{\text{radius}}/\text{radius}(\mathbf{Z}))$ for all $i \in [n]$
    $\mathbf{B}, \mathbf{Z} \leftarrow \mathbf{B}', \mathbf{Z}'$
    `/* Use L-BFGS to optimise Equation (6) */`
    $\mathbf{B}, \mathbf{Z} \leftarrow \arg\min_{\mathbf{B},\mathbf{Z}} \left[\mathscr{L}(\mathbf{X}, \mathbf{y}, \mathbf{B}, \mathbf{Z} \times z_{\text{radius}}/\text{radius}(\mathbf{Z})) + (\text{radius}(\mathbf{Z}) - 1)^2\right]$
  while *not converged*
  $\mathbf{Z} \leftarrow \mathbf{Z} \times z_{\text{radius}}/\text{radius}(\mathbf{Z})$ `/* Set the radius of Z using Equation (3) */`
  Result: $\mathbf{B}$, $\mathbf{Z}$

Function `Escape`($\mathbf{x}'$, $\mathbf{y}'$, $\mathbf{B}$, $\mathbf{Z}$)
  Compute the weight matrix $\mathbf{W}$ from $\mathbf{Z}$ by using Equations (1) and (2)
  $\mathbf{l}_i \leftarrow l(g_i(\mathbf{x}'), \mathbf{y}')$ for all $i \in [n]$ `/* Equation (4) */`
  $k \leftarrow \arg\min_k \sum_{j=1}^{n} \mathbf{W}_{kj}\mathbf{l}_j$
  Result: $\mathbf{B}_{k\cdot}, \mathbf{Z}_{k\cdot}$

Function `Slisemap-new`($\mathbf{x}_{new}$, $\mathbf{y}_{new}$, $\mathbf{X}_{old}$, $\mathbf{y}_{old}$, $\mathbf{B}_{old}$, $\mathbf{Z}_{old}$, z_{radius})
  $\mathbf{b}', \mathbf{z}' \leftarrow$ `Escape`$(\mathbf{x}_{new}, \mathbf{y}_{new}, \mathbf{B}_{old}, \mathbf{Z}_{old})$
  $\mathbf{b}_{new}, \mathbf{z}_{new} \leftarrow \arg\min_{\mathbf{b}',\mathbf{z}'} \mathscr{L}\left(\begin{bmatrix}\mathbf{X}_{old}\\ \mathbf{x}_{new}\end{bmatrix}, \begin{bmatrix}\mathbf{y}_{old}\\ \mathbf{y}_{new}\end{bmatrix}, \begin{bmatrix}\mathbf{B}_{old}\\ \mathbf{b}'\end{bmatrix}, \begin{bmatrix}\mathbf{Z}_{old}\\ \mathbf{z}'\end{bmatrix} \times z_{\text{radius}}/\text{radius}\left(\begin{bmatrix}\mathbf{Z}_{old}\\ \mathbf{z}'\end{bmatrix}\right)\right)$
  Result: $\mathbf{b}_{new}$, $\mathbf{z}_{new}$

Algorithm 1: The SLISEMAP algorithm, where $\mathscr{L}$ is given in Equation (6), $\mathbf{W}$ in Equation (2) and radius($\mathbf{Z}$) in Equation (3). See the text for discussion.

In our algorithm, we keep the radius of the embedding $\mathbf{Z}$ constant by always dividing it by $radius(\mathbf{Z})$ during the optimisation. Due to this normalisation, the loss term $\mathcal{L}()$ does not depend on the radius of $\mathbf{Z}$. Thus, for *numerical stability*, we add a small penalty term $(radius(\mathbf{Z}) - 1)^2$ to the loss (line 8 of Algorithm 1).

For the implementation of "arg min" in Algorithm 1, we use PyTorch (Paszke et al., 2019), which enables us to *optionally* take advantage of GPU acceleration. The optimisation of $\mathbf{B}$ and $\mathbf{Z}$ is performed using the L-BFGS (Nocedal, 1980) optimiser of PyTorch. As explained earlier, in this paper, we assume that the data are real valued and use the white box models and losses of Sect. 3.3 to study regression and classification problems.

In addition to the L-BFGS gradient search, we use an additional heuristic (function `Escape` in Algorithm 1) to help with escaping local optima. The heuristic consists of moving each item (embedding and local model) to the soft neighbourhood, given by $\mathbf{W}$ in Eq. (2), that have the most suitable local models. This process is repeated until no further improvement is found. We empirically validate the advantage of using the escape heuristic in Appendix B.

The pseudocode for Problem 2 (adding new data points to a SLISEMAP solution) is also given in Algorithm 1 (function `Slisemap-new`). Here, we use the same escape heuristic to find a suitable neighbourhood as a starting point and then optimise the embedding and local model for the new data item(s) with PyTorch and L-BFGS.

The source code, published under an open source MIT license, as well as the code needed to replicate all of the experiments in this paper, is available via GitHub (Björklund et al., 2022b).

3.5 Computational complexity

Evaluation of the loss function of Eq. (6) requires at least $O(n^2m)$ iterations for linear regression and $O(n^2mp)$ for multinomial logistic regression. Because, for every local model $O(n)$, the prediction and loss $O(mp)$ must be calculated for every data item $O(n)$. The calculation of the soft neighbourhoods requires $O(n^2d)$ (from calculating the Euclidean distances), but $d < mp$ in most circumstances.

However, this is an iterative algorithm, where Eq. (6) has to be evaluated multiple times. While it is difficult to provide strict running time limits for iterative optimisation algorithms such as L-BFGS—we study this experimentally in Sect. 4—it is obvious that the algorithm may not scale well for very large (n) datasets.

However, usually it is sufficient to subsample $\min(n, n_0)$ data points, where n_0 is a suitably chosen constant, optimise for the loss function (Problem 1), and then add points to the existing solution (Problem 2). By this procedure, the asymptotic complexity of SLISEMAP is linear with respect to the number of data points n. Especially for visualisation purposes, it often makes no sense to compute exact projection for a huge number of data points: visualisations cannot show more data points than there are pixels, so having an extremely accurate solution to the full optimisation problem instead of an approximate solution usually brings little additional benefit. Instead, finding a quick solution for sub-sampled data and adding the necessary number of data points to the embedding works well in practice, as shown in the experiments of Sect. 4.6.

4 Experiments

In the experiments, we usually embed the data into two dimensions ($d = 2$) and normalise data attributes, columns of the data matrix $\mathbf{X}$, to zero mean and unit variance as well as add an intercept term (column of ones) before running SLISEMAP. Furthermore, unless otherwise mentioned, we subsample the large datasets into 1000 data items and run all experiments ten times.

Most datasets have been used in two scenarios, first as normal regression or classification using the definitions from Sect. 3.3, and second in an XAI-inspired scenario where the targets are predictions from black box models, using the alternative formulation from Sect. 3.3 in the case of classification. When the white box model is a linear regression, we use $\lambda = 10^{-4}$ as the regularisation coefficient and $\lambda = 10^{-2}$ for logistic regression. An overview of the datasets and black box models can be seen in Table 1.

As explained earlier, we used PyTorch version 1.11 (Paszke et al., 2019). The runtime experiments were run on a server having an AMD Epyc processor at 2.4 GHz with 4 cores and 16 GB of memory allocated and an NVIDIA Tesla V100 GPU. The code to run the experiments is available via GitHub (Björklund et al., 2022b).

4.1 Datasets

In this section, we describe the datasets used in the experiments. The datasets and the black box models are available from OpenML (https://openml.org) (Vanschoren et al., 2014). A quick summary can be seen in Table 1.

Synthetic data We create synthetic *regression data* (RSYNTH) as follows: given parameters dataset size n (number of data items) and m (number data attributes), as well as $k = 3$ (number of clusters) and $s = 0.25$ (standard deviation of the clusters). We first sample $j \in [k]$ coefficient vectors $\beta_j \in \mathbb{R}^m$ from a normal distribution with zero mean and unit variance and cluster centroids $\mathbf{c}_j \in \mathbb{R}^m$ from a normal distribution with zero mean and standard deviation of s. We then create data items $i \in [n]$ by sampling the cluster index $j_i \in [k]$ uniformly and then generating a data vector $\mathbf{x}_i$ by sampling from a normal distribution with a mean of $\mathbf{c}_{j_i}$ and unit variance. The dependent variable is given by $\mathbf{y}_i = \mathbf{x}_i^T \beta_{j_i} + \epsilon_i$, where ϵ_i is Gaussian noise with zero mean and standard deviation of 0.1.

Air Quality data, cleaned and filtered as in Oikarinen et al. (2021), contains 7355 instances of 12 different air quality measurements, one of which is used as a dependent variable and the others as covariates.

Table 1 An overview of the datasets and black box models used in the experiments

Dataset	Size	Task	Black box model
RSYNTH	$n \times m$	Regression	–
Air quality	7355×11	Regression	Random forest
Boston	506×13	Regression	SVM
Spam	4601×57	Classification	Random forest
Higgs	$98\,049 \times 28$	Classification	Gradient boosting
Covertype	$581\,011 \times 54$	Classification	Logit boost
MNIST	$70\,000 \times 784$	Classification	Convolutional neural network

Boston Housing Dataset collected by the US census service from the Boston Standard Metropolitan Statistical Area in 1970. The size of the dataset is 506 items with 14 attributes, including the median value of owner-occupied homes that is used as the dependent variable.

Boston Housing Dataset collected by the US census service from the Boston Standard Metropolitan Statistical Area in 1970. The size of the dataset is 506 items with 14 attributes, including the median value of owner-occupied homes that is used as the dependent variable.

Spam (Cranor & LaMacchia, 1998) is a UCI dataset containing both spam, i.e., unsolicited commercial email, as well as professional and personal emails. There are 4601 instances with 57 attributes (mostly word frequencies) in the dataset.

Higgs (Baldi et al., 2014) is a UCI dataset containing 11 million simulated collision events for benchmarking classification algorithms. The dependent variable is whether a collision produces Higgs bosons. There are 28 attributes, the first 21 featuring kinematic properties measured by the particle detectors, and the last seven are functions of the first 21.

Covertype is a UCI dataset with over half a million instances, used to classify forest cover type (seven different types, but we only use the first two) from 54 attributes. The areas represent natural forests with minimal human-caused disturbances.

MNIST (Lecun et al., 1998) is the classic machine learning dataset of handwritten digits from 0 to 9. Each digit is represented by a 28×28 greyscale image (784 pixels with integer pixel values between 0 and 255). Due to the large number of pixels, we create a binary classification task by limiting the available digits to 2 and 3 and subsample them to 5000 data items.

4.2 Metrics

To compare different SLISEMAP solutions, we want to be able to objectively measure the performance. To accomplish that, we consider the following metrics.

Loss The most obvious thing to measure is the loss we are trying to minimise; see Eq. 6. However, the loss will change based on the parameters and the size of the dataset.

Cluster Purity For the synthetic dataset, we know the ground truth, which means that we can compare the original clusters to the embedding found by SLISEMAP. If we denote the true cluster id:s as $c_1, \ldots, c_n$, we can measure how well low-dimensional embeddings reconstruct the true clusters:

$$\frac{1}{n}\sum_{i=1}^{n} |\text{k-NN}(i) \cap \{j \mid c_i = c_j\}|/k, \tag{13}$$

where k-NN(i) is the set of k nearest neighbours (of item i) in the embedding space, using Euclidean distance, and $j \in [n]$. A larger value (closer to one) indicates that the dimensionality reduction has found the true clusters.

Fidelity The fidelity of a local model (Guidotti et al., 2019) measures how well it can predict the correct outcome. Using the losses defined in Sect. 3.3, we obtain:

$$\frac{1}{n}\sum_{i=1}^{n} l(g_i(\mathbf{x}_i), \mathbf{y}_i). \tag{14}$$

We are interested not only in how the local models perform on the corresponding data items but also in how well they work for the neighbours in the embedding space, using, e.g., the k nearest neighbours:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{k}\sum_{j\in \mathrm{k-NN(i)}} l(g_i(\mathbf{x}_j),\mathbf{y}_j). \tag{15}$$

A smaller value indicates better fidelity.

Coverage We also want local models that generalise to other data points. Otherwise, it would be trivial to find solutions. The coverage (Guidotti et al., 2019) of a local model can be measured by counting the number of data items that have a loss less than a threshold l_0:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{n}\sum_{j=1}^{n}(l(g_i(\mathbf{x}_j),\mathbf{y}_j)<l_0). \tag{16}$$

This requires us to select the loss threshold l_0. Unless otherwise mentioned, in this paper, we choose the threshold to be the 0.3 quantile of the losses of a global model (without the distance-based weights). Furthermore, we also want this behaviour to be reflected in the low-dimensional embedding. To verify this information, we limit the coverage testing to only the k nearest neighbours:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{k}\sum_{j\in \mathrm{k-NN(i)}}(l(g_i(\mathbf{x}_j),\mathbf{y}_j)<l_0). \tag{17}$$

A larger coverage value (closer to one) is better.

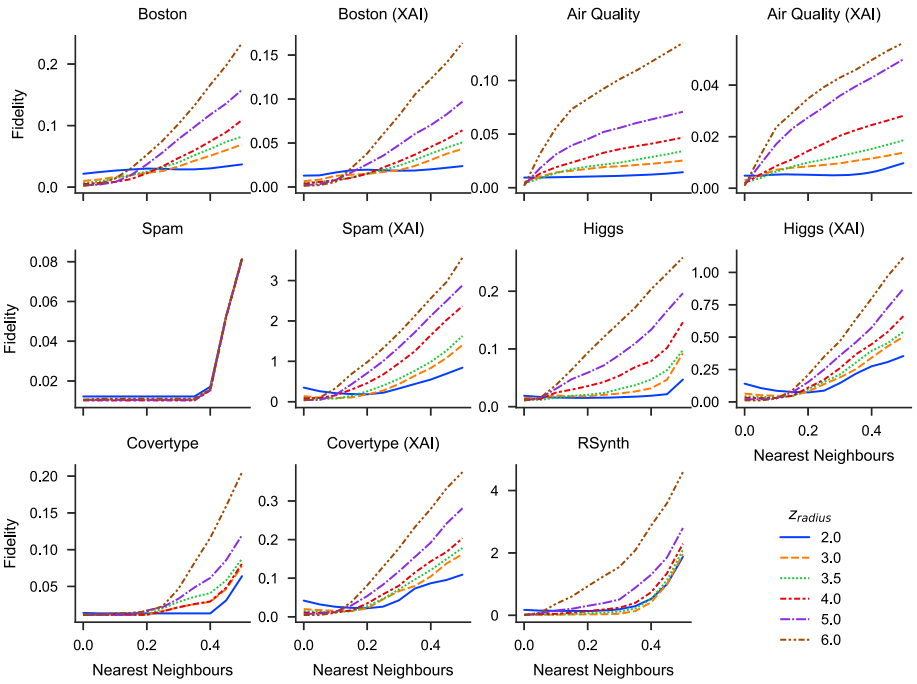

Fig. 3 Fidelity of the local models versus the fraction of nearest neighbours (in the fidelity calculation) for different values of z_{radius}. Smaller fidelity is better, especially for the nearest neighbours. Here, $3 \leq z_{\mathrm{radius}} \leq 4$ results in the best coverage

4.3 Parameter selection

SLISEMAP has one unusual parameter that needs to be selected: z_{radius}. If z_{radius} is too small, then all data items are in the same cluster, resulting in underfitting local models that are almost identical to the global model. However, if z_{radius} is too large, then the neighbourhoods become singular, which causes the local models to overfit.

In Fig. 3, we investigate how different values of z_{radius} affect the fidelity of the local models. Unless the local model is underfitting, the fidelity for the corresponding data item should be close to zero. Then, as the number of nearest neighbours grows, the fidelity should stay as low as possible for as long as possible to avoid overfitting. Based on these results, z_{radius} values from three to four seem to work well for all datasets.

We also consider how the coverage of the local models depends on the z_{radius}. The coverage plots can be seen in Appendix A and support the same conclusion as the fidelity results. Thus, we use $z_{\text{radius}} = 3.5$ as the default value for all the other experiments in this paper.

4.4 Visualisations of the datasets

While fidelity and coverage can be used for the quantitative analysis of z_{radius}, there is still room for a qualitative comparison to account for subjective preferences. In Fig. 4, we plot the low-dimensional embeddings for different z_{radius} values. At small values of z_{radius}, all points converge to the same cluster, as expected. With large values, the points form smaller and smaller clusters, potentially leading to overfitting. Based on Fig. 4, z_{radius} values between three and four seem optimal, which matches the conclusions from above.

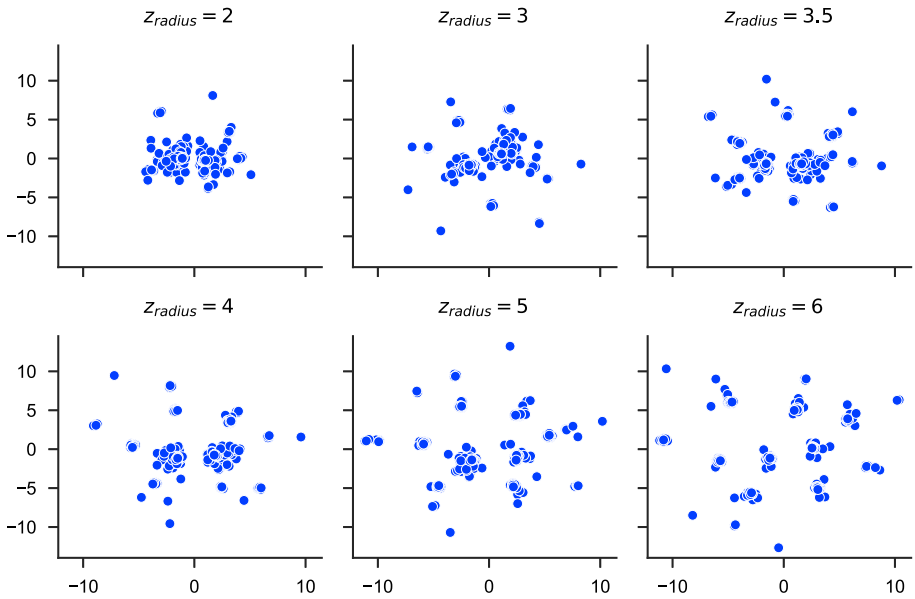

Fig. 4 The low-dimensional embedding of the BOSTON dataset with different values for z_{radius}. Large values (bottom right) lead to sparse solutions with small clusters and potentially overfit local models. Small values of z_{radius} (top left) create a single dense cluster in the centre, with all (non-)local models being almost identical

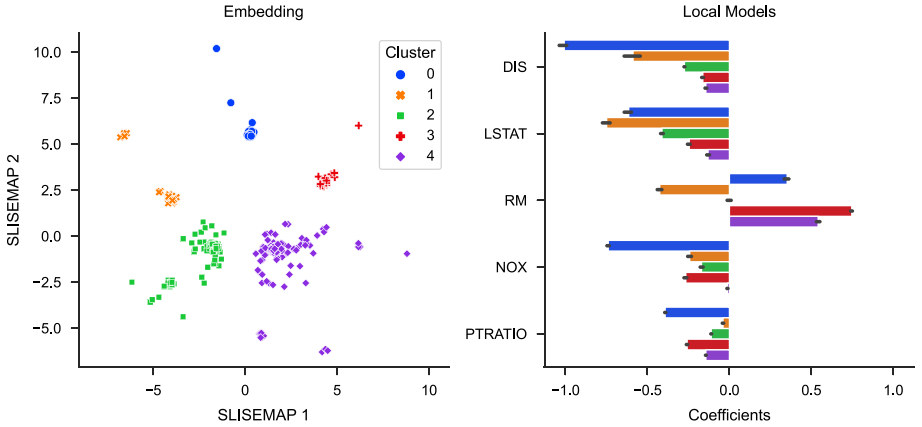

Fig. 5 Clustering the local models (right) to see if they correspond to clusters in the embedding (left). This also shows that the clusters in the embedding (left) have distinct local models (right). The barplot (right) only shows the five most important attributes of the BOSTON dataset

With SLISEMAP, we obtain not only an embedding but also local models for the data items. Data items that are nearby in the embedding space should have similar local models. We can verify this by clustering the local models independently of the embeddings and comparing these clusters to the structure in the embedding. Furthermore, models far apart in the embedding should look different due to the different local weights.

In Fig. 5, we cluster the coefficients of the local models using k-means clustering (on the BOSTON dataset). Here, we see that the clusters in the local models clearly match clusters in the embedding and that the clusters have different local models. Looking at the local models, we notice something curious: the number of rooms (RM) has a positive coefficient for most of the data, but in one cluster it is even negative! Investigating this cluster reveals that this cluster represents industrial locations with a high property tax (see density plots in Appendix C), making large homes less desirable. This insight could, e.g., be used in city planning and construction decisions.

A plot of the MNIST data is shown in Fig. 2 in the introduction, where we can see that some local models focus heavily on the bottom curve of 3:s, while others compare the differences between the pixels in the centre and the pixels just below the centre.

4.5 Uniqueness

In SLISEMAP, the embedding is influenced by the local models. Thus, if multiple local models are suitable for a particular data item, then the optimal embedding might be ambiguous. Some overlap between the local models is expected, and neighbouring (in the embedding) local models should be especially similar, due to the distance-based kernels in the loss function, Eq. 6. We also expect the hyperplanes of the local models to intersect, and any data items at these intersections will fit both models equally well.

In Fig. 6, we select seven data items from the BOSTON dataset and plot scatterplots of the embedding, where the colour of each dot represents how suitable that local model is for the selected data item. We see that not all local models suit all data items, i.e., the local models are actually local. Furthermore, neighbouring points tend to have the most suitable local models, as expected. However, some data items fit well into multiple neighbourhoods.

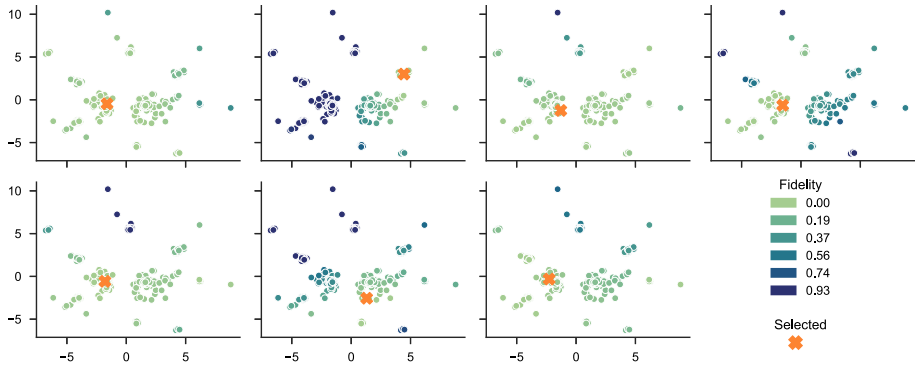

Fig. 6 A SLISEMAP embedding for the BOSTON dataset. The embedding is plotted seven times with different data items selected. The points in the embedding are coloured based on how well the corresponding local model fits the selected data item. Some data items only fit local models that are nearby in the embedding (the same neighbourhood), while some data items are more general

These data items with multiple potential neighbourhoods make the solutions non-unique, since there are multiple local optima with almost equally good losses. However, as shown in Fig. 5, the local models in the different neighbourhoods are different, and this is important for the data items in Fig. 6 matching only a single neighbourhood.

4.6 Subset sampling

With large datasets, the quadratic scaling of SLISEMAP, see Sect. 3.5, can become problematic. One solution is to run SLISEMAP on a random subset of the data, and then, post hoc, add unseen data items whenever necessary (see Sect. 3.2). With larger subsets, we expect better results, but with diminishing returns after the dataset is sufficiently covered.

To investigate how much data are needed, we randomly select 1000 data items from the large datasets to be unseen test data and train SLISEMAP solutions on increasing numbers of data items sampled from the remaining data. Then, we add the unseen data items, using

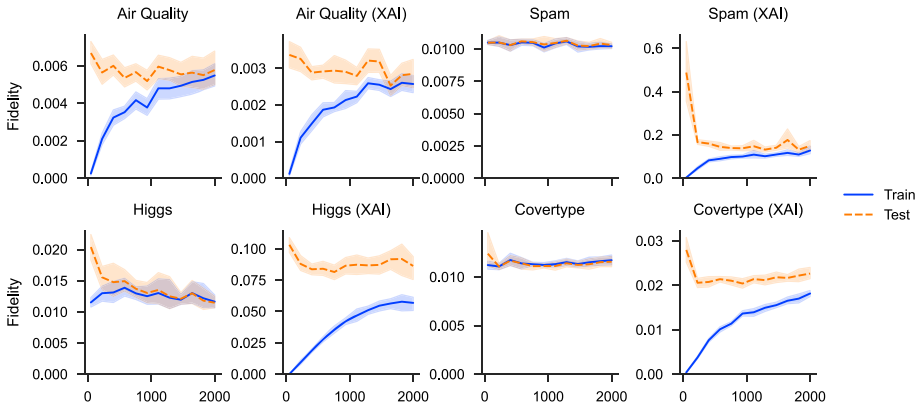

Fig. 7 Adding new data items to SLISEMAP solutions trained on subsampled datasets. With a sufficiently large training dataset, the fidelity of unseen test data matches that of the training data. For most of these datasets, only a couple of hundreds of initial data items are required. Lower fidelity for the test data is better

`Slisemap-new` from Algorithm 1. We repeat this process ten times for each dataset and compare the fidelity, Eq. (14), between the training data and the test data.

The results can be seen in Fig. 7. If the training dataset is too small, then the local models tend to overfit, but for most datasets, only a couple of hundred data items are needed to stabilise the results. This also coincides with the fidelities of the unseen test data approaching the fidelities of the training data.

4.7 Higher dimensional embeddings

In most experiments discussed in this paper, we use a two-dimensional embedding ($d = 2$). This is because a two-dimensional embedding is easy to visualise, which we consider to be an important use-case for the embedding. However, SLISEMAP is not limited to only two dimensions, which we demonstrate in this section.

Using the same fidelity and coverage metrics as in Sect. 4.3, we can find the best z_{radius} value for higher dimensional embeddings. In Fig. 8 and Appendix D, we demonstrate that the same default parameter value of $z_{\text{radius}} = 3.5$, which works well for two-dimensional embeddings, is also suitable for higher dimensions.

With two-dimensional embeddings, the intercluster distances are only independent for up to three clusters. This means that we expect higher dimensional embeddings to produce slightly lower losses if there are more than three clusters. In Fig. 9, we compare the losses for different numbers of dimensions. For some datasets, we indeed see minor improvements

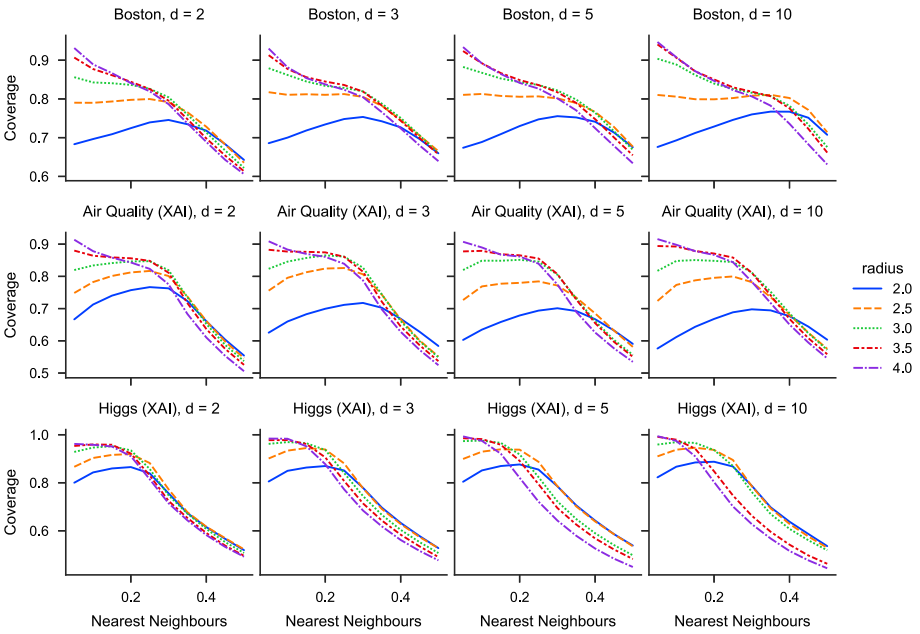

Fig. 8 Coverage of the local models versus the fraction of nearest neighbours (in the coverage calculation) for different values of z_{radius} and different numbers of embedding dimensions d. As the threshold for the coverage, we use the 0.3 quantile of the losses from a global model. Larger coverage is better, especially for the nearest neighbours. Here, $3 \leq z_{\text{radius}} \leq 3.5$ results in the best coverage, even for higher dimensional embeddings. The full plot is available in Appendix D

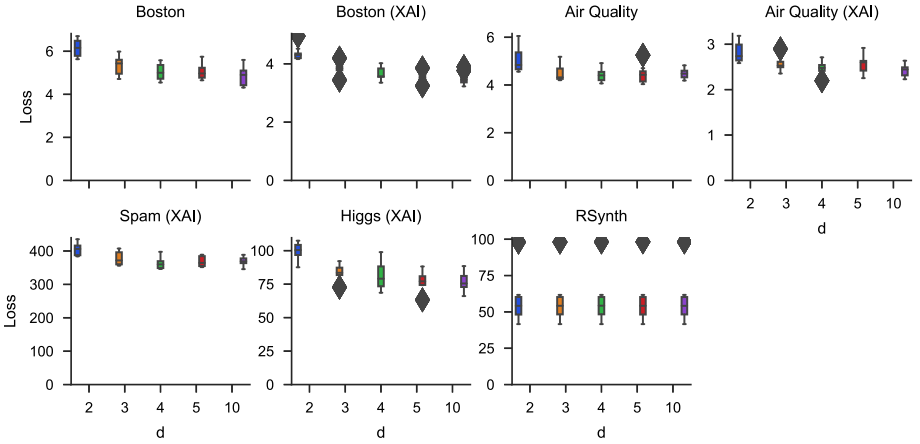

Fig. 9 Comparing losses for different numbers of embedding dimensions d. With higher-dimensional embeddings, we expect either minor improvements to the loss due to more flexible distances between multiple clusters or that the loss stays roughly the same

in the loss with increasing dimensionality. But, for example, in RSYNTH we know that there are only three clusters, so higher dimensional embeddings offer no advantage.

4.8 GPU acceleration

Since we implement SLISEMAP using PyTorch, the calculations can be accelerated using a GPU. Running SLISEMAP on a GPU should be faster than running on a CPU, especially for larger datasets. In Fig. 10, we apply SLISEMAP on RSYNTH datasets with different sizes, both with and without GPU acceleration. The GPU implementation has some overhead, making it slower for small datasets (less than 400×10) but substantially faster for larger datasets.

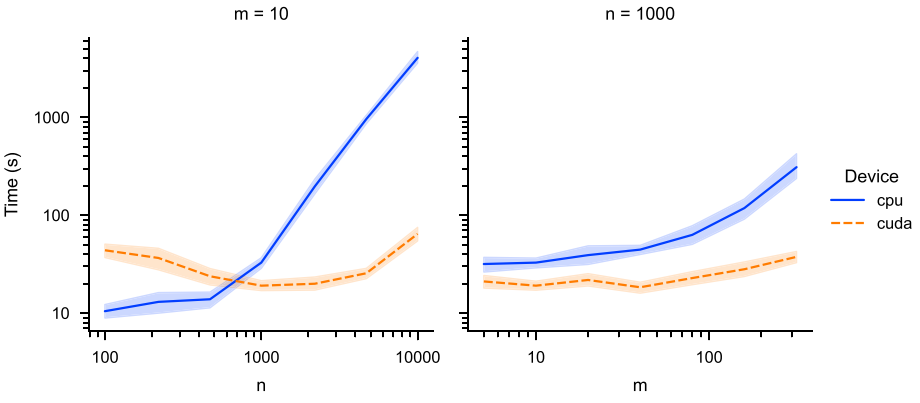

Fig. 10 Runtimes for different dataset sizes (using the RSYNTH dataset). GPU acceleration (cuda) brings some overhead but offers better parallelisation on large datasets. Note the logarithmic scale of the axis

4.9 Comparison to dimensionality reduction methods

The feature that differentiates SLISEMAP from other dimensionality reduction methods is that SLISEMAP provides both a low-dimensional embedding and local models. To demonstrate that doing this optimisation simultaneously is necessary, we take the embeddings from other dimensionality reduction methods and fit local models post hoc (essentially running SLISEMAP with a fixed $\mathbf{Z}$ given by the dimensionality reduction methods).

We use the following dimensionality reduction methods from the `Scikit-learn` package (Pedregosa et al., 2011) for the comparison: PCA, LLE (Roweis & Saul, 2000), MLLE (Zhang & Wang, 2006), MDS (Kruskal, 1964), ISOMAP (Tenenbaum et al., 2000), and t-SNE (van der Maaten, 2014). We also consider UMAP (McInnes et al., 2020).

A selection of the results can be seen in Table 2 (results for all datasets can be found in Appendix F). Since none of the other methods consider the relationship between $\mathbf{X}$ and $\mathbf{y}$ (most do not even use $\mathbf{y}$), their post hoc local models are, unsurprisingly, nonoptimal. However, the downside of using SLISEMAP is the additional time required for convergence. An empirical scalability comparison of the dimensionality reduction methods can be seen in Appendix E.

4.10 Comparison to local explanation methods

If we have access to a black box model, we can use SLISEMAP to find local and interpretable approximations of that black box model. In this section, we investigate how good the approximations are by checking both how local and how general the local models are. We also compare against other model-agnostic, local explanation methods. Furthermore, SLISEMAP finds all local models simultaneously, which could provide a speed benefit.

Of the local, model-agnostic, approximating explanations methods mentioned in Sect. 2, SLISEMAP is most closely related to SLISE Björklund et al. (2019). SLISE uses robust regression (Björklund et al., 2022) on data that have been centred on the selected data item to produce the local approximation. LIME (Ribeiro et al., 2016) creates a neighbourhood of synthetic data by mutating the selected data item (and using the black box model to obtain predictions). To increase interpretability LIME, normally, discretise continuous variables into binary variables (e.g., into quantiles). Then, LIME fits a least squares linear model to the synthetic neighbourhood to form the local approximation. SHAP (Lundberg & Lee, 2017) tries to estimate the Shapley value of keeping a variable in the selected data item versus changing it. This is conceptually quite similar to the discretisation in LIME. The model-agnostic variants of SHAP generally accomplishes this by creating variants of the selected data item where some of the variables are sampled from the dataset. These Shapley values are then used as the local approximation.

In addition to the methods outlined above, SLISEMAP, SLISE, SHAP, and LIME (with and without discretisation), we also consider a global model as a reference. The global models allow us to check that the local approximations are indeed local (better fidelity than the global model) and how general the approximations are (by comparing the coverage). As the threshold for measuring coverage as well as the *error tolerance* parameter in SLISE, we use the 0.3 quantile of the losses of the global model. The results can be seen in Table 3.

Table 2 Comparing SLISEMAP against other dimensionality reduction methods.

Dataset	Method	Loss	Fidelity	Fidelity NN	Coverage NN	Time (s)
	PCA	67.85 ± 7.32	0.05 ± 0.01	0.07 ± 0.01	0.33 ± 0.01	9.73 ± 7.47
	Spectral embedding	74.81 ± 8.34	0.06 ± 0.01	0.07 ± 0.01	0.33 ± 0.01	8.16 ± 7.63
	LLE	68.44 ± 7.17	0.05 ± 0.01	0.07 ± 0.01	0.33 ± 0.02	8.55 ± 3.20
	MLLE	71.51 ± 8.46	0.06 ± 0.01	0.07 ± 0.01	0.33 ± 0.01	7.90 ± 2.55
	MDS	68.08 ± 7.59	0.05 ± 0.01	0.07 ± 0.01	0.33 ± 0.02	22.88 ± 9.23
	Non-metric MDS	78.77 ± 8.18	0.06 ± 0.01	0.08 ± 0.01	0.30 ± 0.02	$\mathbf{4.78 \pm 1.29}$
	Isomap	67.83 ± 7.64	0.05 ± 0.01	0.07 ± 0.01	0.33 ± 0.02	7.81 ± 2.69
	t-SNE	74.71 ± 7.93	0.06 ± 0.01	0.07 ± 0.01	0.33 ± 0.01	10.00 ± 3.46
	UMAP	78.00 ± 8.22	0.07 ± 0.01	0.07 ± 0.01	0.32 ± 0.01	10.56 ± 1.66
	Supervised UMAP	81.43 ± 8.69	0.08 ± 0.01	0.08 ± 0.01	0.31 ± 0.01	$\mathbf{5.06 \pm 0.79}$
SPAM (XAI): 1000×57	Slisemap	$\mathbf{451.05 \pm 24.87}$	$\mathbf{0.10 \pm 0.02}$	$\mathbf{0.26 \pm 0.14}$	$\mathbf{0.91 \pm 0.02}$	172.56 ± 68.68
	PCA	2700.08 ± 100.82	1.65 ± 0.11	2.31 ± 0.11	0.36 ± 0.02	18.35 ± 4.87
	Spectral embedding	2522.60 ± 82.28	1.36 ± 0.06	2.07 ± 0.07	0.41 ± 0.02	17.48 ± 4.36
	LLE	3075.05 ± 233.23	2.20 ± 0.50	3.09 ± 0.41	0.33 ± 0.03	19.22 ± 5.55
	MLLE	3474.37 ± 112.08	3.15 ± 0.13	3.46 ± 0.33	0.29 ± 0.02	$\mathbf{15.66 \pm 7.36}$
	MDS	2401.03 ± 39.46	1.02 ± 0.04	2.11 ± 0.07	0.38 ± 0.02	77.63 ± 15.93
	Non-metric MDS	2926.17 ± 85.26	1.41 ± 0.07	2.70 ± 0.09	0.35 ± 0.01	24.51 ± 4.44
	Isomap	2559.64 ± 106.99	1.38 ± 0.12	2.17 ± 0.16	0.38 ± 0.02	24.55 ± 9.94
	t-SNE	2523.66 ± 64.02	1.22 ± 0.04	2.14 ± 0.07	0.39 ± 0.02	23.93 ± 4.61
	UMAP	3352.64 ± 190.68	2.40 ± 0.27	2.92 ± 0.32	0.32 ± 0.02	19.96 ± 3.51
	Supervised UMAP	3455.06 ± 200.77	2.63 ± 0.31	3.03 ± 0.23	0.31 ± 0.02	$\mathbf{14.03 \pm 2.70}$
RSYNTH: 400×15	Slisemap	$\mathbf{58.46 \pm 15.41}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.02 \pm 0.02}$	$\mathbf{1.00 \pm 0.00}$	11.99 ± 4.81
	PCA	2841.63 ± 739.62	3.23 ± 0.81	7.45 ± 1.97	0.37 ± 0.02	$\mathbf{0.33 \pm 0.03}$
	Spectral embedding	2921.21 ± 836.71	3.35 ± 1.02	7.62 ± 2.19	0.36 ± 0.02	0.39 ± 0.02
	LLE	3151.40 ± 933.08	4.47 ± 1.62	8.58 ± 2.41	0.33 ± 0.02	0.53 ± 0.06
	MLLE	3376.26 ± 823.38	7.04 ± 1.40	8.50 ± 2.03	0.33 ± 0.01	0.61 ± 0.14

Table 2 continued

Dataset	Method	Loss	Fidelity	Fidelity NN	Coverage NN	Time (s)
	MDS	2774.25 ± 758.80	2.89 ± 0.78	7.18 ± 2.01	0.37 ± 0.02	3.84 ± 0.96
	Non-metric MDS	3174.01 ± 844.25	3.54 ± 0.93	8.41 ± 2.14	0.33 ± 0.02	0.53 ± 0.02
	Isomap	2938.62 ± 860.11	3.36 ± 0.93	7.73 ± 2.36	0.36 ± 0.02	0.45 ± 0.04
	t-SNE	2987.75 ± 815.80	3.51 ± 0.90	7.71 ± 2.03	0.35 ± 0.03	1.19 ± 0.02
	UMAP	3773.83 ± 1086.93	8.31 ± 2.48	8.75 ± 2.54	0.32 ± 0.01	5.62 ± 0.20
	Supervised UMAP	3741.48 ± 1058.81	8.16 ± 2.42	8.66 ± 2.42	0.33 ± 0.01	2.28 ± 0.02

Here we use 20% as the number of nearest neighbours, and the running times are without GPU-acceleration. The best results are in bold. The full table is available in Appendix F
The embeddings $\mathbf{Z}$ are given by the dimensionality reduction methods while the local model coefficients $\mathbf{B}$ are optimised post-hoc (using the SLISEMAP loss)

Table 3 Comparison of the local white box models given by SLISEMAP, SLISE, SHAP, LIME, and LIME with no discretisation

Dataset	Method	Fidelity	Coverage	Time (s)
BOSTON (XAI) 404×13	Slisemap	0.00 ± 0.00	0.35 ± 0.01	17.85 ± 5.03
	SLISE	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.46 \pm 0.02}$	70.23 ± 2.16
	SHAP	$\mathbf{0.00 \pm 0.00}$	0.13 ± 0.01	167.58 ± 7.50
	LIME	0.26 ± 0.02	0.14 ± 0.01	1587.75 ± 31.26
	LIME (nd)	0.16 ± 0.01	0.21 ± 0.01	48.65 ± 1.09
	Global	0.11 ± 0.01	0.30 ± 0.00	$\mathbf{0.01 \pm 0.00}$
AIR QUALITY (XAI) 1000×11	Slisemap	0.00 ± 0.00	0.27 ± 0.01	110.64 ± 52.63
	SLISE	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.35 \pm 0.01}$	805.92 ± 12.71
	SHAP	0.01 ± 0.00	0.08 ± 0.00	438.18 ± 4.54
	LIME	0.23 ± 0.05	0.09 ± 0.00	3786.65 ± 53.35
	LIME (nd)	0.09 ± 0.01	0.26 ± 0.02	78.58 ± 0.50
	Global	0.08 ± 0.01	0.30 ± 0.00	$\mathbf{0.03 \pm 0.00}$
SPAM (XAI) 1000×57	Slisemap	0.16 ± 0.03	0.23 ± 0.01	277.70 ± 45.89
	SLISE	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.57 \pm 0.01}$	607.47 ± 64.91
	SHAP	$\mathbf{0.00 \pm 0.00}$	0.15 ± 0.01	1927.36 ± 9.44
	LIME	3.08 ± 0.21	0.20 ± 0.01	4482.63 ± 61.03
	LIME (nd)	9.56 ± 0.27	0.09 ± 0.01	205.86 ± 1.42
	Global	3.10 ± 0.12	0.30 ± 0.00	$\mathbf{0.32 \pm 0.15}$
HIGGS (XAI) 1000×28	Slisemap	0.05 ± 0.01	0.28 ± 0.01	238.95 ± 78.29
	SLISE	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.41 \pm 0.01}$	536.44 ± 75.22
	SHAP	$\mathbf{0.00 \pm 0.00}$	0.24 ± 0.01	492.45 ± 13.74
	LIME	0.85 ± 0.07	0.27 ± 0.01	8639.86 ± 167.58
	LIME (nd)	1.34 ± 0.05	0.27 ± 0.01	156.91 ± 2.85
	Global	1.19 ± 0.06	0.30 ± 0.00	$\mathbf{0.03 \pm 0.00}$

A global model is included as reference. The error tolerance for SLISE and the coverage is selected such that the global model has a coverage of 0.3, and the running times are without GPU-acceleration. Smaller fidelity and larger coverage are better, the best results are marked with bold

By definition, SLISE an SHAP have perfect fidelity for the data item corresponding to the local model, with SLISEMAP not far behind. The global model is obviously not local and, thus, should have the worst fidelity. However, there is nothing in the LIME procedure that ensures that the local approximation matches the selected data item. This results in the fidelity of LIME being comparable to the global model.

One of the advantages of SLISE is specifically optimising the subset size, which results in outstanding coverage. The local models in SLISEMAP are affected by the low-dimensional embedding. This reduced flexibility results in lower coverage than SLISE but better coverage than both LIME and SHAP. Both SHAP and LIME create synthetic neighbourhoods, which results in local models that are more difficult to generalise to real data items, reducing the coverage.

By computing all the local approximations at the same time, SLISEMAP tends to be faster than the methods doing it one-by-one, the exception being LIME with no discretisation. Furthermore, SLISEMAP also finds a low-dimensional embedding that can be used to visualise and compare different data items, different local approximations, and how they relate to each other.

5 Conclusions

In this paper, we present a novel supervised manifold embedding method, SLISEMAP, that embeds data items into a lower-dimensional space such that nearby data items are modelled by the same white box model. Therefore, in addition to reducing the dimensionality of the data, SLISEMAP creates a visualisation that can be used to globally explore and explain black box classification and regression models.

We show that the state-of-the-art dimensionality reduction methods, unsurprisingly, cannot be used to explain classifiers or regression models. On the other hand, the state-of-the-art tools used to explain black box models typically only provide local explanations for single examples, whereas SLISEMAP gives an overview of all local explanations.

Interesting future work would be to explore how SLISEMAP visualisations can be used to better understand data, both with and without a black box model, and to help build better models. For example, if a SLISEMAP visualisation could show that some group of data items should be handled differently. Future work could also explore how to use SLISEMAP to detect anomalous behaviours, such as outliers or concept drift. Finally, the scaling of SLISEMAP could be improved by, e.g., using stochastic optimisation or prototypes.

The source code for SLISEMAP, published under an open source MIT license, as well as the code needed to replicate all of the experiments in this paper, is available via GitHub (Björklund et al., 2022b).

Appendices

A Additional parameter selection results

In Fig. 11, we see how the coverage depends on the choice of value for z_{radius}. The ideal value would be one where the coverage starts high and stays high as the number of nearest neighbours grows. If the coverage starts small, then we are probably underfitting, and if the coverage quickly drops, then we are probably overfitting. A z_{radius} value between three and four seems to be a good choice, which supports the conclusions in Sect. 4.3.

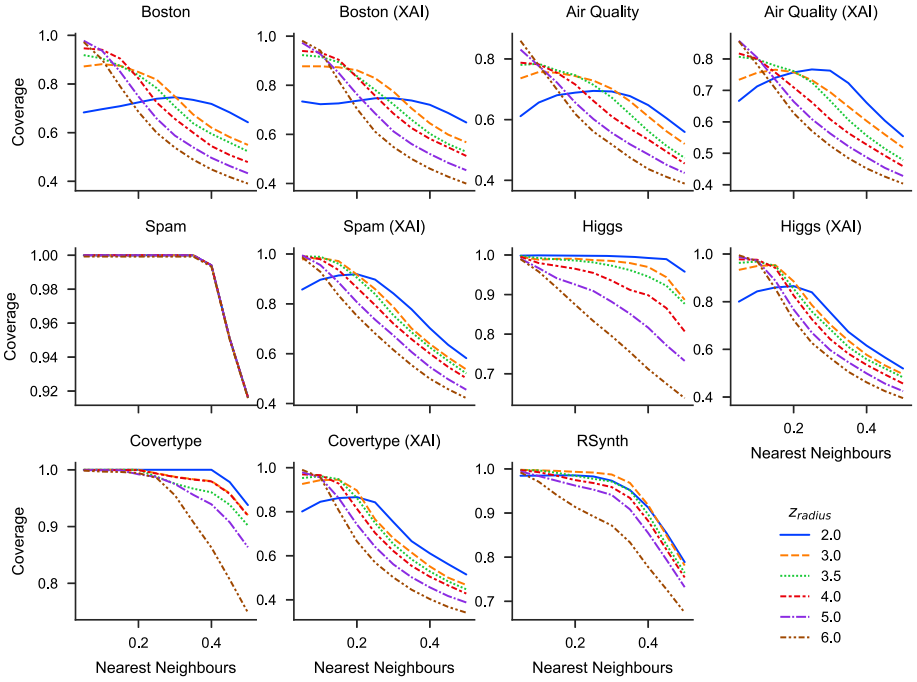

Fig. 11 Coverage of the local models versus the fraction of nearest neighbours (in the coverage calculation) for different values for z_{radius}. As the threshold for the coverage, we use the 0.3 quantile of the losses from a global model. Larger coverage is better, especially for the nearest neighbours. Here, $3 \leq z_{\text{radius}} \leq 4$ results in the best coverage

B Escape heuristic

In Sect. 3.4, we describe a "escape" heuristic that we use to avoid getting stuck in a local optimum, which should yield better solutions. In Table 4, we evaluate whether this is necessary. Using no heuristic would mean drastically faster running times. However, the solutions are nonoptimal compared to the full SLISEMAP solutions.

Table 4 Comparing SLISEMAP with and without the escape heuristic

Dataset	Method	Loss	Fidelity NN	Coverage NN	Cluster Purity	Time (s)
BOSTON: 404×13	Slisemap	$\mathbf{7.34 \pm 0.48}$	$\mathbf{0.03 \pm 0.01}$	$\mathbf{0.84 \pm 0.03}$		18.76 ± 5.30
	No escape	$\mathbf{7.65 \pm 0.55}$	0.05 ± 0.01	0.78 ± 0.04		$\mathbf{2.35 \pm 0.58}$
BOSTON (XAI): 404×13	Slisemap	$\mathbf{5.21 \pm 0.33}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.84 \pm 0.02}$		15.99 ± 4.17
	No escape	5.62 ± 0.53	0.03 ± 0.01	0.80 ± 0.03		$\mathbf{2.33 \pm 0.50}$
AIR QUALITY: 1000×11	Slisemap	$\mathbf{7.29 \pm 0.99}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.75 \pm 0.07}$		85.85 ± 22.77
	No escape	$\mathbf{7.38 \pm 0.75}$	0.03 ± 0.00	$\mathbf{0.73 \pm 0.06}$		$\mathbf{13.88 \pm 2.15}$
AIR QUALITY (XAI): 1000×11	Slisemap	$\mathbf{4.11 \pm 0.36}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.76 \pm 0.04}$		118.95 ± 43.55
	No escape	$\mathbf{4.28 \pm 0.35}$	0.01 ± 0.00	$\mathbf{0.75 \pm 0.05}$		$\mathbf{11.24 \pm 3.31}$
SPAM: 1000×57	Slisemap	$\mathbf{49.90 \pm 0.14}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$		102.27 ± 49.00
	No escape	70.84 ± 2.63	0.05 ± 0.01	0.95 ± 0.01		$\mathbf{38.91 \pm 2.10}$
SPAM (XAI): 1000×57	Slisemap	$\mathbf{451.05 \pm 24.87}$	$\mathbf{0.26 \pm 0.14}$	$\mathbf{0.91 \pm 0.02}$		172.56 ± 68.68
	No escape	597.57 ± 47.22	0.65 ± 0.20	0.86 ± 0.03		$\mathbf{16.48 \pm 3.36}$
HIGGS: 1000×28	Slisemap	$\mathbf{54.55 \pm 5.10}$	$\mathbf{0.02 \pm 0.01}$	$\mathbf{0.99 \pm 0.01}$		386.45 ± 182.21
	No escape	218.88 ± 85.72	0.20 ± 0.09	0.56 ± 0.28		$\mathbf{43.81 \pm 4.55}$
HIGGS (XAI): 1000×28	Slisemap	$\mathbf{113.98 \pm 8.94}$	$\mathbf{0.10 \pm 0.02}$	$\mathbf{0.85 \pm 0.02}$		185.68 ± 48.97
	No escape	188.48 ± 16.79	0.37 ± 0.05	0.72 ± 0.02		$\mathbf{12.89 \pm 2.04}$
COVERTYPE: 1000×54	Slisemap	$\mathbf{55.10 \pm 2.83}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$		110.65 ± 36.01
	No escape	66.83 ± 1.69	0.02 ± 0.00	0.98 ± 0.01		$\mathbf{37.82 \pm 2.25}$
COVERTYPE (XAI): 1000×54	Slisemap	$\mathbf{69.84 \pm 2.21}$	$\mathbf{0.03 \pm 0.00}$	$\mathbf{0.86 \pm 0.03}$		215.81 ± 73.36
	No escape	74.96 ± 5.25	0.06 ± 0.02	0.82 ± 0.05		$\mathbf{19.24 \pm 2.84}$
RSYNTH: 400×15	Slisemap	$\mathbf{58.46 \pm 15.41}$	$\mathbf{0.02 \pm 0.02}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.92 \pm 0.02}$	11.99 ± 4.81
	No escape	495.20 ± 79.90	2.53 ± 0.58	0.77 ± 0.02	0.38 ± 0.02	$\mathbf{2.80 \pm 0.25}$

Here we use 20% as the number of nearest neighbours, and the best results are in bold

Not using the heuristic would be substantially faster, but result in much worse solutions

C Density plots for the clusters

In Sect. 4.4, we qualitatively investigate a SLISEMAP solution for the BOSTON dataset. We find five clusters with different local models. To further study these clusters, we plot density plots for the clusters and variables in the dataset. The plots can be seen in Fig. 12. For example, we see that cluster 1 contains more industrial (INDUS) locations than average as well as better access to highways (RAD).

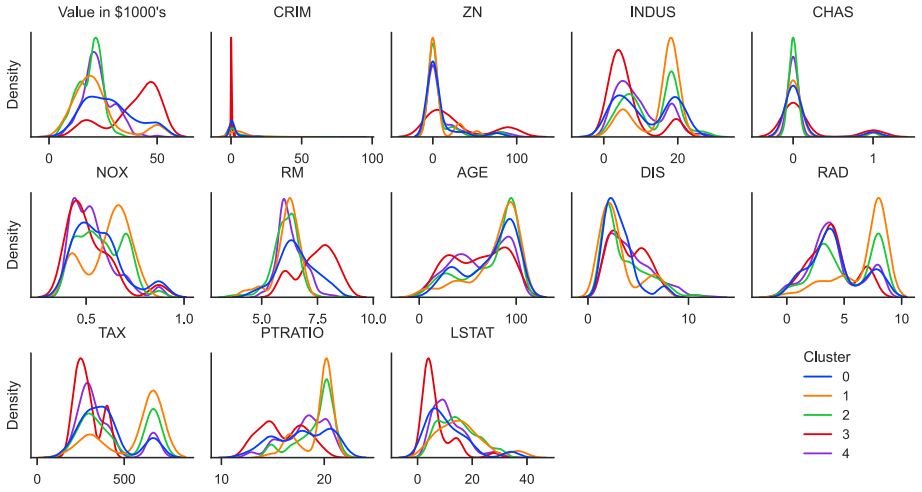

Fig. 12 Density plots for the BOSTON dataset with clusters from the local models given by SLISEMAP

D Higher-dimensional parameter selection results

When using SLISEMAP with embeddings of higher dimensions than two, in Sect. 4.7, we need to select new values for the parameter z_{radius}. For this, we employ the same procedure as in Sect. 4.3 and Appendix A. The results for the fidelity can be seen in Fig. 13, and the results for coverage can be seen in Fig. 14. These results support using $3.0 \leq z_{\text{radius}} \leq 3.5$ for all datasets and different numbers of embedding dimensions. Thus, we use the same default value, $z_{\text{radius}} = 3.5$, for higher dimensions as we do for two dimensions.

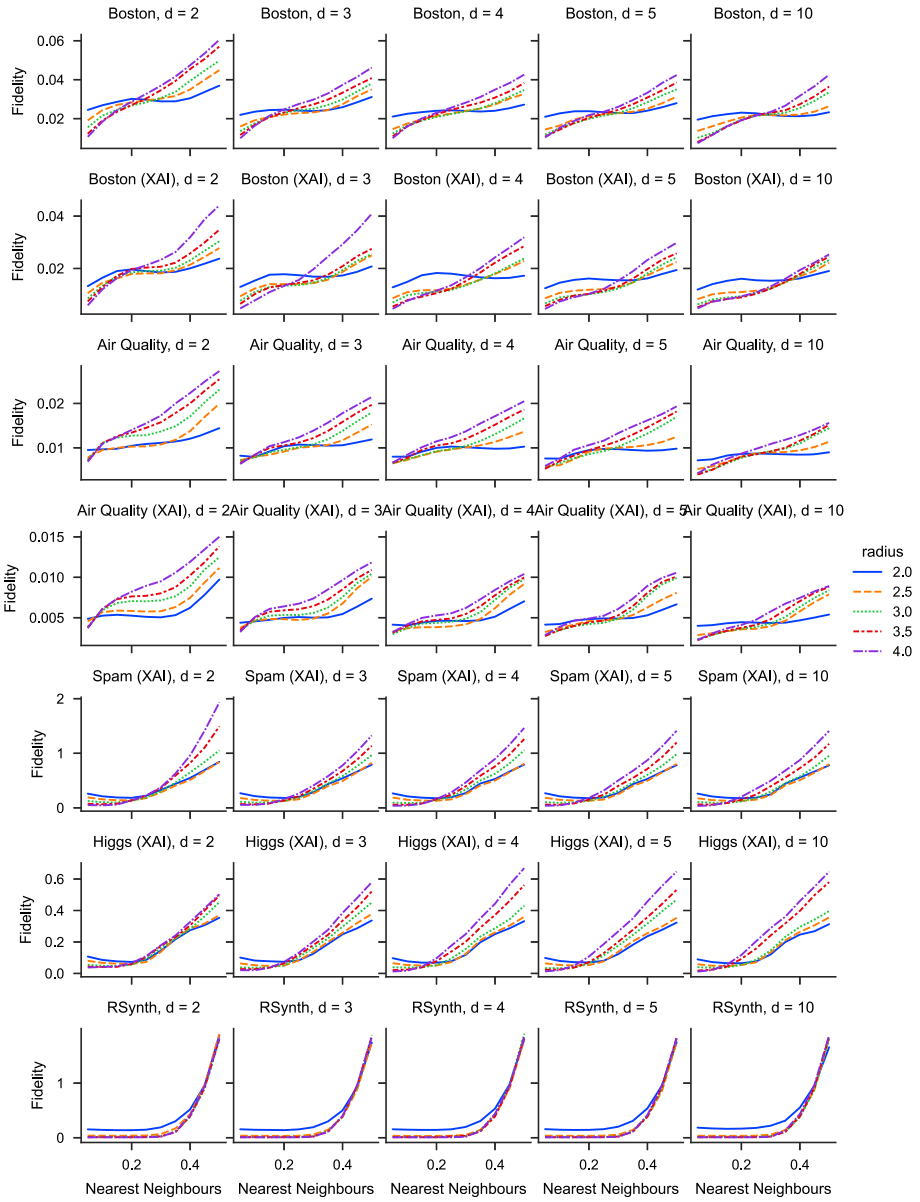

Fig. 13 Fidelity of the local models versus the fraction of nearest neighbours (in the fidelity calculation) for different values of z_{radius} and different numbers of embedding dimensions d. Smaller fidelity is better, especially for the nearest neighbours. Here, $3 \leq z_{\text{radius}} \leq 3.5$ results in the best fidelity, even for higher dimensional embeddings

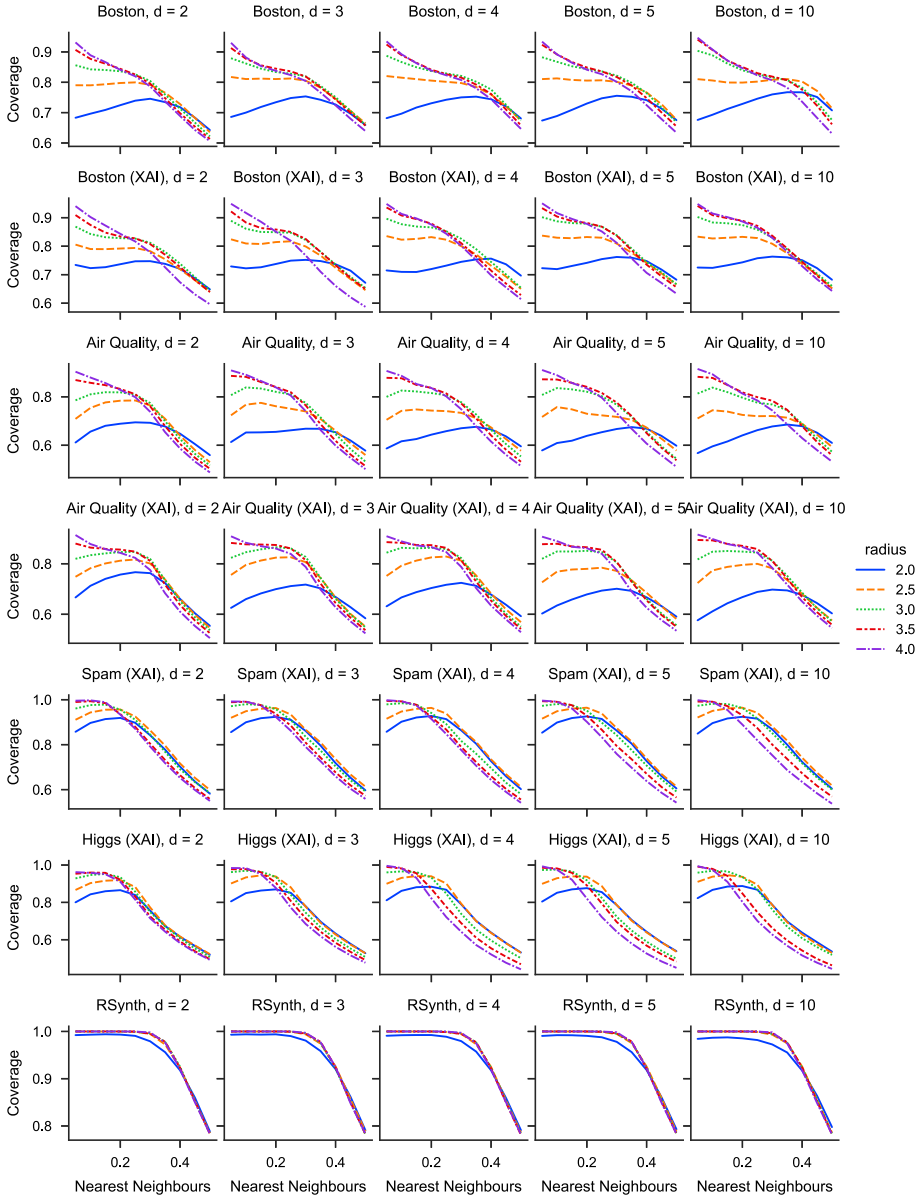

Fig. 14 Coverage of the local models versus the fraction of nearest neighbours (in the coverage calculation) for different values for z_{radius} and different numbers of embedding dimensions d. As the threshold for coverage, we use the 0.3 quantile of the losses from a global model. Larger coverage is better, especially for the nearest neighbours. Here, $3 \leq z_{\text{radius}} \leq 3.5$ results in the best coverage, even for higher dimensional embeddings

E Scaling dimensionality reduction methods

In Sect. 4.8 we compare how SLISEMAP scales with the size of the dataset when running on a GPU versus a CPU. In Fig. 15 we do the same but compare (CPU) SLISEMAP to other dimensionality reduction methods. Here we also use the RSYNTH dataset, and increase the number of data items (left) or the number of variables (right) respectively. Most methods scale superlinearly with the number of data items, except for UMAP and t-SNE that use stochastic updates (as a form subsampling). With increasing an increasing number of dimensions the methods based on distances have an almost constant scaling (beside the initial distance calculation). SLISEMAP is generally an order of magnitude slower than the other methods, but produce more than just a low-dimensional embedding (i.e. the local models).

F Additional dimensionality reduction results

In Sect. 4.9, we compare SLISEMAP to other dimensionality reduction methods by post hoc training local models on the embeddings. In this appendix are additional comparisons to more datasets. We also include spectral embedding (Belkin & Niyogi, 2003), nonmetric MDS (Kruskal, 1964), and supervised UMAP (McInnes et al., 2018) in the methods. In Table 5, we specifically investigate the synthetic dataset and find that SLISEMAP is the only method able to reconstruct the ground truth clusters. In Table 6 is the full results for all the real datasets. In conclusion, SLISEMAP is the slowest of the methods but also the only one to provide reasonable local models.

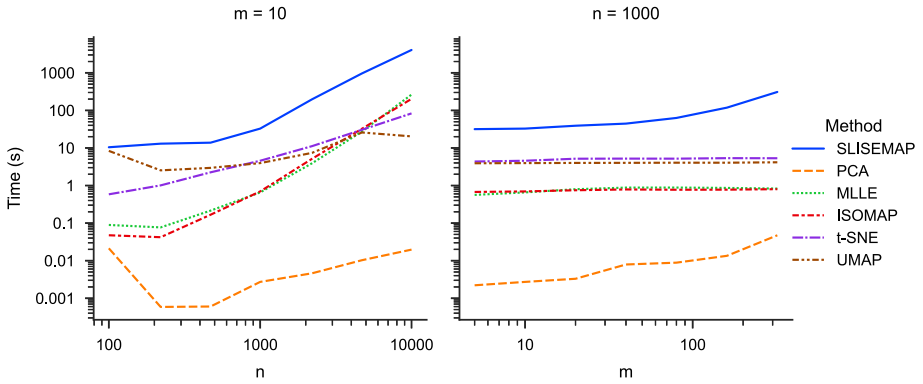

Fig. 15 The scalability of the different dimensionality reduction methods on the RSYNTH dataset is compared empirically. UMAP uses stochastic updates to handle an increasing number of data items (n), and methods that are based on distances (i.e., not PCA and SLISEMAP) are not really affected by increasing the number of variables (m). Note the logarithmic scales

Table 5 Comparing SLISEMAP against other dimensionality reduction methods on the synthetic dataset

Dataset	Method	Loss	Fidelity	Fidelity NN	Coverage NN	Cluster Purity	Time (s)
RSYNTH: 100 × 5	SLISEMAP	**9.04 ± 5.01**	**0.02 ± 0.02**	**0.09 ± 0.10**	**0.97 ± 0.05**	**0.80 ± 0.17**	4.57 ± 1.41
	PCA	270.07 ± 111.80	1.22 ± 0.51	2.99 ± 1.23	0.41 ± 0.06	0.38 ± 0.02	0.37 ± 0.03
	Spectral embedding	294.93 ± 124.35	1.42 ± 0.55	3.15 ± 1.45	0.41 ± 0.05	0.37 ± 0.02	0.35 ± 0.01
	LLE	314.58 ± 131.32	1.75 ± 0.80	3.53 ± 1.44	0.38 ± 0.05	0.37 ± 0.02	0.46 ± 0.05
	MLLE	330.45 ± 135.46	2.38 ± 0.90	3.78 ± 1.26	0.35 ± 0.05	0.37 ± 0.02	0.52 ± 0.09
	MDS	275.08 ± 113.00	1.10 ± 0.46	3.03 ± 1.26	0.41 ± 0.05	0.38 ± 0.02	1.70 ± 0.40
	Non-metric MDS	306.97 ± 134.73	1.08 ± 0.50	3.43 ± 1.54	0.39 ± 0.07	0.36 ± 0.02	**0.33 ± 0.01**
	Isomap	290.58 ± 121.49	1.32 ± 0.60	3.28 ± 1.41	0.40 ± 0.06	0.37 ± 0.02	0.43 ± 0.04
	t-SNE	303.27 ± 115.83	1.51 ± 0.56	3.15 ± 1.25	0.39 ± 0.05	0.37 ± 0.02	0.56 ± 0.01
	UMAP	397.93 ± 155.15	3.34 ± 1.32	3.67 ± 1.38	0.33 ± 0.04	0.37 ± 0.02	5.52 ± 0.09
	Supervised UMAP	406.15 ± 160.76	3.56 ± 1.46	3.80 ± 1.52	0.32 ± 0.02	0.37 ± 0.02	2.04 ± 0.02
RSYNTH: 200 × 10	SLISEMAP	**39.59 ± 21.69**	**0.04 ± 0.03**	**0.19 ± 0.26**	**0.96 ± 0.06**	**0.74 ± 0.18**	11.15 ± 2.92
	PCA	921.91 ± 249.50	1.82 ± 0.52	5.08 ± 1.43	0.34 ± 0.02	0.36 ± 0.02	0.37 ± 0.04
	Spectral embedding	982.53 ± 288.10	1.94 ± 0.61	5.29 ± 1.59	0.34 ± 0.02	0.37 ± 0.02	**0.35 ± 0.01**
	LLE	1070.14 ± 342.64	3.13 ± 1.20	5.97 ± 1.95	0.33 ± 0.02	0.34 ± 0.01	0.50 ± 0.04
	MLLE	1091.79 ± 309.72	4.62 ± 1.36	5.74 ± 1.58	0.32 ± 0.03	0.35 ± 0.00	0.53 ± 0.09
	MDS	928.48 ± 275.71	1.64 ± 0.51	4.96 ± 1.49	0.35 ± 0.02	0.37 ± 0.02	2.33 ± 0.61
	Non-metric MDS	1019.36 ± 285.04	1.76 ± 0.42	5.48 ± 1.52	0.32 ± 0.03	0.34 ± 0.00	0.42 ± 0.01
	Isomap	957.34 ± 263.46	1.86 ± 0.56	5.12 ± 1.33	0.34 ± 0.02	0.36 ± 0.02	0.42 ± 0.03
	t-SNE	989.12 ± 289.49	2.11 ± 0.63	5.16 ± 1.51	0.34 ± 0.02	0.35 ± 0.01	0.73 ± 0.01
	UMAP	1270.96 ± 371.85	5.46 ± 1.62	5.83 ± 1.73	0.32 ± 0.02	0.36 ± 0.01	5.81 ± 0.34
	Supervised UMAP	1281.29 ± 371.56	5.43 ± 1.48	5.92 ± 1.64	0.31 ± 0.02	0.36 ± 0.01	2.15 ± 0.06
RSYNTH: 400 × 15	SLISEMAP	**58.46 ± 15.41**	**0.01 ± 0.00**	**0.02 ± 0.02**	**1.00 ± 0.00**	**0.92 ± 0.02**	11.99 ± 4.81
	PCA	2841.63 ± 739.62	3.23 ± 0.81	7.45 ± 1.97	0.37 ± 0.02	0.40 ± 0.03	**0.33 ± 0.03**
	Spectral embedding	2921.21 ± 836.71	3.35 ± 1.02	7.62 ± 2.19	0.36 ± 0.02	0.40 ± 0.02	0.39 ± 0.02
	LLE	3151.40 ± 933.08	4.47 ± 1.62	8.58 ± 2.41	0.33 ± 0.02	0.35 ± 0.01	0.53 ± 0.06

Table 5 continued

Dataset	Method	Loss	Fidelity	Fidelity NN	Coverage NN	Cluster Purity	Time (s)
	MLLE	3376.26 ± 823.38	7.04 ± 1.40	8.50 ± 2.03	0.33 ± 0.01	0.35 ± 0.01	0.61 ± 0.14
	MDS	2774.25 ± 758.80	2.89 ± 0.78	7.18 ± 2.01	0.37 ± 0.02	0.40 ± 0.03	3.84 ± 0.96
	Non-metric MDS	3174.01 ± 844.25	3.54 ± 0.93	8.41 ± 2.14	0.33 ± 0.02	0.34 ± 0.00	0.53 ± 0.02
	Isomap	2938.62 ± 860.11	3.36 ± 0.93	7.73 ± 2.36	0.36 ± 0.02	0.39 ± 0.02	0.45 ± 0.04
	t-SNE	2987.75 ± 815.80	3.51 ± 0.90	7.71 ± 2.03	0.35 ± 0.03	0.37 ± 0.02	1.19 ± 0.02
	UMAP	3773.83 ± 1086.93	8.31 ± 2.48	8.75 ± 2.54	0.32 ± 0.01	0.37 ± 0.01	5.62 ± 0.20
	Supervised UMAP	3741.48 ± 1058.81	8.16 ± 2.42	8.66 ± 2.42	0.33 ± 0.01	0.38 ± 0.01	2.28 ± 0.02
RSYNTH: 800 × 20	SLISEMAP	**166.54 ± 37.91**	**0.01 ± 0.00**	**0.02 ± 0.03**	**1.00 ± 0.00**	**0.94 ± 0.02**	22.84 ± 9.56
	PCA	8298.25 ± 1542.92	5.46 ± 1.08	10.58 ± 2.00	0.38 ± 0.03	0.45 ± 0.04	**1.31 ± 0.94**
	Spectral embedding	8472.20 ± 1491.20	5.62 ± 1.03	10.77 ± 1.98	0.39 ± 0.03	0.45 ± 0.04	**1.10 ± 0.22**
	LLE	9589.32 ± 1613.10	7.26 ± 1.26	12.80 ± 2.21	0.31 ± 0.01	0.34 ± 0.00	1.50 ± 0.64
	MLLE	10487.73 ± 1975.77	11.96 ± 2.52	12.77 ± 2.25	0.31 ± 0.01	0.35 ± 0.01	1.43 ± 0.36
	MDS	8399.70 ± 1632.89	5.45 ± 1.15	10.64 ± 2.19	0.38 ± 0.03	0.43 ± 0.03	16.86 ± 5.92
	Non-metric MDS	10054.49 ± 1621.43	7.08 ± 1.13	13.13 ± 2.10	0.31 ± 0.01	0.34 ± 0.00	2.24 ± 0.28
	Isomap	9045.78 ± 1601.37	6.14 ± 1.10	11.69 ± 2.18	0.35 ± 0.03	0.42 ± 0.04	**1.28 ± 0.39**
	t-SNE	9198.38 ± 1625.66	6.44 ± 1.18	11.76 ± 2.08	0.35 ± 0.02	0.38 ± 0.03	3.69 ± 0.84
	UMAP	10941.08 ± 1855.82	12.04 ± 2.13	12.73 ± 2.24	0.31 ± 0.01	0.41 ± 0.03	8.32 ± 1.06
	Supervised UMAP	10829.11 ± 1857.73	11.60 ± 2.13	12.47 ± 2.21	0.32 ± 0.01	0.42 ± 0.03	4.39 ± 0.71
RSYNTH: 1000 × 25	SLISEMAP	**239.47 ± 36.94**	**0.01 ± 0.00**	**0.01 ± 0.00**	**1.00 ± 0.00**	**0.95 ± 0.01**	27.51 ± 4.06
	PCA	12141.25 ± 1919.31	6.32 ± 1.13	12.29 ± 1.95	0.40 ± 0.03	0.47 ± 0.02	**1.09 ± 0.14**
	Spectral embedding	12335.77 ± 2183.05	6.50 ± 1.30	12.50 ± 2.41	0.40 ± 0.04	0.46 ± 0.03	1.36 ± 0.15
	LLE	14291.92 ± 2287.89	8.87 ± 1.38	15.20 ± 2.42	0.32 ± 0.01	0.34 ± 0.00	2.11 ± 0.17
	MLLE	15422.12 ± 2321.97	14.00 ± 2.06	14.98 ± 2.20	0.32 ± 0.01	0.35 ± 0.01	2.02 ± 0.54
	MDS	12278.10 ± 1808.29	6.42 ± 0.95	12.34 ± 1.86	0.38 ± 0.03	0.42 ± 0.03	24.06 ± 4.50
	Non-metric MDS	14677.33 ± 2251.64	8.26 ± 1.26	15.34 ± 2.33	0.32 ± 0.01	0.34 ± 0.00	3.97 ± 2.36
	Isomap	13354.91 ± 2031.37	7.30 ± 1.27	13.80 ± 2.17	0.36 ± 0.03	0.41 ± 0.03	1.58 ± 0.12

Table 5 continued

Dataset	Method	Loss	Fidelity	Fidelity NN	Coverage NN	Cluster Purity	Time (s)
	t-SNE	12902.31 ± 1934.43	6.96 ± 1.05	13.12 ± 2.05	0.36 ± 0.03	0.40 ± 0.03	4.23 ± 0.55
	UMAP	16117.32 ± 2418.49	14.14 ± 2.23	14.91 ± 2.25	0.32 ± 0.01	0.42 ± 0.02	9.30 ± 2.04
	Supervised UMAP	16041.85 ± 2333.70	13.71 ± 1.91	14.69 ± 2.17	0.32 ± 0.01	0.43 ± 0.02	4.68 ± 0.35

Here we use 20% as the number of nearest neighbours, and the running times are without GPU-acceleration. Note that this datasets is constructed such that you need to utilise **y** in order to find the known clusters. The best results are highlighted with bold

The embeddings **Z** are given by the dimensionality reduction methods while the local model coefficients **B** are optimised post-hoc (using the SLISEMAP loss)

Table 6 Comparing SLISEMAP against other dimensionality reduction methods on real datasets.

Dataset	Method	Loss	Fidelity	Fidelity NN	Coverage NN	Time (s)
BOSTON: 404×13	SLISEMAP	$\mathbf{7.34 \pm 0.48}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.03 \pm 0.01}$	$\mathbf{0.84 \pm 0.03}$	18.76 ± 5.30
	PCA	58.24 ± 3.17	0.08 ± 0.00	0.13 ± 0.01	0.40 ± 0.02	1.64 ± 0.16
	Spectral embedding	59.97 ± 3.89	0.11 ± 0.01	0.14 ± 0.01	0.37 ± 0.02	1.04 ± 0.14
	LLE	63.25 ± 4.55	0.14 ± 0.01	0.16 ± 0.01	0.36 ± 0.02	0.71 ± 0.13
	MLLE	59.17 ± 5.08	0.13 ± 0.01	0.26 ± 0.10	0.37 ± 0.02	1.36 ± 0.33
	MDS	54.70 ± 3.46	0.07 ± 0.00	0.13 ± 0.01	0.41 ± 0.02	5.22 ± 0.78
	Non-metric MDS	86.09 ± 6.04	0.10 ± 0.01	0.22 ± 0.02	0.30 ± 0.01	$\mathbf{0.65 \pm 0.04}$
	Isomap	53.76 ± 3.31	0.09 ± 0.01	0.14 ± 0.01	0.38 ± 0.03	1.56 ± 0.26
	t-SNE	60.68 ± 3.83	0.09 ± 0.01	0.15 ± 0.01	0.39 ± 0.02	2.58 ± 0.30
	UMAP	66.06 ± 3.99	0.12 ± 0.01	0.15 ± 0.01	0.38 ± 0.03	6.54 ± 0.57
	Supervised UMAP	66.88 ± 4.41	0.13 ± 0.01	0.15 ± 0.01	0.39 ± 0.02	2.97 ± 0.30
BOSTON (XAI): 404×13	SLISEMAP	$\mathbf{5.21 \pm 0.33}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.84 \pm 0.02}$	15.99 ± 4.17
	PCA	33.39 ± 2.07	0.04 ± 0.00	0.07 ± 0.01	0.48 ± 0.03	1.38 ± 0.08
	Spectral embedding	40.28 ± 1.45	0.07 ± 0.00	0.09 ± 0.00	0.42 ± 0.02	1.08 ± 0.10
	LLE	47.78 ± 7.07	0.09 ± 0.02	0.12 ± 0.01	0.40 ± 0.02	0.80 ± 0.30
	MLLE	34.92 ± 4.15	0.07 ± 0.01	0.15 ± 0.04	0.41 ± 0.02	1.25 ± 0.19
	MDS	30.00 ± 1.22	0.03 ± 0.00	0.06 ± 0.00	0.50 ± 0.01	5.08 ± 1.11
	Non-metric MDS	69.88 ± 3.34	0.07 ± 0.00	0.18 ± 0.01	0.30 ± 0.01	$\mathbf{0.64 \pm 0.05}$
	Isomap	32.04 ± 2.08	0.04 ± 0.00	0.09 ± 0.01	0.46 ± 0.02	1.44 ± 0.25

Table 6 continued

Dataset	Method	Loss	Fidelity	Fidelity NN	Coverage NN	Time (s)
	t-SNE	40.15 ± 1.21	0.05 ± 0.00	0.09 ± 0.00	0.43 ± 0.01	2.34 ± 0.20
	UMAP	43.83 ± 3.94	0.07 ± 0.00	0.10 ± 0.01	0.43 ± 0.02	6.42 ± 0.40
	Supervised UMAP	45.10 ± 5.36	0.08 ± 0.01	0.10 ± 0.01	0.42 ± 0.02	2.79 ± 0.16
AIR QUALITY: 1000×11	SLISEMAP	$\mathbf{7.29 \pm 0.99}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.75 \pm 0.07}$	85.85 ± 22.77
	PCA	67.85 ± 7.32	0.05 ± 0.01	0.07 ± 0.01	0.33 ± 0.01	9.73 ± 7.47
	Spectral embedding	74.81 ± 8.34	0.06 ± 0.01	0.07 ± 0.01	0.33 ± 0.01	8.16 ± 7.63
	LLE	68.44 ± 7.17	0.05 ± 0.01	0.07 ± 0.01	0.33 ± 0.02	8.55 ± 3.20
	MLLE	71.51 ± 8.46	0.06 ± 0.01	0.07 ± 0.01	0.33 ± 0.01	7.90 ± 2.55
	MDS	68.08 ± 7.59	0.05 ± 0.01	0.07 ± 0.01	0.33 ± 0.02	22.88 ± 9.23
	Non-metric MDS	78.77 ± 8.18	0.06 ± 0.01	0.08 ± 0.01	0.30 ± 0.02	$\mathbf{4.78 \pm 1.29}$
	Isomap	67.83 ± 7.64	0.05 ± 0.01	0.07 ± 0.01	0.33 ± 0.02	7.81 ± 2.69
	t-SNE	74.71 ± 7.93	0.06 ± 0.01	0.07 ± 0.01	0.33 ± 0.01	10.00 ± 3.46
	UMAP	78.00 ± 8.22	0.07 ± 0.01	0.07 ± 0.01	0.32 ± 0.01	10.56 ± 1.66
	Supervised UMAP	81.43 ± 8.69	0.08 ± 0.01	0.08 ± 0.01	0.31 ± 0.01	$\mathbf{5.06 \pm 0.79}$
AIR QUALITY (XAI): 1000×11	SLISEMAP	$\mathbf{4.11 \pm 0.36}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.76 \pm 0.04}$	118.95 ± 43.55
	PCA	35.22 ± 3.63	0.03 ± 0.00	0.03 ± 0.00	0.35 ± 0.01	7.23 ± 1.26
	Spectral embedding	38.99 ± 4.66	0.03 ± 0.00	0.04 ± 0.00	0.35 ± 0.01	$\mathbf{5.63 \pm 1.20}$
	LLE	36.59 ± 4.19	0.03 ± 0.00	0.04 ± 0.00	0.34 ± 0.01	$\mathbf{6.55 \pm 2.07}$
	MLLE	37.62 ± 4.28	0.03 ± 0.00	0.04 ± 0.00	0.35 ± 0.01	7.67 ± 3.13

Table 6 continued

Dataset	Method	Loss	Fidelity	Fidelity NN	Coverage NN	Time (s)
	MDS	35.29 ± 3.72	0.03 ± 0.00	0.03 ± 0.00	0.36 ± 0.01	33.94 ± 12.78
	Non-metric MDS	41.21 ± 5.40	0.03 ± 0.00	0.04 ± 0.01	0.30 ± 0.01	7.34 ± 2.54
	Isomap	35.20 ± 3.68	0.03 ± 0.00	0.03 ± 0.00	0.35 ± 0.01	9.43 ± 2.52
	t-SNE	38.67 ± 4.58	0.03 ± 0.00	0.04 ± 0.00	0.35 ± 0.01	8.97 ± 2.78
	UMAP	41.23 ± 5.28	0.04 ± 0.00	0.04 ± 0.00	0.34 ± 0.01	11.03 ± 1.66
	Supervised UMAP	41.29 ± 5.34	0.04 ± 0.00	0.04 ± 0.00	0.33 ± 0.01	$\mathbf{6.01 \pm 0.83}$
SPAM: 1000×57	SLISEMAP	$\mathbf{49.90 \pm 0.14}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	102.27 ± 49.00
	PCA	224.82 ± 4.00	0.17 ± 0.01	0.19 ± 0.01	0.69 ± 0.02	52.99 ± 9.21
	Spectral embedding	207.47 ± 4.51	0.16 ± 0.00	0.18 ± 0.01	0.71 ± 0.02	56.67 ± 10.12
	LLE	273.74 ± 7.99	0.25 ± 0.02	0.27 ± 0.01	0.35 ± 0.17	$\mathbf{30.16 \pm 17.48}$
	MLLE	283.45 ± 1.53	0.28 ± 0.00	0.28 ± 0.00	0.15 ± 0.12	$\mathbf{19.29 \pm 18.12}$
	MDS	233.66 ± 2.77	0.17 ± 0.01	0.20 ± 0.01	0.63 ± 0.02	101.13 ± 41.29
	Non-metric MDS	284.04 ± 1.53	0.27 ± 0.00	0.28 ± 0.00	0.34 ± 0.02	$\mathbf{21.03 \pm 16.72}$
	Isomap	213.59 ± 4.86	0.16 ± 0.01	0.18 ± 0.01	0.69 ± 0.02	45.86 ± 13.07
	t-SNE	207.01 ± 5.09	0.15 ± 0.01	0.18 ± 0.01	0.68 ± 0.02	46.74 ± 6.92
	UMAP	256.87 ± 10.13	0.20 ± 0.01	0.22 ± 0.01	0.53 ± 0.06	49.26 ± 13.75
	Supervised UMAP	69.71 ± 23.99	0.02 ± 0.00	0.02 ± 0.01	0.99 ± 0.01	52.50 ± 13.28

Dataset	Method	Loss	Fidelity	Fidelity NN	Coverage NN	Time (s)
SPAM (XAI): 1000×57	SLISEMAP	$\mathbf{451.05 \pm 24.87}$	$\mathbf{0.10 \pm 0.02}$	$\mathbf{0.26 \pm 0.14}$	$\mathbf{0.91 \pm 0.02}$	172.56 ± 68.68
	PCA	2700.08 ± 100.82	1.65 ± 0.11	2.31 ± 0.11	0.36 ± 0.02	18.35 ± 4.87
	Spectral embedding	2522.60 ± 82.28	1.36 ± 0.06	2.07 ± 0.07	0.41 ± 0.02	17.48 ± 4.36
	LLE	3075.05 ± 233.23	2.20 ± 0.50	3.09 ± 0.41	0.33 ± 0.03	19.22 ± 5.55
	MLLE	3474.37 ± 112.08	3.15 ± 0.13	3.46 ± 0.33	0.29 ± 0.02	$\mathbf{15.66 \pm 7.36}$
	MDS	2401.03 ± 39.46	1.02 ± 0.04	2.11 ± 0.07	0.38 ± 0.02	77.63 ± 15.93
	Non-metric MDS	2926.17 ± 85.26	1.41 ± 0.07	2.70 ± 0.09	0.35 ± 0.01	24.51 ± 4.44
	Isomap	2559.64 ± 106.99	1.38 ± 0.12	2.17 ± 0.16	0.38 ± 0.02	24.55 ± 9.94
	t-SNE	2523.66 ± 64.02	1.22 ± 0.04	2.14 ± 0.07	0.39 ± 0.02	23.93 ± 4.61
	UMAP	3352.64 ± 190.68	2.40 ± 0.27	2.92 ± 0.32	0.32 ± 0.02	19.96 ± 3.51
	Supervised UMAP	3455.06 ± 200.77	2.63 ± 0.31	3.03 ± 0.23	0.31 ± 0.02	$\mathbf{14.03 \pm 2.70}$
HIGGS: 1000×28	SLISEMAP	$\mathbf{54.55 \pm 5.10}$	$\mathbf{0.01 \pm 0.00}$	0.02 ± 0.01	0.99 ± 0.01	386.45 ± 182.21
	PCA	272.96 ± 1.68	0.21 ± 0.00	0.26 ± 0.00	0.47 ± 0.04	$\mathbf{39.49 \pm 5.00}$
	Spectral embedding	273.49 ± 3.30	0.20 ± 0.01	0.26 ± 0.01	0.50 ± 0.05	$\mathbf{38.67 \pm 12.46}$
	LLE	274.42 ± 2.91	0.21 ± 0.01	0.27 ± 0.01	0.45 ± 0.07	$\mathbf{41.70 \pm 5.80}$
	MLLE	282.10 ± 2.94	0.26 ± 0.01	0.28 ± 0.01	0.31 ± 0.13	44.61 ± 8.69
	MDS	272.83 ± 2.30	0.20 ± 0.01	0.26 ± 0.00	0.50 ± 0.03	111.24 ± 26.22
	Non-metric MDS	277.48 ± 2.21	0.20 ± 0.01	0.26 ± 0.00	0.47 ± 0.03	47.12 ± 20.36
	Isomap	273.36 ± 3.29	0.21 ± 0.01	0.26 ± 0.01	0.49 ± 0.06	43.38 ± 12.98
	t-SNE	274.02 ± 3.54	0.20 ± 0.01	0.26 ± 0.01	0.50 ± 0.05	47.52 ± 7.16
	UMAP	287.73 ± 2.51	0.27 ± 0.01	0.28 ± 0.01	0.30 ± 0.11	$\mathbf{26.69 \pm 15.38}$

Table 6 continued

Dataset	Method	Loss	Fidelity	Fidelity NN	Coverage NN	Time (s)
	Supervised UMAP	74.12 ± 17.22	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	42.54 ± 9.91
HIGGS (XAI): 1000×28	SLISEMAP	$\mathbf{113.98 \pm 8.94}$	$\mathbf{0.04 \pm 0.00}$	$\mathbf{0.10 \pm 0.02}$	$\mathbf{0.85 \pm 0.02}$	185.68 ± 48.97
	PCA	742.42 ± 37.94	0.43 ± 0.02	0.73 ± 0.04	0.35 ± 0.01	5.68 ± 1.82
	Spectral embedding	744.08 ± 36.40	0.41 ± 0.02	0.76 ± 0.03	0.35 ± 0.01	$\mathbf{4.81 \pm 1.33}$
	LLE	852.65 ± 65.06	0.62 ± 0.08	0.89 ± 0.07	0.32 ± 0.02	$\mathbf{5.24 \pm 1.37}$
	MLLE	878.56 ± 49.85	0.77 ± 0.07	0.87 ± 0.05	0.32 ± 0.02	$\mathbf{4.79 \pm 0.84}$
	MDS	714.88 ± 33.53	0.37 ± 0.02	0.72 ± 0.03	0.36 ± 0.01	45.55 ± 9.82
	Non-metric MDS	847.28 ± 49.05	0.48 ± 0.03	0.88 ± 0.05	0.32 ± 0.01	$\mathbf{5.39 \pm 1.60}$
	Isomap	767.62 ± 41.30	0.46 ± 0.03	0.79 ± 0.04	0.34 ± 0.01	5.96 ± 1.04
	t-SNE	781.60 ± 42.49	0.44 ± 0.03	0.80 ± 0.04	0.34 ± 0.01	6.20 ± 0.93
	UMAP	946.18 ± 54.98	0.87 ± 0.05	0.90 ± 0.05	0.31 ± 0.00	10.28 ± 2.29
	Supervised UMAP	947.67 ± 55.75	0.88 ± 0.05	0.91 ± 0.05	0.31 ± 0.01	$\mathbf{5.14 \pm 1.77}$
COVERTYPE: 1000×54	SLISEMAP	$\mathbf{55.10 \pm 2.83}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	110.65 ± 36.01
	PCA	277.74 ± 1.67	0.25 ± 0.00	0.26 ± 0.00	0.55 ± 0.03	$\mathbf{49.00 \pm 17.60}$
	Spectral embedding	280.70 ± 3.26	0.25 ± 0.01	0.27 ± 0.00	0.47 ± 0.06	$\mathbf{53.53 \pm 11.50}$
	LLE	280.52 ± 3.03	0.26 ± 0.01	0.27 ± 0.01	0.47 ± 0.07	$\mathbf{55.00 \pm 11.14}$
	MLLE	283.49 ± 2.03	0.26 ± 0.01	0.27 ± 0.00	0.49 ± 0.09	$\mathbf{57.72 \pm 20.50}$
	MDS	277.15 ± 1.76	0.24 ± 0.00	0.27 ± 0.00	0.49 ± 0.03	140.04 ± 37.21
	Non-metric MDS	289.03 ± 1.16	0.27 ± 0.01	0.28 ± 0.00	0.34 ± 0.04	$\mathbf{47.23 \pm 27.84}$
	Isomap	275.04 ± 2.42	0.24 ± 0.00	0.27 ± 0.00	0.51 ± 0.04	$\mathbf{58.53 \pm 12.97}$
	t-SNE	271.09 ± 1.89	0.23 ± 0.00	0.27 ± 0.00	0.48 ± 0.04	61.10 ± 13.71

Table 6 continued

Dataset	Method	Loss	Fidelity	Fidelity NN	Coverage NN	Time (s)
	UMAP	274.85 ± 2.00	0.24 ± 0.00	0.27 ± 0.00	0.48 ± 0.05	$\mathbf{52.72 \pm 7.74}$
	Supervised UMAP	156.68 ± 28.05	0.06 ± 0.02	0.09 ± 0.03	0.91 ± 0.04	$\mathbf{53.95 \pm 8.53}$
COVERTYPE (XAI): 1000×54	SLISEMAP	$\mathbf{69.84 \pm 2.21}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.03 \pm 0.00}$	$\mathbf{0.86 \pm 0.03}$	215.81 ± 73.36
	PCA	324.37 ± 21.30	0.26 ± 0.02	0.28 ± 0.02	0.32 ± 0.01	$\mathbf{12.56 \pm 3.34}$
	Spectral embedding	321.61 ± 22.26	0.25 ± 0.02	0.30 ± 0.03	0.32 ± 0.01	$\mathbf{15.67 \pm 4.47}$
	LLE	336.89 ± 11.54	0.28 ± 0.01	0.34 ± 0.02	0.30 ± 0.01	$\mathbf{12.40 \pm 4.18}$
	MLLE	327.49 ± 19.67	0.27 ± 0.02	0.37 ± 0.18	0.31 ± 0.01	$\mathbf{14.94 \pm 5.54}$
	MDS	307.56 ± 16.67	0.21 ± 0.01	0.27 ± 0.02	0.33 ± 0.01	61.04 ± 10.45
	Non-metric MDS	321.57 ± 16.29	0.16 ± 0.01	0.29 ± 0.02	0.33 ± 0.01	19.10 ± 5.41
	Isomap	318.53 ± 18.38	0.24 ± 0.02	0.28 ± 0.02	0.32 ± 0.01	$\mathbf{13.26 \pm 3.56}$
	t-SNE	321.57 ± 16.86	0.23 ± 0.02	0.29 ± 0.02	0.32 ± 0.01	16.91 ± 4.30
	UMAP	329.05 ± 17.01	0.25 ± 0.02	0.29 ± 0.02	0.32 ± 0.01	23.32 ± 4.25
	Supervised UMAP	329.91 ± 19.46	0.26 ± 0.02	0.29 ± 0.02	0.32 ± 0.01	18.09 ± 4.70

Here we use 20% as the number of nearest neighbours, and the running times are without GPU-acceleration. The best results are highlighted with bold

The embeddings **Z** are given by the dimensionality reduction methods while the local model coefficients **B** are optimised post-hoc (using the SLISEMAP loss)

Here we use 20% as the number of nearest neighbours, and the running times are without GPU-acceleration. The best results are highlighted with bold, continued from Table 6

The embeddings **Z** are given by the dimensionality reduction methods while the local model coefficients **B** are optimised post-hoc (using the SLISEMAP loss)

Author contributions The authors Anton Björklund, Jarmo Mäkelä, and Kai Puolamäki have all contributed to all parts of the research (theory, experiments, and writing).

Funding Open Access funding provided by University of Helsinki including Helsinki University Central Hospital. Computational resources provided by Finnish Grid and Cloud Infrastructure (2022). Anton Björklund is supported by the Doctoral Programme in Computer Science at University of Helsinki, Jarmo Mäkelä is supported by Academy of Finland (decision 320182) and Future Makers program of Technology Industries of Finland Centennial Foundation and the Jane and Aatos Erkko Foundation, and Kai Puolamäki is supported by the Academy of Finland (decision 346376).

Data availability All datasets found in this paper can be downloaded from https://openml.org.

Code availability The source code for the algorithm and all the experiments are available under an open source MIT License from GitHub (Björklund et al., 2022b).

Declarations

Conflict of interest The authors declare that they have no conflicts of interest.

Ethical approval This paper is computational in nature, it uses only synthetic or previously published reference datasets, does not involve any human participants, and has no other outstanding ethical concerns.

Consent for participation and publication Not applicable.

References

Adler, P., Falk, C., Friedler, S. A., Nix, T., Rybeck, G., Scheidegger, C., et al. (2018). Auditing black-box models for indirect influence. *Knowledge and Information Systems, 54*(1), 95–122. https://doi.org/10.1007/s10115-017-1116-3.

Ali, S. M., & Silvey, S. D. (1966). A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society: Series B (Methodological), 28*(1), 131–142. https://doi.org/10.1111/j.2517-6161.1966.tb00626.x.

Anbtawi, W. (2019). A 3D Playground for t-SNE With explainable classification. PhD thesis, American University of Beirut, Beirut, Lebanon.

Anders, F., Chiappini, C., Santiago, B. X., Matijevič, G., Queiroz, A. B., Steinmetz, M., & Guiglion, G. (2018). Dissecting stellar chemical abundance space with t-SNE. *Astronomy & Astrophysics, 619,* A125. https://doi.org/10.1051/0004-6361/201833099.

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., & Müller, K. R. (2010). How to explain individual classification decisions. *Journal of Machine Learning Research, 11*(61), 1803–1831.

Baldi, P., Sadowski, P., & Whiteson, D. (2014). Searching for exotic particles in high-energy physics with deep learning. *Nature Communications, 5*(1), 4308. https://doi.org/10.1038/ncomms5308.

Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation, 15*(6), 1373–1396. https://doi.org/10.1162/089976603321780317.

Bibal, A., Vu, V.M., Nanfack, G., & Frénay, B. (2020). Explaining t-SNE embeddings locally by adapting LIME. In: 28th European Symposium on artificial neural networks, computational intelligence and machine learning, ESANN 2020, Bruges, Belgium, October 2–4, 2020, pp 393–398, https://www.esann.org/sites/default/files/proceedings/2020/ES2020-105.pdf

Björklund, A., Henelius, A., Oikarinen, E., Kallonen, K., & Puolamäki, K. (2019). Sparse robust regression for explaining classifiers. In *Discovery Science*, vol 11828, Springer International Publishing, Cham, pp 351–366, https://doi.org/10.1007/978-3-030-33778-0_27

Björklund, A., Henelius, A., Oikarinen, E., Kallonen, K., & Puolamäki, K. (2022). Robust regression via error tolerance. *Data Mining and Knowledge Discovery*. https://doi.org/10.1007/s10618-022-00819-2.

Björklund, A., Mäkelä, J., & Puolamäki, K. (2022b). SLISEMAP: Combine supervised dimensionality reduction with local explanations. https://github.com/edahelsinki/slisemap

Cheng, M. Y., & Wu, H. T. (2013). Local linear regression on manifolds and its geometric interpretation. *Journal of the American Statistical Association, 108*(504), 1421–1434. https://doi.org/10.1080/01621459.2013.827984.

Cranor, L. F., & LaMacchia, B. A. (1998). Spam. *Communications of the ACM, 41*(8), 74–83. https://doi.org/10.1145/280324.280336.

Cunningham, J. P., & Ghahramani, Z. (2015). Linear dimensionality reduction: Survey, insights, and generalizations. *Journal of Machine Learning Research, 16*(89), 2859–2900.

Datta, A., Sen, S., & Zick, Y. (2016). Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE symposium on security and privacy (SP), IEEE, San Jose, CA*, pp 598–617, https://doi.org/10.1109/SP.2016.42

Diaz-Papkovich, A., Anderson-Trocmé, L., & Gravel, S. (2021). A review of UMAP in population genetics. *Journal of Human Genetics, 66*(1), 85–91. https://doi.org/10.1038/s10038-020-00851-4.

Finnish Grid and Cloud Infrastructure (2022). Finnish grid and cloud infrastructure.

Fisher, A., Rudin, C., & Dominici, F. (2019). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research, 20*(177), 1–81.

Fong, R.C., & Vedaldi, A. (2017). Interpretable Explanations of black boxes by meaningful perturbation. In *2017 IEEE international conference on computer vision (ICCV)*, IEEE, Venice, pp 3449–3457, https://doi.org/10.1109/ICCV.2017.371

Goldstein, A., Kapelner, A., Bleich, J., & Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics, 24*(1), 44–65. https://doi.org/10.1080/10618600.2014.907095.

Goodman, B., & Flaxman, S. (2017). European Union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine, 38*(3), 50–57. https://doi.org/10.1609/aimag.v38i3.2741.

Guidotti, R., Monreale, A., Ruggieri, S., Pedreschi, D., Turini, F., & Giannotti, F. (2018). Local Rule-Based Explanations of Black Box Decision Systems. arXiv:1805.10820

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2019). A survey of methods for explaining black box models. *ACM Computing Surveys, 51*(5), 1–42. https://doi.org/10.1145/3236009.

Hajderanj, L., Weheliye, I., & Chen, D. (2019). A New Supervised t-SNE with Dissimilarity Measure for Effective Data Visualization and Classification. In *Proceedings of the 2019 8th international conference on software and information Engineering*, ACM, Cairo Egypt, pp 232–236, https://doi.org/10.1145/3328833.3328853

Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer.

Henelius, A., Puolamäki, K., Boström, H., Asker, L., & Papapetrou, P. (2014). A peek into the black box: Exploring classifiers by randomization. *Data Mining and Knowledge Discovery, 28*(5–6), 1503–1529. https://doi.org/10.1007/s10618-014-0368-8.

Henelius, A., Puolamäki, K., & Ukkonen, A. (2017). Interpreting classifiers through attribute interactions in datasets. arXiv:1707.07576

Kang, B., García García, D., Lijffijt, J., Santos-Rodríguez, R., & De Bie, T. (2021). Conditional t-SNE: More informative t-SNE embeddings. *Machine Learning, 110*(10), 2905–2940. https://doi.org/10.1007/s10994-020-05917-0.

Kobak, D., & Berens, P. (2019). The art of using t-SNE for single-cell transcriptomics. *Nature Communications, 10*(1), 5416. https://doi.org/10.1038/s41467-019-13056-x.

Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika, 29*(1), 1–27. https://doi.org/10.1007/BF02289565.

Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., & Müller, K. R. (2019). Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications, 10*(1), 1096. https://doi.org/10.1038/s41467-019-08987-4.

Laugel, T., Renard, X., Lesot, M.J., Marsala, C., & Detyniecki, M. (2018). Defining Locality for Surrogates in Post-hoc Interpretablity. arXiv:1806.07498

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324. https://doi.org/10.1109/5.726791.

Levine, Y., Lenz, B., Dagan, O., Ram, O., Padnos, D., Sharir, O., Shalev-Shwartz, S., Shashua, A., & Shoham, Y. (2020). SenseBERT: Driving some sense into BERT. In *Proceedings of the 58th annual meeting of the association for computational linguistics*. Association for Computational Linguistics, pp 4656–4667, https://doi.org/10.18653/v1/2020.acl-main.423

Liese, F., & Vajda, I. (2006). On divergences and informations in statistics and information theory. *IEEE Transactions on Information Theory, 52*(10), 4394–4412. https://doi.org/10.1109/TIT.2006.881731.

Lundberg, S.M., & Lee, S.I. (2017). A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems, Curran Associates, Inc., vol 30, https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

McInnes, L., Healy, J., Saul, N., & Großberger, L. (2018). UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software, 3*(29), 861. https://doi.org/10.21105/joss.00861.

McInnes, L., Healy, J., & Melville, J. (2020). UMAP: Uniform manifold approximation and projection for dimension reduction. arXiv:1802.03426

Mead, A. (1992). Review of the development of multidimensional scaling methods. *The Statistician, 41*(1), 27. https://doi.org/10.2307/2348634.

Molnar, C. (2019). *Interpretable machine learning: A guide for making black box models interpretable*. Lulu.

Nelles, O., Fink, A., & Isermann, R. (2000). Local linear model trees (LOLIMOT) toolbox for nonlinear system identification. *IFAC Proceedings Volumes, 33*(15), 845–850. https://doi.org/10.1016/S1474-6670(17)39858-0.

Nocedal, J. (1980). Updating quasi-Newton matrices with limited storage. *Mathematics of Computation, 35*(151), 773–782. https://doi.org/10.1090/S0025-5718-1980-0572855-7.

Oikarinen, E., Tiittanen, H., Henelius, A., & Puolamäki, K. (2021). Detecting virtual concept drift of regressors without ground truth values. *Data Mining and Knowledge Discovery, 35*(3), 726–747. https://doi.org/10.1007/s10618-021-00739-7.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, Curran Associates, Inc., vol 32, https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research, 12*(85), 2825–2830.

Ribeiro, M.T., Singh, S., & Guestrin, C. (2016), "Why Should I Trust You?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, ACM, San Francisco California USA, pp 1135–1144, https://doi.org/10.1145/2939672.2939778

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. *Proceedings of the AAAI Conference on Artificial Intelligence, 32,* 1527–1535.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science, 290*(5500), 2323–2326. https://doi.org/10.1126/science.290.5500.2323.

Samek, W., Montavon, G., Vedaldi, A., Hansen, L. K., & Muller, K. R. (2019). Explainable AI: Interpreting, explaining and visualizing deep learning, lecture notes in computer science, vol 11700. *Springer International Publishing, Cham,*. https://doi.org/10.1007/978-3-030-28954-6

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2020). Grad-CAM: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision, 128*(2), 336–359. https://doi.org/10.1007/s11263-019-01228-7. arXiv:1610.02391

Shapley, L. S. (1951). Notes on the N-Person Game: II: The value of an N-person game. *RAND Corporation*. https://doi.org/10.7249/RM0670.

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science, 290*(5500), 2319–2323. https://doi.org/10.1126/science.290.5500.2319.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B (Methodological)., 58*(1), 267–288.

van der Maaten, L. (2014). Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research, 15*(93), 3221–3245.

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research, 9*(86), 2579–2605.

Vanschoren, J., van Rijn, J. N., Bischl, B., & Torgo, L. (2014). OpenML: Networked science in machine learning. *ACM SIGKDD Explorations Newsletter, 15*(2), 49–60. https://doi.org/10.1145/2641190.2641198.

Zhang, Z., & Wang, J. (2006). MLLE: Modified locally linear embedding using multiple weights. In: Advances in Neural information processing systems, MIT Press, vol 19, https://proceedings.neurips.cc/paper/2006/file/fb2606a5068901da92473666256e6e5b-Paper.pdf.