SLISEMAP: Explainable Dimensionality Reduction

Björklund, Anton

2022

Björklund , A , Mäkelä , J & Puolamäki , K 2022 ' SLISEMAP: Explainable Dimensionality Reduction ' . < https://arxiv.org/abs/2201.04455 >

http://hdl.handle.net/10138/341567

submittedVersion

SLISEMAP: Explainable Dimensionality Reduction

Anton Björklund · **Jarmo Mäkelä** · **Kai Puolamäki**

the date of receipt and acceptance should be inserted later

Abstract Existing explanation methods for black-box supervised learning models generally work by building local models that explain the models behaviour for a particular data item. It is possible to make global explanations, but the explanations may have low fidelity for complex models. Most of the prior work on explainable models has been focused on classification problems, with less attention on regression.

We propose a new manifold visualization method, SLISEMAP, that at the same time finds local explanations for all of the data items and builds a two-dimensional visualization of model space such that the data items explained by the same model are projected nearby. We provide an open source implementation of our methods, implemented by using GPU-optimized PyTorch library. SLISEMAP works both on classification and regression models.

We compare SLISEMAP to most popular dimensionality reduction methods and some local explanation methods. We provide mathematical derivation of our problem and show that SLISEMAP provides fast and stable visualizations that can be used to explain and understand black box regression and classification models.

Keywords Manifold visualization · explainable AI

1 Introduction

In the past 20 years one of the major developments in the area of unsupervised learning is the development of manifold visualization methods. The trend that started from ISOMAP in 2000 (Tenenbaum et al., 2000) has resulted to hundreds of methods to be developed, popular examples of which include methods such as t-SNE (van der Maaten and Hinton, 2008) and UMAP (McInnes et al., 2018a). The manifold visualization methods have proven to be invaluable tool in exploring and understanding complex data sets, for example such that occur in genetics (Kobak and Berens, 2019; Diaz-Papkovich et al., 2021).

Another recent development is explainable artificial intelligence (XAI), where the objective is to understand and explore black box supervised learning algorithms, see Guidotti

A. Björklund · J. Mäkelä · K. Puolamäki
University of Helsinki, Helsinki, Finland

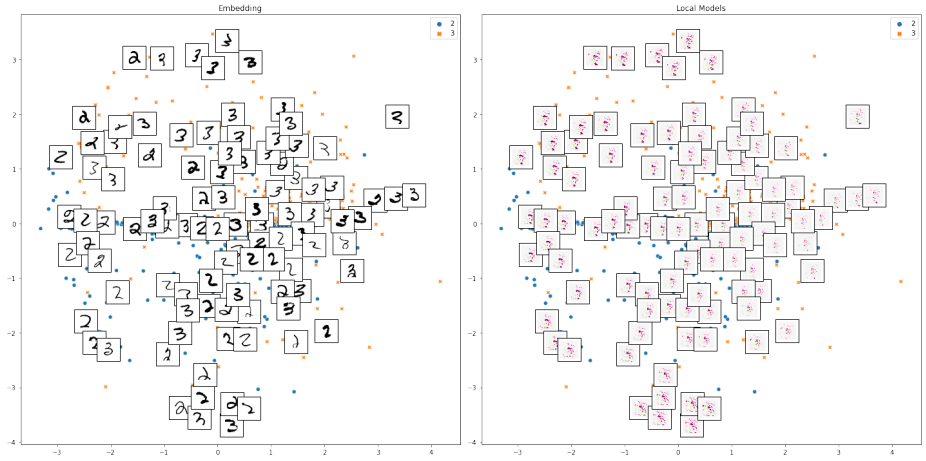

Fig. 1 SLISEMAP visualization of the MNIST dataset.

et al. (2019) for a recent survey. However, most of the work in XAI has been related to classification, there are fewer examples on regression problems. The explanation methods can be roughly divided into global methods and local methods.

Global methods try to explain the global behaviour of classifiers by constructing a global understandable surrogate model that approximates the complex classifier. The drawback of the global approach is that for complex enough classifiers there is no simple surrogate model that would replicate the classifier with a reasonable fidelity.

An alternative approach is not even try global explanations, but only try to make a simple model that explains how a particular data point is classified. These methods are called local: their advantage is that it is often possible to give high-fidelity interpretable local explanations, but obvious disadvantage being that an explanation that is good for one data point may be useless for most of the other data points.

In this paper, we will combine the above two developments, namely the manifold visualizations and local explanations to obtain *global* explanations *and* manifold visualizations of the model spaces of arbitrary black box supervised learning algorithms.

Our idea is straightforward: we want to find an embedding of data points into (typically) two-dimensional plane such that the supervised learning model of the data points that are nearby in the embedding are explained by the same interpretable model. 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 models. At the same time, our method produces a visualization of the data where the data points that are being classified (or regressed) with the same rules are shown nearby.

As an example, Figure 1 (left) shows visualization of a model that classifies 2 vs. 3 in the classic MNIST data set. The images are projected into two-dimensional plane such that the classifier for images projected nearby can to a good fidelity be approximated by the same white-box linear model. The local white-box models are shown in Figure 1 (right). For example the classifier separates the 2s and 3s at the bottom right corner mainly by the ”peak” in the middle of number 3. The SLISEMAP visualization has been scaled so that the points at distance $\lesssim 1$ away in the embedding belong to the same soft neighbourhood and their classification is explained by similar white-box model.

The contributions of this paper are the following: (i) We define a criterion for an embedding that shows local explanations and give efficient algorithm to find such embeddings. (ii) We show experimentally that our method is scalable and results into informative and useful visualizations and global white box models that can be used to explain and understand the supervised learning model. (iii) We compare our contribution to manifold visualization methods and comparable local explanation methods.

2 Related work

In this section we briefly review the explainable AI and dimensionality reduction methods.

2.1 Explainable AI

The interpretation of machine learning models can be generally divided into 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 (Ribeiro et al., 2016, 2018; Fong and Vedaldi, 2017; Lundberg and Lee, 2017; Guidotti et al., 2018); please see Guidotti et al. (2019) for a recent survey and references. On a global level, the scope of interpretation is on understanding *how* the model has produced the predictions where as *why* is usually beyond human comprehension due to model complexity. On this level we can examine, e.g., which features affect the predictions most (Fisher et al., 2019) and what interactions there are between features (Goldstein et al., 2014; Henelius et al., 2014, 2017).

On the other hand, it is possible to examine and understand *how* and *why* the model produce predictions locally since these can be approximated with interpretable simple models. Instead of focusing on single use cases, we are interested in locally interpretable methods that can be used for any type of model (model-agnostic) and do not require any model modifications (post-hoc). One of the first such methods was LIME (Ribeiro et al., 2016), which generates the interpretations for user-defined areas of interest by perturbing the data and training a weighted model based on predictions. Another similar method is SHAP (Lundberg and Lee, 2017), which utilises weights based on Shapley value estimation (Shapley, 1951). Both of these methods generate the local explanations based on perturbed data and this process is non-trivial (Guidotti et al., 2018; Laugel et al., 2018; Molnar, 2019). A third method called SLISE (Björklund et al., 2019), utilises only existing data, and finds the largest set of data items that can be approximated (up to a given accuracy) by a sparse linear model, by ignoring outliers; 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 indefinite features or combining multiple features into single elements and thus make the data more interpretable. There are advantages of utilising dimensional reduction as it removes correlated features in the data and allows for easier visualization, e.g., in two dimensions, but combined features can also become less interpretable and some information will be inevitably lost. The simplest dimensional reduction techniques are unsupervised 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 and Ghahramani, 2015). Other approaches include truncated singular value decomposition (SVD) (Belkin and Niyogi, 2003), locally linear embedding and modified version (LLE, MLLE) (Roweis and Saul, 2000; Zhang and Wang, 2006), spectral embedding (Belkin and Niyogi, 2003) and multidimensional scaling (MDS) (Kruskal, 1964), global-distance preserving MDS (Mead, 1992), ISOMAP (Tenenbaum et al., 2000), t-SNE (van der Maaten and Hinton, 2008), and UMAP (McInnes et al., 2018a). Lately, some supervised methods have also become available, based on t-SNE (Kang et al., 2021; Hajderanj et al., 2019) and included in the UMAP implementation (McInnes et al., 2018b). So far, to our knowledge, there has not been a method that would directly combine explainable AI and dimensionality reduction.

3 Definitions and algorithms

3.1 Definitions and problems

The dataset consists of n pairs of points $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$, where $\mathbf{x}_i \in \mathscr{X}$ are the *covariates* and $\mathbf{y}_i \in \mathscr{Y}$ are *responses*. $\mathscr{X}$ and $\mathscr{Y}$ are the domains of the covariates $\mathbf{x}_i$ and responses $\mathbf{y}_i$, respectively. In this paper, and in our software implementation, we restrict ourselves to real spaces, $\mathscr{X} = \mathbb{R}^m$ and $\mathscr{Y} = \mathbb{R}^p$, but the derivations in this subsection are general.

The goal is to find a local *white-box* model $g_i : \mathscr{X} \to \mathscr{Y}$ for every data point $(\mathbf{x}_i, \mathbf{y}_i)$. More specifically, we write $\tilde{\mathbf{y}}_{ij} = g_i(\mathbf{x}_j)$. Again, while the derivation is general, in this paper we focus on cases where the white-box algorithm, g, is either a linear projection (for regression problems) or multinomial logistic regression (for classification problems), as defined in Section 3.3.

If we have access to a trained *black-box* supervised learning algorithm $f : \mathscr{X} \to \mathscr{Y}$, then we can also consider using its estimates $\hat{\mathbf{y}}_i = f(\mathbf{x}_i)$ instead of $\mathbf{y}_i$. This would mean that the local models g_i are local approximations of the black-box model. These approximations can then also be used to explain the predictions of the black-box model (Björklund et al., 2019).

Additionally, we want to find a lower-dimensional embedding $\mathbf{Z}_i$, where $\mathbf{Z} \in \mathbb{R}^{n \times d}$, for every data point $(\mathbf{x}_i, \mathbf{y}_i)$, such that neighbouring data points in the embedding space have similar local models g_i. For the sake of visualisation, $\mathbf{Z}_i$ is typically 2-dimensional ($d = 2$).

Denote by $\mathbf{D}_{ij}$ the Euclidean distance between the points $\mathbf{Z}_i$ and $\mathbf{Z}_i$ in the embedding, where

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

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 further define a loss function $l : \mathscr{Y} \times \mathscr{Y} \to \mathbb{R}_{\geq 0}$ for the white-box models. Here we use the shorthand notation $\mathbf{L}_{ij} = l(\tilde{\mathbf{y}}_{ij}, \mathbf{y}_j) = l(g_i(\mathbf{x}_j), \mathbf{y}_j)$. In this work, we use quadratic losses (for regression) and Hellinger distances between multinomial distributions (for classification), which we define later in Section 3.3.

Recall that we want that all points in the (soft) neighbourhood of point $\mathbf{Z}_i$ to be modelled well by the local white-box model g_i. Mathematically, this is can be formalized as minimizing the following weighted loss:

$$\mathscr{L}_i = \sum_{j=1}^{n} \mathbf{W}_{ij} \mathbf{L}_{ij}, \tag{3}$$

where each local model g_i has its own set of weights $\mathbf{W}_i$, of which $\mathbf{W}_{ii}$ is the largest (due to $\mathbf{D}_{ii} = 0$). This is what makes the models *local*. We obtain our final loss function by summing over all losses given by Equation (3). We summarize 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 function g, loss function l, and constants $\lambda_z > 0$ and $\lambda_{lasso} \geq 0$, find the parameters for $g_1, \ldots, g_n$ and embedding $\mathbf{Z} \in \mathbb{R}^{n \times d}$ that minimize the loss given by

$$\mathscr{L} = \sum_{i=1}^{n} \sum_{j=1}^{m} \mathbf{W}_{ij} \mathbf{L}_{ij} + \lambda_z \sum_{i=1}^{n} \sum_{k=1}^{d} \mathbf{Z}_{ik}^2 + \lambda_{lasso} \sum_{i=1}^{N} \sum_{p=1}^{P} \left| \mathbf{B}_{ip} \right| \tag{4}$$

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

We have added a regularization term proportional to λ_z, in order to limit the range of values in the embedding $\mathbf{Z}$; otherwise, there would be a trivial solution where all data points are infinitely far away from each other in the embedding. The regularization term is quadratic, which means that the embedding is invariant under rotation, i.e., we can freely rotate the embedding.

The λ_z parameter essentially regularizes the size of the neighbourhood. If λ_z is small the neighbourhoods are small as well. At the limit of $\lambda_z = 0$ the points are infinitely far away from each other in the embedding $\mathbf{Z}$ and the neighbourhood of a point consists only of the point itself. On the other hand, if λ_z is large then the neighbourhoods are also large. At the limit $\lambda_z \to \infty$ all of the points will be compressed to the origin in the embedding $\mathbf{Z}$ and, hence, all points will be estimated by the same local model.

We have additionally added for numerical stability a small Lasso regularization term, which we set to $\lambda_{lasso} = 10^{-4}$ for the remainder of the paper.

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 problems 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 $\mathbf{Z}'_{n+1} \in \mathbb{R}^d$ such that the loss of Equation (4) is minimized, when n is incremented by one, g_{n+1} is added to the set of local models, and $\mathbf{Z}$ is replaced by $\mathbf{Z}'$.

Solving of the 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 vanish asymptotically at the limit of large n. We study this difference experimentally in Section 4.

3.3 Slisemap for regression and classification

While the definitions in Section 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 $\mathscr{X} = \mathbb{R}^m$. 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 to 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 $\mathscr{Y} = \mathbb{R}$. The white-box regression model is given by a linear function

$$g_R(\mathbf{x}, \mathbf{b}) = \mathbf{x}^T \mathbf{b}, \tag{5}$$

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

$$l_R(\tilde{\mathbf{y}}, \mathbf{y}) = (\tilde{\mathbf{y}} - \mathbf{y})^2, \tag{6}$$

where $\tilde{\mathbf{y}} = g_R(\mathbf{x}, \mathbf{b})$.

The linear regression model g_R is being 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 $\mathbf{B} \in \mathbb{R}^{n \times m}$, then the parameters being optimised in Problem 1 are $\mathbf{B}$ and $\mathbf{Z}$.

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 $\mathscr{Y} = \{\mathbf{y} \in \mathbb{R}^p_{\geq 0} \mid \sum_{i=1}^p \mathbf{y}_i = 1\}$. Multinomial logistic regression can be parametrized 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),

$$\tilde{\mathbf{y}}_i = g_C(\mathbf{x}, \mathbf{b})_i = \begin{cases} \frac{\exp\left(\mathbf{x}^T \mathbf{b}_{((i-1)m+1):(im)}\right)}{1 + \sum_{j=1}^{p-1} \exp\left(\mathbf{x}^T \mathbf{b}_{((j-1)m+1):(jm)}\right)} & \text{if } i < p \\ \frac{1}{1 + \sum_{j=1}^{p-1} \exp\left(\mathbf{x}^T \mathbf{b}_{((j-1)m+1):(jm)}\right)} & \text{if } i = p \end{cases}, \tag{7}$$

where we have used $\mathbf{b}_{a:b}$ to denote a $(b-a+1)$-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 $\mathbf{B} \in \mathbb{R}^{n \times (p-1)m}$.

For the loss function any distance measure between multinomial probabilities, such as Kullback-Leibler (KL) divergence, would do. Here we however choose the numerically more stable squared Hellinger distance (Ali and Silvey, 1966; Liese and Vajda, 2006),

$$l_C(\tilde{\mathbf{y}}, \mathbf{y}) = \frac{1}{2} \sum_{i=1}^{p} \left(\sqrt{\tilde{\mathbf{y}}_i} - \sqrt{\mathbf{y}_i} \right)^2 = 1 - \sum_{i=1}^{p} \sqrt{\tilde{\mathbf{y}}_i \mathbf{y}_i}. \tag{8}$$

The squared Hellinger distance is symmetric and bounded in interval $[0, 1]$, unlike the KL which is not symmetric nor 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.

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

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 ($p = 2$) that was used in Björklund et al. (2019). 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 the SLISEMAP for regression with quadratic loss, as above. The alternative formulation has advantage that it may converge in some cases where the multinomial logistic regression does not. Using a logit transformation followed by a linear model matches the behaviour of, e.g., SHAP (Lundberg and 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). The parameters for the local models $\mathbf{B}$ are initialised by sampling from a normal distribution with zero mean and unit variance. Then we optimise the values by minimising the loss in Equation (4).

Function `Slisemap(`$\mathbf{X}$, $\mathbf{y}$, λ_z`)`
  $\mathbf{B} \sim \mathbb{N}(0,1)$
  $\mathbf{Z} \leftarrow \mathrm{PCA}(\mathbf{X})$
  $\mathbf{B}, \mathbf{Z} \leftarrow \arg\min_{B,Z} \mathscr{L}(\mathbf{X}, \mathbf{y}, \mathbf{B}, \mathbf{Z}, \lambda_z)$
  while *not converged* **do**
    $\mathbf{B}, \mathbf{Z} \leftarrow$ `Escape(`$\mathbf{X}, \mathbf{y}, \mathbf{B}, \mathbf{Z}$`)`
    $\mathbf{B}, \mathbf{Z} \leftarrow \arg\min_{B,Z} \mathscr{L}(\mathbf{X}, \mathbf{y}, \mathbf{B}, \mathbf{Z}, \lambda_z)$
  Result: $\mathbf{B}$, $\mathbf{Z}$

Function `Escape(`$\mathbf{X}$, $\mathbf{y}$, $\mathbf{B}$, $\mathbf{Z}$`)`
  $\mathbf{W} \leftarrow$ `Softmax(`$-D, 1$`)`
  for $i \leftarrow 1$ **to** n **do**
    $k \leftarrow \arg\min_k \sum_{j=1}^{n} W_{kj} L_{ji}$
    $\mathbf{B}'_i, \mathbf{Z}'_i \leftarrow \mathbf{B}_k, \mathbf{Z}_k$
  Result: $\mathbf{B}'$, $\mathbf{Z}'$

Function `Slisemap-new(`$\mathbf{X}_{new}$, $\mathbf{y}_{new}$, $\mathbf{X}_{old}$, $\mathbf{y}_{old}$, $\mathbf{B}_{old}$, $\mathbf{Z}_{old}$, λ_z`)`
  $\mathbf{B}_{\text{new}}, \mathbf{Z}_{\text{new}} \leftarrow$ `Escape(`$\mathbf{X}_{\text{new}}, \mathbf{y}_{\text{new}}, \mathbf{B}_{\text{old}}, \mathbf{Z}_{\text{old}}$`)`
  $\mathbf{B}_{\text{new}}, \mathbf{Z}_{\text{new}} \leftarrow \arg\min_{B_{\text{new}}, Z_{\text{new}}} \mathscr{L}([\begin{smallmatrix}\mathbf{X}_{\text{old}}\\ \mathbf{X}_{\text{new}}\end{smallmatrix}], [\begin{smallmatrix}\mathbf{y}_{\text{old}}\\ \mathbf{y}_{\text{new}}\end{smallmatrix}], [\begin{smallmatrix}\mathbf{B}_{\text{old}}\\ \mathbf{B}_{\text{new}}\end{smallmatrix}], [\begin{smallmatrix}\mathbf{Z}_{\text{old}}\\ \mathbf{Z}_{\text{new}}\end{smallmatrix}], \lambda_z)$
  Result: $\mathbf{B}_{\text{new}}$, $\mathbf{Z}_{\text{new}}$

Algorithm 1: The SLISEMAP algorithm, where $\mathscr{L}, W, L$ are given in Equation (4). The algorithm uses a heuristic to escape local optima in search of better solutions, and iterates until the loss stops improving.

Since the problem is non-convex, we are likely to only find a local optimum. Thus, we use an heuristic approach to escape local optima. In Section 4.7 we empirically evaluate the value of the escape heuristic by comparing results with and without it. The heuristic consists of moving each item (embedding and local model) to the soft neighbourhood, given by $\mathbf{W}$ in Equation (2), that have the most suitable local models. This is repeated until no further improvement is found.

The pseudocode for Problem 2 (adding new data points to a SLISEMAP solution) is also given in Algorithm 1. 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).

For the implementation 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) optimizer of PyTorch. As explained earlier, in this paper we assume that the data is real valued and use the white box models and losses of Section 3.3 to study regression and classification problems. We solve Problem 2 with PyTorch and L-BFGS as well.

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.[1]

3.5 Computational complexity

Evaluation of the loss function of Equation (4) requires at least $O(n^2m)$ iterations for linear regression and $O(n^2mp)$ for multinomial logistic regression. This is because for every local model $O(n)$ the prediction and loss $O(mp)$ has to 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 Equation (4) has to be evaluated multiple times. While it is difficult to provide strict running time limits for iterative optimization algorithm such as L-BFGS—we study this experimentally in Section 4—it is obvious that the algorithm may not scale well for very large (n) datasets.

Usually it however suffices to sub-sample $\min(n, n_0)$ data points, where n_0 is a suitably chosen constant, optimize for the loss function (Problem 1), and then add points to the existing solution (Section 3.2). By this procedure the asymptotic complexity of SLISEMAP is linear with respect to the number of data points n. Especially for visualization purposes it often makes no sense to compute exact projection for a huge number of data points: visualization cannot show more data points than there are pixels and usually having a extremely accurate solution to the full optimization problem instead of approximate solution brings little additional benefit. Instead, finding a quick solution for a sub-sampled data and adding necessary number of data points to the embedding works well in practice, as shown in the experiments of Section 4.

4 Experiments

In the experiments of this section, we always embed data into two dimensions ($d = 2$) and normalize 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) to the data matrix $\mathbf{X}$ before running SLISEMAP or any other algorithm on the data. Furthermore, unless otherwise mentioned we subsample the large datasets to 1000 data items.

Most datasets have been used in two scenarios, first for normal regression or classification using the definitions from Section 3.3, and second in an XAI inspired scenario where the targets are predictions from black box models, using the alternative formulation from Section 3.3 in case of classification. An overview of the datasets and black box models can be seen in Table 1.

As explained earlier, we use PyTorch version 1.8.1 (Paszke et al., 2019). The runtime experiments have been run on a server having Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz

[1] The Github repository will be published when the paper is accepted.

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

processors with 4 cores and 16GB RAM memory allocated, and a NVIDIA Tesla P100 GPU with 16GB memory. The scripts to run the experiments are available via Github[2].

4.1 Datasets

Below we describe the datasets used in the experiments. A quick summary can be seen in Table 1.

Synthetic data We create synthetic *regression data* (RSYNTH) as follows: given parameters data set size n (number of data items) and m (number data attributes), as well as k (number of clusters) and s standard deviation of the clusters. We first sample from normal distribution with zero mean and unit variance a vector $\beta_j \in \mathbb{R}^m$ and cluster centroids $\mathbf{c}_j \in \mathbb{R}^m$ from normal distribution with zero mean and standard deviation of s, where $j \in [k] = \{1, \ldots, k\}$. We then create data point $i \in [n]$ by first sampling the cluster index $j_i \in [k]$ in random and then generating a data vector $\mathbf{x}_i$ by sampling from a normal distribution with mean of $\mathbf{c}_{j_i}$ and unit variance. The dependent variable is then given by $\mathbf{y}_i = \mathbf{x}_i^T \beta_{j_i} + \varepsilon_i$, where ε_i is Gaussian noise with zero mean and standard deviation of 0.1. Unless otherwise mentioned we use $k = 3$ and $s = 0.25$ for all the experiments in this paper.

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

Boston housing dataset was collected by the U.S Census Service from the Boston Standard Metropolitan Statistical Area in 1970. The size of the dataset is 506 items with 14 attributes, including median value of owner-occupied homes that is used as the dependent variable. The dataset was obtained from openML.

Spam (Cranor and LaMacchia, 1998) is a UCI dataset containing both spam, i.e., unsolicited commercial email, as well as professional and personal, non-spam emails. There are 4601 instances with 57 attributes in the dataset, including a classifier denoting whether the email was spam or not.

[2] The Github repository will be published when the paper is accepted.

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 is producing Higgs bosons or not. There are altogether 28 attributes, the first 21 featuring kinematic properties measured by the particle detectors, and the last seven are functions of the first 21—high-level features derived by physicists to help discriminate between the two classes.

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, located on four wilderness areas in the Roosevelt National Forest of northern Colorado.

MNIST (Deng, 2012) is the classic machine learning dataset of handwritten digits from 0 to 9. Each digit is represented by a 28x28 greyscale image (784 pixels with integer pixel values between 0 and 255). The images are mostly black and white and grey levels are due to anti-aliasing techniques in the normalization algorithm. Due to large number of pixels, we subsample to 4000 data items and create a binary classification task by limiting the available digits to 2 and 3.

4.2 Metrics

To be able 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 Equation (4). However, the loss will change based on parameters such as λ_z and the size of the dataset, which makes it less useful for comparison.

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:

$$\frac{1}{n}\sum_{i=1}^{n} |\text{k-NN}(i) \cap \{j \mid c_i = c_j\}|/k, \tag{9}$$

where k-NN(i) is the set of k nearest neighbours (of item i) in the embedding space, using Euclidean distance of Equation (1), and $j \in 1, \ldots, n$. 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., 2018) measures how well it can predict the correct outcome, using the losses defined in Section 3.3 we get:

$$\frac{1}{n}\sum_{i=1}^{n} l(g_i(\mathbf{x}_i), \mathbf{y}_i). \tag{10}$$

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

$$\frac{1}{n}\sum_{i=1}^{n} \frac{1}{k}\sum_{j \in \text{k-NN}(i)} l(g_i(\mathbf{x}_j), \mathbf{y}_j). \tag{11}$$

A smaller value indicates better fidelity.

Table 2 The default value for λ_z for the different datasets.

Dataset	Default λ_z	XAI λ_z
RSYNTH	0.1	-
Air Quality	0.005	0.001
Boston	0.1	0.001
Covertype	-	0.005
Spam	0.005	0.05
Higgs	0.01	0.01

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., 2018) of a local model can be measured by counting the number of data items that have a loss less than a threshold l_0:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{n}\sum_{j=1}^{n}(l(g_i(\mathbf{x}_j),\mathbf{y}_j) < l_0). \tag{12}$$

This obviously 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 want this behaviour to be reflected in the low-dimensional embedding. For that we limit the coverage testing to only the k nearest neighbours:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{k}\sum_{j\ \in\text{k-NN}(i)}(l(g_i(\mathbf{x}_j),\mathbf{y}_j) < l_0). \tag{13}$$

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

4.3 Parameter selection

SLISEMAP has one parameter that needs to be selected, λ_z, that is used to avoid trivial solutions, where the data items are infinitely far apart in the embedding and the local models overfit on the singular neighbourhoods. λ_z is also used to adjust the density of the embedding. The optimal value varies from dataset to dataset and can also be a bit subjective; *how clustered do you want the embedding to be?* To guide the selection of the value for λ_z we use the *fidelity* and *coverage* metrics from above.

The fidelity results for different values of λ_z can be seen in Figure 2. Here we see that for some values of λ_z the fidelity actually improves when the neighbourhood grows, which means that the local models do not match the corresponding data items. This is a sign that λ_z is probably too large.

The coverage results can be seen in Figure 3. We see that for some values of λ_z the coverage immediately drops as the neighbourhood grows. This is a sign that the value of λ_z is probably too small.

Based on these results we can choose the default values for λ_z for the different datasets. The default values can be seen in Table 2. We use these values for the rest of the experiments in this paper.

4.4 Visualizations of the data sets

While fidelity and coverage can be used to diagnose problematic choices of the value of λ_z, there is still room for some subjectivity that is best explored by plotting the low-dimensional

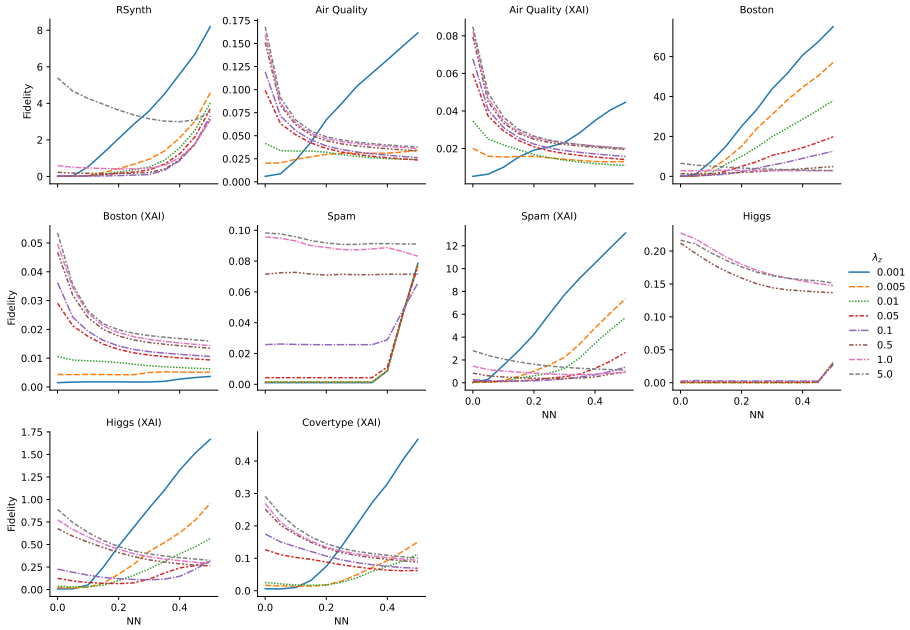

Fig. 2 Local model fidelity on the neighbourhood, with different values for λ_z. Smaller is better, especially with small neighbourhoods.

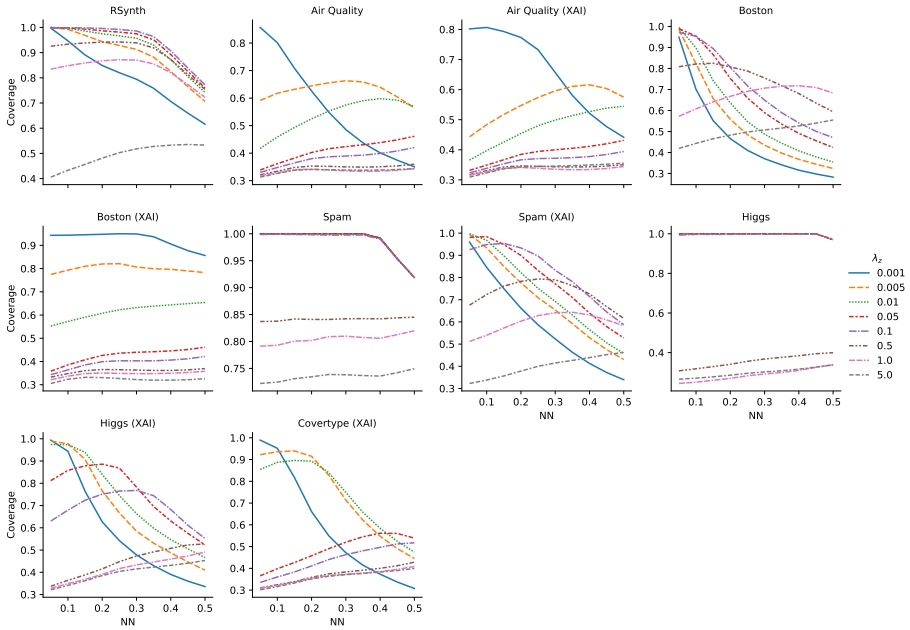

Fig. 3 Local model coverage of the neighbourhood, with different values for λ_z. larger is better.

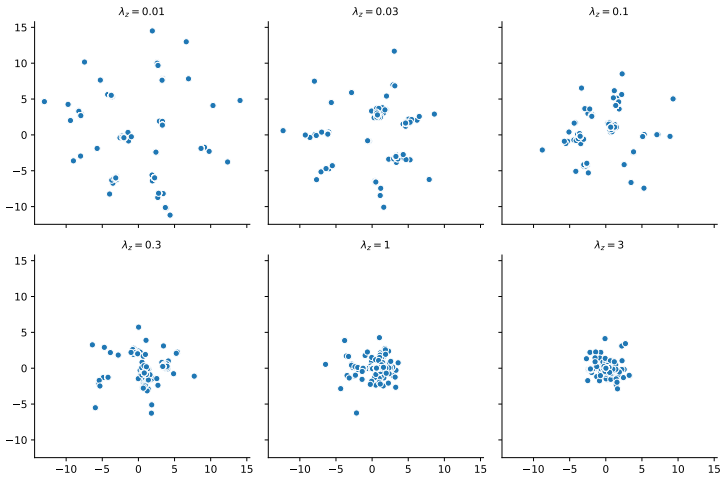

Fig. 4 The low-dimensional embedding of the Boston dataset with different values for λ_z. Small values leads (top left) to sparse solutions with small clusters and potentially overfit local models. Large values of λ_z (bottom right) create a dense cluster in the centre with all (non-)local models being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

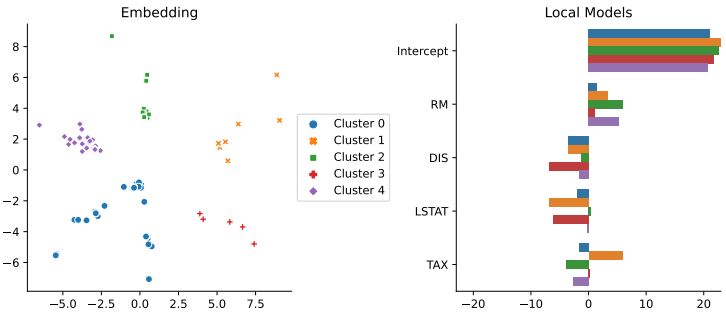

Fig. 5 Clustering the local models to see if they correspond to clusters in the embedding. We also see that the clusters in the embedding have distinct local models.

embeddings for different λ_z values. Figure 4 shows how λ_z affects the embedding. At small values of λ_z we get a relatively sparse solution with many small clusters. However, basing the local models primarily on just a few data items (the ones in the same cluster) might lead to poor generalisation. In the other end of the spectrum, large values of λ_z puts all data items are put into the same cluster. This means that they all have almost the same local model, in which case just fitting a global model (with uniform weights) would work just as well (and be much faster). Additionally, based on this plot we would choose $\lambda_z = 0.1$ or $\lambda_z = 0.3$, which is supported by the results above in Section 4.3.

With SLISEMAP we do not only get 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 check this by clustering the local models independently of the embeddings and check if these clusters corresponds to the structure in the embedding. Furthermore, models far apart in the embedding should look different due to the different local weights.

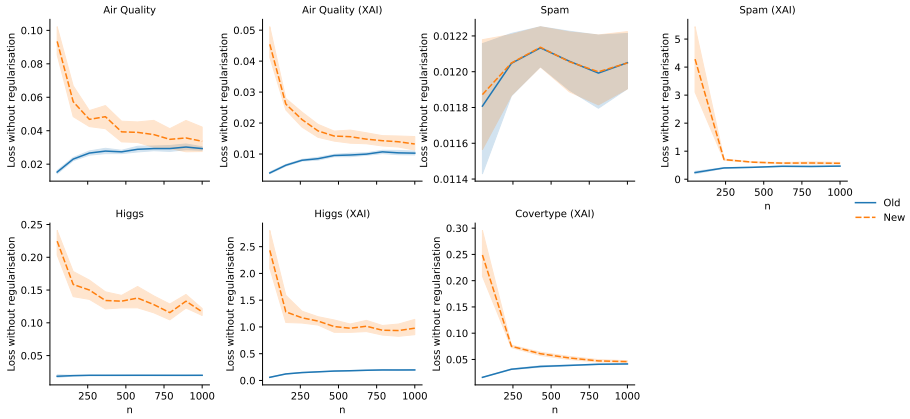

Fig. 6 Adding new data items to a SLISEMAP solution, to see how large the initial set has to be. For most of these datasets just a couple of hundred initial data items are required. Lower is better.

In Figure 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. We also see some differences in the local models (here we only show the cluster centroids): Some local models heavily penalise locations far from the city centre, and some even use a high property tax as an indicator of high value.

A plot of the MNIST data is shown in Figure 1 in the introduction, where we can see that some local models focus heavily on bottom curve of 3:s, while others utilise the (lower) diagonal line of 2:s.

4.5 Subset sampling

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

To investigate how much data is needed we randomly select 2000 data items from the large datasets, of which we let 1000 be left unseen and train SLISEMAP solutions on increasing fractions of the remaining 1000 data items. Then we *individually* add the unseen data items, using `Slisemap-new` from Algorithm 1, and calculate the loss for *only* the unseen data items.

The results can be seen in Figure 6. We see that for most of these datasets just a couple of hundred data items are needed for the results to converge. Since the new items are individually optimised their loss might not match the loss of the existing items, but here we see them get fairly close.

4.6 Runtime and scalability

We tested the time to run SLISEMAP on RSYNTH with different sizes for both CPU and GPU variants of the algorithm. The results can be seen in Figure 7. The GPU implementation has

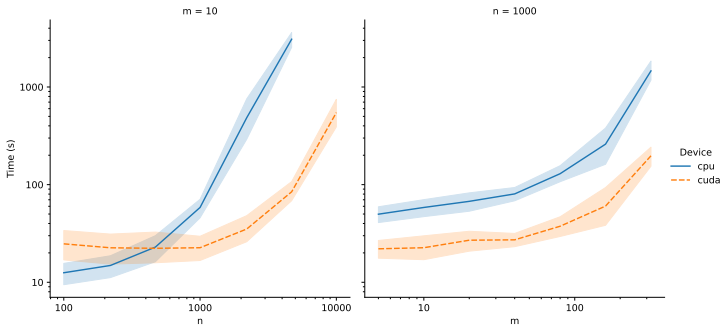

Fig. 7 Runtimes for different dataset sizes (using the RSYNTH dataset). GPU-acceleration (cuda) brings some overhead but offers better parallelisation on large datasets.

some overhead, making it slower for small datasets (less than 400×10), but significantly faster for large datasets (note the logarithmic scale).

4.7 Comparison to dimensionality reduction methods

To compare against other dimensionality reduction methods we use the following methods from the Scikit-learn package (Pedregosa et al., 2011): PCA, Spectral Embedding (Belkin and Niyogi, 2003), LLE (Roweis and Saul, 2000), MLLE (Zhang and Wang, 2006), MDS (Kruskal, 1964), ISOMAP (Tenenbaum et al., 2000), and t-SNE (van der Maaten, 2014). Additionally we use UMAP (McInnes et al., 2018a) and the supervised variant from the umap-learn package (McInnes et al., 2018b).

With the synthetic, RSYNTH, dataset we know the true clusters, so we can use *cluster purity* (Section 4.2) to measure how well the dimensionality reduction methods reconstruct the true clusters. The result can be seen in Table 3. SLISEMAP has the best cluster purity. All the other methods are barely better than random. However, SLISEMAP is slower than most methods, especially on large datasets, due to not using any kind of subsampling and trying to fit local models at the same time.

To demonstrate that the simultaneous local model and embedding search is necessary, we take the embeddings from the different dimensionality reduction methods and fit local models post-hoc (essentially running SLISEMAP with a fixed $\mathbf{Z}$ given by the dimensionality reduction methods). To evaluate the results we use *fidelity* and *coverage*, see Section 4.2. The results can be seen in Figure 8 and Figure 9. As expected, SLISEMAP is better than the other methods.

Included in these results is also a version of SLISEMAP where we do not use the heuristic for escaping local optima. We see that for some datasets the escape is really needed to reach optimal performance, and that it is never worse. However, as can be seen in Table 3, it is a bit slower.

4.8 Comparison to explanations

As eluded to in the introduction, if we have access to a black box model we can use SLISEMAP to find local and interpretable approximations of that black box model. The idea

Table 3 Comparing how well the dimensionality reduction methods reconstruct the ground truth clusters in the RSYNTH dataset. The higher the cluster purity the better the reconstruction. The running times are without GPU-acceleration.

Dataset	Method	Cluster Purity	Time (s)
RSYNTH 200×10	Slisemap	0.844 ± 0.129	7.560 ± 2.642
	Slisemap (no escape)	0.432 ± 0.053	1.501 ± 0.518
	PCA	0.377 ± 0.025	0.053 ± 0.004
	Spectral Embedding	0.379 ± 0.025	0.052 ± 0.003
	LLE	0.351 ± 0.013	0.069 ± 0.006
	MLLE	0.367 ± 0.012	0.111 ± 0.005
	MDS	0.372 ± 0.020	1.543 ± 0.034
	Non-Metric MDS	0.344 ± 0.009	0.138 ± 0.003
	Isomap	0.373 ± 0.029	0.134 ± 0.012
	t-SNE	0.371 ± 0.018	2.715 ± 0.064
	UMAP	0.379 ± 0.028	7.356 ± 0.128
	Supervised UMAP	0.362 ± 0.016	2.964 ± 0.038
RSYNTH 1000×50	Slisemap	0.954 ± 0.012	60.563 ± 17.228
	Slisemap (no escape)	0.456 ± 0.036	14.665 ± 4.728
	PCA	0.586 ± 0.036	0.316 ± 0.046
	Spectral Embedding	0.618 ± 0.025	0.456 ± 0.052
	LLE	0.417 ± 0.034	0.531 ± 0.032
	MLLE	0.409 ± 0.032	0.754 ± 0.047
	MDS	0.432 ± 0.023	31.105 ± 5.883
	Non-Metric MDS	0.335 ± 0.001	3.610 ± 0.713
	Isomap	0.498 ± 0.039	2.088 ± 0.281
	t-SNE	0.415 ± 0.049	13.122 ± 2.976
	UMAP	0.526 ± 0.027	9.373 ± 0.599
	Supervised UMAP	0.527 ± 0.027	5.271 ± 0.313

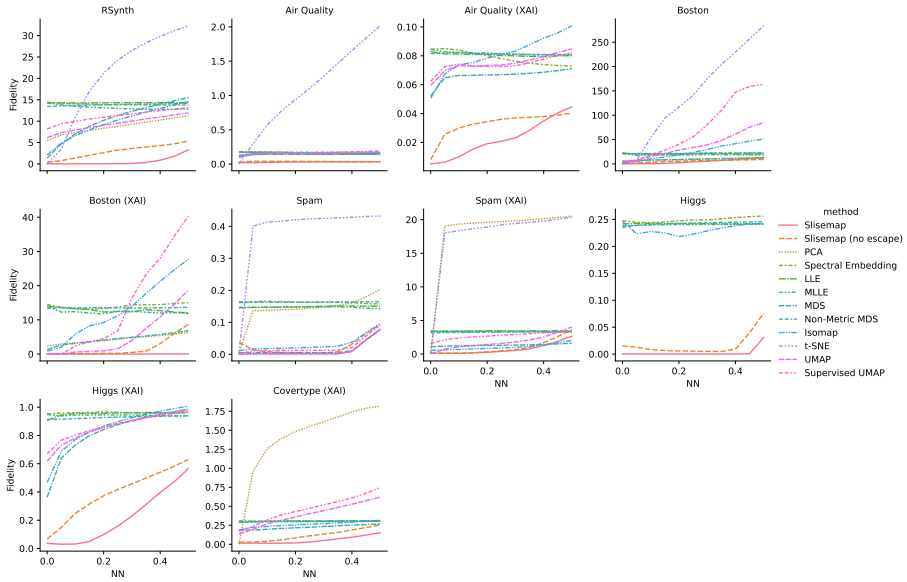

Fig. 8 Fidelity on the neighbourhood, when using different embedding methods before finding the local models. Smaller is better.

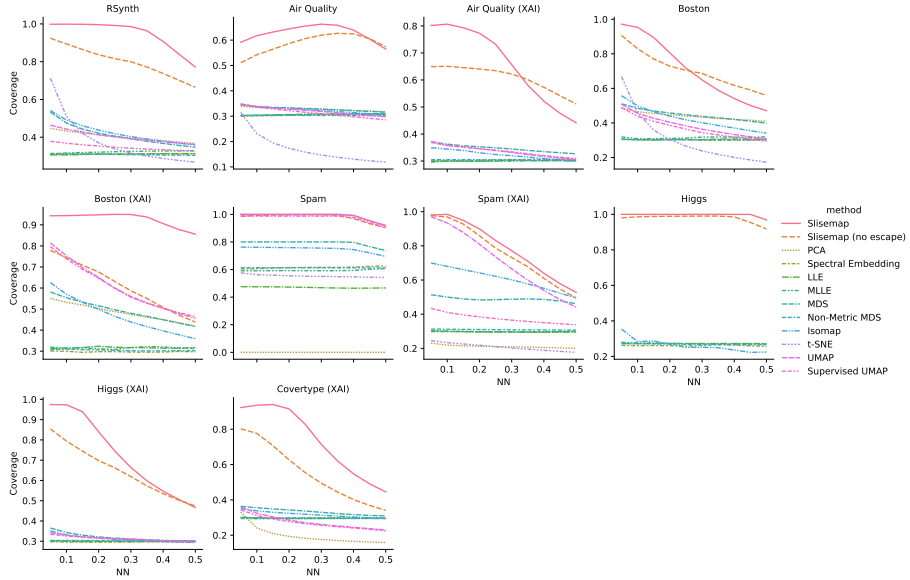

Fig. 9 Coverage of the neighbourhood, when using different embedding methods before finding the local models. Larger is better.

Table 4 Comparing the local models given by SLISEMAP and SLISE, with a global model as a reference. The error tolerance for SLISE and the coverage is selected such that the global model has a coverage of 0.3. Smaller fidelity and larger coverage is better.

Dataset	Method	Fidelity	Coverage	Time (s)
RSYNTH 400×20	SLISEMAP	0.015 ± 0.002	0.447 ± 0.011	28.684 ± 40.276
	SLISE	0.000 ± 0.000	0.498 ± 0.013	130.942 ± 6.815
	Global	11.629 ± 1.396	0.300 ± 0.000	0.019 ± 0.005
BOSTON 400×13	SLISEMAP	1.356 ± 0.140	0.325 ± 0.019	72.221 ± 28.095
	SLISE	0.000 ± 0.000	0.446 ± 0.019	140.076 ± 5.893
	Global	22.625 ± 1.334	0.300 ± 0.000	0.044 ± 0.005
AIR QUALITY 1000×11	SLISEMAP	0.044 ± 0.002	0.311 ± 0.009	133.621 ± 41.316
	SLISE	0.000 ± 0.000	0.353 ± 0.007	3632.525 ± 79.921
	Global	0.172 ± 0.017	0.300 ± 0.000	0.255 ± 0.073

of using local approximations to explain the predictions of a black box model is discussed in, e.g., Ribeiro et al. (2016); Lundberg and Lee (2017); Björklund et al. (2019). The difference to these methods is that SLISEMAP tries to find local approximations for all data items at once, not one at a time, and connect them via a low-dimensional embedding.

Of the local, model-agnostic, approximating explanations methods SLISEMAP is most closely related to SLISE Björklund et al. (2019), hence the name. Thus we want to compare the local approximations of SLISE with those of SLISEMAP. As a reference we also consider a global model. SLISE requires that we specify an *error tolerance* that we choose to be the 0.3 quantile of the losses of the global model. The results can be seen in Table 4, where each dataset has been run ten times with different seeds and subsamples.

By definition SLISE has perfect fidelity, Equation (10), for the data item corresponding to the local model, with SLISEMAP is not far behind. The global model is obviously not local and, thus, have the worst fidelity. One of the things that SLISE is specifically optimising for is the subset size, so it obviously outperforms in the coverage, Equation (12). But, SLISEMAP also performs better than the global model (being fixed at 0.3). However, the drawback of SLISE is that the whole procedure has to be rerun for each data item, which is slow, while SLISEMAP finds them all at once, which significantly faster. SLISEMAP also finds an low-dimensional embedding, which can make it easier to visualise and compare different data items and supervised learning models.

5 Discussion and conclusions

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

We showed that the state-of-the-art manifold dimensionality measures, unsurprisingly, cannot be used to explain classifiers or regression models. On the other hand, the state-of-the-art tools to explain black box models are mostly focused on classifiers and they typically provide only local explanations.

An interesting future work would be to explore how the SLISEMAP visualizations can be used to help build better models and to visualize or adjust model parameters. For example, if SLISEMAP visualization could show that some data items are systematically classified wrongly, which could then be used to improve the underlying supervised learning model with the user feedback.

Declarations

Funding: Computational resources provided by Finnish Grid and Cloud Infrastructure. Anton Björklund is supported by the Doctoral Programme in Computer Science at University of Helsinki, and Jarmo Mäkelä is supported by Academy of Finland (decision 320182).
Conflict of interest: The authors declare that they have no conflicts of interest.
Ethical considerations: This paper is computational in nature, does not involve any human participants, and has no other ethical concerns.
Consent for participation and publication: Not applicable.
Data availability: All datasets found in this paper can be downloaded from `openml.org`.
Code availability: The source code for the algorithm and all the experiments are available from Github,[3]
Authors' contributions: The authors, Anton Björklund, Jarmo Mäkelä, and Kai Puolamäki, have all contributed to all parts of the research (theory, experiments, and writing).

[3] The Github repository will be published when the paper is accepted.

References

Adler P, Falk C, Friedler SA, Nix T, Rybeck G, Scheidegger C, Smith B, Venkatasubramanian S (2018) Auditing black-box models for indirect influence. Knowledge and Information Systems 54(1):95–122, DOI 10.1007/s10115-017-1116-3

Ali SM, Silvey SD (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, DOI 10/ggc3sq

Baehrens D, Schroeter T, Harmeling S, Kawanabe M, Hansen K, Müller KR (2010) How to explain individual classification decisions. Journal of Machine Learning Research 11(61):1803–1831, URL http://jmlr.org/papers/v11/baehrens10a.html

Baldi P, Sadowski P, Whiteson D (2014) Searching for exotic particles in high-energy physics with deep learning. Nature Communications 5(1):4308, DOI 10/f6c8jp

Belkin M, Niyogi P (2003) Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. Neural Computation 15(6):1373–1396, DOI 10/bbr9cw

Björklund A, Henelius A, Oikarinen E, Kallonen K, Puolamäki K (2019) Sparse Robust Regression for Explaining Classifiers. In: Discovery Science, Springer International Publishing, Lecture Notes in Computer Science, vol 11828, pp 351–366, DOI 10.1007/978-3-030-33778-0_27

Cranor LF, LaMacchia BA (1998) Spam! Communications of the ACM 41:74–83

Cunningham JP, Ghahramani Z (2015) Linear Dimensionality Reduction: Survey, Insights, and Generalizations. Journal of Machine Learning Research 16(89):2859–2900, URL http://jmlr.org/papers/v16/cunningham15a.html

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), pp 598–617, DOI 10.1109/SP.2016.42

Deng L (2012) The MNIST database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine 29(6):141–142

Diaz-Papkovich A, Anderson-Trocmé L, Gravel S (2021) A review of UMAP in population genetics. Journal of Human Genetics 66(1):85–91, DOI 10/gmmd69

Finnish Grid and Cloud Infrastructure (2021) Urn:nbn:fi:research-infras-2016072533

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, URL http://jmlr.org/papers/v20/18-760.html, 1801.01489

Fong RC, Vedaldi A (2017) Interpretable Explanations of Black Boxes by Meaningful Perturbation. 2017 IEEE International Conference on Computer Vision (ICCV) pp 3449–3457, DOI 10.1109/iccv.2017.371, 1704.03296

Goldstein A, Kapelner A, Bleich J, Pitkin E (2014) Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation. URL http://arxiv.org/abs/1309.6392, 1309.6392

Guidotti R, Monreale A, Ruggieri S, Turini F, Giannotti F, Pedreschi D (2018) A Survey of Methods for Explaining Black Box Models. DOI 10.1145/3236009

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

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, DOI 10/gn3rtq

Hastie T, Tibshirani R, Friedman J (2009) The Elements of Statistical Learning, 2nd edn. 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):1503–1529, DOI 10.1007/s10618-014-0368-8

Henelius A, Puolamäki K, Ukkonen A (2017) Interpreting Classifiers through Attribute Interactions in Datasets. In: Proceedings of the 2017 ICML Workshop on Human Interpretability in Machine Learning (WHI 2017), URL http://arxiv.org/abs/1707.07576, 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, DOI 10/gn3rtr

Kobak D, Berens P (2019) The art of using t-SNE for single-cell transcriptomics. Nature Communications 10(1):5416, DOI 10/ggdrfz

Kruskal JB (1964) Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika 29(1):1–27, DOI 10/dk9pcw

Laugel T, Renard X, Lesot MJ, Marsala C, Detyniecki M (2018) Defining Locality for Surrogates in Post-hoc Interpretablity. URL http://arxiv.org/abs/1806.07498, 1806.07498

Liese F, Vajda I (2006) On Divergences and Informations in Statistics and Information Theory. IEEE Transactions on Information Theory 52(10):4394–4412, DOI 10/bccwtp

Lundberg SM, Lee SI (2017) A Unified Approach to Interpreting Model Predictions. In: Advances in Neural Information Processing Systems, vol 30, pp 4765–4774, URL http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions

van der Maaten L, Hinton G (2008) Visualizing Data using t-SNE. Journal of Machine Learning Research 9(86):2579–2605, URL http://jmlr.org/papers/v9/vandermaaten08a.html

McInnes L, Healy J, Melville J (2018a) UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. URL http://arxiv.org/abs/1802.03426, 1802.03426

McInnes L, Healy J, Saul N, Grossberger L (2018b) UMAP: Uniform manifold approximation and projection. The Journal of Open Source Software 3(29):861, DOI 10/gf6k3s

Mead A (1992) Review of the Development of Multidimensional Scaling Methods. Journal of the Royal Statistical Society: Series D (The Statistician) 41(1):27–39, DOI 10/bj5z9m

Molnar C (2019) Interpretable Machine Learning. Leanpub, URL https://christophm.github.io/interpretable-ml-book

Nocedal J (1980) Updating quasi-Newton matrices with limited storage. Mathematics of Computation 35(151):773–782, DOI 10/fg4z49

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, pp 8024–8035, URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay É (2011) Scikit-learn: Machine Learning in Python. Journal of Ma-

chine Learning Research 12(85):2825–2830, URL http://jmlr.org/papers/v12/pedregosa11a.html

Ribeiro MT, 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, DOI 10/gfgrbd

Ribeiro MT, Singh S, Guestrin C (2018) Anchors: High-Precision Model-Agnostic Explanations. Proceedings of the AAAI Conference on Artificial Intelligence 32(1), URL https://ojs.aaai.org/index.php/AAAI/article/view/11491

Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500):2323–2326

Shapley LS (1951) Notes on the N-Person Game — II: The Value of an N-Person Game. RAND Corporation, DOI 10.7249/RM0670

Tenenbaum JB, de Silva V, Langford JC (2000) A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science 290(5500):2319–2323, DOI 10/cz8wgk

van der Maaten L (2014) Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research 15(93):3221–3245, URL http://jmlr.org/papers/v15/vandermaaten14a.html

Zhang Z, Wang J (2006) MLLE: Modified Locally Linear Embedding Using Multiple Weights. In: Proceedings of the 19th International Conference on Neural Information Processing Systems, MIT Press, NIPS'06, vol 19, pp 1593–1600, URL https://proceedings.neurips.cc/paper/2006/hash/fb2606a5068901da92473666256e6e5b-Abstract.html