SLISEMAP

SUPERVISED DIMENSIONALITY REDUCTION THROUGH LOCAL EXPLANATIONS

Anton Björklund, Jarmo Mäkelä, Kai Puolamäki.

SLISEMAP: Supervised dimensionality reduction through local explanations.
Machine Learning 112, 1–43 (2023). DOI: 10.1007/s10994-022-06261-1.

https://github.com/edahelsinki/slisemap

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

8th April 2024

These notes contain the spoken parts

SLISEMAP is a new supervised dimensionality reduction method, that can be used to explain black box classification or regression models.

SLISEMAP is open source and available as a Python library.

OUTLINE

- Background
- Problem definition and algorithm
- Examples and experiments
- Future work and summary

- This presentation starts with some background and motivation.
- After that we will look at what we do and how we do it.
- Then some examples, discoveries, and other results.
- We will wrap things up with a future outlook.

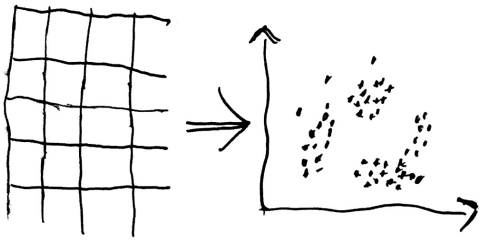
BACKGROUND

Manifold visualisation **+** **Local Explanations**

Lets start by briefly looking at two types of methods that inspired SLISEMAP: Manifold visualisation and local explanations.

BACKGROUND

Manifold visualisation **+** **Local Explanations**

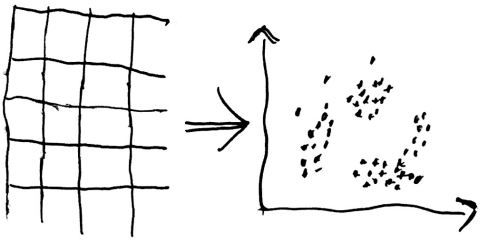


Low dimensional embeddings of high dimensional data.

- The goal of manifold visualisation is to turn a high-dimensional dataset into a lower-dimensional embedding that preserves some of the patterns in the data.
- This is commonly used for visualisations, for example in astronomy, genetics, or linguistics.

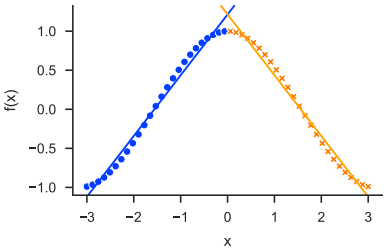
BACKGROUND

Manifold visualisation **+**



Low dimensional embeddings of high dimensional data.

Local Explanations

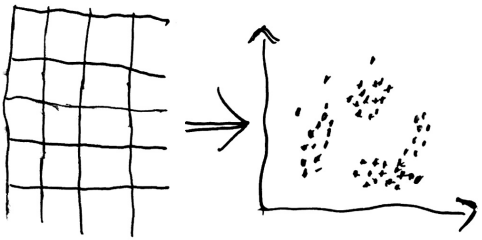


We can approximate this non-linear function with two local linear models.

- Local explanations explain particular predictions from black box machine learning models.
- A common approach for model-agnostic local explanations is to locally approximate the complex model with a simpler one.
- Here is an example of how to use linear models to approximate a more complex function.

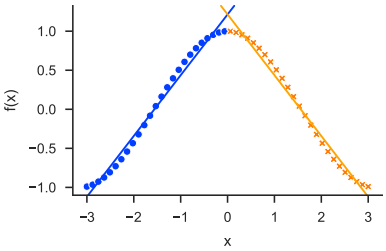
BACKGROUND

Manifold visualisation **+**



Low dimensional embeddings of high dimensional data.

Local Explanations

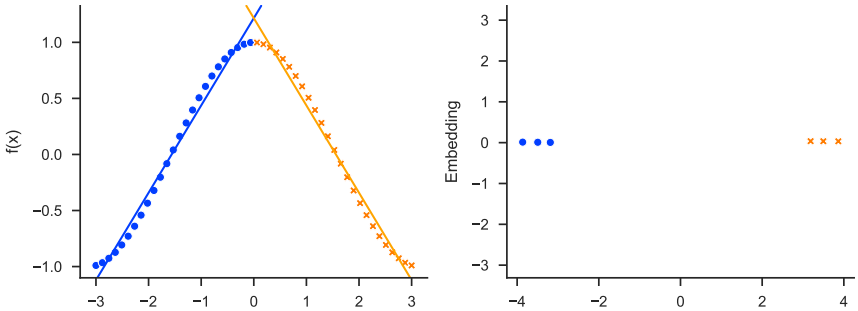


We can approximate this non-linear function with two local linear models.

= *"Find an embedding that groups data items with similar local models".*

With SLISEMAP we combine these two techniques to find an embedding and a local model for every data item, such that data items with similar models end up next to each other in the embedding.

SIMPLE EXAMPLE



SLISEMAP finds two groups of points with different local models.

- Here we apply SLISEMAP to the simple example from the previous slide.
- We see the two expected models.
- And an embedding with two clusters.
- Interesting to note is that the linear models intersect at the top of the curve.
- This means that nearby points are approximated by both linear models, but we will come back to that later.

PROBLEM DEFINITION

- Given a dataset of n items: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$.
- Find the embedding coordinates $\mathbf{z}_1, \ldots, \mathbf{z}_n$ and the local models $g_1, \ldots, g_n$ that minimise:

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{e^{-\|\mathbf{z}_i - \mathbf{z}_j\|_2}}{\sum_{k=1}^{n} e^{-\|\mathbf{z}_i - \mathbf{z}_k\|_2}} l(g_i(\mathbf{x}_j), \mathbf{y}_j)$$

- Where l is a loss function for the local models.
- Under the constraint $\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{d} \mathbf{z}_{ik}^2 = r^2$.

Formally, we express the objective of SLISEMAP as a loss function.
The loss function has two parts:

PROBLEM DEFINITION

- Given a dataset of n items: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$.
- Find the embedding coordinates $\mathbf{z}_1, \ldots, \mathbf{z}_n$ and the local models $g_1, \ldots, g_n$ that minimise:

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{e^{-\|\mathbf{z}_i - \mathbf{z}_j\|_2}}{\sum_{k=1}^{n} e^{-\|\mathbf{z}_i - \mathbf{z}_k\|_2}} l(g_i(\mathbf{x}_j), \mathbf{y}_j)$$

- Where l is a loss function for the local models.
- Under the constraint $\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{d} \mathbf{z}_{ik}^2 = r^2$.

First we have the losses from the local models.

PROBLEM DEFINITION

- Given a dataset of n items: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$.
- Find the embedding coordinates $\mathbf{z}_1, \ldots, \mathbf{z}_n$ and the local models $g_1, \ldots, g_n$ that minimise:

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{e^{-\|\mathbf{z}_i - \mathbf{z}_j\|_2}}{\sum_{k=1}^{n} e^{-\|\mathbf{z}_i - \mathbf{z}_k\|_2}} l(g_i(\mathbf{x}_j), \mathbf{y}_j)$$

- Where l is a loss function for the local models.
- Under the constraint $\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{d} \mathbf{z}_{ik}^2 = r^2$.

- Then the local models are weighted based on distances in the embedding.
- We turn the distances into weights using softmax.
- So, a trivial solution is to push the points in the embedding infinitely far apart...

PROBLEM DEFINITION

- Given a dataset of n items: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$.
- Find the embedding coordinates $\mathbf{z}_1, \ldots, \mathbf{z}_n$ and the local models $g_1, \ldots, g_n$ that minimise:

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{e^{-\|\mathbf{z}_i - \mathbf{z}_j\|_2}}{\sum_{k=1}^{n} e^{-\|\mathbf{z}_i - \mathbf{z}_k\|_2}} l(g_i(\mathbf{x}_j), \mathbf{y}_j)$$

- Where l is a loss function for the local models.
- Under the constraint $\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{d} \mathbf{z}_{ik}^2 = r^2$.

- . . . unless we add a constraint on the radius of the embedding.
- Now, items with similar local approximations form clusters and data items with incompatible local models get pushed apart.

PROBLEM DEFINITION

- Given a dataset of n items: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$.
- Find the embedding coordinates $\mathbf{z}_1, \ldots, \mathbf{z}_n$ and the local models $g_1, \ldots, g_n$ that minimise:

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{e^{-\|\mathbf{z}_i - \mathbf{z}_j\|_2}}{\sum_{k=1}^{n} e^{-\|\mathbf{z}_i - \mathbf{z}_k\|_2}} l(g_i(\mathbf{x}_j), \mathbf{y}_j)$$

- Where l is a loss function for the local models.
- Under the constraint $\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{d} \mathbf{z}_{ik}^2 = r^2$.

- As default local models we use linear regression and logistic regression for classification.
- In both cases we also add lasso and ridge regularisation to avoid overfitting in smaller clusters.
- Lasso regularisation also causes sparse models, which is important for interpretability.

ALGORITHM

1. Initialise $\boldsymbol{z}_i, \ldots, \boldsymbol{z}_n$ with PCA
2. Initialise $g_i, \ldots, g_n$ by optimising $\mathcal{L}$
3. Repeat 4–5 until convergence:
4. Escape local optima by greedily reassigning $\boldsymbol{z}_i, \ldots, \boldsymbol{z}_n$
5. Update $(\boldsymbol{z}_i, g_i), \ldots, (\boldsymbol{z}_n, g_n)$ by optimising $\mathcal{L}$ using LBFGS

- The algorithm for SLISEMAP is fairly simple.
- First we initialise some values using PCA.
- Then we have a two-phased optimisation where we alternate between
- optimising the loss function using LBFGS and
- escaping local optima using a greedy heuristic,
- until we don't find any further improvement.

EXAMPLE CODE

Installation

```
pip install slisemap
```

Code

```
from slisemap import Slisemap
# Use Lasso regularisation
sm = Slisemap(X, y, lasso=0.01)
sm.optimise()
sm.plot(clusters=5, bars=6)
```


- SLISEMAP is implemented in Python, using Pytorch for the optimisation.
- The library can be installed via pip.
- Using SLISEMAP requires only a couple of lines.
- The two most important being the creation of the SLISEMAP object and the optimisation.

EXAMPLE CODE

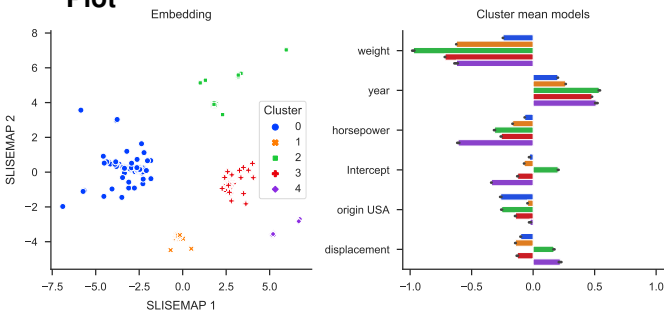
Installation

```
pip install slisemap
```

Code

```
from slisemap import Slisemap
# Use Lasso regularisation
sm = Slisemap(X, y, lasso=0.01)
sm.optimise()
sm.plot(clusters=5, bars=6)
```

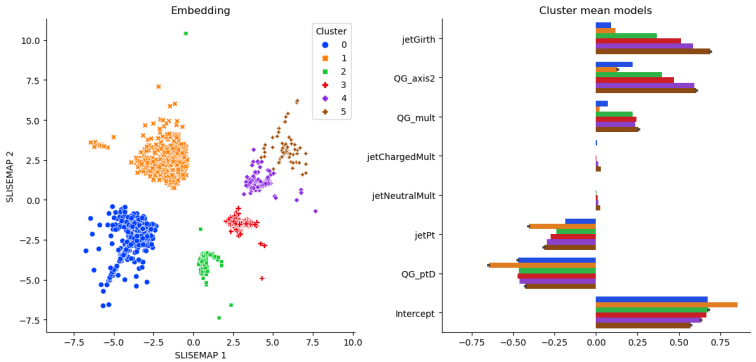
Plot



Cluster the local models to make them easier to read (using a dataset about cars).

- Here we apply SLISEMAP on a dataset of fuel efficiency in cars.
- To make the models easier to read we have clustered the models by their coefficients.
- You can maybe see the small error bars on the coefficients in the barplot.
- This way we don't have to look at all the local models individually.

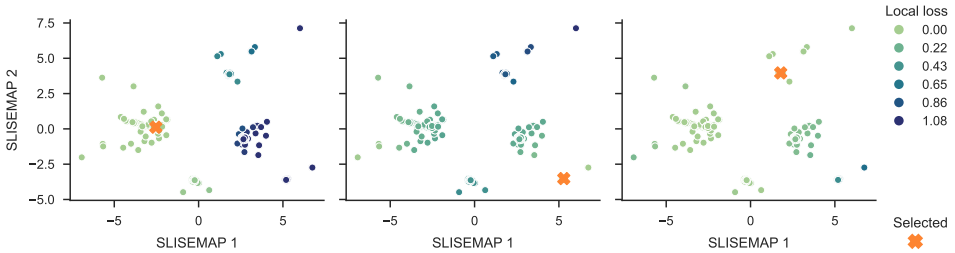
PHYSICS EXAMPLE



Explanations are consistent with quantum chromodynamics theory.

- Here is a second example from high energy physics.
- We use a random forest to classify particle jets.
- The signs of the coefficients are similar in all explanations and consistent with quantum chromodynamics.
- The magnitude differs because of the different composition of the clusters.
- For example, the blue cluster contains high energy jets, where the multiplicity is not a good differentiator.

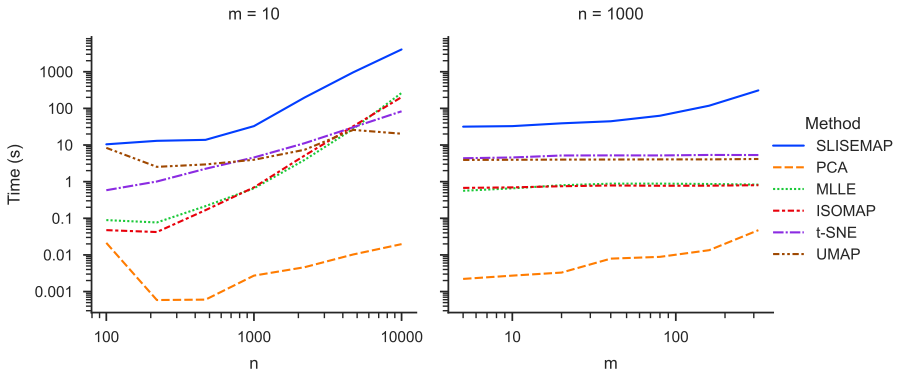
ALTERNATIVE EXPLANATIONS



Some data items have multiple viable local models.

- Going back to the alternative local models from the beginning, but on a real dataset.
- Here we select one data item and colour all other points based on how well that local model works for the selected item.
- We expect local models in the same cluster to work well.
- But we also see that, for some items, models from other clusters have low loss as well.
- This means that one data item might have multiple viable explanations.
- There are natural reasons for this behaviour:
  - Overlapping local models, such as the example from the beginning
  - Correlated variables
  - Explanations with different levels of locality
  - etc..
- With SLISEMAP we can discover and investigate alternative explanations, but an important takeaway from this presentation is that this phenomenon is not unique to SLISEMAP, but inherent in essentially all local explanation methods.

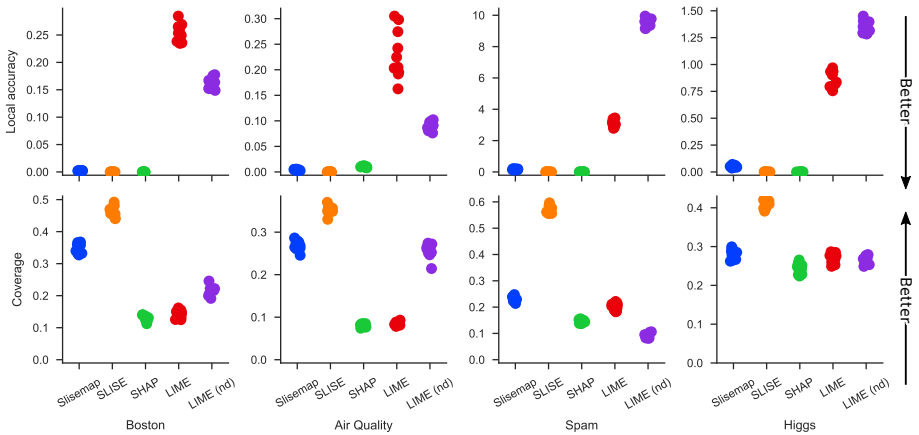
SCALING



Use a GPU and/or subsampling for larger datasets.

- Here we compare SLISEMAP to other dimensionality reduction methods.
- SLISEMAP, like most embedding methods, scales quadratically with the number of items.
- For larger datasets we can use subsampling, which is especially appropriate for visualisations since we are anyway limited by the number of pixels.
- And, with SLISEMAP, new points can be added later to a fixed embedding.
- Furthermore, since SLISEMAP is implemented with PyTorch we can also use a GPU for maybe 2 orders of magnitude faster calculations.

COMPARISON



Inline with some other XAI methods.

- Since the embedding adds additional constraints we need to evaluate the local explanations.
- We compare against some model-agnostic local explanation methods.
- And we find that the local explanations are inline with the other methods using various metrics.

FUTURE WORK

- Investigate particle formation in atmospheric chemistry
- Improve on the scaling
- Investigating alternative explanations

- Our plans for SLISEMAP is to use it on atmospheric molecules to investigate particle formation.
- We also plan a variant with improved scaling.
- Finally, we want to investigate the implications and usefulness of the alternative explanations.

SUMMARY

SLISEMAP is a supervised manifold visualisation method that embeds data items into a lower-dimensional space such that nearby data items share the same white box model.

A. Björklund, J. Mäkelä, K. Puolamäki.
SLISEMAP: Supervised dimensionality reduction through local explanations.
Machine Learning 112, 1–43 (2023).
DOI: 10.1007/s10994-022-06261-1.

https://github.com/edahelsinki/slisemap

- To summarise, SLISEMAP is a new supervised manifold visualisation method that simultaneously finds local models for every data item and constructs an embedding based on the local models.
- Please read our paper for a more in-depth treatment of SLISEMAP.
- And on GitHub you can find documentation for the Python library, as well as tutorial notebooks.
- Thank you for your time and interest.