Reproduction: Explaining Groups of Points in Low-Dimensional Representations

Liang Huang
liang.huang@student.uva.nl

Damiaan Reijnaers
info@damiaanreijnaers.nl

Daniel Van de Pavert
daniel.vandepavert@student.uva.nl

Giguru Scheuer
giguru.scheuer@student.uva.nl

Reproducibility Summary

Scope of Reproducibility

The purpose of the algorithm posed by Plumb et. al is to find human-understandable explanations, in terms of given explainable features of input data, for differences between groups occurring in a lower-dimensional representation of that input data.

Methodology

We have upgraded the original code such that it is compatible with the latest popular code libraries, among which the `Tensorflow 2.x`-library. Furthermore, we have created our own implementation of the algorithm in which we have incorporated additional experiments in order to evaluate the algorithm's relevance in the scope of different dimensionality reduction techniques and differently structured data. We have performed the same experiments as described in the paper using both versions of the code.

Exprimentation was done on a Macbook with 2GHz quad core Intel i5 CPU and 16GB 3733 MHz RAM memory. Depending on the dataset, training can take up to 3 hours for the RNA data.

Results

The ideas and results presented in the original paper were reproducible, both by using the provided code and our own implementation. Our additional experiments have highlighted several limitations of the explanatory algorithm posed by Plumb et. al: the algorithm severely relies on the shape and variance of the clusters present in the data (and, if applicable, the method used to label these clusters), and highly non-linear dimensionality reduction algorithms perform worse in terms of explainability.

What was easy

The authors have shared their code and datasets via Github[1]. The code cleanly separates implementations on different datasets (in different notebooks in separate folders) and the shared core functions. As a result of this separation, one could easily apply the code on their own data set as well given that the VAE based on the external `scvis`-library is used.

What was difficult

The minor difficulties encountered were the missing environment set-up instructions and usage of absolute paths. In short, it took one person approx. 2 days to figure out what would make the code run on their device and replicate.

The authors' artifacts depended on the `Tensorflow`-model created in the python package `scvis 0.1.0`[2]. Since the DR algorithms from `sklearn` are not based on `Tensorflow`, it was required to rewrite all of the original authors

[1] GitHub, *Explaining Low Dim. Representations*, `https://github.com/GDPlumb/ELDR`, accessed on January 29th, 2021

[2] GitHub, *shahcompbio/scvis: Python package for dimension reduction of high-dimensional biological data*, `https://github.com/shahcompbio/scvis`, accessed on January 22nd, 2021

33rd Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

artifacts to compute the corresponding TGTs. Rewriting and reproduction for different DR algorithms and PyTorch on different data sets took a four person team two weeks.

Communication with original authors

No communication with the original authors was required to reproduce their work.

1 Introduction

As AI models are getting more integrated into applications with economical or social implications, the need for *explaining* decisions made by (potentially complex) models is increasing. In many AI applications, big datasets form the basis of a decision-making algorithm (3). As these datasets often involve data of high dimensionality, the dimensionality of data is, in many different artificially intelligent applications, often reduced using *dimensionality reduction* techniques (6).

Data, and consequently the decisions made by an algorithm which are (in)directly based on that data, often involve some sort of 'grouping;' either by utilizing some sort of *clustering method* or by (manually) pre-defined 'clusters of data.' The 'grouping process' may happen before the dimensionality reduction, for which, assuming well-organized data in which the dimensions correspond to explainable 'real-life' *features* of the phenomena measured in the data, the distinguishing characteristics of a certain group becomes relatively explainable as these groups are marked by boundaries in terms of explainable dimensions. However, this process can also happen after the dimensionality of the data has been reduced, which results in the groups being defined in terms of dimensions in a *latent space*. Especially when grouping occurs after dimensionality reduction, different clusters in data often play a key role in the decision-making process of an algorithm – one could for example, when regarding *credit risk modelling*, point out a group in latent space which 'poses a high risk' when provided a loan.

Thus, considering the above mentioned need for explainable machine decisions, in combination with the increasing use of dimensionality reduction algorithms and the reliance on observed data clusters in abstract latent spaces which are not understandable to the human mind, it is of great relevance and importance to develop a method for the purpose of explaining differences between different groups in the light of the application of a certain dimensionality reduction technique.

Elaborating upon the earlier introduced example in credit risk modelling, one could argue that a good explanation for a machine decision is of the kind: *"Your loan was rejected, but if you would have earned €422,86 more per month, your loan would have been accepted."*. This type of explanation is a *counterfactual explanation* – a decision is made on basis of a 'fake,' counterfactual, 'world,' in which features *would have* been shaped differently. The proposed explanatory technique by Plumb et al. (2020) does exactly this: it reverse-engineers the obtained cluster labels in latent space to label the corresponding data in original space; then 'tweaks' an initial cluster by translating the initial cluster in original space, so that it is mapped to (approximately) the same point in latent space. Since the dimensions in original space correspond to explainable features—the characteristics of a certain group—which are understandable for the human mind, a translation, which corresponds to merely adding or subtracting values to each of these features, can be perfectly explained in 'human language.' Since it is desirable for the explanations to be concise, the translation should be as *sparse* as possible – the translation should 'tweak' as few dimensions as possible. The intuition behind this idea is visualized in figure 1.

2 Scope of reproducibility

As follows from the introduction, the authors opted for a counterfactual, sparse explanation for the key differences between groups. They have extended their algorithm to explain differences globally between all groups, which they call Transitive Global Translations (TGT). The main tested contributions are that "TGTs provide a *global* and *counterfactual* explanation between all groups which is mathematically consistent (i.e. symmetrical and transitive)." and that "TGTs overcome the shortcomings of statistical and manual interpretation which do do not use the model that learned the low dimensional representation that was used to define the groups in the first place."

In addition to replicating these claims, we further evaluated these claims by testing compatibility with other dimensionality reduction models than the used variational autoencoder; testing the applicability on different datasets with different internal structures; testing the reproducibility of the experimental DR method, by creating a PyTorch implementation, whereas the original code is written in Tensorflow, because (2) shows that PyTorch is becoming the preferred framework by researchers.

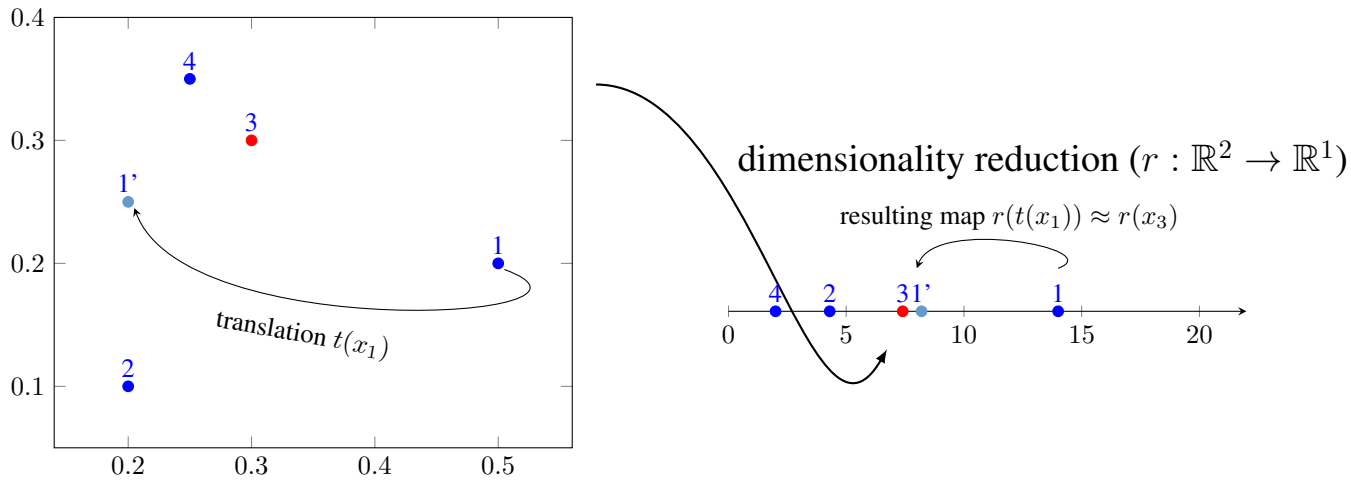


Figure 1: Visualisation of method described by Plumb et. al (2020): the difference between point 1 and 3 is explained each data point can be seen as an individual cluster. Note that although point 1 maps to $1'$, which is far from point 3 in original space, point $1'$ is almost identical to 3 in latent space – this is the goal, as we want to find a point $1'$ which maps 1 as close to 3 in latent space while keeping the translation from 1 to $1'$ as sparse as possible. Note that the shown clusters (1-4) are determined in latent space, and back-engineered to original space.

3 Methodology

For the purpose of reproducing (4), several steps were taken in order to best cover the scope of the original paper with limited resources. The dimensionality reduction algorithm used in the original paper is a Variational Auto-Encoder (VAE), which is a non-linear family of dimensionality reduction algorithms based on (deep) symmetrical neural networks with a *bottleneck* in the middle layers which form the latent representation of the input data. The implemented VAE is provided by an external library, *scvis*, by Ding et al.(1)

Firstly, the provided artifacts[3] were 'modernized' in order for them to run on our equipment using the most recent versions of the involved Python libraries at publication of this paper. The code provided by Plumb et al. and the external library `scvis` were upgraded to be compatible with `tensorflow 2.4.1`[4] and `matplotlib 3.3.3`[5].

Secondly, the original code was rewritten from scratch — both of the `scvis` library and the code complementing the original paper — for the purpose of making computing the explanations independent of the `scvis` library, as the `scvis` library only corresponds to variational autoencoders, while the presented idea in the original paper should work with any kind of dimensionality reduction technique. For the same reason, we have opted to rewrite the VAE architecture of `scvis` in `PyTorch 1.7.1` instead of `TensorFlow`, as PyTorch offers clearer coding structure[6] and is preferred by researchers publishing in the major conferences (2).

Thirdly, we have run our version of the algorithm on different dimensionality reduction algorithms—both linear and non-linear (see section 3.1) — and on different datasets (see section 3.3). Both the reproduced version (using original code) and the rewritten version are further explained in section 3.2.

3.1 Dimensionality reduction (DR) algorithms

Throughout the paper, Plumb et. al mention DR algorithms generally. Only in the section Experimental Results, the VAE by Ding et al. (1) was introduced, which is the only DR method used. As mentioned above, the explanatory method presented by Plumb et. al *should* work for any DR algorithm while effectively treating the algorithm as a 'black box.' Apart from testing this hypothesis, it is relevant to compare 'explanatory performance' on different DR algorithms as different algorithms suit different types of data. Furthermore, an explanation based solely on translations, could

[3]Github, `https://github.com/GDPlumb/ELDR`, accessed on January 22nd, 2021

[4]TensorFlow, *Automatically upgrade code to TensorFlow 2*, `https://www.tensorflow.org/guide/upgrade`, accessed on January 22nd, 2021

[5]Matplotlib, *API Changes*, `https://matplotlib.org/3.3.3/api/api_changes.html`, accessed on January 22nd, 2021

[6]Medium, `https://towardsdatascience.com/pytorch-vs-tensorflow-spotting-the-difference-25c75777377b`, accessed on January 25th, 2021

possibly perform worse in situations wherein data is transformed in a non-linear manner. Especially when methods inherently, non-linearly transform (and warp) the input space. Therefore, as an addition to simply reproducing the implementation of Plumb et al., experiments were done with several commonly known DR algorithms listed below.

3.1.1 Linear methods

We have opted for two different linear DR algorithms: Truncated SVD (TSVD) and Sparse PCA (SPCA). Both TSVD and SPCA reduce the dimentionality in a linear fashion but a key difference between the two is that SPCA centers the data before computing the decomposition where TSVD does not. As a result, TSVD works more efficiently with sparse data.

The objective of Sparse PCA is to find the sparse components that can best reconstruct the data. In many cases, the original PCA extracts the components with dense expressions but these are hard to interpret. However, the sparse vectors extracted by Sparse PCA can naturally match the latent components and it is more explainable.

3.1.2 Non-linear methods

The original authors use an implementation of a Gaussian Variational Autoencoder model for DR called scvis by (1). The probabilistic generative model learns a parametric mapping from the high dimensional space to a lower dimensional embedding. The low dimensional representations by scvis preserve both local and global neighbor structures in the data.

Kernel PCA (KPCA) is a non-linear variant of PCA, which can achieve non-linear dimensionality reduction with the extension of different kernels. The advantage is its ability to identify clusters which are not linear separable. For our experiments, we found using a sigmoid kernel provided the greatest coverage and correctness when the latent space resulting from the KPCA was used in the TGT algorithm.

Finally, we used two manifold bases DR algorithms to provide results supporting our claim of linearity. The first manifold method used is Isomap which can be seen as an extension of Kernel PCA. Isomap maps to a lower dimensional space while maintaining distances between all points. The second manifold method we used is Locally Linear Embedding (LLE). LLE aims to map to a lower dimensional space while maintaining the distance in local neighborhoods. While Isomap could be seen as a extension of KPCA, LLE can be understood as a combination of PCAs ran on local neighborhoods.

As shown in section 4.3, different latent spaces (resulting from applying different processes for the purpose of reducing dimensions) seem to perform better in terms of explainability, which is further discussed in section 5.

3.2 Model descriptions

The original paper aimed to find an explanation for the key differences between a pair of groups in latent space where the explanation is expressed in terms of the original dimensions. This explanation is represented by a counterfactual translation of one of the groups (in the pair) in original space: "*what if* all points in group A, thus $\forall x \in A, x \in \mathbb{R}^d$, would have been translated, so that $\forall x \in A, \delta \in \mathbb{R}^d, x' = x + \delta$? Would group A have been roughly the same as group B *after* the groups have been mapped to latent space, so that $\forall x \in A, \forall y \in B, x \in \mathbb{R}^d, y \in \mathbb{R}^m, r : \mathbb{R}^d \rightarrow \mathbb{R}^m, r(x + \delta) \approx r(y)$?".

As both the original space ($\mathbb{R}^d$) and latent space ($\mathbb{R}^m$) are Euclidean spaces, translations between any pair of groups within a larger number of groups (> 2) can be constructed by composing two translations (a vector addition in this context) or negating a translation (in this context equivalent to vector-scalar multiplication with $\lambda = -1$). Using these operations and a *reference group*, explanations (δ) can be obtained for every possible pair of groups. The intuition behind this idea is illustrated in figure 2. Points in figure 2 do not directly correspond to the groups for which we want to find differences, since these locations will be approximated using sparse translations (see figure 1). Moreover, this linearity enables the possibility to measure the difference between the points in the two groups in latent space using the l_2-norm of the squared differences, while it is also possible to measure the *sparsity* of δ using l_1-regularization, which directly relates to the comprehensibility of the explanation posed by δ.

In order to obtain adequate components for the δ-vector between a group and the reference group, all δ-vectors can be initialized to zero, and optimized by adjusting the components of δ corresponding to two randomly chosen groups in the negative direction of the gradient of a loss function which accounts for the above mentioned constraints. This loss function is formulated in equation 9 of the original paper. While Plumb et. al decide to incorporate the calculation of the gradient as an 'extension' of the `scvis` package, which we have reproduced using the provided code, we have decided to analytically compute the gradient using `SciPy 1.6.0` (using the `optimize.approx_fprime`-method). This implementation choice is in line with our aim to rewrite all code from scratch in order to obtain a 'stand-alone'

division between the explanatory algorithm, which is the main idea of the original paper, and various 'black-box' dimensionality reduction algorithms, including the variational auto-encoder (the only algorithm Plumb et. al experiment with), as introduced in the beginning of this section.

The model performance is measured by two separate, but related (see section 4.3), metrics defined by Plumb et al.: *correctness*, which measures the degree of which projected points are actually *in the vicinity* of points they should map to (which they, in a sense, should 'imitate') and *coverage*, which measures the degree to which projected points *cover* the target group.

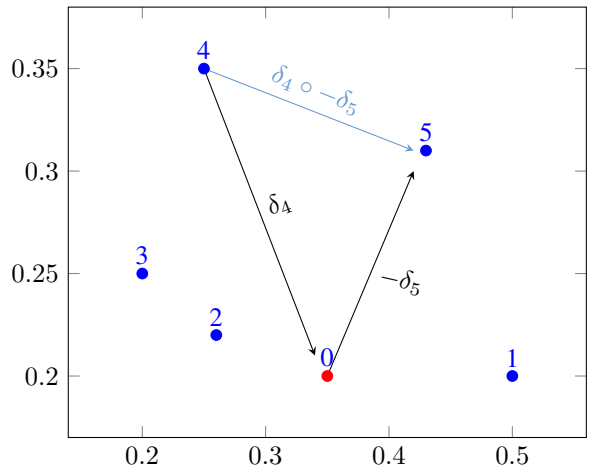


Figure 2: Visualisation of the intuition behind the composition of two translations with regard to a reference group 0, of which one is negated. Note that in this example every data point represents a whole group.

3.3 Datasets

In the upgraded-code model we decided to reproduce the explanations on all provided datasets, including the synthetic and corrupted versions. The datasets used in the experiments in the original paper are the Heart Disease dataset[7]; the Boston Housing set[8]; the Iris dataset[9]; and the single-cell RNA dataset (5). In our from-scratch model, we have incorporated all provided datasets except for the latter, which was due to lack of time. However, experiments were run on three additional UCI datasets: Seeds[10], Wine[11] and Glass[12]. The additional datasets allowed us to further understand how different structures in data can influence the explainability using different dimensionality reduction algorithms. Not all datasets can be reduced by all techniques: all three added datasets do not yield eigencomponents for a sigmoidial Kernel-PCA procedure.

3.4 Hyperparameters

One constraint on the resulting explanations is that they must be sparse. The sparsity level is formally defined by k; representing the number of dimensions in the original space (with 'real-life' features) used as the main components of the translation which forms the explanation. The hyperparameter k is enforced *after* the learning process by truncating the δ-vector. To enforce sparsity *during* the learning process, the δ-vector is being l_1-regularized by a term λ, as follows from (4, eq. 9) and was inspired by the field of *compressed sensing* (see the next section). Lastly, hyperparameter ϵ defines a threshold for the metrics *correctness* and *coverage*.

3.5 Experimental setup and Computational requirements

We have reproduced the experiments presented in the paper by Plumb et. al, by using the upgraded `Tensorflow`-code, as explained in the introduction of this section. Additionally, we have ran our from-scratch code on three additional datasets and six other dimensionality reduction algorithms. All experiments are ran for five trials for eleven different values of λ (evenly spaced between 0 and 5), for which the best combination of δ-vectors is chosen for every different value of k (k is evenly spaced between 1 and the number of dimensions d in the original space, with a step size of 1 for $d \leq 5$, and 2 otherwise).

Following the provided code by Plumb et. al, ϵ is set to 1. This poses a problem if the data in latent space is of low variance (especially when $< \epsilon$), as the performance measures, introduced in section 3.2, would unjustly report very high scores. We have solved this problem by re-scaling the data in latent space so that a variance of 10 is preserved (which amounts to a standard deviation of ≈ 3.16). We have deliberately chosen to not utilize a method for removing *outliers* prior to defining the factor with which to rescale the data in latent space, as to preserve 'the spirit of the data,' especially since the internal structure of some datasets contain outliers which potentially correspond to groups with a significantly different character, as opposed to outliers resulting from noisy measurements. We further discuss this

[7] UCI Machine Learning Repository, *Heart Disease Data Set*, `https://archive.ics.uci.edu/ml/datasets/heart+disease`, accessed on January 29th, 2021; which also applies to the following four:

[8] *Boston Housing*, `https://archive.ics.uci.edu/ml/machine-learning-databases/housing/`

[9] *Iris Dataset*, `https://archive.ics.uci.edu/ml/datasets/iris`

[10] *seeds Data Set*, `https://archive.ics.uci.edu/ml/datasets/seeds`

[11] *Wine Data Set*, `https://archive.ics.uci.edu/ml/datasets/Wine`

[12] *Glass Identification Data Set*, `https://archive.ics.uci.edu/ml/datasets/glass+identification`

choice in section 5. However, as we failed to preserve an adequate amount of variance when performing *TSVD* (see section 3.1) in the *Glass*-dataset (see section 3.3) using this method, we manually scaled the data in this latent space, for this particular dataset, with an additional factor of 20.

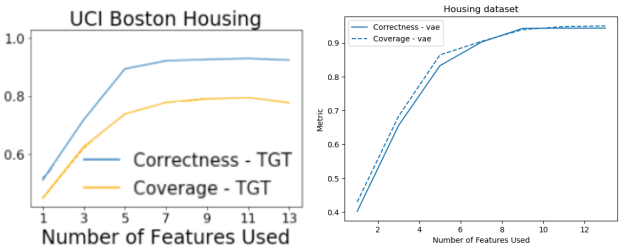
Exprimentation was done on a Macbook with 2GHz quad core Intel i5 CPU and 16GB 3733 MHz RAM memory. The original code, the 'modernized' version of the original code and the code for the from scratch implementation can be found at `https://github.com/giguru/fact-ai`.

Digression: creative inspiration from another field

In many of today's application, *compressing* plays a key role. Almost every photo we see, or song that we listen to, is a *regenerated* version from a saved *compressed* version of that photo or song. In this sense, we could perceive the saved version as the 'sparse' version, from which we would like to regenerate the original. Often, this can be done with only minimal loss, for example when *Fourier series*[13] are utilized. Reflecting this idea on the original paper by Plumb et al., one could say that the groups in the latent space is the 'sparse representation' of these same groups in original space—which were not labeled before being mapped to latent space—from which the original groups are being reproduced. In a way that requires some creative thinking, this is exactly what we are attempting to do.

4 Results

Using our upgraded `TensorFlow`-code and the provided pre-trained VAE models, the obtained results are identical to those provided by the codebase of the paper for all datasets and methods[14]. After retraining all VAE models and re-learning all explanations, using the same upgraded `TensorFlow`-code, we obtain very similar results[15]. The results are not identical, as the models learn slightly different representations, in which the manually selected clusters also differ. This indicates that the used cluster generation techniques influences the model's ability to explain based on a translation. An example arises with the Iris dataset: Plumb et. al yield 0.833 coverage, while we get only 0.66. This difference caused by a different organization of clusters propagates further into the results for the corrupted data[16]. Except for small differences arising from this same issue, we have successfully reproduced the results for all other datasets.



(a) Plumb et. al (b) From-scratch implementation

Figure 3: Comparison of results for Housing data on VAE

When running the Housing, Iris and Heart datasets on our own from-scratch implementation of the explanation algorithm using our own implementation of `scvis`, we obtain either similar (see figure 3) or even better results than Plumb et. al do. This once again indicates that the method for dimensionality reduction does impact the results.

For different datasets, and different dimensionality reduction algorithms which map the original datasets to different latent spaces, we have encountered the same problem related to the inability of explaining pairs of groups of which the clusters have a different standard deviation, as outlined by the original paper in figures 4-6 (4, p. 4). This problem occurs, among others, in figure 5f in Appendix A (and the corresponding corresponding measures in figure 6f).

4.1 Additional results not present in the original paper

As (4, p. 6) pointed out, the metrics *correctness* or *coverage* are closely related. If the translations between arbitrary groups are symmetrical, the former translation—which is the technical representation of the explanation—is the negative of the latter (and vice-versa). Thus, when applying different **linear** dimensionality reduction algorithms, both metrics

[13] Fourier series are equivalent to *Taylor series* in complex space. In real space, Fourier series can approximate *any* function be only using sinusoids. The storage of the coefficients of these sinusoids are an efficient way to store a signal (function), since the original function can be generated based on these coefficients solely.

[14] To compare, open the *.ipynb* in all folders (except 'code,' 'scvis,' 'MiscFigures,' and 'Integrated-Gradients-master') on both repositories: `https://github.com/GDPlumb/ELDR` and `https://github.com/giguru/fact-ai/tree/main/ELDR-TF2.x_(pre_trained_models)`

[15] Now, compare with `https://github.com/giguru/fact-ai/tree/main/ELDR-TF2.x_(newly_trained_models)`

[16] See: `https://github.com/giguru/fact-ai/blob/main/ELDR-TF2.x_(newly_trained_models)/Iris/run.ipynb`

will exactly equal each other, as follows from the results shown below, in figure 6 (along all different values for k, the graphs conveying correctness and coverage are on top of each other for all linear algorithms) and figure 4 (note the color map plot for correctness is the 'transpose' of the plot for coverage, and vice-versa).

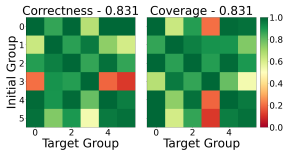


Figure 4: Metrics plot for Housing on PCA, $k = 5$

An additional observation is that topology-based dimensionality reduction techniques are less suited for data compression, when the compressed data consequently needs to be explained using translation-based counterfactual explanations, such as presented by Plumb et al. Locally Linear Embedding and Isomapping are reduction methods based on manifold mapping: a one- or two-manifold is embedded in 2 dimensional space, and points of higher-dimensional space are mapped onto this manifold. This often yields problematic data structures. For instance, when the data is represented using a line (a one-manifold) embedded in two-dimensional space in which the one-dimensional line is twirled. Since, we are only considering translations (and no rotations) in this explanatory method, the method is unable to adequately explain differences between clusters present on manifolds in latent space (which are inherently twirled when embedded in a space higher than the manifold).

Furthermore, we have observed that certain datasets might yield constant results when varying the sparsity-constraint. In this case, the compression from high-dimensional to two-dimensional space was (nearly) *lossless* – a single component suffices to explain the difference between any group. This phenomenon is shown in figure 6e in Appendix A; showing that the chemical composition of Wine actually depends on only one latent variable: when using PCA, 99.98% of the variance is explained by only two components (of which 99.81% by the first component). Note that the input space of the Wine dataset spans 13 dimensions.

5 Discussion

We have been able to successfully reproduce and upgrade the code provided by Plumb. et al, which is the implementation of technique presented in (4). We were able to reproduce results by running the pre-loaded models, after re-training these models, and by rewriting the whole algorithm from scratch.

The metrics correctness and coverage depend on the mapping into the latent space. The limitation of the algorithm is the lack of variable freedom: only translations can be used to explain differences between groups. By utilizing different DR techniques, we have shown that not all DR algorithms produce latent spaces in which translations suffice for an explanation between clusters – especially manifold-based algorithms require a more sophisticated type of explanation using for instance rotation and scaling.

A possible solution would be to generate an explanation in terms of a matrix (so that $\boldsymbol{x'} = \boldsymbol{M}\boldsymbol{x} + \boldsymbol{\delta}$ instead of $\boldsymbol{x'} = \boldsymbol{x} + \boldsymbol{\delta}$). However, explanations using translations are directly interpretable for humans as the dimensions of our datasets correspond to explainable features. Incorporating, for example, rotations would come at the cost of explainability in layman's terms.

Furthermore, by using different datasets, we have shown that different DR algorithms do a better job in terms of explainability between clusters. However, another limitation is that data might be structured in different ways which produces different types of clusters. It directly follows from our results that this explanation method heavily relies on the generated clusters, and more importantly, its shapes and variance in the latent space, which is not desirable.

In section 3.5, we opted for an approach in which the latent spaces are dynamically scaled for the purpose of achieving a certain amount of variance in the data, as to obey the fixed hyperparameter ϵ which was introduced in section 3.4. A more robust method would be to dynamically define ϵ based on the variance in the latent space depending on the dataset and, potentially, a proper handling of its outliers.

As VAEs are sampling points in latent space from a distribution which entails that an identical point $\boldsymbol{x}$ in the input space which is repeatedly mapped to latent space will yield different different mappings. It would be interesting to further investigate to what extent explanatory algorithms can 'handle' this variance – to what extent such algorithms could find a stable and unchanging explanation in an ever-changing counterfactual world.

5.1 What was easy

Replicating, i.e. using the original authors code, can be done in 2 working days by a single person. The hardest part of the set up was the obtaining the right version of packages. The paper provides clear examples and results of the effect of various levels of sparsity on the datasets 6 and the pairwise explanation of the metrics. The only latent space available in the paper itself was of the single-cell RNA dataset (5) but due to the output buffer function of Ipython notebooks

and the preview function of GitHub we were able to find the latent space representations of the other datasets used. Furthermore, the code was mostly well written, properly commented and the various files were understandably ordered into folders.

5.2 What was difficult

Although the code required for computing the explanations was separated from the implementations in other folders in which it is applied on the different datasets, Plumb et. al decided to perform experiments using a VAE, and intertwined all code for generating the explanations with the `scvis`-library. This caused the explanatory model to rely on `TensorFlow`, while the idea presented by Plumb et. al does not require deep learning. It is preferred to have more independent components of the model, since the explanatory model should work with other DR algorithms; including those which do not require deep learning as well (see section 3). Only because Plumb et. al chose to use a TensorFlow VAE, his entire repository relied on the deep-learning framework. For that reason, we opted to rewrite everything from scratch, and to provide the dimensionality reduction methods on a stand-alone basis. Since we are aim to reproduce the results provided by the original paper, we also had to rewrite the external `scvis`-library in `PyTorch` (for the sake of keeping clean, separated and object-oriented code). This was very time-consuming as both frameworks are fundamentally different (dynamic vs static graph definition). Simply rewriting line by line was not an option. Rewriting required extensive reading into the documentation of both frameworks, and the `scvis` documentation (1). The whole process of rewriting everything from scratch took approximately two weeks.

Lastly, results for the Bipolar could only be replicated with the provided model configuration file. Although we have retrained the model using the upgraded originally provided code, we have chosen to exclude this dataset for the experiments due to lack of computation power and time.

5.3 Communication with original authors

A short interaction occurred with the author of the paper, in order to gain more insight in the required external libraries as these were not listed in any documentation[17]. Although the author responded very quickly, the issue had already been solved in the meantime.

References

[1] DING, J., CONDON, A., AND SHAH, S. P. Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nature communications 9*, 1 (2018), 1–13.

[2] HE, H. The state of machine learning frameworks in 2019. *The Gradient* (2019).

[3] O'LEARY, D. E. Artificial intelligence and big data. *IEEE Intelligent Systems 28*, 2 (2013), 96–99.

[4] PLUMB, G., TERHORST, J., SANKARARAMAN, S., AND TALWALKAR, A. Explaining groups of points in low-dimensional representations, 2020.

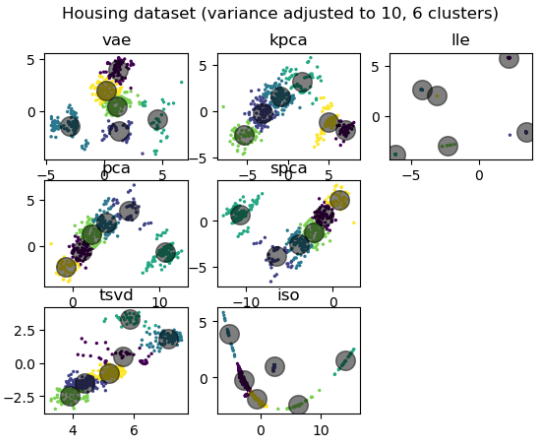
[5] SHEKHAR, K., LAPAN, S. W., WHITNEY, I. E., TRAN, N. M., MACOSKO, E. Z., KOWALCZYK, M., ADICONIS, X., LEVIN, J. Z., NEMESH, J., GOLDMAN, M., ET AL. Comprehensive classification of retinal bipolar neurons by single-cell transcriptomics. *Cell 166*, 5 (2016), 1308–1323.

[6] VAN DER MAATEN, L., POSTMA, E., AND VAN DEN HERIK, J. Dimensionality reduction: a comparative. *J Mach Learn Res 10*, 66-71 (2009), 13.

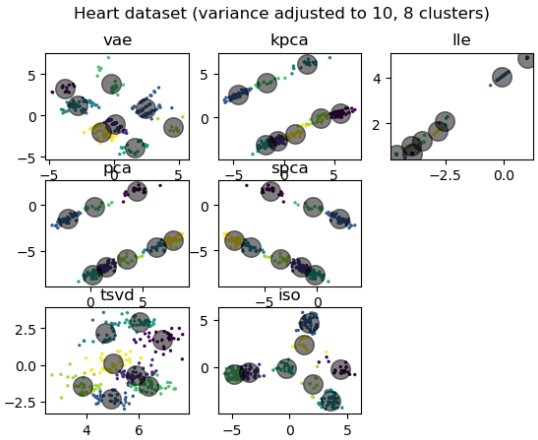
[17] GitHub, *ModuleNotFoundError: No module named 'vae'*, `https://github.com/GDPlumb/ELDR/issues/1`, accessed on January 29th, 2021

Appendices

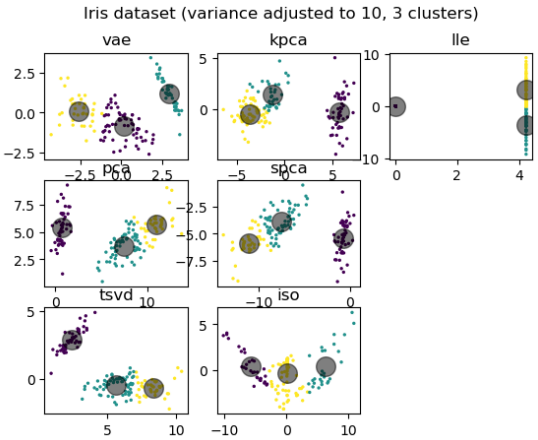
A Additional latent spaces and corresponding measures



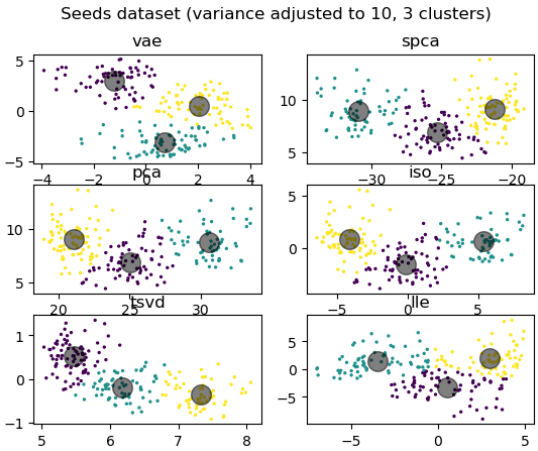
(a) Latent spaces for the Housing dataset



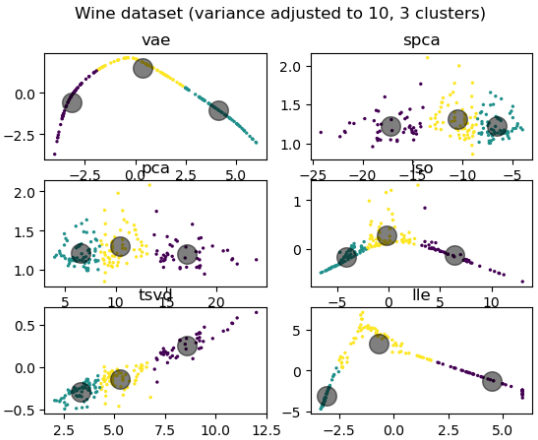
(b) Latent spaces for the Heart dataset



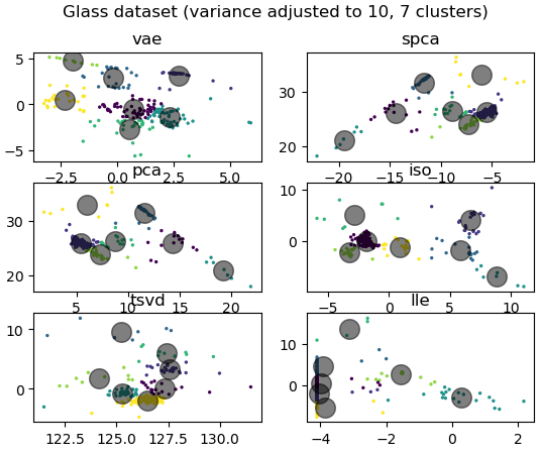
(c) Latent spaces for the Iris dataset



(d) Latent spaces for the Seeds dataset

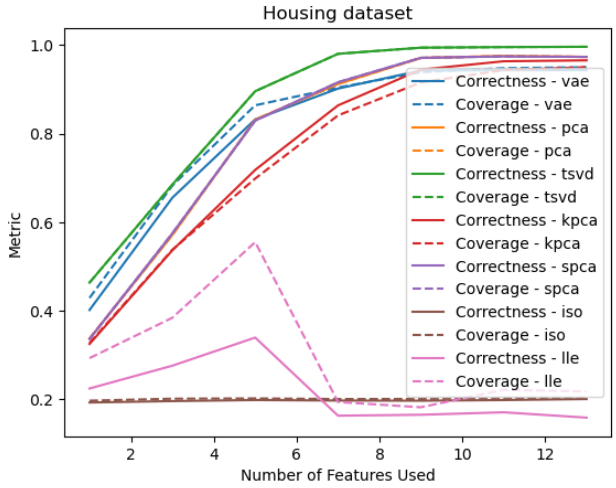


(e) Latent spaces for the Wine dataset

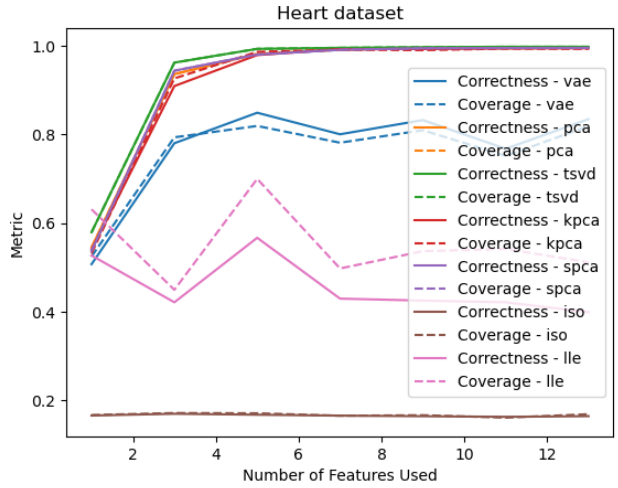


(f) Latent spaces for the Glass dataset

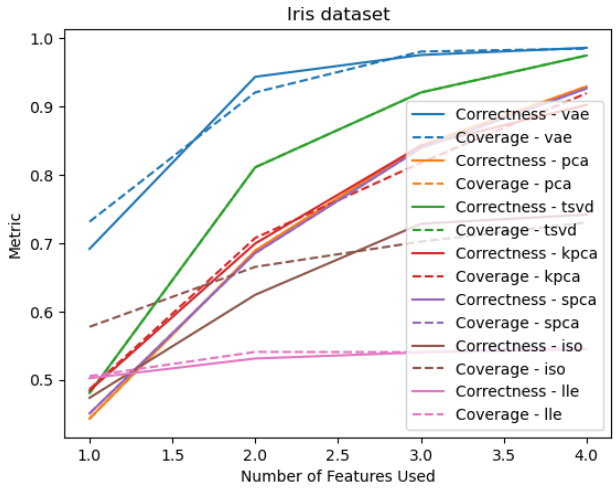
Figure 5: Visualization of the two-dimensional latent spaces in which higher dimensional is compressedly stored after a dimensionality reduction technique has been utilized for different datasets.



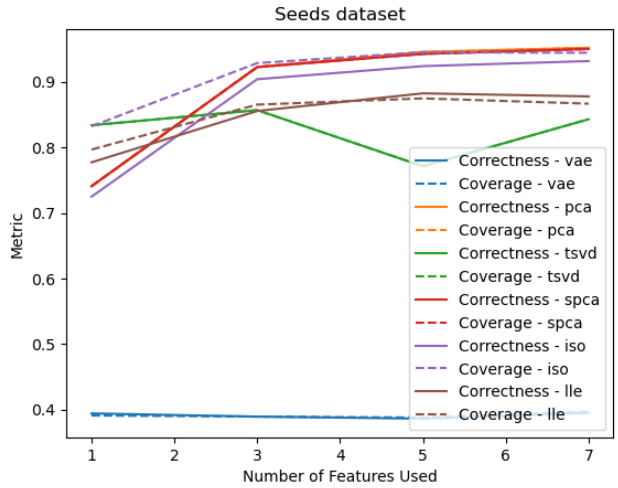
(a) Measures for Housing dataset (all algorithms)



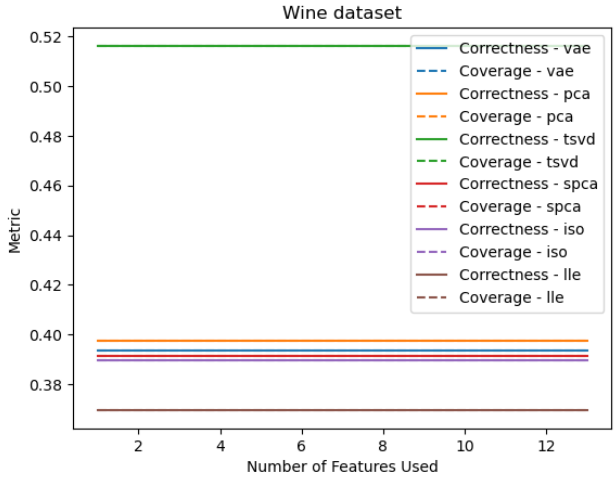
(b) Measures for Heart dataset (all algorithms)



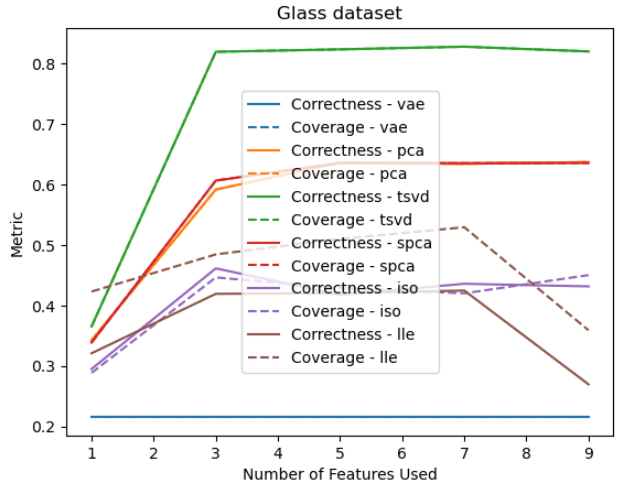
(c) Measures for Iris dataset (all algorithms)



(d) Measures for Seeds dataset (note: no KPCA)



(e) Measures for Wine dataset (note: no KPCA)



(f) Measures for Glass dataset (note: no KPCA)

Figure 6: Measures for coverage and correctness for all involved dimensionality reduction algorithms on all datasets.