Projection Navigation In Extremely Large Datasets (PNIELD)

J. F. Kruiger[1,2], A. C. Telea[1], C. Hurter[2]
[1]University of Groningen, the Netherlands
[2]ENAC, France

November 27, 2017

1 Introduction

Multidimensional projections (MPs) can be used to explore high-dimensional datasets. However, when the number of observations in the dataset becomes very large, MP techniques can become very slow or inaccurate.

Our top-level goal is to make multidimensional projection exploration scalable with respect to the number of observations.

We want to make MPs more scalable in the following aspects:

Projectability: As the number of observations N in an n-dimensional dataset increases, the chances that the dataset lies on a (near) 2D manifold decreases. Making a trustworthy projection of such a dataset is more difficult or even infeasible. That is, the dataset becomes less 'projectable'. Of course, also if the dimensionality n increases, the projectability tends to decrease.

Computational scalability: When there is more data, the evaluation of the projection is going to be slower. Parallelization is often challenging, and will still require many resources—especially if an accurate projection is expected.

Visual scalability: As N increases, then at some point there will simply be insufficient space on the screen to show all projected points. There are several options to solve this problem. One way is to aggregate information from multiple points in the projection, and show the aggregated information. Another way is to only show a representative part of the data, and provide navigation mechanisms. This second approach is the one we took, as explained below.

A framework is needed that improves the projectability and computational scalability of what is being projected, and new navigation methods can help in this regard.

1.1 Contributions

We aim to solve the projectability and computational scalability problems by proposing a framework called PNIELD.

In a nutshell, PNIELD uses subsampling to improve projectability and computational scalability, while offering exploration of the full dataset by means of novel interaction operations to navigate in nD.

Section 2 treats related work, and Section 3 explains our framework.

2 Related work

In this section, a number of preliminaries are explained. First, we will establish what an MP is. A number of MP techniques are explained in more detail. Subsequently, nearest-neighbor algorithms will be discussed shortly. Finally, subsampling will be treated, as it is central to our framework. Along the way, the necessary notation and definitions are established.

2.1 Multidimensional projections

We define an n-dimensional dataset with N observations[1] as a set

$$X = \left\{\mathbf{x}^{(i)}\right\}_{i=1}^{N},$$

where $\mathbf{x}^{(i)} \in \mathbb{R}^n$ are observations in nD:

$$\mathbf{x}^{(i)} = \begin{bmatrix} x_1^{(i)} & x_2^{(i)} & \cdots & x_n^{(i)} \end{bmatrix}.$$

These can be interpreted as points in nD space. As can be seen, parentheses in superscript ($\cdot^{(i)}$) identify the observation, while subscript ($\cdot_i$) identifies the component of a vector.

A multidimensional projection (MP) technique aims to project an nD dataset to an mD dataset, where $m \ll n$. For visualization purposes, m is usually 2 or 3.

An MP technique can be seen as a function that maps sets of nD observations to mD points:

$$\mathcal{P} \colon S_N(\mathbb{R}^n) \to S_N(\mathbb{R}^m),$$

where $S_N(A)$ denotes the set of N-sized subsets of A. Applying $\mathcal{P}$ to a dataset $X = \left\{\mathbf{x}^{(i)}\right\}_{i=1}^{N}$ results in a set of mD points:

$$\mathcal{P}(X) = Y = \left\{\mathbf{y}^{(i)}\right\}_{i=1}^{N}, \quad \text{with } \mathbf{y}^{(i)} \in \mathbb{R}^m .$$

Many MP techniques need parameters. If this is the case, these are supplied as additional arguments to the function $\mathcal{P}$.

[1] Also known as measurements or samples.

MP techniques are designed to preserve relevant patterns from nD to mD. Examples of such patterns are: pairwise distances between observations, neighborhoods, clusters, local variations in neighborhoods, and so forth.

In the following sections, a few MP techniques are explained.

2.1.1 Feature selection

Feature selection can be seen as a (trivial) MP technique. Given a set of observations, it selects only the m features specified in the parameter space:

$$\mathcal{P}_{\text{FS}}\left(X, \{d_1, \ldots, d_m\}\right) = \left\{[x^{(i)}_{d_1}, \ldots, x^{(i)}_{d_m}]\right\}_{i=1}^{N}, \text{ where } 1 \leq d_i \leq n.$$

This is equivalent to projecting the observations to the axes of dimensions $d_1, \ldots, d_m$, and simply ignoring all other dimensions.

2.1.2 Principal component analysis (PCA)

PCA projects observations along the principal components of the data. The principal components are orthogonal directions in nD that describe the shape of the dataset in terms of variance. The projection (along m principal components) is given by:

$$\mathcal{P}_{\text{PCA}}\left(X, m\right) = \left\{\left[\frac{\mathbf{e}^{(1)} \cdot \mathbf{x}^{(i)}}{\|\mathbf{e}^{(1)}\|}, \ldots, \frac{\mathbf{e}^{(m)} \cdot \mathbf{x}^{(i)}}{\|\mathbf{e}^{(m)}\|}\right]\right\}_{i=1}^{N}$$

The principal components $\mathbf{e}^{(i)}$ can be found by computing the covariance matrix of the data matrix $\mathbf{X}$. The data matrix $\mathbf{X}$ is constructed by vertically stacking the observation (row) vectors:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(1)} \\ \vdots \\ \mathbf{x}^{(N)} \end{bmatrix}.$$

It is assumed that the dataset is preprocessed such that it is zero-centered and has unit variance[2] in each dimension.

The covariance matrix $\mathbf{\Sigma}$ of $\mathbf{X}$ is then given by the elements:

$$\begin{aligned} \Sigma_{i,j} &= \frac{1}{N} \sum_{k=1}^{N} X_{k,i} X_{k,j} \\ &= \frac{1}{N} \sum_{k=1}^{N} x^{(k)}_i x^{(k)}_j, \end{aligned}$$

where $X_{i,j}$ denote the elements in the data matrix $\mathbf{X}$.

The principal components $\mathbf{e}^{(i)}$ are the eigenvectors of $\mathbf{\Sigma}$, ordered (in descending order) by their corresponding eigenvalues. That is, $\mathbf{e}^{(1)}$ has the largest

[2]If the dimensions in X have different units—which is often the case—it is advisable to scale each dimension such that it has unit variance.

eigenvalue, hence it is the component that accounts for the largest part of variance in the data.

PCA is successful in explaining the variance in the projections, but—as it is a linear projection—it is unable to 'unfold' more complex patterns in the data, such as in the *swiss roll* dataset [5].

2.1.3 Multidimensional scaling (MDS)

Often, it is useful to have the pairwise distances in mD represent the distances in nD as well as possible. MDS is a class of MP techniques that aims to preserve relative pairwise distances.

Metric MDS (MMDS) algorithms minimize a so-called *stress* function to obtain a projection:

$$\mathcal{P}_{\text{MMDS}}(X, m) = Y = \left\{\mathbf{y}^{(i)}\right\}_{i=1}^{N} \subset \mathbb{R}^m,$$
$$\text{where } Y \text{ minimizes } \sigma(Y) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left(d_n(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) - d_m(\mathbf{y}^{(i)}, \mathbf{y}^{(j)})\right)^2.$$

Here, d_n and d_m are suitable distance functions in nD and mD, respectively. The minimization is usually done by *stress majorization* (*e.g.*, the SMACOF algorithm [3]).

2.1.4 Locally affine multidimensional projection (LAMP)

LAMP [4] is a special kind of MP technique, as it uses k special *landmark points* to make the projection. The landmarks are chosen (usually randomly) from the set of observations:

$$X_{\text{LM}} \subset X, \quad |X_{\text{LM}}| = k.$$

X_{LM} is projected using a 'third-party' projection technique (*e.g.*, MMDS), resulting in a set of projected landmarks Y_{LM}. Since k can be quite small ([4] suggests $\mathcal{O}(\sqrt{N})$), the projection of the landmarks is not very expensive.

Given the landmarks and their projections, the rest of the observations ($X \setminus X_{\text{LM}}$) is projected by a set of locally affine transformations. For more details, refer to [4]. What is important is that observations from $X \setminus X_{\text{LM}}$ are approximately projected close to similar landmark points. That is, the landmarks steer where the non-landmarks are projected.

$$\mathcal{P}_{\text{LAMP}}(X, (X_{\text{LM}}, Y_{\text{LM}})) = Y, \quad Y_{\text{LM}} \subset Y.$$

2.2 Nearest neighbors (NN)

In many techniques that handle high-dimensional data it is useful to reason about neighbors or neighborhoods in nD.

Very often, a search for the k nearest points in a dataset X to a query point $\mathbf{q}$ is done:

$$\begin{aligned} \mathcal{N}^{\mathcal{K}}(\mathbf{q}, X, k) = X' \subset X, &\text{ with } |X'| = k, \\ &\text{such that } d(\mathbf{x}', \mathbf{q}) \leq d(\mathbf{x}, \mathbf{q}), \forall \mathbf{x}' \in X', \forall \mathbf{x} \in X \setminus X', \end{aligned}$$

where d is a suitable distance function.

A related type of search is for a set of points within a given radius r from $\mathbf{q}$:

$$\mathcal{N}^{\mathcal{R}}(\mathbf{q}, X, r) = \{\mathbf{x} \in X \mid d(\mathbf{x}, \mathbf{q}) \leq r\}.$$

When n becomes very large, the only feasible options for NN searches are *approximate* NN (ANN) searches that make a trade-off between accuracy and computational costs. For a recent benchmark of ANN algorithms, refer to [2].

Our implementation uses *Annoy*[3] [1], which is an ANN library that uses a forest of tree structures with random projections.

For our framework, it is useful to be able to search for nearest neighbors to a *set* of points $Q = \left\{\mathbf{q}^{(i)}\right\}_{i=1}^{M}$:

$$\begin{aligned} \mathcal{N}^{\mathcal{K}}_{\text{pointset}}(Q, X, k) = X' \subset X, &\text{with } |X'| = k, \\ &\text{such that } \min_{\mathbf{q} \in Q}(d(\mathbf{x}', \mathbf{q})) \leq \min_{\mathbf{q} \in Q}(d(\mathbf{x}, \mathbf{q})), \\ &\forall \mathbf{x}' \in X', \forall \mathbf{x} \in X \setminus X'. \end{aligned}$$

which is the set of k points from X that are closest to (any point in) Q. The definition of $\mathcal{N}^{\mathcal{R}}_{\text{pointset}}$ follows similarly.

2.3 Subsampling

With large datasets, subsampling is an increasingly important problem: often, it is simply infeasible to consider all data. Since it is useful to have a one-to-one correspondence to original observations, we restrict ourselves to subsampling where the original observations are used:

$$\mathcal{S}(X) \subset X.$$

The domain of the subsampling function is as follows:

$$\mathcal{S}\colon S_N(\mathbb{R}^n) \to S_{N_{\text{S}}}(\mathbb{R}^n),$$

where $S_i(A)$ denotes the set of i-sized subsets of A. N_{S} denotes the number of samples. If additional parameters are needed (such as the number of samples), these are supplied as addition function arguments.

A straightforward way of performing subsampling on a dataset $X = \left\{\mathbf{x}^{(i)}\right\}_{i=1}^{N}$ is by picking $N_{\text{S}} < N$ random samples:

$$\mathcal{S}_{\text{RND}}(X, N_{\text{S}}) = \left\{\mathbf{x}^{(a_i)}\right\}_{i=1}^{N_{\text{S}}},$$

where a_i are unique random elements from $\{1, \ldots, N\}$.

[3] *Annoy* is an acronym for *Approximate Nearest Neighbors Oh Yeah.*

3 Navigation methods

This section describes our framework in detail. In particular, we explain how our framework enables interactive exploration of a generic nD dataset X, that potentially contains many millions of observations.

Note that many operations may also be done with other techniques than the ones suggested here. For example, instead of random sampling, one could also use more advanced sampling methods that aim for a uniform coverage in nD.

3.1 Initial projection

Parameters:

- N_{V}: Number of points visible on screen. This parameter will also be respected by the zoom-in/zoom-out operations.

Since X contains many observations, we want to subsample it before making a projection:

$$X_{\mathrm{V}} = \mathcal{S}_{\mathrm{RND}}(X, N_{\mathrm{V}}),$$

where N_{V} is the number of observations that will be projected and visualized at a time.

The first 2D projection is evaluated on this subset of observations:

$$Y_{\mathrm{V}} = \mathcal{P}_{\mathrm{MMDS}}(X_{\mathrm{V}}, 2),$$

which can be shown on the screen.

3.2 Zoom-in

Parameters:

- N_{LM}: Number of landmarks to retain.
- k: Neighborhood size for region of interest (ROI) selection in 2D. (Determines zoom factor.)

As the initial projection—as described in the previous subsection—does not show all the data, we provide interactive level-of-detail exploration if the data by *nD zoom-in*.

Given X_{V} the currently visible data (with $|X_{\mathrm{V}}| = N_{\mathrm{V}}$) and Y_{V} its projection, the user selects a focus point $\mathbf{q} \in \mathbb{R}^2$, *e.g.*, at the mouse location. We next select all observations $X_{\mathrm{ROI}} \subset X_{\mathrm{V}}$ whose projections in Y_{ROI} are the k-nearest neighbors of $\mathbf{q}$ *in the 2D space*, where k defines the zoom level - *e.g.*, setting k to 90% of N_{V} yields a zoom of roughly 10%. Points outside X_{ROI} are discarded. There is now room for $N_{\mathrm{V}} - k$ more points, so we compute the set X_{c} of $N_{\mathrm{V}} - k$ observations from $X \setminus X_{\mathrm{ROI}}$ that are closest to X_{ROI}. Next, we define the new set of observations $X_{\mathrm{ZI}} = X_{\mathrm{ROI}} \cup X_{\mathrm{c}}$, and project it using as landmarks X_{LM} a set of N_{LM} randomly chosen points from X_{ROI}, *i.e.*,

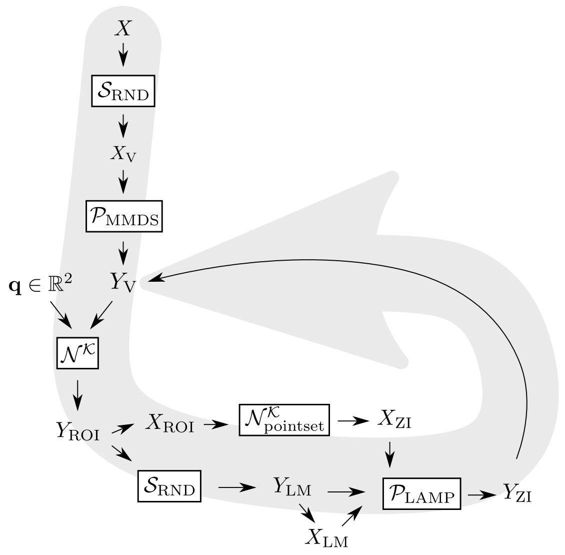
X
$\mathcal{S}_{\text{RND}}$
X_{V}
$\mathcal{P}_{\text{MMDS}}$
$\mathbf{q} \in \mathbb{R}^2$
Y_{V}
$\mathcal{N}^{\mathcal{K}}$
Y_{ROI}
X_{ROI}
$\mathcal{N}^{\mathcal{K}}_{\text{pointset}}$
X_{ZI}
$\mathcal{S}_{\text{RND}}$
Y_{LM}
$\mathcal{P}_{\text{LAMP}}$
Y_{ZI}
X_{LM}

Figure 1: nD zoom-in pipeline. First, the initial projection is made by projecting a subsampling of the original data. Second, a point of interest $\mathbf{q}$ is selected by the user, and a trivial neighborhood search is done in 2D to retrieve the region of interest (ROI) Y_{ROI}. X_{ROI}—the nD observations corresponding to Y_{ROI}—are used to search for relevant points in nD. The results of this search are projected by using landmarks from Y_{ROI} to obtain the new view.

$X_{\text{LM}} = \mathcal{S}_{\text{RND}}(X_{\text{ROI}}, N_{\text{LM}})$. To preserve visual continuity, the projection Y_{LM} of the landmarks is not re-computed, but is set to the points from Y_{ROI} that already mapped the observations in X_{LM}.

Fig. 1 shows a schematic view of the nD zoom-in pipeline, and Fig. 2 shows the schematic user interaction.

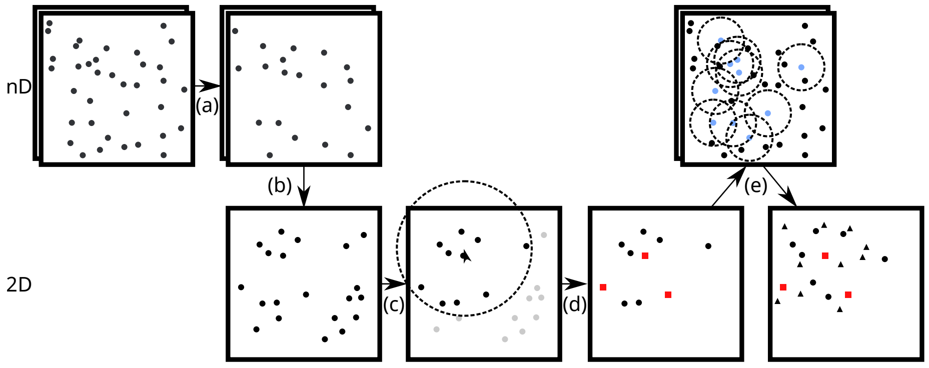

Figure 2: nD zoom-in illustrated. (a) Subsampling the dataset $X \subset \mathbb{R}^n$. (b) Projecting $\mathcal{S}_{\mathrm{RND}}(X, N_{\mathrm{V}})$ to 2D. (c) User selects ROI in 2D. (d) Landmarks are sampled from ROI points. (e) $\mathbb{R}^n$ observations are selected as nearest-neighbors of observations mapped to ROI points. Newly selected points are projected with the other remaining points using landmarks from (d).

3.3 Zoom-out

Since we are providing a zoom-in functionality, it is natural to also have a zoom-out functionality. A trivial way of 'zooming out' is simply going back to the previous view after a zoom-in. However, we also made attempts at zooming out by using the actual nD data, rather than the history.

Note (written november 2017), the goal of doing a zoom-out based on the actual data is to enable exploration; a user is more likely to find interesting features in the data if the user does not get stuck in the same subset of the data. Therefore, an alternative to zoom-out is a compromise between history-based zoom-out and data-based zoom-out, and this can happen as follows: A view of the data (in general) uses sampling to determine the subset of the data that is shown. When going back to the previous view, the compromise method uses the same parameters, but a *different sampling* than the previous (history-based) view, so that the user sees new data.

In this section, X_{V} is understood to be a zoomed-in nD dataset. That is, it is the set of high-dimensional observations corresponding to the points on the screen after a (series of) zoom-in operation(s). All strategies discussed here aim to find a set of observations X_{ZO} (of size N_{V}, the limit of the number of points on screen) that is natural after a zoom-out operation. This set of observations can be visualized (by means of multidimensional projections) on screen next. Indeed, as with zoom-in, landmarks can be used to provide continuity.

3.3.1 Strategy 1: Neighbor search, then subsampling (S1)

Parameters:

- k: Number of neighbors for the neighborhood search.

Given the current (zoomed in) dataset X_{V} (with N_{V} points), we perform multiple neighborhood queries on X (the full dataset), using $\mathbf{x} \in X_{\mathrm{V}}$ as query, unioning the result. This results in a dataset X_{nn} with the k closest points from X for all N_{V} points in X_{V}. The dataset X_{nn} has (at most) kN_{max} points. This is an upper bound, because the neighborhood search results can—and almost always do—have overlap. The lower bound is k, but this is very unlikely.

The dataset X_{nn} is very likely to be too large (it has $> N_{\mathrm{V}}$ points, which is intended), so it is subsampled to have only N_{V} points, and this is the resulting zoomed-out dataset X_{ZO}.

Figure 3 illustrates the pipeline.

Problems:

- Since X can be very dense, the dataset X_{nn} might not even contain samples outside of the domain of the previous view X_{V}. (Increasing k is possible, but this will make it very slow, and it might take very large values of k to reach observations that are outside of the domain of X_{V}.)

$$\begin{tikzcd}
X_{\mathrm{V}} \arrow[d] & \\
\boxed{\mathcal{N}(\mathbf{x}, X, k)} \arrow[d, shift left=3] \arrow[d, shift left=1] \arrow[d, shift right=1] \arrow[d, shift right=3] & \forall \mathbf{x} \in X_{\mathrm{V}} \\
\boxed{\bigcup} \arrow[d] & \\
X_{\mathrm{nn}} \arrow[d] & \\
\boxed{\mathcal{S}_{\mathrm{RND}}(X_{\mathrm{nn}}, N_{\mathrm{V}})} \arrow[d] & \\
X_{\mathrm{ZO}} &
\end{tikzcd}$$

Figure 3: ZO-S1: First, a neighborhood search is done for every point in X_{V}. The results are aggregated by taking the union, which is subsequently subsampled to contain the required N_{V} observations.

3.3.2 Strategy 2: Subsampling, then neighbor search (S2)

Parameters:

- a_{ZO}: Zoom-out factor. For example, if $a_{\text{ZO}} = 1.2$ we expect to see about 20% more of the space.
- f: Some condition that checks if the operation zoomed out too little or too much. Signature: $f(X_{\text{V}}, X_{\text{nn}}) \in \mathbb{R}$. Returns 0 if the zoom-out was optimal, negative values if the zoom-out was too weak, positive values if the zoom-out was too strong.
- A search strategy to find the optimal N_{S}.

In this zoom-out strategy, the first step is to subsample X (the full dataset) with N_{S} ($\geq N_{\text{V}}$) points: $X_{\text{S}} = \mathcal{S}_{\text{RND}}(X, N_{\text{S}})$. After that, we perform a neighborhood search in X_{S}, using X_{V} as query: $X_{\text{nn}} = \mathcal{N}_{\text{pointset}}(X_{\text{V}}, X_{\text{S}})$. Now, X_{nn} is the dataset that contains the N_{V} points from X_{S} that are closest to any point in X_{V}. One of three things can be the case:

- X_{S} was sampled too densely: This is detected by observing that $f(X_{\text{V}}, X_{\text{nn}}) < 0$ (see below for definition), i.e., we have zoomed out too little. In this case: Try again, but with a lower N_{S}.
- X_{S} was sampled too sparsely: This is detected by observing that $f(X_{\text{V}}, X_{\text{nn}}) > 0$, i.e., we have zoomed out too much. In this case: Try again, but with a larger N_{S}.
- X_{S} was sampled correctly: This is detected by observing that $f(X_{\text{V}}, X_{\text{nn}}) = 0$ (or within some tolerance). In this case: Great. We are done, and we can use X_{nn} as the result X_{ZO}.

For the condition f that checks if the zoom-out was succesful, two things have been considered:

- f_1: Evaluate ratio between the radii of the bounding hyperspheres of X_{V} and X_{nn}, and return how much it deviates from a_{ZO} (the zoom-out factor).
- f_2: Count how many points from X_{nn} fall outside of X_{V}'s bounding hypersphere, and return how much the ratio of the number of points outside *vs.* number of points inside and outside deviates from a_{ZO} (the zoom-out factor).

Both conditions return negative values if the zoom-out is too weak (it may even be a zoom-in), and positive values if the zoom-out is too strong.

For the search strategy, N_{S} is initialized with N_{V}, and (assuming that it is too small) it is doubled every time to find the upper bound of N_{S}. When the upper bound is found, binary search is used to find the optimal value.

Problems:

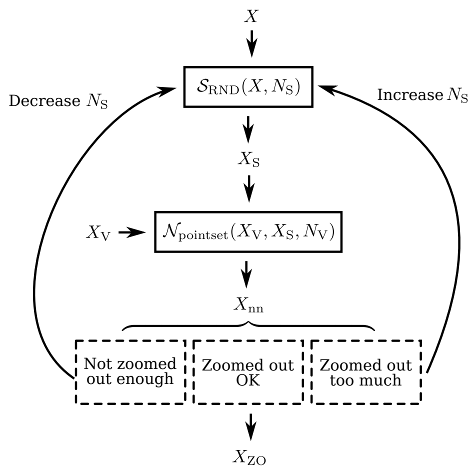
X
$\mathcal{S}_{\mathrm{RND}}(X, N_{\mathrm{S}})$
Decrease N_{S}
Increase N_{S}
X_{S}
X_{V}
$\mathcal{N}_{\mathrm{pointset}}(X_{\mathrm{V}}, X_{\mathrm{S}}, N_{\mathrm{V}})$
X_{nn}
Not zoomed out enough
Zoomed out OK
Zoomed out too much
X_{ZO}

Figure 4: ZO-S2: First, the full dataset is subsampled. Then, the current observations (X_{V}) is used as a query to search for neighbors in the subsampled dataset. If the result not good (depending on f), try again with a denser/sparser sampling.

- The condition $f(X_{\mathrm{V}}, X_{\mathrm{nn}})$ is checked on randomly subsampled datasets (X_{S}). This may not be reliable enough to use in practice. (For better reliability, we can try more subsamplings X_{S} of the same size N_{S}, and aggregate the condition values f somehow, but this is probably too costly.)
- Searching for the ideal value of N_{S} like this is too costly.
- It has been observed many times that N_{S} was not large enough, and it kept being incremented by the search strategy, eventually using the entire dataset for X_{S}.
- This method seems to zoom out very slowly when X_{V} is on a very zoomed-in scale. But when X_{V} becomes less zoomed-in, one step instantly zooms out to the coarsest level. (See Figure 7) This means that—on that scale—it is very hard to zoom out just a little bit. This may have to do with the 'curse of dimensionality'. (See also the problems strategy 3, below.)

3.3.3 Strategy 3: Neighbor search with generated query points (S3)

The first step here is to sample the bounding hypersphere of X_{V}. This results in a (generated) dataset X_{hs_0}. The next step is to perform a search in X to find the N_{V} points that are nearest to any point in X_{hs_0}. This results in the dataset X_{nn_0}. Now, it is very likely that many points from X_{nn_0} are also inside the bounding hypersphere of X_{V} (which we do not want, because we want to zoom out). So, we increase the radius of the hypersphere, and sample it again, resulting in a dataset X_{hs_1}. We try if this results in a better X_{nn_1}. If not, try again, etc.

Parameters:

- Some strategy for adapting the generated query points to search 'outward'.

Problems:

- Since the 'surface' of an n-sphere grows with R^{n-1}, where R is the radius, it is very costly to sample densely enough on a hypersphere, and even more if we increase R during the outward search strategy.

3.3.4 Strategy 4 (S4)

Parameters:

- N_{S}: Number of samples for the subsampling of the full dataset.
- a_{new}: Fraction of points that are guaranteed to be outside of the current domain.

We start with determining the bounding hypersphere of X_{V}, we denote this $\bigcirc(X_{\mathrm{V}})$. Next, we subsample X_{V} to have $(1 - a_{\mathrm{new}}) \cdot N_{\mathrm{V}}$ samples, and call this X_{ZO}. After this, we pick $a_{\mathrm{new}} \cdot N_{\mathrm{V}}$ new samples (and add them to X_{ZO}) as follows:

1. Sample X: $X_{\mathrm{S}} = \mathcal{S}(X, N_{\mathrm{S}})$.
2. From X_{S}, select the point $\mathbf{p}$ (could also be plural, that should be faster) that is closest to any point in X_{V}, but outside of $\bigcirc(X_{\mathrm{V}})$ that is not already in X_{ZO}.
3. Add $\mathbf{p}$ to X_{ZO}.
4. Go back to step 1, unless we already have enough new points.

After this, X_{ZO} is the set that has a fraction a_{new} of points that *are guaranteed to be* outside of the previous domain, and a fraction of $1 - a_{\mathrm{new}}$ points from inside the previous domain.
Problems:

- After a while, it becomes very slow to find points $\mathbf{p}$ that meet the requirements. It also happens—especially when zooming out from a just barely zoomed in level, and with small values of N— that these points are never found (or probably do not even exist), and the algorithm does not terminate. It really depends on how the observations are distributed.
- Even though the new points are guaranteed to be outside of the previous domain, it often happens that the zoom-out operation is too weak and nothing (seemingly) changes in one zoom-out operation. This was observed with $a_{\mathrm{new}} = 0.5$.

N	n	Preprocessing (s)	ZI (s)	ZO-S1 (s)	ZO-S2 (s)	ZO-S4 (s)
10K	30	0.26	0.26	0.76	$\sim 1-2$	–
100K	30	2.01	0.27	1.15	~ 3	–
1M	30	23.49	0.27	1.80	–	21.94
5M	50	132.90	0.29	2.35	–	105.87

Table 1: Navigation timings on Gaussian blob datasets with different sizes and dimensionalities. ZO-S1 has been tested with $k = 1000$, ZO-S2 with f_1 and $a_{\mathrm{ZO}} = 1.2$, and ZO-S4 with $N_{\mathrm{S}} = 1000$. Some entries are empty (–), indicating that the algorithm did not terminate.

4 Results

The zoom-in and zoom-out operations have mostly been tested on a synthetic dataset containing observations sampled from 10 different nD Gaussian distributions with different means, but equal variance.

Table 1 shows timings for navigation in Gaussian blob datasets with different sizes and dimensionalities. It can be seen that for the zoom-in operation, the main bottleneck is the preprocessing for the NN evaluations. Other than that, it scales well.

As Fig. 5 shows, the zoom-in operation works quite intuitively. Figs. 6 and 7 illustrate the problems that two of the zoom-out operations encounter.

References

[1] Erik Bernhardsson. Annoy. `https://github.com/spotify/annoy`, 2013.

[2] Erik Bernhardsson. ann-benchmarks. `https://github.com/erikbern/ann-benchmarks`, 2017.

[3] Jan De Leeuw and Patrick Mair. Multidimensional scaling using majorization: Smacof in r. *Department of Statistics, UCLA*, 2011.

[4] Paulo Joia, Danilo Coimbra, Jose A Cuminato, Fernando V Paulovich, and Luis G Nonato. Local affine multidimensional projection. *IEEE TVCG*, 17(12):2563–2571, 2011.

[5] D Surendran. Swiss roll dataset. `http://people.cs.uchicago.edu/~dinoj/manifold/swissroll.html`, 2004.

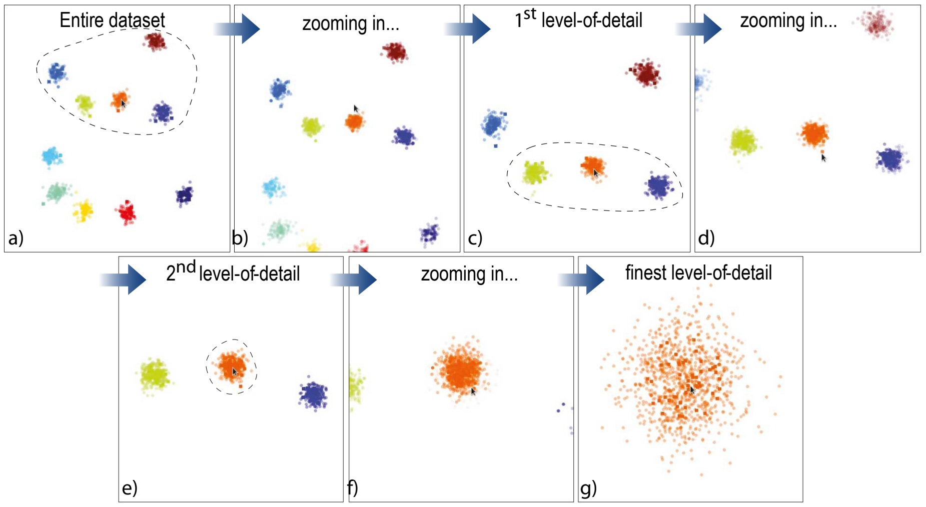

Figure 5: nD zoom-in in action. From the projection of X_{V} (a), we zoom three times to get details in the orange cluster, yielding views (c), (e), and (g). As we zoom, points are added on-demand—(g) has about $N_{\mathrm{V}} = 1000$ orange points as compared to only about 100 in (a). Images (b), (d), and (f) show intermediate interpolation stages during the zooming. Dashed lines show the approximate regions of interest (ROIs) Y_{ROI}.

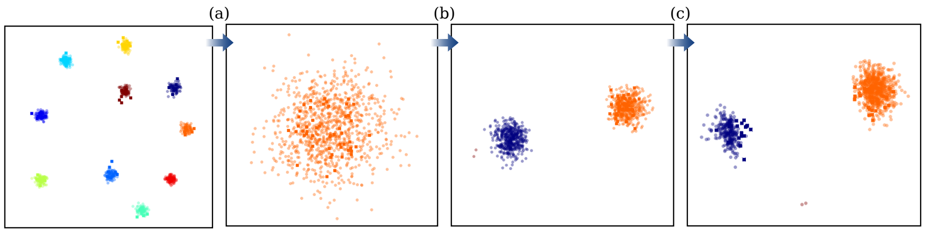

Figure 6: ZO-S1. (a) Zoom-in (as in Fig. 5). (b) Zoom-out operation: It is natural that the blue cluster—that was close in the original projection—is in the zoomed out view. (c) Subsequent zoom-out operations do not zoom out further.

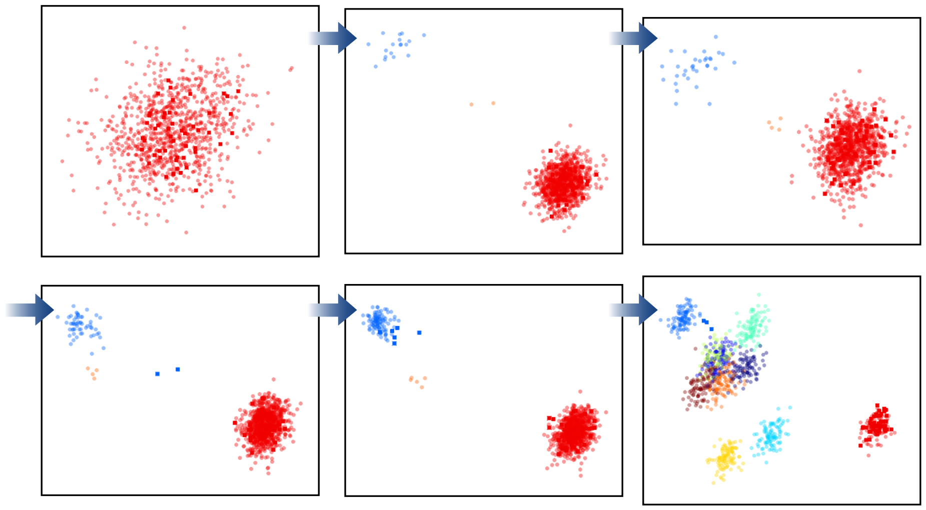

Figure 7: ZO-S2. The first zoom-out operations are very slow, whereas the last operation instantly increases the domain to the entire dataset.