Dimensionality Reduction and Sorting Tool GUI user manual

Jordan Sorokin (jorsorokin@gmail.com) *

This documents provides a detailed overview of the basic usage, features, and current limitations of the Dimensionality Reduction & Sorting Tool (DRST) GUI. Much of this GUI was inspired by existing spike-sorting software (see KlustaKwik, KiloSort, and MSort for some examples), but was created as a response to the limited flexibility present in most of these existing packages, and as a means to directly interface with the Neo-Matlab hierarchy. Most features have been extensively tested, but I cannot guarantee it will be fool-proof for your needs. Please send me an email if any questions or issues arise.

Contents

*This work was possible thanks to funding by the NIH

1 Overview

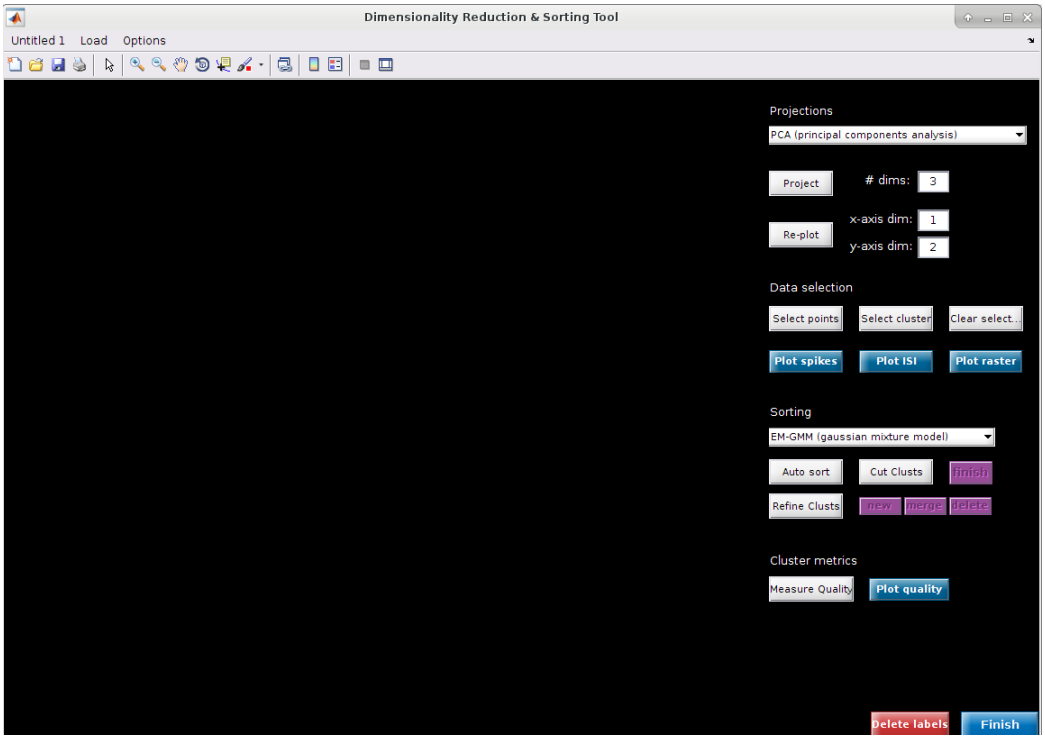
The Dimensionality Reduction and Sorting Tool (DRST) GUI is a general-purpose tool for both dimensionality reduction and clustering, with a strong focus on flexibility and feature design often limited in other such software. While it may not be the snappiest of programs, it is exceedingly easy to use as it is written in pure MATLAB and does not require specific file formats. Some of the main features (as of January, 2018) in the DRST are:

1. intuitive and flexible I/O, independent of specific file or data formats
2. both linear and non-linear methods for dimensionality reduction (thanks to Van der Maaten's MATLAB drtoolbox)
3. both parametric and non-parametric clustering algorithms
4. extensive options for clustering, including automatic cluster # identification, hierarchical clustering, outlier detection & elimination, and methods for evaluating cluster quality
5. re-clustering and plotting of subsets of data
6. integrated plotting of raw data, projections, rasters, inter-spike interval histograms, and cluster quality
7. manual cluster-cutting in conjunction with automatic clustering

8. capable of GPU-acceleration for plotting and dimensionality reduction

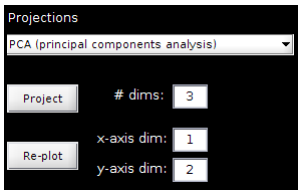
9. fully integrated with the *Neo-MATLAB* package for facilitating sorting and bookkeeping of large datasets

1.1 Main interface



This is the main interface of the DRST GUI. The buttons / lists on the right are the means for projecting, plotting, and clustering your data. Following dimensionality reduction, the projected data will be plotted in the large dark region of the GUI. This is the main plotting window for selecting individual points and/or clusters.

Projections All dimensionality reduction is performed with the first set of tools. The default dimensionality reduction method is PCA, but you can select a preferred method by clicking on the drop-down list. You can also specify the number of dimen-

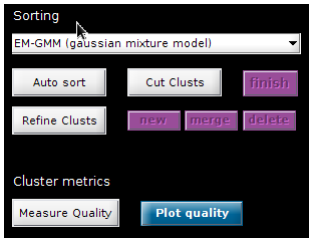


sions to project onto by changing the number in the *nDims* box. To do the actual projections, click the *Project data* button. Doing so will perform dimensionality reduction and plot the first two dimensions of the projected data onto the main figure. You can change the "viewpoint" of the projected data by editing the *x-axis dim* and *y-axis dim* edit boxes and then clicking the *re-plot* button.

Data selection You can select individual points or clusters in the main projection window via the *data selection* set of tools. Clicking *Select points* allows you to draw a lasso around individual points, while clicking *Select cluster* allows you to select an entire cluster by simply clicking on any point belonging to that cluster. You can select multiple clusters or individual points by simply clicking *Select points* or *Select cluster* repeatedly. Selected points will appear highlighted in the main projection figure, and clicking *Clear selection* will un-select any currently selected points.

Plotting and data selection are tightly integrated in the DRST GUI. One can plot the raw waveforms, rasters, and inter-spike intervals of currently selected data points by clicking on the corresponding buttons. Note that this is a dynamic process: these plots will be cleared following data un-selection via the *Clear selection* button.

Sorting Cluster sorting is controlled by the *Sorting* set of buttons. The drop-down list contains various automatic clustering algorithms (defaulting to the Expectation-Maximization for Gaussian Mixture Models), some of which are non-parametric algorithms (spectral clustering, DBSCAN, and HDBSCAN). Automatic clustering is performed by clicking the *Auto sort* button, and automatic cluster refinement can then be performed by clicking the *Refine clusts* button. To manually delete, merge, or create new clusters, click the *Manual sort* button, which will un-hide the manual clustering options. Further sorting options are available by clicking on the *options* menu in the menu bar, which opens a new window with various sorting parameters. See section 3 (Clustering) for details.



1.2 Inputs / outputs

All of the following inputs into the DRST GUI are optional:

1. **data**
   The raw data that may be projected / clustered. The data can be a 2D matrix or 3D tensor. For a 2D matrix, the data should be formated as: $X \in R^{DxN}$, where D equals the original dimensionality of the data points, and N equals the total number of data points. For a 3D tensor, the data should be formated as: $X \in R^{DxNxC}$, where D and N are the same as above, and C equals the number of channels simultaneously recording the data points.

   Note that these are typical interpretations of data formats for spike-sorting. However, the GUI will still provide meaningful results even if D, N, and C do not represent the values described above, provided that you understand what they do represent. For instance, in an RNA-seq data set, one may have D represent the number of genes per sample, N represent the total number of samples, and C represent different cell compartments from which the samples were extracted.

2. **projection**
   Pre-computed projections of the original dataset can be supplied directly to avoid the overhead of re-computing the projections. The projection matrix should be formated as R^{NxK}, where N is the number of data points, and K is the reduced number of dimensions. Both *data* and *projection* inputs need not be supplied to the GUI; however it is important to note that if only the latter is supplied, one cannot re-project the points or plot the raw waveforms.

3. **times**
   The *times* input is an N-dimensional vector of spike-times (one time for each of the N action potentials) used for plotting the inter-spike intervals and rasters of spikes.

4. **trials**
   Another N-dimensional vector specifying which trial each spike belongs to (if any). This is used in both the ISI and raster plots.

5. **labels**
   One can input an N-dimensional vector of cluster labels from a previous clustering result to visualize the clustering and, possibly, alter the clusters / re-cluster.

6. **location**
   In some multi-channel recording configurations (i.e. silicon probes or MEAs), providing the physical location of the channels as another "feature" in the projected space can help separate clusters. The *location* input of the GUI should be of size NxL, where L represents the number of physical dimensions of channel locations (i.e. [X,Y,Z])

7. **mask**
   In the case of very high channel count recordings, such as those from dense multi-electrode silicon probes, projecting the data directly often fails to separate clusters as, for any given action potential, only a small subset of channels actually record the voltage waveform of the action potential. This results in a projection space dominated mainly by noisy channels. In this case, providing a *masking matrix* of size CxN can help resolve clusters. In essence, the masking matrix M is a weight matrix with each element $M_{i,j} \in [0, 1]$, representing the *prominence* of spike j on channel i. For more information, see the *Neo-Matlab* documentation and the *Masked-EM* algorithm.

Calling the GUI from the MATLAB command line is an easy one-liner. Inputs are supplied via name-value pairs (i.e. **sortTool('data',X,'projection',P)**). Optionally, one may load the optional inputs through the **load** menu item in the GUI itself. Clicking *load* will open a data loading window. There you can select the type of input and the name of the variable in the MATLAB workspace to load. Note that you must have the variables loaded into the MATLAB workspace, or have them saved as separate files in a .mat file in an accessible folder.

1.3 Outputs

The DRST GUI has 3 optional outputs.

labels
An $Nx1$ vector of cluster IDs for the data points. Any $label_i = 0$

indicates an "unclusterable" data point, or a point whos label was manually deleted.

projection
As described above, the *projection* output will be of size NxK, with $K < D$.

R
The third output is a structure, R, containing the following fields:

mapping - a structure containing the information from the mapping of the original data to the lower-dimensional subspace

projMethod - the mapping method

sortMethod - the sorting method

sortModel - a sorting model, if applicable (i.e. the means, covariances, and weights found by EM-GMM,...)

probabilities - an $Nx1$ vector or NxQ matrix of probabilities of each point i belonging to one of the Q clusters

2 Dimensionality Reduction

One of the most powerful aspects of the DRST GUI is the abundance of projection methods available, as "typical" methods such as PCA and ICA, often implemented in other software, may fail to separate clusters well. The projection methods can be broken into two main categories: *linear* and *non-linear*. Linear methods have the advantage of being (typically) faster to compute and containing a simple means to project new data onto the same subspace. However, linear methods fails if the data actually lie on a non-linear manifold, (i.e. points lying on a 3D sphere cannot be linearly projected onto a 2D subspace without distorting the distances between points).

Non-linear methods are typically slower to compute, often having to calculate the nearest neighbors of each point, however they can preserve distances between points that lie on curved manifolds, which often leads to

much better cluster separation than with linear methods. Unfortunately, it is difficult to embed new points onto the same subspace using non-linear methods, although there are ways to approximate the embedding for some methods (see the **drtoolbox** in the GitHub repo).

2.1 Linear

The linear projection methods use weighted combinations of the original data points to project the data onto a lower dimensional subspace. The linear methods in the DRST GUI are:

Principal components analysis (PCA)

Independent components analysis (ICA)

Neighborhood components analysis (NCA)

Local linear embedding (LLA)

Hessian local linear embedding (HLLA)

Neighborhood preserving embedding (NPE)

2.2 Non-linear

Non-linear methods first transform the original dataset, often by computing a graph from pairwise distances between points, then projecting points from the transformation using various methods. New points can be approximately embedded into the subspace created by these methods. Other non-linear methods use iterative procedures to minimize some cost function given the data set. New points generally cannot be embedded using these methods.

The non-linear, non-iterative methods are:

Kernel principal components analysis (KPCA)

Probabilistic principal components analysis (pPCA)

Locality preserving projection (LPP)

Stochastic proximity embedding (SPE)

Maximum variance unfolding (MVU) and fastMVU

Laplacian eigenmaps (LE)

Difusion maps (DM)

While the iterative methods are:

Stochastic neighborhood embedding (SNE)

t-distributed stochastic neighborhood embedding (tSNE)

Sammon mapping

Autoencoders

Isomap

3 Clustering

Beyond projecting the original data, the DRST GUI is also capable of clustering the data points into distinct clusters. This can be done automatically or manually (by clicking on the *Auto sort* and *Cut Clusts* buttons, respectively). Although most clustering algorithms require a pre-determined number of clusters set by the user, there are methods built into the GUI to automatically search for the correct number. This can be done in the sorting options (see below), or by choosing a clustering algorithm that automatically finds the correct number to begin with (HDBSCAN). Clustering algorithms are generally divided into *parametric* and *non-parametric* methods, each of which has its own benefits and limitations...

3.1 Parametric

Parametric algorithms try and force the clusters to meet pre-determined distributions. The most well known parametric algorithm is *K-Means*, which tries to fit spherically-Gaussian clusters to the dataset. Parametric clustering algorithms have the advantage of being arithmetically tractable, well-studied, and providing an intuitive model output that can be used to cluster future points and evaluate the clusters themselves. The parametric clustering algorithms in the GUI are:

K-means (Km)
As briefly described above, K-means fits K-Gaussian clusters to the data, each of which is spherical. K-means is fast, but may fail if the data do not fall into well-defined "blobs".

Expectation maximization for Gaussian mixture models (EM-GMM)
A fast and widely used clustering algorithm that fits K-Gaussian clusters to the data. The clusters need not be spherical, but can be elongated and overlapping. The algorithm outputs the means, covariances, and weights of the Gaussian distributions that it finds, as well as the probabilities of each of the points belonging to each of the K clusters.

Masked EM-GMM (mEM-GMM)
In the case of high-channel count data, using a masking matrix M can help avoid the collapse of the Gaussian clusters into a single, "noise cluster" due to the large number of "noise channels" present.

Variational bayes for Gaussian mixture models (VB)
Similar in theory to the EM-GMM algorithm in that it fits K-Gaussian clusters to the data, however it does so using a probabilistic, Bayesian framework. For some applications, this may be advantageous as the cluster assignment may be more intuitive. Additionally, the algorithm is able to find an appropriate number of clusters (given a starting value).

Masked VB (mVB)
Similar in scope to the mEM-GMM algorithm, but for the VB algorithm.

Expectation maximiaztion for t-distributed mixture model (EM-TMM)
The EM algorithm is a general-purpose method for iterating over conditional distributions of unknown parameters, and fitting the data to the discovered parameters. In the EM-GMM algorithm, those parameters are the means and covariances of the Gaussian clusters. In the EM-TMM algorithm, the EM algorithm is applied to the t-distribution, meaning the parameters are means, covariances, and the alpha, beta, and gamma parameters of a Gamma distribution. This is still under development, but will be useful when clusters have long tails.

3.2 Non-parametric

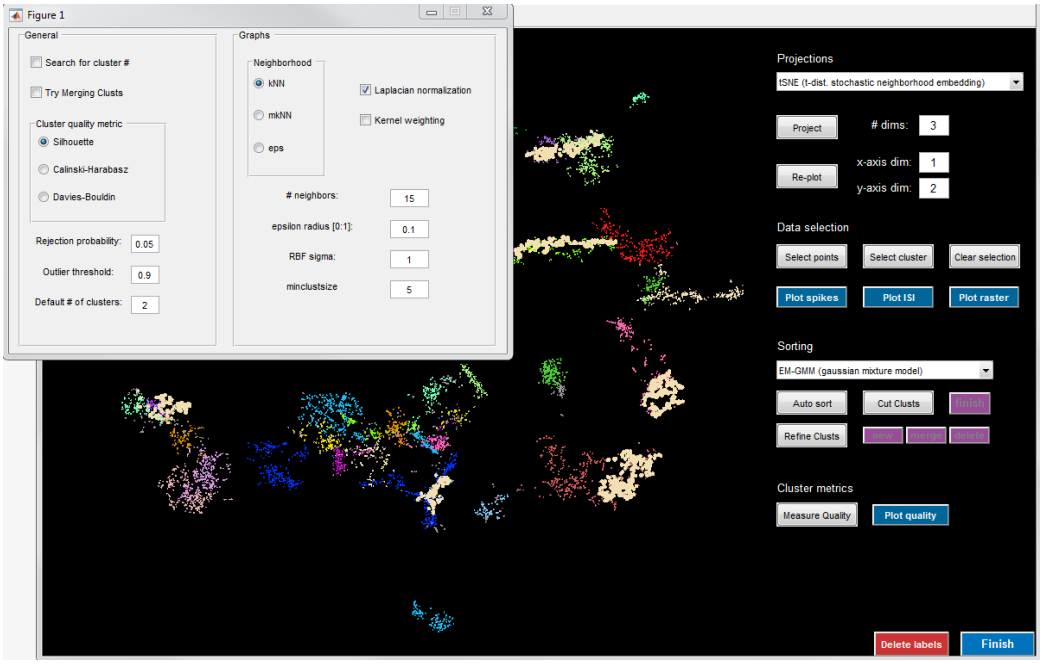
Non-parametric methods generally transform the data (usually via pair-wise distances of some kind) and try to cluster the data in the transformed space. They have the advantage of using information from the distribution of the data itself, rather than trying to force pre-determined distributions onto the data. They tend to be slightly slower (although DBSCAN is quite fast), but are robust to odd shapes in the data manifold. Specifically, if the projected points lie on curved surfaces, most parametric clustering algorithms will fail to accurately model the clusters.

Density-based clustering for applications with noise (DBSCAN)
One of the most well-recognized non-parametric clustering algorithms. DBSCAN clusters data based on the density profile of the full dataset. Pairwise distances are calculated, and a "graph" connecting dense regions and separating sparse regions in the dataset is constructed. Links that are too large (regions too far apart in the density-space) are cut, and the resulting "nodes" become the clusters. Additionally, points may be deemed outliers if they sit too far away from the dense region of any cluster.

Hierarchical DBSCAN (HDBSCAN)
A relatively recent algorithm, invented by the same authors as DBSCAN. The HDBSCAN algorithm builds a hierarchical cluster tree from the data and cuts links between clusters in a way that optimizes the "robustness" of the resulting clusters. HDBSCAN is significantly slower than DBSCAN as a full hierarchy is developed, however it can automatically detect the "optimal" number of stable clusters. Additionally, it doesn't rely on global threshold for cutting links between nodes, but can do so locally.

Spectral Clustering
A graph is built from the data by finding nearest-neighbors of the points, and then an eigen-decomposition (similar to PCA) is performed on the transformed graph, followed by clustering of eigenvectors of the graph. Although spectral clustering requires a pre-determined number of clusters, it can be more robust to odd cluster shapes as it does not try and fit specific parameters to the data.

3.3 Sorting options



One is able to fine tune various sorting hyperparameters in the DRST GUI, which are specific to the sorting routine selected in the main Interface. In the screenshot above, the sorting options window is shown above the main GUI interface. Clicking on the *options* menu in the GUI opens up this window. The sorting options window is divided into two main panels, *General* and *Graphs*.

3.3.1 General

This panel controls general sorting routine options that are applicable to more than one sorting algorithm.

Search for cluster #
Allows the GUI to call an automatic cluster # identifying algorithm for parametric clustering procedures (K-means, EM-GMM...). Note that this box must be *unchecked* if using DBSCAN, HDBSCAN, or Spectral Clustering.

Try Merging Clusts
If checked, both automatic splitting and merging of clusters will be

attempted by pressing the *refine clusts* button in the main GUI interface. Normally, only splitting is performed as merging of clusters is more computationally demanding.

Cluster quality metric
This subpanel controls which method is used when measuring cluster quality. Each method has its own advantages and disadvantages, and the user should be wary of interpreting the cluster quality depending on the combination of the selected method and the distribution of the data.

Rejection probability
For all clustering methods besides HDBSCAN, if any data point assigned to a cluster has a probability of belonging to that cluster less than the value in this box, that point will be flagged as an outlier.

Outlier threshold
This is specifically for the HDBSCAN algorithm. As this number approaches 1.0, fewer points will be flagged as outliers.

Default # of clusters
For parametric algorithms, (and Spectral clustering), this value controls the # of clusters that the sorting algorithms will try to discover from the data.

3.3.2 Graphs

The *Graphs* panel provides additional control over hyperparameters used in all graph-based clustering algorithms (DBSCAN, HDBSCAN, and Spectral Clustering).

Neighborhood
This panel allows the user to choose between different implementations of nearest-neighbor identification of each data point. *kNN* is the typical *k-Nearest Neighbor* interpretation of a neighborhood, where k-neighbors of each point are found by sorting a distance matrix for each point. The *mkNN* is a slightly different approach that produces non-symmetric nearest neighbor queries. Finally, the *eps* implements a radius of a fixed size for each point, where any other data point lying within that hyper-sphere is considered a nearest-neighbor.

Laplacian normalization
If checked, the laplacian matrix formed during Spectral clustering is normalized prior to the eigenvector decomposition. For most purposes, this should be left checked.

Kernel weighting
If checked, the adjacency matrix formed during Spectral clustering will be transformed using an RBF kernel, which can help form smoother boundaries between clusters.

 # neighbors
This determines the number of neighboring points to query for each data point. The larger the number, the more "global" the clustering scheme, and generally, the fewer the # of identified clusters by HDBSCAN and DBSCAN. However, too small and clusters may become spurious due to local density variations due to noise. For HDBSCAN, this is best left between 3-5, while for DBSCAN and Spectral clustering, 10-20 tends to work well. This is only applicable if *kNN* or *mkNN* are selected.

epsilon radius
Determines the radius of the hyper-sphere (as a percentage of the total volume occupied by the data) for querying nearest-neighbors. This is only applicable if *eps* is selected.

RBF sigma
The variance (width) of the RBF kernel used when transforming the adjacency matrix during Spectral clustering. This is only applicable if *Kernel weighting* is checked.

minclustsize
This determines the lower bound of the # of points for any valid cluster found by HDBSCAN during the hierarchical search. The smaller the number, the greater the # of clusters will be realized in the data set at the risk of over-clustering and producing more outliers.

4 Plotting

All plots within the DRST are all integrated with one another, and reflect instantaneous cluster assignment changes. For instance, assigning points to a new cluster is automatically reflected by a change in the color of the same data in open subplots (waveform, rasters, etc).

4.1 Raw waveforms

If raw waveforms are passed into the GUI under the optional input *data*, the waveforms can be displayed by clicking on the *Plot spikes* button. A new figure will pop up, with as many subplots as the size of the third dimension of the input data (if it is a tensor). For spike sorting, this could be interpreted as the number of channels on the recording electrode. Any highlighted point in the main DRST interface will have its raw waveforms plotted on these subplots, color coded by the assigned cluster.

4.2 Inter-spike intervals

Similarly, if the times of the spikes are provided to the GUI, then clicking on *plot ISI* will open a new figure, with the ISI histogram displayed for the selected highlighted points. Each unique cluster will have its own ISI represented by the same color.

4.3 Rasters

As with the ISI plot, providing the times of the spikes allows the GUI to display a raster plot of the highlighted spikes, color coded by cluster. If the optional input *trials* is provided, there will be as many rows in this plot as trials.

4.4 Cluster quality

Towards the bottom right of the DRST interface, one can click on the *plot quality* button to open a figure showing the cluster quality of each individual cluster (top axes) and the global clustering quality (bottom axes). Throughout the sorting process, one can click on the *Measure quality* button while

the quality plot is open, and the plot will update according to each new clustering scheme. The history of the global cluster quality will be displayed, allowing the user to reference previous clustering schemes.

In addition to plotting the history, clicking on *Measure quality* stores the cluster labels into a growing matrix. One can then analyze the cluster quality history outside of the GUI for choosing the final labels. Note that the label history and quality history will not be stored automatically for each new cluster scheme, but will only be saved for each click of the *Measure quality* button.

5 Known issues

All features in the DRST GUI have been extensively tested with both simulated and real data sets. However, there are some limitations as of of Jan 1, 2018. These are:

- Limited control on hyperparameters of non-linear dimensionality reduction algorithms (such as eigen solver, perplexity of tSNE).
- Projections onto the 2D axes of the DRST do not combine multiple dimensions, thus forcing viewpoints to a limited subset of possible views.
- Features are all z-normalized, although other transformations can be added.
- Checking *Search for cluster #* in the *sort options* menu causes an error if sorting with DBSCAN, HDBSCAN, or Spectral Clustering.
- Cluster # searching maxed to 15; thus it is better to use this parameter for subsets of data.

6 Developer

Added functionality to DRST is welcome and encouraged! Please submit a pull request for any added functionality you think would be generally useful for others.