VisPipeline Components
User Manual

Jose Gustavo de Souza Paiva

November 13, 2013

Contents

1 Data Source Format

There are two different types of Data Sources which can be used in the system: Points File (.data files) and Distance Matrix File (.dmat files). For all the details about these data sources, please consult the PEx user manual, available in `http://www.lcad.icmc.usp.br/~paulovic/pex/repository/pex_manual.zip`.

2 Classification Components

2.1 Training/Test set builder

This component is used to split a data set in two disjoint subsets, that can be used as training and test sets.

- Location: Sampling/
- Pipeline Example: **TrainingTestBuilder.pip**.
- Input:
  - points matrix: data set to be split, represented by a Points File, as presented in Section 1.
- Output:
  - matrix: subset to be used as training/test set, represented by a Points File, as presented in Section 1;
  - matrix: subset to be used as training/test set, represented by a Points File, as presented in Section 1.
- Component Parameters:
  - **Samples**: number of samples to be used for the training set. The remaining instances will compose the test set;
  - **Method**: method used to select the samples. The available methods are:
    - **Random**: the instances are randomly selected from the data set;
    - **K-means Clustering**: the data set is clustered using K-means clustering technique, and the instances are sampled from each obtained cluster;
    - **X-means Clustering**: the data set is clustered using X-means clustering technique, and the instances are sampled from each obtained cluster;
    - **Label Clustering**: the samples are collected from the groups that represent the classes of the data set, according to the classes scheme of the data set. This option require that the data set is previously labeled.
  - **Centroid Nearest Samples**: check this if you want that, considering the clusters obtained from the clustering procedure, the sampling process selects only the nearest instances of these clusters centroids. This option is not available if the **Random** is selected as sampling method;
  - **Overall Cluster Samples**: check this if you want that, considering the clusters obtained from the clustering procedure, the sampling process selects half of the samples that are the nearest instances of these clusters centroids, and half that are the furthest instances of these clusters centroids. This option is not available if the selected sampling method is **Random**;
  - **Number of Clusters**: number of clusters to be obtained from the clustering procedure. This option is available only if the selected sampling method is **K-means**;

  - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances in the clustering method.

    The available dissimilarity measures are:

    - City-block;
    - Cosine-based dissimilarity;
    - Euclidean;
    - Histogram Log;
    - Extended Jaccard;
    - Infinity norm;
    - Dynamic Time Warping (DTW);
    - Max Moving Euclidean;
    - Min Moving Euclidean.

2.2 Locally Weighted Projection Regression Classifier (LWPR)

This component is used to classify a data set using the **Locally Weighted Projection Regression Classifier (LWPR)** [14] classification technique.

- Location: Classification/PLS/
- Pipeline Example: **LWPRClassification.pip**.
- Input:
  - points matrix: data set to be classified, represented by a Points File, as presented in Section 1.
- Output:
  - matrix: classified data set, represented by a Points File, as presented in Section 1.
- Component Parameters:
  - **LWPR Training Parameters**: the LWPR algorithm implemented in the system is a Java wrapper to the LWPR library, and the entire documentation about its specific parameters are available in `http://wcms.inf.ed.ac.uk/ipab/slmc/research/lwpr/lwpr-doc.pdf`;
  - **Training Set File name**: the training data set file, represented by a Points File, as presented in Section 1. This option is available only when **Use Model** option is not checked;
  - **Method**: the method used by the classification model. There are two options, **One Against All** and **Multiclass Matrix**. More details about each method can be found in [8];
  - **Use Model**: check this if you want to use a previously created model to classify the data set. It is necessary to inform the correspondent classification model file;
  - **Save Generated Model**: check this if you want to save the classification model created by this classification process to a file or set of files. This option is available only when **Use Model** option is not checked.

2.3 Partial Least Squares Classifier (PLS)

This component is used to classify a data set using the **Partial Least Squares (PLS)** [17] classification technique.

- Location: Classification/PLS/
- Pipeline Example: **PLSClassification.pip**.
- Input:
  - points matrix: data set to be classified, represented by a Points File, as presented in Section 1.
- Output:
  - matrix: classified data set, represented by a Points File, as presented in Section 1.
- Component Parameters:
  - **Training Set File name**: the training data set file, represented by a Points File, as presented in Section 1. This option is available only when **Use Model** option is not checked;
  - **Method**: the method used by the classification model. There are two options, **One Against All** and **Multiclass Matrix**. More details about each method can be found in [8];
  - **Factors**: number of factors to be used in the creation of the PLS classification model(s). This is a parameter of the algorithm;
  - **Use Sampling**: check this if you want to sample the data set and use these samples to feed train the classification model. This functionality requires the following parameters:
    - **Sampling Clustering method**: clustering method used in the sampling process. K-means and X-means clustering techniques are available;
    - **Samples**: number of samples to be collected in the sampling process;
    - **Save Samples**: check this if you want to save the samples collected in the sampling process to a Points File;
    - **Centroid Nearest Samples**: check this if you want that, considering the clusters obtained from the clustering procedure, the sampling process selects only the nearest instances of these clusters centroids;
    - **Overall Cluster Samples**: check this if you want that, considering the clusters obtained from the clustering procedure, the sampling process selects half of the samples that are the nearest instances of these clusters centroids, and half that are the furthest instances of these clusters centroids;
    - **Number of Clusters**: number of clusters to be obtained from the clustering procedure. This option is available only if the selected sampling clustering method is **K-means**;
    - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances in the clustering method. The available dissimilarity measures are the same as show on the Section 2.1.
  - **Use Model**: check this if you want to load a previously created PLS dimensionality reduction model. For the **One Against All** option, one model file is stored for each class. Thus, if this type of model is selected, it is necessary to inform all .yml model files, correspondent to each class, or the directory containing these model files. For **Multiclass Matrix** option, a single model .yml file is created, and all information used by the system is stored in this file. Thus, if this type of model is selected, it is necessary to inform the .yml file of the correspondent model.
  - **Save Generated Model(s)**: check this if you want to save the model(s) created for the dimensionality reduction process, to be used in future reductions.

2.4 Support Vector Machine

This component is used to classify a data set using the **Support Vector Machine (SVM)** [13] classification technique.

- Location: Classification/SVM/
- Pipeline Example: **SVMClassification.pip**.
- Input:
    - points matrix: data set to be classified, represented by a Points File, as presented in Section 1.
- Output:
    - matrix: classified data set, represented by a Points File, as presented in Section 1;
    - distance: classified data set, represented by a Distance Matrix File, as presented in Section 1.
- Component Parameters:
    - **Use Model**: check this if you want to use a previously created model to classify the data set. it is necessary to inform the correspondent classification model file;
    - **Save Generated Model**: check this if you want to save the classification model created by this classification process to a file. This option is available only when **Use Model** option is not checked.
    - **Save Documents X Terms Matrix**: check this if you want to save the classified data set, represented by a Points File, as presented in Section 1.
    - **Save Distance Matrix**: check this if you want to save the classified data set, represented by a Distance Matrix File, as presented in Section 1.
    - **Training Set File name**: the training data set file, represented by a Points File, as presented in Section 1. This option is available only when **Use Model** option is not checked;
    - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances. The available dissimilarity measures are the same as show on the Section 2.1.
    - **SVM Training Parameters**: the SVM algorithm implemented in the system is based in the **libsvm** library, and the entire documentation about its specific parameters are available in`http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

2.5 Self Organizing Map Classifier (SOM)

This component is used to classify a data set using the **Self Organizing Maps (SOM)** [4] classification technique.

- Location: Classification/SOM/
- Pipeline Example: **SOMClassification.pip**.
- Input:
    - points matrix: data set to be classified, represented by a Points File, as presented in Section 1.
- Output:
    - matrix: classified data set, represented by a Points File, as presented in Section 1.

- Component Parameters:
  - **Training Set File name**: the training data set file, represented by a Points File, as presented in Section 1. This option is available only when **Use Model** option is not checked;
  - **Grid Size**: the dimensionality of the rectangular grid. This is a parameter of the algorithm;
  - **Number of Iterations**: the number of iterations used by the algorithm to construct the grid. This is a parameter of the algorithm;
  - **Initial Learning Rate**: the initial learning rate used to adjust the components of the grid. This is a parameter of the algorithm;
  - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances. The available dissimilarity measures are the same as show on the Section 2.1.
  - **Use Model**: check this if you want to use a previously created model to classify the data set. it is necessary to inform the correspondent classification model file;
  - **Save Generated Model**: check this if you want to save the classification model created by this classification process to a file or set of files. This option is available only when **Use Model** option is not checked.

2.6 Class Match/Graph Class Match Coordination

This component is used to perform the classification evaluation methodology named **Class Matching**, that creates a layout that presents the matching instances as green circles, and non-matching instances as red circles. This layout is useful to highlight the influence of the similarity amongst instances, and their neighborhood on the classification results. Use Class Match Coordination with projection-based layouts, and Graph Class Match Coordination with graph-based layouts, such as Tree layouts.

- Location: Coordination/
- Pipeline Example: **ClassMatching.pip**.
- Input:
  - coordination: the layout to insert the coordination between points. At least two layouts are required to construct the coordination.
- Output: none.
- Component Parameters: none.

2.7 Classifier Analysis

This component is used to perform the classification evaluation using a confusion matrix and classical evaluation measures, such number of Matching/Non-matching objects, as well as average values of Accuracy, Precision, Recall and F1 measures.

- Location: Classification/Classifier Analysis/
- Pipeline Example: **ClassifierAnalysis.pip**.
- Input:
  - Classified points matrix: classified data set, represented by a Points File, as presented in Section 1;

  - Classified distance matrix: classified data set, represented by a Distance Matrix File, as presented in Section 1.

- Output: none.
- Component parameters:
  - **Ground Truth**: the data set file containing the ground truth classification, represented by a Points File or a Distance Matrix File, as presented in Section 1;
  - **Window Label**: the label of the window containing the analysis. This parameter is optional.

2.8 ROC Graphic

This component is used to perform the classification evaluation using the ROC Graphic. The average ROC point, and the ROC points for each class are shown.

- Location: Classification/Classifier Analysis/
- Pipeline Example: **ROCAnalysis.pip**.
- Input:
  - Classified points matrix: classified data set, represented by a Points File, as presented in Section 1;
  - Classified distance matrix: classified data set, represented by a Distance Matrix File, as presented in Section 1.
- Output: none.
- Component parameters:
  - **Ground Truth**: the data set file containing the ground truth classification, represented by a Points File, as presented in Section 1;
  - **Window Label**: the label of the window containing the analysis. This parameter is optional.

3 Dimensionality Reduction Components

3.1 Locally Weighted Projection Regression (LWPR)

This component is used to reduce the dimensionality of a data set using the **Locally Weighted Projection Regression (LWPR)** [14] technique, and the **One Against All** method [8].

- Location: Transformation/Dimension Reduction/
- Pipeline Example: **LWPRReduction.pip**.
- Input:
  - points matrix: data set whose dimensionality will be reduced, represented by a Points File, as presented in Section 1;
  - Combined Inputs:
    - points matrix: data set whose dimensionality will be reduced, represented by a Points File, as presented in Section 1.

    - samples: samples to be used to train the dimensionality reduction model, represented by a Points File, as presented in Section 1.

- Output:
  - matrix: reduced data set, represented by a Points File, as presented in Section 1.
- Component Parameters:
  - **LWPR Training Parameters**: the LWPR algorithm implemented in the system is a Java wrapper to the LWPR library, and the entire documentation about its specific parameters are available in `http://wcms.inf.ed.ac.uk/ipab/slmc/research/lwpr/lwpr-doc.pdf`;
  - **Use Model**: check this if you want to use a previously created model to reduce the data set. It is necessary to inform the correspondent dimensionality reduction model file;
  - **Save Generated Model**: check this if you want to save the dimensionality reduction model created by this dimensionality reduction process to a file or set of files. This option is available only when **Use Model** option is not checked.

3.2 Partial Least Squares Supervised Reduction (PLS)

This component is used to reduce the dimensionality of a data set using the **Partial Least Squares (PLS)** [17] technique, using an external samples file [8].

- Location: Transformation/Dimension Reduction/
- Pipeline Example: **SupervisedPLSReduction.pip**.
- Input:
  - points matrix: data set whose dimensionality will be reduced, represented by a Points File, as presented in Section 1;
  - Combined Inputs:
    - points matrix: data set whose dimensionality will be reduced, represented by a Points File, as presented in Section 1.
    - samples: samples to be used to train the dimensionality reduction model, represented by a Points File, as presented in Section 1.
- Output:
  - matrix: reduced data set, represented by a Points File, as presented in Section 1.
- Component Parameters:
  - **Method**: the method used by the dimensionality reduction model. There are two options, **One Against All** and **Multiclass Matrix**. More details about each method can be found in [8];
  - **Factors**: number of factors to be used in the creation of the PLS classification model(s). This is a parameter of the algorithm;
  - **Use Model**: check this if you want to load a previously created PLS dimensionality reduction model. For the **One Against All** option, one model file is stored for each class. Thus, if this type of model is selected, it is necessary to inform all .yml model files, correspondent to each class, or the directory containing these model files. For **Multiclass Matrix** option, a single model .yml file is created, and all information used by the system is stored in this file. Thus, if this type of model is selected, it is necessary to inform the .yml file of the correspondent model.

  - **Save Generated Model(s)**: check this if you want to save the model(s) created for the dimensionality reduction process, to be used in future reductions.

3.3 Partial Least Squares Semi-Supervised Reduction (PLS)

This component is used to reduce the dimensionality of a data set using the **Partial Least Squares (PLS)** [17] technique, and performing a sampling process on the data set to be reduced to construct the dimensionality reduction model [8].

- Location: Transformation/Dimension Reduction/
- Pipeline Example: **SemiSupervisedPLSReduction.pip**.
- Input:
  - points matrix: data set whose dimensionality will be reduced, represented by a Points File, as presented in Section 1.
- Output:
  - matrix: reduced data set, represented by a Points File, as presented in Section 1.
- Component Parameters:
  - **Method**: the method used by the dimensionality reduction model. There are two options, **One Against All** and **Multiclass Matrix**. More details about each method can be found in [8];
  - **Factors**: number of factors to be used in the creation of the PLS classification model(s). This is a parameter of the algorithm;
  - **Sampling**: clustering method used in the sampling process. K-means and X-means clustering techniques are available;
  - **Samples**: number of samples to be collected in the sampling process;
  - **Centroid Nearest Samples**: check this if you want that, considering the clusters obtained from the clustering procedure, the sampling process selects only the nearest instances of these clusters centroids;
  - **Overall Cluster Samples**: check this if you want that, considering the clusters obtained from the clustering procedure, the sampling process selects half of the samples that are the nearest instances of these clusters centroids, and half that are the furthest instances of these clusters centroids;
  - **Number of Clusters**: number of clusters to be obtained from the clustering procedure. This option is available only if the selected sampling clustering method is **K-means**;
  - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances in the clustering method. The available dissimilarity measures are the same as show on the Section 2.1.

4 Layout Analysis Components

4.1 Correlation Coefficient

This component is used to perform an evaluation of the layout created by a visualization technique, using the Correlation Coefficient analysis [5].

- Location: Data Analysis/
- Pipeline Example: **CorrelationCoefficient.pip**.
- Input:
  - topic tree model: topic tree layout to which the analysis will be performed. More than one layout can be attached;
  - distance matrix: data set to which the analysis will be performed, represented by a Distance Matrix File, as presented in Section 1. More than one layout can be attached;
  - points matrix: data set to which the analysis will be performed, represented by a Points File, as presented in Section 1. More than one layout can be attached;
  - topic projection model: topic projection layout to which the analysis will be performed. More than one layout can be attached;
  - tree model: tree layout to which the analysis will be performed. More than one layout can be attached;
  - projection model: projection layout to which the analysis will be performed. More than one layout can be attached.
- Output: none.
- Component Parameters:
  - **Source File**: the data set file, represented by a Points File or a Distance Matrix file, as presented in Section 1. This measure is calculated based on the neighborhood of the instances on the original space. As the layout contains only the information about the neighborhood in the visualization space, it is necessary to recovery the original feature values to obtain the original neighborhood;
  - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances. The available dissimilarity measures are the same as show on the Section 2.1.
  - **Use Edges Weights (Tree models)**: using the similarity tree layout, the distance between two instances is considered as the sum of the edge values that compose the minimum path between them; using a projection layout, the distance between two instances is the Euclidean Distance on the visualization area. Check this to take into account the edges weights when calculating the distances between instances on the layout. If not checked, all the edge weights are assumed 1.
  - **Title**: the label of the window containing the analysis. This parameter is optional.

5 Sampling Components

5.1 Random Sampling reader

This component is used to randomly collect a set of samples from a data set.

- Location: Sampling/
- Pipeline Example: **RandomSampling.pip**.
- Input:
  - distance matrix: data set from which the samples will be collected, represented by a Distance Matrix File, as presented in Section 1;

  - points matrix: data set from which the samples will be collected, represented by a Points File, as presented in Section 1.
- Output:
  - distance matrix: set of samples extracted from the data set, represented by a Distance Matrix File, as presented in Section 1;
  - points matrix: set of samples extracted from the data set, represented by a Points File, as presented in Section 1.
- Component Parameters:
  - **Max Number of Instances**: the number of samples to be collected. This quantity may not be reached if **Equal Number of Instances per Class** is checked;
  - **Equal Number of Instances per Class**: check this if you want that the same number of instances are collected for each class. This option require that the data set is previously labeled.

5.2 Sampling reader

This component is used to collect a set of samples from a data set.

- Location: Sampling/
- Pipeline Example: **SamplingReader.pip**.
- Input:
  - points matrix: data set from which the samples will be collected, represented by a Points File, as presented in Section 1.
- Output:
  - points matrix: set of samples extracted from the data set, represented by a Points File, as presented in Section 1.
- Component Parameters:
  - **Samples**: number of samples to be used for the training set. The remaining instances will compose the test set;
  - **Method**: method used to select the samples. The available methods are:
    - **Random**: the instances are randomly selected from the data set;
    - **K-means Clustering**: the data set is clustered using K-means clustering technique, and the instances are sampled from each obtained cluster;
    - **X-means Clustering**: the data set is clustered using X-means clustering technique, and the instances are sampled from each obtained cluster;
    - **Label Clustering**: the samples are collected from the groups that represent the classes of the data set, according to the classes scheme of the data set. This option require that the data set is previously labeled.
  - **Centroid Nearest Samples**: check this if you want that, considering the clusters obtained from the clustering procedure, the sampling process selects only the nearest instances of these clusters centroids. This option is not available if the **Random** is selected as sampling method;

  - **Overall Cluster Samples**: check this if you want that, considering the clusters obtained from the clustering procedure, the sampling process selects half of the samples that are the nearest instances of these clusters centroids, and half that are the furthest instances of these clusters centroids. This option is not available if the selected sampling method is **Random**;
  - **Number of Clusters**: number of clusters to be obtained from the clustering procedure. This option is available only if the selected sampling method is **K-means**;
  - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances in the clustering method. The available dissimilarity measures are the same as show on the Section 2.1.

6 Projection Components

6.1 Partial Least Squares New Projection (PLS)

This component is used to project the instances of a data set in two dimensions, using the **Partial Least Squares (PLS)** [17] technique.

- Location: Projection/Technique
- Pipeline Example: **PLSProjection.pip**.
- Input:
  - points matrix: data set to be projected, represented by a Points File, as presented in Section 1.
  - Combined Inputs:
    - points matrix: data set to be projected, represented by a Points File, as presented in Section 1.
    - samples: samples to be used to train the projection model, represented by a Points File, as presented in Section 1.
- Output:
  - matrix: projected data set, represented by a Points File, as presented in Section 1.
- Component Parameters:
  - **Method**: the method used by the projection model. There are two options, **One Against All** and **Multiclass Matrix**. More details about each method can be found in [8];
  - **Factors**: number of factors to be used in the creation of the PLS classification model(s). This is a parameter of the algorithm;
  - **Sampling**: clustering method used in the sampling process. K-means and X-means clustering techniques are available. This option is available only when **Use Model** option is not checked;
  - **Samples**: number of samples to be collected in the sampling process. This option is available only when **Use Model** option is not checked;
  - **Centroid Nearest Samples**: check this if you want that, considering the clusters obtained from the clustering procedure, the sampling process selects only the nearest instances of these clusters centroids. This option is available only when **Use Model** option is not checked;
  - **Overall Cluster Samples**: check this if you want that, considering the clusters obtained from the clustering procedure, the sampling process selects half of the samples that are the nearest instances of these clusters centroids, and half that are the furthest instances of these clusters centroids. This option is available only when **Use Model** option is not checked;

  - **Number of Clusters**: number of clusters to be obtained from the clustering procedure. This option is available only if the selected sampling clustering method is **K-means**;
  - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances in the clustering method. The available dissimilarity measures are the same as show on the Section 2.1. This option is available only when **Use Model** option is not checked;
  - **Use Model**: check this if you want to load a previously created PLS dimensionality reduction model. For the **One Against All** option, one model file is stored for each class. Thus, if this type of model is selected, it is necessary to inform all .yml model files, correspondent to each class, or the directory containing these model files. For **Multiclass Matrix** option, a single model .yml file is created, and all information used by the system is stored in this file. Thus, if this type of model is selected, it is necessary to inform the .yml file of the correspondent model.
  - **Save Generated Model(s)**: check this if you want to save the model(s) created for the dimensionality reduction process, to be used in future reductions. This option is available only when **Use Model** option is not checked.

6.2 Self Organizing Map Projection (SOM)

This component is used to project the instances of a data set in two dimensions, using the **Self Organizing Map (SOM)** [4] technique.

- Location: Projection/Technique
- Pipeline Example: **SOMProjection.pip**.
- Input:
  - points matrix: data set to be projected, represented by a Points File, as presented in Section 1.
- Output:
  - matrix: projected data set, represented by a Points File, as presented in Section 1.
- Component Parameters:
  - **Training Set File name**: the training data set file, represented by a Points File, as presented in Section 1. This option is available only when **Use Model** option is not checked;
  - **Grid Size**: the dimensionality of the rectangular grid. This is a parameter of the algorithm. This option is available only when **Use Model** option is not checked;
  - **Number of Iterations**: the number of iterations used by the algorithm to construct the grid. This is a parameter of the algorithm. This option is available only when **Use Model** option is not checked;
  - **Initial Learning Rate**: the initial learning rate used to adjust the components of the grid. This is a parameter of the algorithm. This option is available only when **Use Model** option is not checked;
  - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances. The available dissimilarity measures are the same as show on the Section 2.1;
  - **Use Model**: check this if you want to use a previously created model to classify the data set. it is necessary to inform the correspondent classification model file;
  - **Save Generated Model**: check this if you want to save the classification model created by this classification process to a file or set of files. This option is available only when **Use Model** option is not checked.

6.3 Neighbor-Joining based Projection

This component is used to project the instances of a data set in two dimensions, using the **Neighbor Joining Technique (NJ)** [9] technique to construct a tree, use a radial layout technique [12] to position the instances on the visualization plane, and remove all the edges.

- Location: Projection/Technique
- Pipeline Example: **NJBasedProjection.pip**.
- Input:
  - points matrix: data set to be projected, represented by a Points File, as presented in Section 1;
  - distance matrix: data set to be projected, represented by a Distance Matrix File, as presented in Section 1.
- Output:
  - matrix: projected data set, represented by a Points File, as presented in Section 1.
- Component Parameters:
  - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances. The available dissimilarity measures are the same as show on the Section 2.1;
  - **Leaf Promotion**: check this to perform the leaf promotion procedure [7] after the tree construction;
  - **Tree Construction Algorithm**: the tree construction algorithm. The available techniques are: the original algorithm [9], Rapid Neighbor Joining algorithm [10] or Fast Neighbor Joining algorithm [2].

6.4 RadViz Projection

This component is used to project the instances of a data set in two dimensions, using the **RadViz** [6] technique.

- Location: Projection/Technique
- Pipeline Example: **RadVizProjection.pip**.
- Input:
  - points matrix: data set to be projected, represented by a Points File, as presented in Section 1.
- Output:
  - matrix: projected data set, represented by a Points File, as presented in Section 1.
- Component Parameters:
  - **Anchor Ordering**: the ordering of the axes that define the anchors, representing each data set dimension. The available options are: **None**, in which the anchors will follow the original sequence of the data set, **Correlation**, in which the anchors will be order by the correlation amongst the dimensions, and **Covariance**, in which the anchors will be order by the covariance amongst the dimensions.

7 Simple Tree Components

7.1 Simple Tree Model

This component is used to generate the complete layout structure from a tree structure, including information about each instance and the edges that connect them.

- Location: Simple Tree/Basics/
- Pipeline Example: **NJAlgorithms.pip**.
- Input:
  - simple tree: object containing the elements that compose the tree structure, including the tree nodes and edges.
- Output:
  - simple tree model: object containing the elements that compose the layout structure, including the tree instances and edges information.
- Component Parameters:
  - **Name**: name of the layout, used to identify it in evaluation analysis procedures. This parameter is optional.

7.2 Simple Radial Layout

This component is used to organize the layout in the visualization plane according to a radial layout [1].

- Location: Simple Tree/Layout/
- Pipeline Example: **NJAlgorithms.pip**.
- Input:
  - simple tree model: object containing the elements that compose the layout structure, including the tree instances and edges information.
- Output:
  - simple tree model: object containing the elements that compose the layout structure, including the tree instances and edges information, after the application of the radial layout procedure, with each instance containing information about its position in the visualization plane.
- Component Parameters: none.

7.3 Simple tree Viewer

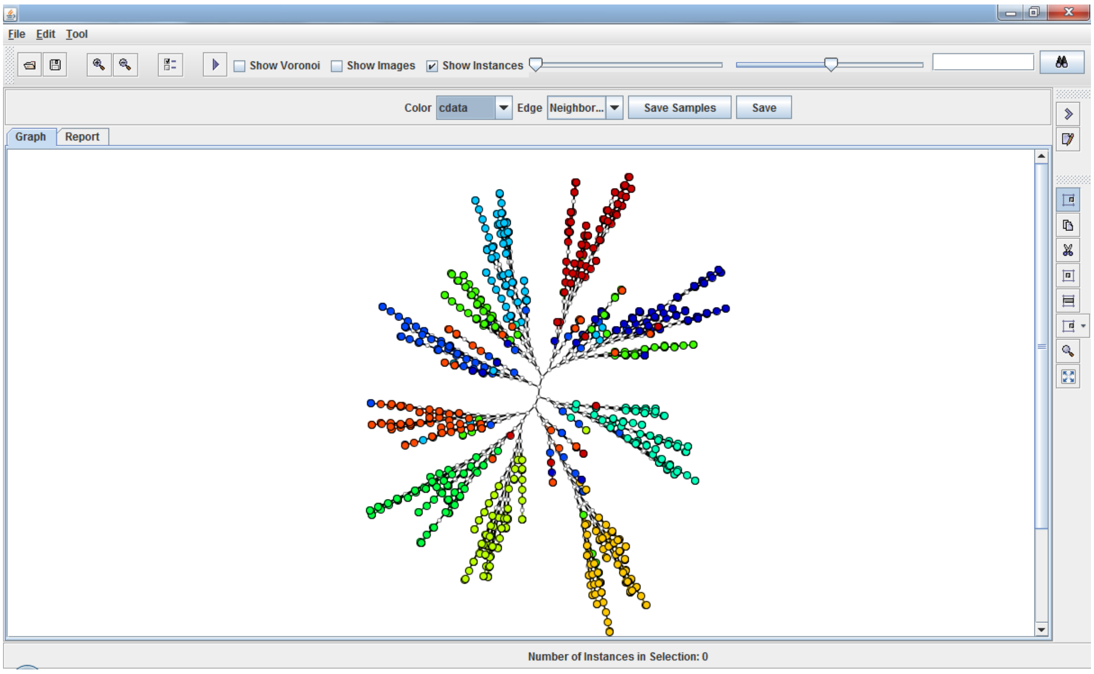
This component is used to construct a window to display the generated layout, allowing the visualization and exploration of a data set. The details about the functionalities of this window are presented in this section.

- Location: Simple Tree/View/
- Pipeline Example: **NJAlgorithms.pip**.

- Input:
  - simple tree model: object containing the elements that compose the layout structure, including the tree instances and edges information.
- Output: none.
- Component Parameters:
  - **Frame title**: name/description to be displayed as the window title. This parameter is optional.

Visualization Window

Figure 1 presents the visualization and exploration window, in which it is possible to visualize and interact with the tree layout. This section details all the interaction tools presented in this window.



☐ Show Voronoi ☐ Show Images ☑ Show Instances

Color cdata Edge Neighbor...

Number of Instances in Selection: 0

Figure 1: Visualization window.

- **Top Menu Functionalities**: performs several procedures using the layout points:

  Open Layout: Open a previously created layout.

  Save Layout: Save the current layout into a file.

  Export: Export the layout to a 2D Points file, as presented in Section 1, or to an image file (PNG, SVG, PDF).

  Clean Layout: Cancel the layout instance selection, setting the number of selected instances to 0.

Delete Points: Delete points from the current layout.

Memory Check: Provide information about the memory being used by the system.

Import Scalars: Import scalars to the current layout.

Export Scalars: Export the current scalar to a file.

Join Scalars: merge two scalar schemes to one.

Clustering Multidimensional Data: Performs a clustering procedure on the layout displayed in the Visualization area, using the original instances features.

Silhouette Coefficient: Analyze the current layout using the Silhouette measure [11].

Correlation Coefficient: Analyze the current layout using the Correlation measure [5].

Support Vector Machine Classification: Classify the layout displayed in the Visualization area using a classifier based on the **Support Vector Machine** [13] technique.

Create/Update LWPR Model: Creates/Update a LWPR classification model, using a set of instances selected in the layout displayed in the Visualization area as a training set.

LWPR Classifier: Classify the layout displayed in the Visualization area using a classifier based on the **Locally Weighted Projection Regression** [14].

Classification Evaluation: evaluates the results of a classification process, using the ground truth of a data set.

Calculate Hierarchical Clustering: Performs a hierarchical clustering procedure on the layout displayed in the Visualization area, using the instances positions as features. The available techniques are: Complete Link, Average Link [11], and Ward method [15].

Stress Curve: Calculates the stress curve associated with the layout displayed in the Visualization area.

Create Projection: Extracts the edges of the layout displayed in the Visualization area, creating a projection layout.

Tool Options: Customize layout properties.

- **Visualization Area**: displays the data set employing a similarity tree visualization technique. This area is divided in two tabs. The first tab, called **Layout** shows the layout generated by the visualization technique, representing the instances of a data set as circles, with their color representing their classes, according to a scalar scheme (if available). If the similarity tree technique is used, there are a set of white small circles representing the virtual nodes. The second tab, called **Report**, show informations about the data set and the layout, such as the number of instances, number of features, etc. The interaction tools directly related to the Visualization area are detailed as follows:

Color: scalar scheme to be used to color each instance of the layout in the visualization area. The default scalar scheme is '**cdata**', that represents the classes information loaded from the Points file. If there is no information about classes in the Points file, the '**cdata**' displays all instances with a same color. If you do not want to use any scalar scheme, select the item '**...**'.

Edge: this option represents the edge scheme to be displayed in the visualization area. The default value is '**Neighbor-Joining**'. If you do not want to use any edge scheme, select the item '**...**'.

Save Samples: samples a set of instances from the data set displayed in the Visualization area.

Save: save the data set displayed in the Visualization area, using the selected scalar scheme, to a Points File or a Distance Matrix.

Navigation in Visualization area: in Visualization area, it is possible to interact to the layout displayed in the Visualization area in several ways, described as follows:

- **Label Display**: passing the mouse over the circle that represents an instance will automatically display its label, as shown in Figure 2. By default, the name of the file that represents the instance is shown. However, it is possible to configure this functionality to display a correspondent image thumbnail, if the data set contains image files. To configure the label of each instance, please consult Section **Tool Preferences**;

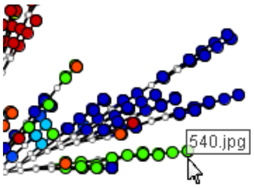


Figure 2: Label display.

- **Instance Selection**: clicking the left button of the mouse over a a circle that represents an instance will select it. When an instance is selected, all remaining instances become translucent, to highlight this selection, as shown in Figure 3. To configure the translucency degree of non-selected instances, please consult Section **Tool Preferences**;

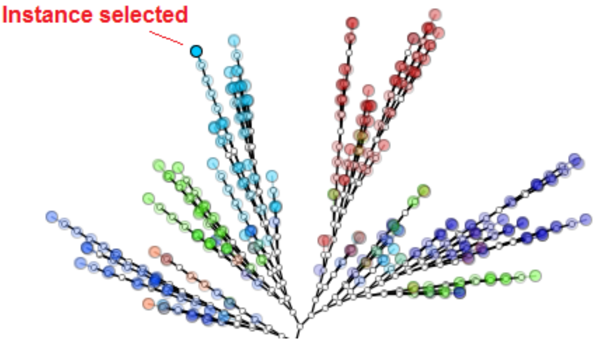


Figure 3: Instance selection.

- **Group Selection**: pressing the left button of the mouse and dragging it over the visualization area will create a rectangular selection, and all the instances inside the rectangle will be selected, as shown in Figure 4a. Pressing the right button of the mouse and dragging it over the visualization area will create a free selection area, and all the instances inside this area will be selected, as shown in Figure 4b. In both cases, release the mouse will confirm the selection. All the instances that are not selected become translucent, to highlight this selection. To configure the translucency degree of non-selected instances, please consult Section **Tool Preferences**. The number of instances in selection will be displayed in the bottom of the screen;

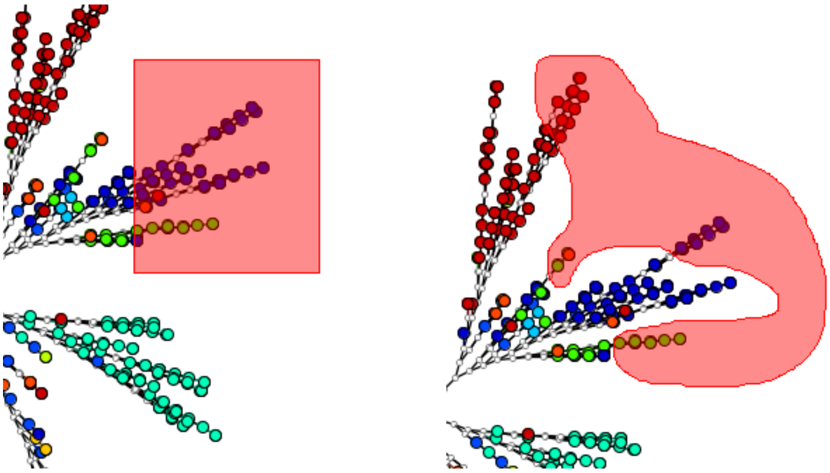

(a) Rectangular selection. (b) Free selection.

Figure 4: Types of instance selection.

- **Tree Selection**: clicking the left button of the mouse in a internal instance will automatically select all its child instances. The number of instances in selection will be displayed in the bottom of the screen;
- **Selective Selection**: holding the CTRL key and clicking the left button of the mouse in the instances on the layout displayed in the Visualization area will select them, one by one. Clicking them again will cancel the selection. The number of instances in selection will be displayed in the bottom of the screen;
- **Cancel Selection**: after an instance of group of instances is selected, clicking the right button of the mouse in any part of the Visualization area will cancel the entire selection. The number of instances in selection will be set to 0;
- **Move instances**: after an instance or group of instances is selected, pressing the left button of the mouse in one of the selected instances and dragging it over the Visualization area will move all the selected instances.

The functionalities related to the layout are presented and detailed in Table 1. The Tool configuration is detailed in the next section.

- **Tool Preferences**: allows the configuration of several properties of the layout displayed in the Visualization area. Figure 5 shows the Tool Preferences dialog screen.

  The following functionalities can be configured:

Table 1: Visualization area functionalities.

Functionality	Short Description
Basic Operations buttons	
📂	Open a layout to be displayed in the Visualization area.
💾	Save the layout displayed in the Visualization area to a file.
🔍	Zoom in the current layout.
🔍	Zoom out the current layout.
☑	Customize layout properties.
▶	Apply a Force Based Layout Procedure [12] to the current layout.
Show Voronoi	Check this to create and display the Voronoi diagram for the layout displayed in the Visualization area.
Show Images	Check this to display the instances of the data set displayed in the Visualization area as image thumbnails. The images option is available only when an image data set is being displayed.
Show Instances	Check this to display the instances of the data set displayed in the Visualization area.
Control Cells Grouping slider	Use this slider to control the grouping level of the Voronoi diagram. This option is available only when **Show Voronoi** is selected.
Control Image Transparency slider	Use this slider to control the transparency of the images displayed in the mosaic of the Voronoi diagram. This option is available only when **Show Voronoi** and **Show images** are selected, as well when an image data set is being displayed.

High/Poor Quality: the quality of the layout displayed in the Visualization area. Selecting High quality may slow down the interaction with the layout depending on the machine capabilities.

Label font size: size of the font displayed in an instance label, displayed when the mouse is over it. This option is available only when 'file name' is selected as **Title** option.

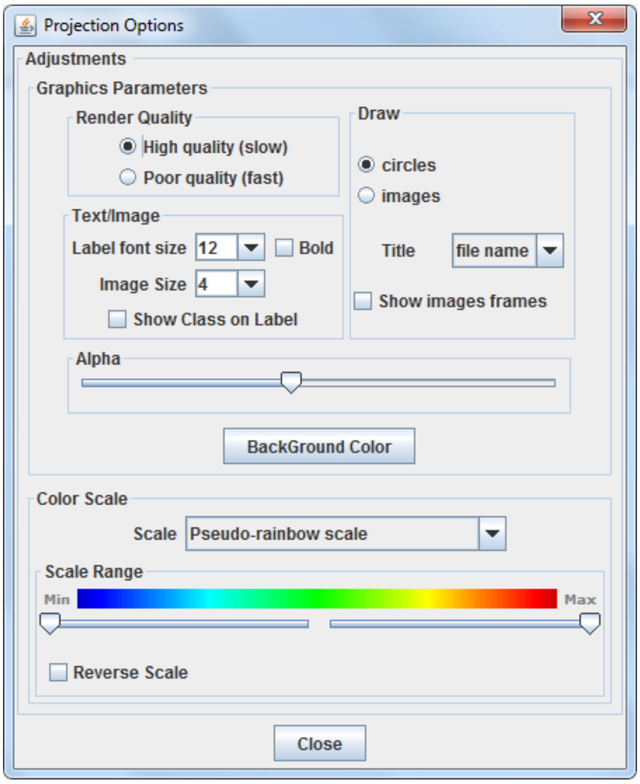
Bold: check this if you want the instances labels to be displayed in bold, when the mouse is over it. This option is available only when 'file name' is selected as **Title** option.

Image size: size of the instance correspondent image thumbnail, when the mouse is over it. This option is available only when 'images' is selected as **Draw** option, and when an image data set is being displayed.

Show Class on Label: check this if you want the instances classes are displayed with the instances labels, displayed when the mouse is over it. This option is available only when 'file name' is selected as **Title** option.

Draw: select between 'circle' or 'images' to display the instances as circles or image thumbnails, respectively. The images option is available only when an image data set is being displayed.

Title: select between 'file name' or 'image' to display the instances labels as its file names or image thumbnails, respectively, when the mouse is over it. The images option is available only when an image data set is being displayed.



● High quality (slow)
○ Poor quality (fast)
● circles
○ images
Label font size 12 ☐ Bold
Image Size 4
☐ Show Class on Label
Title file name
☐ Show images frames
Scale Pseudo-rainbow scale
☐ Reverse Scale

Figure 5: Tool Preferences dialog screen.

Show images frames: check this if you want the instances classes colors are displayed as the frame of the images thumbnails. This option is available only when 'images' is selected as **Draw** option, and when an image data set is being displayed.

Alpha: translucency degree of the non-selected instances in the layout.

Background Color: background color of the Visualization area.

Color Scale: color scale scheme used to map the instances classes to the color of the correspondent circles in the Visualization area. Using the default color scale, when a data set file is loaded into the Visualization area the range of classes scheme is mapped to a a color scale scheme, with the lowest class number mapped to blue, the highest class number mapped to red, and the remaining classes interpolated between these colors. Different color scales will map these colors to different color ranges. It is possible to select predefined color scales, modify the range of a selected color scale, or reverse it.

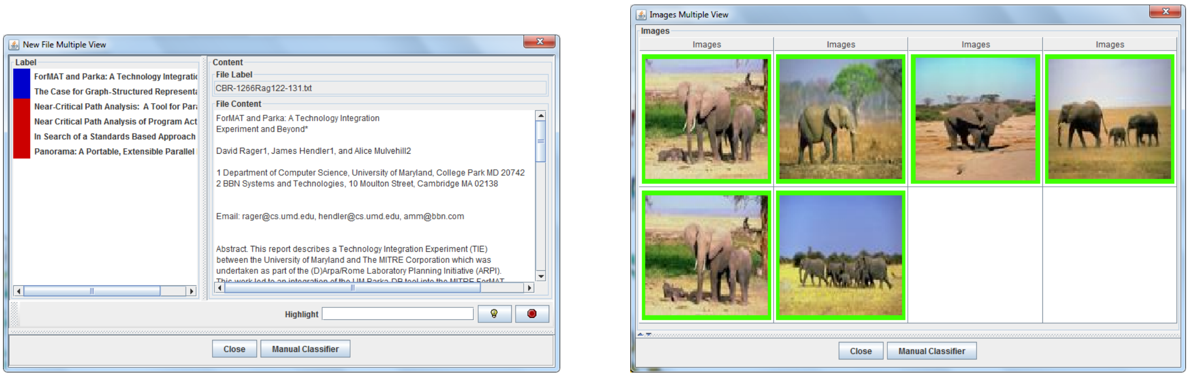
- **Layout Explorer Menu**: allows direct manipulation of the layout displayed in the Visualization area. Table 2 presents these interaction tools, described as follows:

  Select Points/Tree: allows the selection of instances in the layout. This is the default interaction tool.

Table 2: Layout explorer functionalities.

Functionality	Short Description
Layout Explorer buttons	
⬚	Select instances from the current layout.
📋	View content of instances from the current layout.
✂	Cut instances from the current layout to a new layout.
⊡	Create a new label scheme for the current layout.
☰	Select all instances of a class from the current layout.
⬚ ▾	Select a node and display only its neighbourhood.
🔍	Display and coordinate a selected branch on a new window
⤢	Display the content of a cluster tree node on a new window

View Content: allows the inspection of the content of an instance or group of instances. Figure 6 shows the Content visualization window, for text documents (6a) and images (6b). You may be prompted to load the data set files, if it had not been loaded before by other functionality. This interaction tool allows the manual classification of instances.


New File Multiple View
Label
ForMAT and Parka: A Technology Integratio
The Case for Graph-Structured Representa
Near-Critical Path Analysis: A Tool for Para
Near Critical Path Analysis of Program Act
In Search of a Standards Based Approach
Panorama: A Portable, Extensible Parallel
Content
File Label
File Content
Highlight
Close
Manual Classifier
Images Multiple View
Images
Images
Images
Images
Images
Close
Manual Classifier

CBR-1266Rag122-131.txt

ForMAT and Parka: A Technology Integration
Experiment and Beyond*

David Rager1, James Hendler1, and Alice Mulvehill2

1 Department of Computer Science, University of Maryland, College Park MD 20742
2 BBN Systems and Technologies, 10 Moulton Street, Cambridge MA 02138

Email: rager@cs.umd.edu, hendler@cs.umd.edu, amm@bbn.com

Abstract. This report describes a Technology Integration Experiment (TIE)
between the University of Maryland and The MITRE Corporation which was
undertaken as part of the (D)Arpa/Rome Laboratory Planning Initiative (ARPI).

(a) Text content visualization.

(b) Image content visualization.

Figure 6: Types of content visualization.

Split: allows the visualization of a specific part of the layout, by selecting this part. A new Visualization area is created, containing only this selection, and maintaining the instances organization of the original layout.

Label group of instances: allows a manual labeling of groups of instances, and consequently the creation of a new class scheme. Each selection performed on the layout will label all instances inside this selection with a single class. This functionality is useful to create labels for a non-labeled data set, or to redefine a previously label scheme. Every time a manual group labeling process is started, a new class scheme is created. To each class label created a color is automatically associated. Also, every time the focus changes from one group to another, the labels are recomputed, to accommodate the new class to the class range. Using the default color scheme, the first label created will always be blue, and the last will always be red. As many labels as necessary can be assigned. Figure 7 shows an example of a manual labeling process.

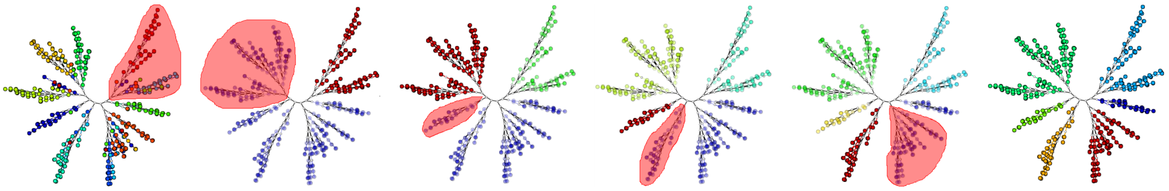

(a) First labeling. (b) Second labeling. (c) Third labeling. (d) Fourth labeling. (e) Fifth labeling. (f) Final Result.

Figure 7: Manual classification process.

View Instances of a Class: allows the selection of all instances of a specific class, just selecting one instance in the layout that belongs to this class.

Partial Tree View: allows the display of the neighbourhood of a selected node. The original tree view is replaced by one in which only the neighbourhood of the selected node is visible. This view mode is useful when the tree is too crowded, and there is no enough visual space to display all nodes at the same time. If one clicks at the arrow in the selection button, a menu is displayed to let the user set the radius size of the node neighbourhood, or cancel this partial view.

Coordinated Branch Zoom: allows the selection of a tree branch to show the content on a new coordinated window. All changes on the zoomed view are also reflected on the original branch (changes like node removal, layout changes, etc.).

Cluster Node Expansion: allows the display of a cluster node content on a new window. The content of the cluster node is displayed as a new sub-tree on a new window. Thus the user can explore the nodes and sub-cluster nodes of the new tree visualization.

7.4 Neighbor-Joining Algorithms for Simple Tree

This component is used to create a similarity tree structure using the **Neighbor Joining (NJ)** [9] algorithm.

- Location: Simple Tree/Technique
- Pipeline Example: **NJAlgorithms.pip**.
- Input:
  - points matrix: data set whose tree will be constructed, represented by a Points File, as presented in Section 1;
  - distance matrix: data set whose tree will be constructed, represented by a Distance Matrix File, as presented in Section 1.
- Output:

  - simple tree: object containing the elements that compose the tree structure, including the tree nodes and edges.

- Component Parameters:
  - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances. The available dissimilarity measures are the same as show on the Section 2.1;
  - **Leaf Promotion**: check this to perform the leaf promotion procedure [7] after the tree construction;
  - **NJ Algorithm**: the tree construction algorithm. The available techniques are: the original algorithm [9], Rapid Neighbor Joining algorithm [10] and Fast Neighbor Joining algorithm [2].

7.5 Hierarchical Clustering Tree

This component is used to create a tree representation for the result of an hierarchical clustering application to a data set.

- Location: Simple Tree/Technique
- Pipeline Example: **HierarquicalTree.pip**.
- Input:
  - points matrix: data set whose tree will be constructed, represented by a Points File, as presented in Section 1;
  - distance matrix: data set whose tree will be constructed, represented by a Distance Matrix File, as presented in Section 1.
- Output:
  - simple tree: object containing the elements that compose the tree structure, including the tree nodes and edges.
- Component Parameters:
  - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances. The available dissimilarity measures are the same as show on the Section 2.1;
  - **Clustering Type**: the hierarchical clustering algorithm. The available techniques are: Complete Link, Complete Link, Average Link [11], and Ward method [15];
  - **Leaf Promotion**: check this to perform the leaf promotion procedure [7] after the tree construction.

7.6 Multi-Scale Neighbor-Joining

This component is used to create a similarity tree structure using the **Multi-Scale Neighbor Joining (NJ)** algorithm.

- Location: Simple Tree/Technique
- Pipeline Example: **MNJTree.pip**.
- Input:

  - points matrix: data set whose tree will be constructed, represented by a Points File, as presented in Section 1;
  - Combined Inputs:
    - points matrix:data set whose tree will be constructed, represented by a Points File, as presented in Section 1.
    - points clusters: file containing information about the data set clusters.

- Output:
  - simple tree: object containing the elements that compose the tree structure, including the tree nodes and edges.

- Component Parameters:
  - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances. The available dissimilarity measures are the same as show on the Section 2.1;
  - **NJ Algorithm**: the tree construction algorithm. The available techniques are: the original algorithm [9], Rapid Neighbor Joining algorithm [10] and Fast Neighbor Joining algorithm [2];
  - **Leaf Promotion**: check this to perform the leaf promotion procedure [7] after the tree construction;
  - **Use Classes as Initial Clusters**: check this if you want to use the original classes of the data set as the initial clusters, that is, no initial clustering algorithm will be performed. This option require that the data set is previously labeled.
  - **Number Clusters**: number of clusters to be obtained from the clustering procedure. This option is available only if the selected sampling clustering method is **K-means**, **Bissecting K-means**, **K-medoids**, or **Hierarchical**;
  - **Clustering Type**: the hierarchical clustering algorithm. The available techniques are: Complete Link, Complete Link, Average Link [11], and Ward method [15]. This option is available only if the selected sampling clustering method is **Hierarchical**;
  - **Clustering method**: clustering method used by the procedure. The available clustering methods are:
    - K-means;
    - Bissecting K-means;
    - K-medoids;
    - Hierarchical;
    - X-means;
    - DBScan;
    - IsoData;
  - **Colapse Tree**: Check this to display the tree colapsed by the groups obtained by the clustering procedure. In this case, the tree will be displayed in levels, and each level represents the application of the clustering procedure in a group of the data set;
  - **Max Cluster Size**: maximum number of instances in each group. This is a threshold value that determines when a group of instances needs to be divided again in the process. This is a parameter of the algorithm.

7.7 Simple Tree Reading/Writing Components

The following components read previously created simple tree layouts from a file:

- Simple Tree model Binary reader;
- Simple Tree model XML reader.

Simple tree layouts can be writen in XML files (.xml) or binary files (.bmodel). There is no difference on the contents of these two types of files.

- Location: Simple Tree/Input
- Pipeline Examples: **TreeBinaryReader.pip** (binary layout), and **TreeXMLReader.pip** (XML layout).
- Input: none.
- Output:
  - Simple Tree Model: object containing the elements that compose the tree layout structure.
- Component Parameters:
  - **File name**: the previously created simple tree layout file.

The following components write a simple tree layout to a file:

- Simple Tree model Binary writer;
- Simple Tree model XML writer.

- Location: Simple Tree/Output
- Pipeline Examples: **TreeBinaryWriter.pip** (binary layout), and **TreeXMLWriter.pip** (XML layout).
- Input: none.
  - Simple Tree Model: object containing the elements that compose the tree layout structure.
- Output: none.
- Component Parameters:
  - **File name**: the file to sabe the simple tree layout file.

7.8 Newick Tree Reader Components

This component is used to read a previously created similarity tree from a file, using the Newick representation format [3].

- Location: Simple Tree/Input
- Pipeline Example: **NewickReader.pip**.
- Input: none.
- Output:

  - simple tree: object containing the elements that compose the tree structure, including the tree nodes and edges.

- Component Parameters:
  - **File name**: the previously created simple tree layout file;
  - **Leaf Promotion**: check this to perform the leaf promotion procedure [7] after the tree construction.

8 Distance Matrix Perturber

This component is used to execute the distance matrix perturbation procedure. This procedure modify the distances values in a distance matrix, approximating the instances that belong to the same class by a factor value, and moving away distances that belong to different classes by another factor value.

- Location: Distance/Perturb/
- Pipeline Example: **DmatPerturber.pip**.
- Input:
  - points matrix: data set whose distance matrix will be perturbed, represented by a Points File, as presented in Section 1;
  - distance matrix: distance matrix to be perturbed, represented by a Distance Matrix File, as presented in Section 1.
- Output:
  - Hashmap: A set of distance matrices representing the gradual values perturbation.
- Component Parameters:
  - **Number of Distance Matrices**: the number of distance matrices to be generated by the process;
  - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances. The available dissimilarity measures are the same as show on the Section 2.1;
  - **% Approximating Factor**: the factor value used to approximate instances that belong to the same class;
  - **% Distancing Factor**: the factor value used to move away instances that belong to different classes;
  - **Save Distance Matrices**: check this to save the generated distance matrices to correspondent files.

9 Layout sets Components

9.1 Neighbor-Joining Algorithms Set Builder

This component is used to create a set of Neighbor Joining trees, one for each distance matrix informed.

- Location: Simple Tree/Technique

- Pipeline Example: **DmatPerturber.pip**.
- Input:
  - distance matrices: the set of distance matrices whose trees will be constructed. These matrices can be informed one bye one, or in set represented by a Hashmap with the keys representing the names/desctiptions, and the values representing the matrices.
- Output:
  - ArrayList: A set of simple tree layouts constructed for each distance matrix.
- Component Parameters:
  - **Leaf Promotion**: check this to perform the leaf promotion procedure [7] after the tree construction;
  - **NJ Algorithm**: the tree construction algorithm. The available techniques are: the original algorithm [9], Rapid Neighbor Joining algorithm [10] and Fast Neighbor Joining algorithm [2].

9.2 Simple Tree Set View Frame

This component is used to display a set of Neighbor Joining trees in a single window. This visualization window provides the same functionalities presented in the **Visualization Window** section, with an extra slider component (Figure 8). This component can be used to switch between the layouts from the set.

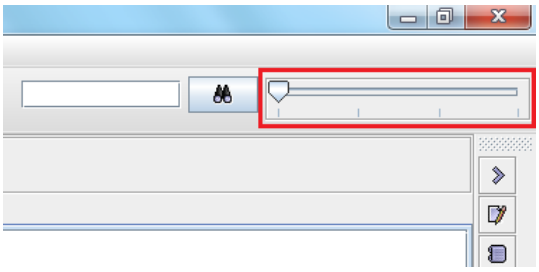


Figure 8: Layout switch slider.

- Pipeline Example: **DmatPerturber.pip**.
- Input:
  - **Coordinator**: coordinator component, to coordinate a visualization and exploration with other layout;
  - **Original Distance Matrix**: data set whose trees will be constructed, represented by a Distance Matrix File, as presented in Section 1. As the layout contains only the information about the position of the instances in the visualization space, it is necessary to recovery the original feature values to obtain the original distances;
  - **tree models set**: the set of simple tree layouts to be displayed;
  - **Original Data Matrix**: data set whose trees will be constructed, represented by a Points File, as presented in Section 1. As the layout contains only the information about the distances in the visualization space, it is necessary to recovery the original feature values to obtain the original distances.
- Component Parameters: none.

10 NINJA Algorithm Components

This section details the components related to the creation of Neighbor-Joining trees using the NINJA algorithm [16].

10.1 Ninja Distance Matrix File Converter

This component is used to convert a Distance Matrix file, from the format presented in Section 1 to the one used in the NINJA software[1].

- Location: Distance/Calculator
- Pipeline Example: **NinjaDistanceMatrixFileConverter.pip**.
- Input:
  - points matrix: data set to be converted, represented by a Points File, as presented in Section 1;
  - distance matrix: data set to be converted, represented by a Distance Matrix File, as presented in Section 1;
  - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances. The available dissimilarity measures are the same as show on the Section 2.1.
- Output: none.
- Component Parameters:
  - **File name**: output NINJA distance matrix file (.dist file);
  - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances in the clustering method. The available dissimilarity measures are the same as show on the Section 2.1.

10.2 Ninja NJ Tree

This component is used to create a similarity tree structure using the **NINJA** [16] algorithm.

- Location: Simple Tree/Technique
- Pipeline Example: **NinjaTree.pip**.
- Input:
  - points matrix: data set whose tree will be constructed, represented by a Points File, as presented in Section 1;
  - distance matrix: data set whose tree will be constructed, represented by a Distance Matrix File, as presented in Section 1.
- Output:
  - simple tree: object containing the elements that compose the tree structure, including the tree nodes and edges.

---

[1] http://nimbletwist.com/software/ninja/

- Component Parameters:
  - **Dissimilarity Measure**: the Dissimilarity measure used to calculate the distance amongst the instances. The available dissimilarity measures are the same as show on the Section 2.1;
  - **Leaf Promotion**: check this to perform the leaf promotion procedure [7] after the tree construction;
  - **Use External Memory**: check this if you want to use the secondary memory to store intermediate steps of the algorithm. This option is useful in cases where the data set is very large.

Bibliography

[1] C. Bachmaier, U. Brandes, and B. Schlieper. *Drawing phylogenetic trees*. Springer, 2005.

[2] I. Elias and J. Lagergren. Fast neighbor joining. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580, pages 1263–1274, 2005.

[3] J. Felsenstein, J. Archie, W. Day, W. Maddison, C. Meacham, F. Rohlf, and D. Swofford. The newick tree format, 1986.

[4] T. Kohonen. *Self-organizing maps*, volume 30. Springer, 2001.

[5] J. Lee Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988.

[6] L. Nováková and O. Štěpánková. Multidimensional clusters in radviz. In *Proceedings of the 6th WSEAS International Conference on Simulation, Modelling and Optimization*, pages 470–475. World Scientific and Engineering Academy and Society (WSEAS), 2006.

[7] J. Paiva, L. Florian, H. Pedrini, G. Telles, and R. Minghim. Improved similarity trees and their application to visual data classification. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2459–2468, 2011.

[8] J. G. S. Paiva, W. R. Schwartz, H. Pedrini, and R. Minghim. Semi-supervised dimensionality reduction based on partial least squares for visual analysis of high dimensional data. *Computer Graphics Forum*, 31(3pt4):1345–1354, 2012.

[9] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.

[10] M. Simonsen, T. Mailund, and C. N. Pedersen. Rapid neighbour-joining. In *Proceedings of WABI 2008*, pages 113–122, Karlsruhe, Germany, September 2008.

[11] P. N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[12] E. Tejada, R. Minghim, and L. G. Nonato. On improved projection techniques to support visual exploration of multidimensional data sets. *Information Visualization*, 2(4):218–231, 2003.

[13] V. Vapnik. *The nature of statistical learning theory*. springer, 2000.

[14] S. Vijayakumar, A. D'Souza, and S. Schaal. Incremental online learning in high dimensions. *Neural Computation*, 17:2602–2634, 2005.

[15] J. H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[16] T. J. Wheeler. Large-scale neighbor-joining with ninja. In *Algorithms in Bioinformatics*, pages 375–389. Springer, 2009.

[17] H. Wold. Partial least squares. In *Encyclopedia of Statistical Sciences*, volume 6, pages 581–591. Wiley, New York, NY, USA, 1985.