TUBE: Command Line Program Calls

March 15, 2009

Contents

Chapter 1

Command Line Program Calls

This document is a short user guide for the programs developed in the thesis **Tree-based Density Estimation: Algorithms and Applications** [1]. It lists for each application introduced in the thesis relevant examples of java program calls of the corresponding classes from the command line. All programs for this thesis have been implemented in Java and are based on the WEKA machine learning tool set [2]. (Note, all command lines given in this appendix perform only simple 10-fold cross-validation but most experiments performed for the work in this thesis used 10×10 cross-validation using the WEKA experiment environment.)

Chapter 2

Program Calls Used in Application Discretization

2.1 Drawing Histograms

Drawing a 10 Bin Equal-Width Histogram Output data for gnuplot to print an equal-width histogram with 10 bins (`-B 10`); the input file is `gauss.arff` (`-i gauss.arff`) and the output file suffix is `gauss-B10` (`-X gauss-B10`).

```
java weka.estimators.EqualWidthEstimator -i gauss.arff -B 10 \
-V 8 -X gauss-B10

[plot with gnuplot]
plot 'gauss-B10-0EW.hist' title "EW 10 bins" with lines
```

Drawing a 10 Bin Equal-Frequency Histogram

```
java weka.estimators.EqualFrequencyEstimator -i gauss.arff -B 10 \
-V 8 -X gauss-B10

[plot with gnuplot]
plot 'gauss-B10-0EF.hist' title "EF 10 bins" with lines
```

Drawing a TUBE Histogram with a fixed number of bins

```
java weka.estimators.TUBEstimator -i gauss.arff -B 10 \
-V 8 -X gauss-B10

[plot with gnuplot]
plot 'gauss-B10-V8-0LL.hist' title "TUBE 10 bins" with lines
```

Drawing a TUBE Histogram, TUBE computes the number of bins using cross-validation

```
java weka.estimators.TUBEstimator -i gauss.arff -B 10 \
-V 8 -X gauss

[plot with gnuplot]
plot 'gauss-0LL.hist' title "TUBE CV" with lines
```

2.2 Discretizing

(TUBE) TUBE discretization with cross-validation for the number of bins (`default`).

```
java weka.estimators.TUBEstimator -i iris01.arff
```

(EW-10) Equal-width discretization with ten bins.

```
java weka.estimators.EqualWidthEstimator -i iris01.arff -B 10
```

(EWcvB) Equal-width discretization with cross-validation for the number of bins (`- Y`); the maximum number of bins is set to 100 (`-B 100`).

```
java weka.estimators.EqualWidthEstimator -i iris01.arff -Y -B 100
```

(EWcvBO) Equal-width discretization with cross-validation for the origin of the bins (`- Z`) and the number of bins (`- Y`); the maximum number of bins is set to 100 (`-B 100`).

```
java weka.estimators.EqualWidthEstimator -i iris01.arff -Y -Z -B 100
```

(EF-10) Equal-frequency discretization with ten bins.

```
java weka.estimators.EqualFrequencyEstimator -i iris01.arff -B 10
```

Chapter 3

Program Calls Used in Application Naive Bayes

Introduction: Naive Bayes classifier used with varying estimators In WEKA the standard implementation of a naive Bayes classifier is the class NaiveBayes. NaiveBayes uses Gaussian distributions for density estimation. The class NaiveBayesParametrized substitutes the Gaussian estimators with an estimator given in the parameter `-E`.

```
java weka.classifiers.bayes.NaiveBayesParametrized -t iris.arff \
-E "weka.estimators.EqualWidthEstimator -B 10"
```

(Gauss) Naive Bayes classifier using Gaussian distribution.

```
java weka.classifiers.bayes.NaiveBayes -t iris.arff
```

(TUBE-CV) Naive Bayes classifier using TUBE histograms; the number of bins is cross-validated (`default`); the maximum number of bins is 100 (`-B 100`).

```
java weka.classifiers.bayes.NaiveBayesParametrized -t iris.arff \
-E "weka.estimators.TUBEstimator -B 100"
```

(EW-CV) Naive Bayes classifier using equal-width histograms; the number of bins is cross-validated (`-Z`); the maximum number of bins is 100 (`-B 100`).

```
java weka.classifiers.bayes.NaiveBayesParametrized -t iris.arff \
-E "weka.estimators.EqualWidthEstimator -Z -B 100"
```

(TUBE-02) Naive Bayes classifier using TUBE histograms; TUBE with the cut distance set to 0.1 (`-Z 0.1`) and the minimal bin width set to 0.2 (`-L -U 2`); the number of bins is cross-validated (`default`); the maximum number of bins is 100 (`-B 100`).

```
java weka.classifiers.bayes.NaiveBayesParametrized -t iris.arff \
-E "weka.estimators.TUBEstimator -L -Z 0.1 -U 2 -B 100 "
```

(TUBE-15) Naive Bayes classifier using TUBE histograms; TUBE with the cut distance set to 0.1 (`-Z 0.1`) and the minimal bin width set to 0.2 (`-L -U 2`); the number of bins is cross-validated (`default`); the maximum number of bins is 15 (`-B 15`).

```
java weka.classifiers.bayes.NaiveBayesParametrized -t iris.arff \
-E "weka.estimators.TUBEstimator -L -Z 0.1 -U 2 -B 15"
```

(TUBE-EW) Naive Bayes classifier using TUBE histograms; TUBE estimatior with the cut distance and the minimal bin width set using equal-width heuristic (`-L -U 4`); the number of bins is cross-validated (`default`); the maximum number of bins is 100 (`-B 100`).

```
java weka.classifiers.bayes.NaiveBayesParametrized -t iris.arff \
-E "weka.estimators.TUBEstimator -L -U 4 -B 100 "
```

(EW-15) Naive Bayes classifier using equal-width histograms; the number of bins is fixed to 15 (`-B 15`).

```
java  weka.classifiers.bayes.NaiveBayesParametrized -t iris.arff \
-E "weka.estimators.EqualWidthEstimator -B 15"
```

(EW-30) Naive Bayes classifier using equal-width histograms; the number of bins is fixed to 30 (`-B 30`).

```
java  weka.classifiers.bayes.NaiveBayesParametrized -t iris.arff \
-E "weka.estimators.EqualWidthEstimator -B 30"
```

Chapter 4

Program Calls Used for Presentation Methods

4.1 Bin Lists

Bin List Generated Using the MultiTUBE Estimator java -Xmx1500M weka.estimators.MultiTUBE -i musk1-propositional.arff -V17 -B 10 -N ¿musk1-B10-V17.binpos

4.2 Bin Lists for Two-Class Problems

Bin List Generated Using MultiTUBEClusterer

```
java -Xmx1500M weka.clusterers.MultiTUBEClusterer \
-t musk1_propositional.arff -T musk1-propositional.arff  -V 19 -L last -C 1 -X \
-E "weka.estimators.MultiTUBE -B 50 -N" >musk1-B50-C1-V19.binlist
```

Bin List After Using the Mixing of Binnings Method

```
java -Xmx1500M weka.clusterers.MultiTUBEClusterer \
-t musk1_propositional.arff -T musk1-propositional.arff  -V 19 -L last -C 3 -Y 2
-E "weka.estimators.MultiTUBE -B 10 -N" >musk1-B50-C3-Y2-V19.binlist
```

4.3 Bin Position Overview

```
java -Xmx1500M weka.clusterers.MultiTUBEClusterer \
-t musk1_propositional.arff -T musk1-propositional.arff  -V 9 -L last -C 1 -X \
-E "weka.estimators.MultiTUBE -B 10 -N" >musk1-B10-C1-V9.binpos
```

Chapter 5

Program Calls Used in Application Clustering

5.1 Generating the Example Datasets

Introduction: WEKA Subspace Data Generator For the evaluation of the application clustering all datasets used were generated using WEKA's subspace data generator (generating arff datasets). The class SubspaceCluster has parameters to set the number of attributes (`-a 4`), if a class attribute is generated or not (`-c`), the percentage of generated noise (`-P 10` for 10 percent noise), the output file name for the generated arff-file (`-o test.arff`) and the value range for all attributes that are not listed in the attribute list of the cluster (`-s -10.0,10.0`).

Several clusters can be defined with each a `-C` parameter list containing: the type of cluster specified (`-G` for Gaussian, `-A` for uniformly distributed), the dimensions of the cluster (`-D ..,..,...` for uniform distributions the minimal and the maximal value of the range, for Gaussian distributions the mean value and the standard deviation are given), and the number of instances in this cluster (`-N 100..200`).

In the example below the WEKA subspace clusterer generates two clusters. The first cluster is normally distributed around the mean point $(2.0, 4.0, .., ..)$ and has the standard deviation 1.0 in both attributes. This cluster is clustered in the subspace of the first two attributes only and is normally distributed in the last two attributes in the range $[-10.0, 10.0]$. A value between 100 and 200 is randomly selected as number of instances in this cluster.

The second cluster is a subspace cluster in the third and fourth attribute and is there randomly uniform distributed between 8.0 and 9.0 in attribute 3 and between 6.0 and 7.0 in attribute 4. The first two attribute values of the instances in this cluster are uniformly distributed in the range $[-10.0, 10.0]$. For this second cluster 300 instances are generated in this fashion.

```
java weka.datagenerators.clusterers.SubspaceCluster  -P 10 -c -a 4 \
-s -10.0,10.0 -o test.arff \
-C "-G 1,2  -D 2.0,1.0,4.0,1.0 -N 100..200" \
-C "-A 3,4  -D 8.0,9.0,6.0,7.0 -N 300..300"
```

Example 1: Dataset with three clusters in fifteen dimensions, each with three relevant attributes.

```
java weka.datagenerators.clusterers.SubspaceCluster  -P 10 -c -a 15 \
-s -3.4,12.6 -o appcluster_multi_3_Vers2.arff \
-C "-G 1,2,3  -D 2.0,1.0,4.0,1.0,1.0,1.0 -N 200..200" \
-C "-G 1,2,4  -D 8.0,1.0,0.0,1.0,4.0,1.0 -N 200..200"  \
-C "-G 2,3,4  -D 10.0,1.0,7.0,1.0,9.0,1.0 -N 200..200"
```

Example 2: Dataset with three clusters in four dimensions, each with three relevant attributes.

```
java weka.datagenerators.clusterers.SubspaceCluster  -P 10 -c -a 4 \
   -s -2.3,12.6 -o appcluster_multi_3_Vers3.arff \
-C "-G 1,2,3  -D 2.0,1.0,4.0,1.0,1.0,1.0 -N 200..200" \
-C "-G 1,2,4  -D 8.0,1.0,0.0,1.0,4.0,1.0 -N 200..200"  \
-C "-G 2,3,4  -D 10.0,1.0,7.0,1.0,9.0,1.0 -N 200..200"
```

Example 3: Dataset with eight clusters in four dimensions, with three relevant attributes each.

```
java weka.datagenerators.clusterers.SubspaceCluster  -P 10 -c -a 15 \
    -s 0.0,10.0 -o appcluster_multi_3_Vers4.arff \
-C "-G 1,2,3  -D 2.0,1.0,4.0,1.0,8.0,1.0 -N 500..500" \
-C "-G 1,2,4  -D 4.0,1.0,8.0,1.0,4.0,1.0 -N 500..500" \
-C "-G 2,3,4  -D 2.0,1.0,2.0,1.0,8.0,1.0 -N 500..500" \
-C "-G 1,2,3  -D 8.0,1.0,4.0,1.0,4.0,1.0 -N 500..500" \
-C "-G 1,2,4  -D 2.0,1.0,8.0,1.0,2.0,1.0 -N 500..500" \
-C "-G 2,3,4  -D 2.0,1.0,8.0,1.0,2.0,1.0 -N 500..500" \
-C "-G 1,2,3  -D 4.0,1.0,4.0,1.0,2.0,1.0 -N 500..500" \
-C "-G 1,2,4  -D 8.0,1.0,8.0,1.0,8.0,1.0 -N 500..500"
```

Example 4: Example with diagonally set clusters. 2 datasets.

```
java weka.datagenerators.clusterers.SubspaceCluster  -P 10 -c -a 2 \
     -o appcluster_multi_3_Vers11.arff \
-C "-G 1,2  -D 2.0,1.0,2.0,1.0 -N 200..200" \
-C "-G 1,2  -D 7.0,1.0,2.0,1.0 -N 200..200" \
-C "-G 1,2  -D 12.0,1.0,2.0,1.0 -N 200..200"

java weka.datagenerators.clusterers.SubspaceCluster  -P 10 -c -a 2 \
     -o appcluster_multi_3_Vers12.arff \
-C "-G 1,2  -D 2.0,1.0,2.0,1.0 -N 200..200" \
-C "-G 1,2  -D 7.0,1.0,2.0,1.0 -N 200..200" \
-C "-G 1,2  -D 12.0,1.0,4.5,1.0 -N 200..200" \
-C "-G 1,2  -D 4.5,1.0,7.0,1.0 -N 200..200"
```

Example 5: Datasets with Oblong-shaped Clusters. .

```
java weka.datagenerators.clusterers.SubspaceCluster  -P 10 -c -a 2 \
    -s 0.0,10.0 -o appcluster_multi_3_Vers8.arff \
-C "-A 1,2  -D 2.0,3.0,2.0,3.0  -N 500..500" \
-C "-A 1,2  -D 2.0,3.0,8.0,14.0  -N 500..500"
```

Example 6: Dataset with Non-Convex Cluster.

```
java weka.datagenerators.clusterers.SubspaceCluster  -P 10 -c -a 2 \
-o appcluster_multi_3_Vers9.arff \
-C "-A 1,2  -D 2.0,3.0,1.0,6.0  -N 500..500" \
-C "-A 1,2  -D 3.0,8.0,5.0,6.0  -N 500..500" \
-C "-A 1,2  -D 7.0,8.0,1.0,5.0  -N 500..500" \
-C "-A 1,2  -D 0.0,10.0,-1.0,8.0  -N 100..100"
```

5.2 Clustering with TUBE Clusterer

Introduction: TUBE Clusterer The class weka.clusterers.MultiTUBEClusterer clusters the data given in arff format (`-t test.arff`), ignoring the attribute given with its index as the `-L` parameter. Before the clustering the multi-dimensional space is discretized using the multidimensional TUBE estimator (`MultiTUBE`). The parameters of the MultiTUBE discretizer can be set via the `-E` parameter.

```
java weka.clusterers.MultiTUBEClusterer -t test.arff -L last \
 -E "weka.estimators.MultiTUBE -B 10"
```

Evaluation of the TUBE Clusterer All tests done to evaluate the TUBE clusterer used the default setting of 100 bins for the discretization step.

```
java weka.clusterers.MultiTUBEClusterer -t test.arff -L last
```

Chapter 6

Program Calls Used in Application Multiple-Instance Learning

6.1 Using TUBE Clusters for MI Classification

Introduction: TUBE Multiple-instance Classifier TUBEMIC performes clustering to find a cluster defining the positive concept area. In order to do this it uses D_{diff} (`-Y 2`) instead of the density for the bin ‘height’. Bins are only accepted as part of the positive concept cluster if their density is above a certain threshold (`-P 90` sets the threshold to 90 percent). Also, in all examples TUBEMIC uses the mixing of binnings (`-C 3`). The number of bins is set with the parameter list of the MultiTUBE estimator in the `-E` parameter.

(TMIC-90) TUBE multiple-instance classifier using 90 percent as density threshold (`-P 90`); the mixing of binnings (`-C 3`) was performed with 10 bins (`-B 10`) and uses D_{diff} (`-Y 2`) instead of the density for the bin ‘height’.

```
java weka.classifiers.mi.TUBEMIC -t eastwest_relational.arff -P 90 \
-C "weka.clusterers.MultiTUBEClusterer -L last -C 3 -Y 2 \
-E \"weka.estimators.MultiTUBE -B 10\" "
```

(TMIC-70) and (TMIC-60) Same as TMIC-90 but with different threshold settings (`-P 70` and `-P 60`).

(MIDD) The results of the TUBE multiple-instance classifier were compared with the Multiple-instance Diverse Density classifier

```
java weka.classifiers.mi.MIDD -t eastwest_relational.arff
```

6.2 Improving the Efficiency of the Diverse Density Algorithm

Introduction: Diverse Density Augmented with TUBE The standard Diverse Density classifier is implemented in the class MIDD. The TUBEDD class implements the Diverse Density (MIDD) classifier augmented with TUBE. The TUBE method computes starting points for the Diverse Density method in two ways: as the centre point of the modes (`default`) bins or as randomly selected training points from the mode bins (`-S 6`).

Mixing of Binnings Method The TUBE augmented DD method uses the mixing of binnings method, which is implemented in the TUBE clusterer's class MultiTUBEClusterer (`-C 3`, and for using D_{diff}: `-Y 2`). The number of bins is given with the specification of a MultiTUBE estimator (`-E "weka.estimators.MultiTUBE -B 5"`).

(TUBE-5) Diverse Density augmented with TUBE: using five bins (`-B 5`) in the histograms generated for the mixing of binnings (`-C 3 -Y 2`) and using the centre point of the mode (`default`) as starting point.

```
java weka.classifiers.mi.TUBEDD -t eastwest_relational.arff \
-C "weka.clusterers.MultiTUBEClusterer -L last -C 3 -Y 2 \
-E \"weka.estimators.MultiTUBE -B 5\" "
```

(TUBE-10) Diverse Density augmented with TUBE: using ten bins (`-B 10`) in the histograms generated for the mixing of binnings (`-C 3 -Y 2`) and using the centre point of the mode (`default`) as starting point.

```
java weka.classifiers.mi.TUBEDD -t eastwest_relational.arff \
-C "weka.clusterers.MultiTUBEClusterer -L last -C 3 -Y 2 \
-E \"weka.estimators.MultiTUBE -B 10\" "
```

(TUBE-10-rand) Diverse Density augmented with TUBE: using ten bins (`-B 10`) in the histograms generated for the mixing of binnings (`-C 3 -Y 2`), and using a training instance randomly selected from the concept area (`-S 6`) as starting point.

```
java weka.classifiers.mi.TUBEDD -t eastwest_relational.arff -S 6 \
-C "weka.clusterers.MultiTUBEClusterer -L last -C 3 -Y 2 \
-E \"weka.estimators.MultiTUBE -B 10\" "
```

(MIDD) Multiple-instance Diverse Density classifier.

```
java weka.classifiers.mi.MIDD -t eastwest_relational.arff
```

Bibliography

[1] Gabi Schmidberger. *Tree-based Density Estimation: Algorithms and Applications; (not yet published)*. PhD thesis, Department of Computer Science, University of Waikato, 2009.

[2] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, San Francisco, 2 edition, 2005.