

WEKA's Knowledge Flow Interface

CSCI 682: Data Mining

Knowledge Discovery in Databases

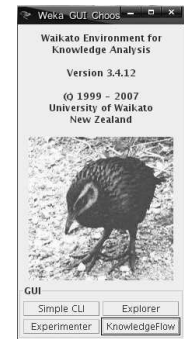
Dr. Juliano

Department of Computer Science
California State University, Chico
Chico, CA 95929-0410



Overview

- Knowledge Flow Interface
 - WEKA components are selected from a tool bar, positioned a layout canvas, and connected into a directed graph to model a complete system that processes and analyzes data



The Knowledge Flow Environment



Data Sources

- used to indicate where data is coming from
- supports various file types and sources
- configurable for
 - file name of data source
 - dataset or instance (incremental) loading



Data Sinks

- used to indicate where data is going to be written
- supports various file types and sources
- configurable for
 - file name of data source



Filters

- used to preprocess data prior to classification or learning
- supports both supervised and unsupervised filters
- configurable depending on filter type





Classifiers

- supports all classification algorithms presented in the textbook
- parameters are configurable depending on classification algorithm



Clusterers

- supports all clustering algorithms presented in the textbook
- parameters are configurable depending on clustering algorithm



Evaluation

- used to configure both inputs to and outputs from algorithms
- supports various algorithm performance evaluators
- output format fairly “standardized”



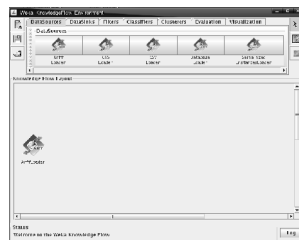
Visualization

- used to visually display outputs
- supports performance and summaries
- comparable to options from *Explorer* interface



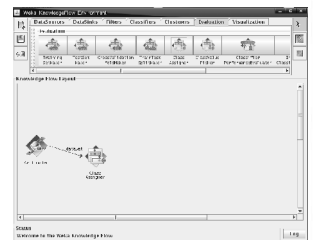
Example: Decision Tree Classifier

- 1) Specify a data source
 - select *ARFFLoader*
 - right-click the *ARFFLoader* icon to configure by specifying the location of the dataset



Example: Decision Tree Classifier

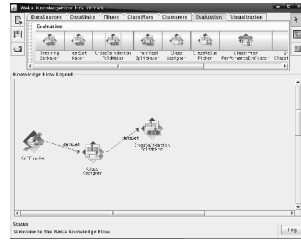
- 2) Specify which attribute is the class
 - select *ClassAssigner* from the *Evaluation* panel
 - right-click the *ARFFLoader* icon and select *dataset* to connect to *ClassAssigner*
 - right-click the *ClassAssigner* to configure the target class



Example: Decision Tree Classifier

3) Specify cross-validation

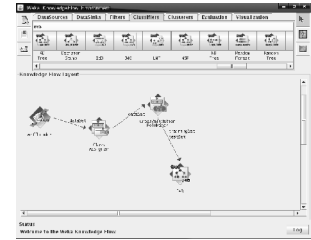
- select *CrossValidation FoldMaker* from the *Evaluation* panel
- right-click the *ClassAssigner* icon and select *dataset* to connect to *CVFoldMaker*
- right-click *CrossValidation FoldMaker* to configure ...



Example: Decision Tree Classifier

4) Specify decision tree algorithm

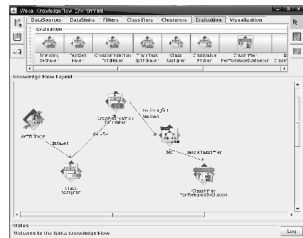
- select *J48* from the *Classifiers* panel
- right-click the *CVFoldMaker* icon and select *trainingSet* to connect to *J48*; repeat, but select *testSet* to connect to *J48*



Example: Decision Tree Classifier

5) Specify evaluation

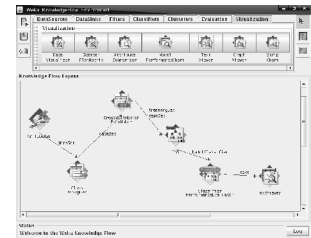
- select *ClassifierPerformance Evaluator* from the *Evaluation* panel
- right-click the *J48* icon and select *batchClassifier* to connect to *ClassifierPerformance Evaluator*



Example: Decision Tree Classifier

6) Specify evaluation output

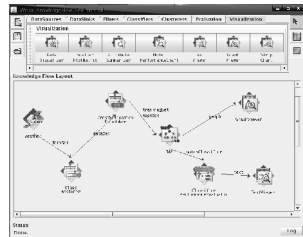
- select *TextViewer* from the *Visualization* panel
- right-click the *ClassifierPerformance Evaluator* icon and select *text* to connect to *TextViewer*



Example: Decision Tree Classifier

7) To allow viewing of decision trees per fold

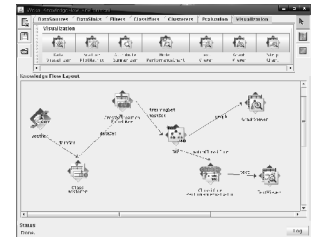
- select *GraphViewer* from the *Visualization* panel
- right-click the *J48* icon and select *graph* to connect to *GraphViewer*



Example: Decision Tree Classifier

8) Run experiments

- right-click *ARFFloader* icon and select *Start loading*
- right-click *TextViewer* icon for textual summary
- right-click *GraphViewer* icon to view decision tree for each fold
- reconfigure and re-run ...



Example: Incremental Learning

- Classifiers:
 - *AODE*, Bayes averaged, one-dependence estimators
 - *NaiveBayesUpdateable*, incremental naïve Bayes classifier that learns one instance at a time
 - *Winnow*, mistake-driven perceptron with multiplicative updates
 - instance-based learners: *IB1*, *IBk*, *KStar*, *LWL*



Example: Incremental Learning

- Filters:

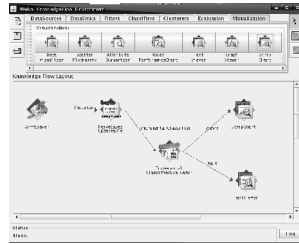
<i>Add</i>	<i>NumericTransform</i>
<i>AddExpression</i>	<i>Obfuscate</i>
<i>Copy</i>	<i>Remove</i>
<i>FirstOrder</i>	<i>RemoveType</i>
<i>MakeIndicator</i>	<i>RemoveWithValues</i>
<i>MergeTwoValues</i>	<i>SparseToNonSparse</i>
<i>NonSparseToSparse</i>	<i>SwapValues</i>
<i>NumericToBinary</i>	



Example: Incremental Learning

- Configure an experiment based on the figure to the right ...

Note: Although incremental learning algorithms can process data files that are too large to fit in memory, many instance-based learners store entire dataset internally.



Example: Incremental Learning

- Strip charts
 - scrolling data plot showing both the *accuracy* and the *root mean-squared probability error* against time

