

Chapter 1

WEKA

A Machine Learning Workbench for Data Mining

Eibe Frank, Mark Hall, Geoffrey Holmes, Richard Kirkby, Bernhard Pfahringer, Ian H. Witten

Department of Computer Science, University of Waikato, Hamilton, New Zealand

{eibe, mhall, geoff, rkirkby, bernhard, ihw}@cs.waikato.ac.nz

Len Trigg

Reel Two, P O Box 1538, Hamilton, New Zealand

len@reeltwo.com

Abstract The Weka workbench is an organized collection of state-of-the-art machine learning algorithms and data preprocessing tools. The basic way of interacting with these methods is by invoking them from the command line. However, convenient interactive graphical user interfaces are provided for data exploration, for setting up large-scale experiments on distributed computing platforms, and for designing configurations for streamed data processing. These interfaces constitute an advanced environment for experimental data mining. The system is written in Java and distributed under the terms of the GNU General Public License.

Keywords: machine learning software, data mining, data preprocessing, data visualization, extensible workbench

Experience shows that no single machine learning method is appropriate for all possible learning problems. The universal learner is an idealistic fantasy. Real datasets vary, and to obtain accurate models the bias of the learning algorithm must match the structure of the domain.

The Weka workbench is a collection of state-of-the-art machine learning algorithms and data preprocessing tools. It is designed so that users can quickly try out existing machine learning methods on new datasets

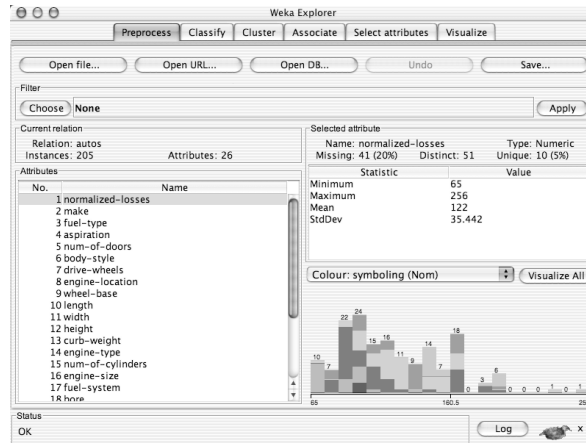


Figure 1.1. The Explorer interface.

in very flexible ways. It provides extensive support for the whole process of experimental data mining, including preparing the input data, evaluating learning schemes statistically, and visualizing both the input data and the result of learning. This has been accomplished by including a wide variety of algorithms for learning different types of concepts, as well as a wide range of preprocessing methods. This diverse and comprehensive set of tools can be invoked through a common interface, making it possible for users to compare different methods and identify those that are most appropriate for the problem at hand.

The workbench includes methods for all the standard data mining problems: regression, classification, clustering, association rule mining, and attribute selection. Getting to know the data is a very important part of data mining, and many data visualization facilities and data preprocessing tools are provided. All algorithms and methods take their input in the form of a single relational table, which can be read from a file or generated by a database query.

Exploring the data

The main graphical user interface, the “Explorer,” is shown in Figure 1.1. It has six different panels, accessed by the tabs at the top, that correspond to the various data mining tasks supported. In the “Preprocess” panel shown in the Figure, data can be loaded from a file or extracted from a database using an SQL query. The file can be in CSV format, or in the system’s native ARFF file format. Database access is

provided through Java Database Connectivity, which allows SQL queries to be posed to any database for which a suitable driver exists. Once a dataset has been read, various data preprocessing tools, called “filters,” can be applied—for example, numeric data can be discretized. In the Figure the user has loaded a data file and is focusing on a particular attribute, *normalized-losses*, examining its statistics and a histogram.

Through the Explorer’s second panel, called “Classify,” classification and regression algorithms can be applied to the preprocessed data. Classification algorithms typically produce decision trees or rules, while regression algorithms produce regression curves or regression trees. This panel also enables users to evaluate the resulting models, both numerically through statistical estimation and graphically through visualization of the data and examination of the model (if the model structure is amenable to visualization). Users can also load and save models.

The third panel, “Cluster,” enables users to apply clustering algorithms to the dataset. Again the outcome can be visualized, and, if the clusters represent density estimates, evaluated based on the statistical likelihood of the data. Clustering is one of two methodologies for analyzing data without an explicit target attribute that must be predicted. The other one comprises association rules, which enable users to perform a market-basket type analysis of the data. The fourth panel, “Associate,” provides access to algorithms for learning association rules.

Attribute selection, another important data mining task, is supported by the next panel. This provides access to various methods for measuring the utility of attributes, and for finding attribute subsets that are predictive of the data. Users who like to analyze the data visually are supported by the final panel, “Visualize.” This presents a color-coded scatter plot matrix, and users can then select and enlarge individual plots. It is also possible to zoom in on portions of the data, to retrieve the exact record underlying a particular data point, and so on.

The Explorer interface does not allow for incremental learning, because the Preprocess panel loads the dataset into main memory in its entirety. That means that it can only be used for small to medium sized problems. However, some incremental algorithms are implemented that can be used to process very large datasets. One way to apply these is through the command-line interface, which gives access to all features of the system. An alternative, more convenient, approach is to use the second major graphical user interface, called “Knowledge Flow.” Illustrated in Figure 1.2, this enables users to specify a data stream by graphically connecting components representing data sources, preprocessing tools, learning algorithms, evaluation methods, and visualization tools. Using

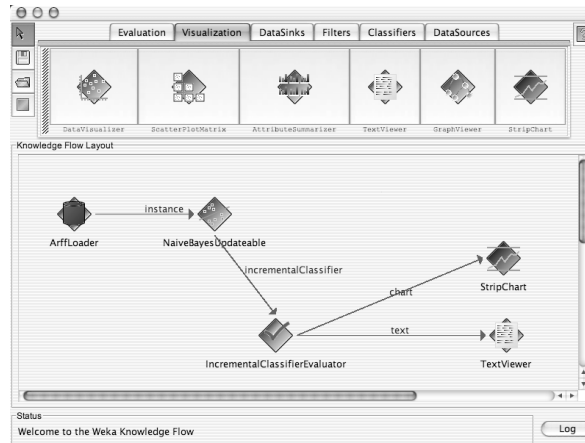


Figure 1.2. The Knowledge Flow interface.

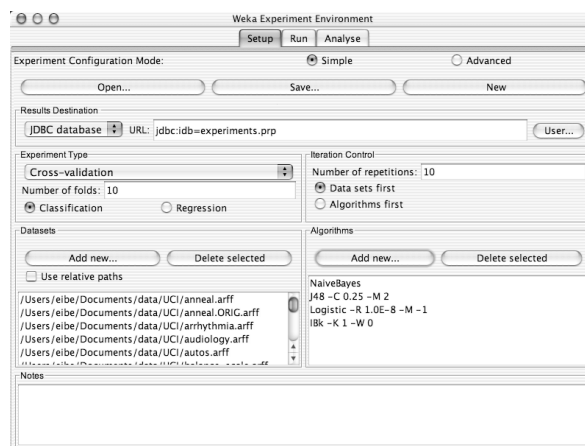


Figure 1.3. The Experimenter interface.

it, data can be loaded and processed incrementally by those filters and learning algorithms that are capable of incremental learning.

An important practical question when applying classification and regression techniques is to determine which methods work best for a given problem. There is usually no way to answer this question *a priori*, and one of the main motivations for the development of the workbench was to provide an environment that enables users to try a variety of learning techniques on a particular problem. This can be done interactively in

the Explorer. However, to automate the process Weka includes a third interface, the “Experimenter,” shown in Figure 1.3. This makes it easy to run the classification and regression algorithms with different parameter settings on a corpus of datasets, collect performance statistics, and perform significance tests on the results. Advanced users can also use the Experimenter to distribute the computing load across multiple machines using Java Remote Method Invocation.

Methods and algorithms

Weka contains a comprehensive set of useful algorithms for a panoply of data mining tasks. These include tools for data engineering (called “filters”), algorithms for attribute selection, clustering, association rule learning, classification and regression. In the following subsections we list the most important algorithms in each category. Most well-known algorithms are included, along with a few less common ones that naturally reflect the interests of our research group.

An important aspect of the architecture is its modularity. This allows algorithms to be combined in many different ways. For example, one can combine bagging, boosting, decision tree learning and arbitrary filters directly from the graphical user interface, without having to write a single line of code. Most algorithms have one or more options that can be specified. Explanations of these options and their legal values are available as built-in help in the graphical user interfaces. They can also be listed from the command line. Additional information and pointers to research publications describing particular algorithms may be found in the internal Javadoc documentation.

Classification. Implementations of almost all main-stream classification algorithms are included. Bayesian methods include naive Bayes, complement naive Bayes, multinomial naive Bayes, Bayesian networks, and AODE. There are many decision tree learners: decision stumps, ID3, a C4.5 clone called “J48,” trees generated by reduced error pruning, alternating decision trees, and random trees and forests thereof. Rule learners include OneR, an implementation of Ripper called “JRip,” decision tables, single conjunctive rules, and Prism. There are several separating hyperplane approaches like support vector machines with a variety of kernels, logistic regression, voted perceptrons, Winnow and a multi-layer perceptron. There are many lazy learning methods like IB1, IBk, lazy Bayesian rules, KStar, and locally-weighted Learning.

As well as the basic classification learning methods, so-called “meta-learning” schemes enable users to combine instances of one or more of the basic algorithms in various ways: bagging, boosting (including

the variants AdaboostM1 and LogitBoost), and stacking. A method called “FilteredClassifier” allows a filter to be paired up with a classifier. Classification can be made cost-sensitive, or multi-class, or ordinal-class. Parameter values can be selected using cross-validation.

Regression. There are implementations of many regression schemes. They include multiple and simple linear regression, piecewise regression, a multi-layer perceptron, support vector regression, locally-weighted learning, decision stumps, regression and model trees (M5) and rules (M5rules). The standard instance-based learning schemes IB1 and IBk can be applied to regression problems (as well as classification problems). Moreover, there are additional meta-learning schemes that apply to regression problems, such as additive regression and regression by discretization.

Clustering. At present, only a few standard clustering algorithms are included: KMeans, EM for naive Bayes models, farthest-first clustering, and Cobweb. This list is likely to grow in the near future.

Association rule learning. The standard algorithm for association rule induction is Apriori, which is implemented in the workbench, and there is also Tertius, which can extract first-order rules.

Attribute selection. Both wrapper and filter approaches to attribute selection are supported. A wide range of filtering criteria are implemented, including correlation-based feature selection, the chi-square statistic, gain ratio, information gain, symmetric uncertainty, and a support vector machine-based criterion. There are also a variety of search methods: forward and backward selection, best-first search, genetic search, and random search. Additionally, principal components analysis can be used to reduce the dimensionality of a problem.

Filters. Processes that transform instances and sets of instances are called “filters,” and they are classified according to whether they make sense only in a prediction context (called “supervised”) or in any context (called “unsupervised”). We further split them into “attribute filters,” which work on one or more attributes of an instance, and “instance filters,” which work on entire instances.

Unsupervised attribute filters include adding a new attribute, adding a cluster indicator, adding noise, copying an attribute, discretizing a numeric attribute, normalizing or standardizing a numeric attribute, making indicators, merging attribute values, transforming nominal to binary values, obfuscating values, swapping values, removing attributes,

replacing missing values, turning string attributes into nominal ones or word vectors, computing random projections, and processing time series data. Unsupervised instance filters transform sparse instances into non-sparse instances and vice versa, randomize and resample sets of instances, and remove instances according to certain criteria.

Supervised attribute filters include support for attribute selection, discretization, nominal to binary transformation, and re-ordering the class values. Finally, supervised instance filters resample and subsample sets of instances to generate different class distributions—stratified, uniform, and arbitrary user-specified spreads.

System architecture

In order to make its operation as flexible as possible, the workbench was designed with a modular, object-oriented architecture that allows new classifiers, filters, clustering algorithms and so on to be added easily. A set of abstract Java classes, one for each major type of component, were designed and placed in a corresponding top-level package.

All classifiers reside in subpackages of the top level “classifiers” package and extend a common base class called “Classifier.” The Classifier class prescribes a public interface for classifiers and a set of conventions by which they should abide. Subpackages group components according to functionality or purpose. For example, filters are separated into those that are supervised or unsupervised, and then further by whether they operate on an attribute or instance basis. Classifiers are organized according to the general type of learning algorithm, so there are subpackages for Bayesian methods, tree inducers, rule learners, etc.

All components rely to a greater or lesser extent on supporting classes that reside in a top level package called “core.” This package provides classes and data structures that read data sets, represent instances and attributes, and provide various common utility methods. The core package also contains additional interfaces that components may implement in order to indicate that they support various extra functionality. For example, a classifier can implement the “WeightedInstancesHandler” interface to indicate that it can take advantage of instance weights.

A major part of the appeal of the system for end users lies in its graphical user interfaces. In order to maintain flexibility it was necessary to engineer the interfaces to make it as painless as possible for developers to add new components into the workbench. To this end, the user interfaces capitalize upon Java’s introspection mechanisms to provide the ability to configure each component’s options dynamically at runtime. This frees the developer from having to consider user interface issues

when developing a new component. For example, to enable a new classifier to be used with the Explorer (or either of the other two graphical user interfaces), all a developer need do is follow the Java Bean convention of supplying “get” and “set” methods for each of the classifier’s public options.

Applications

Weka was originally developed for the purpose of processing agricultural data, motivated by the importance of this application area in New Zealand. However, the machine learning methods and data engineering capability it embodies have grown so quickly, and so radically, that the workbench is now commonly used in all forms of data mining applications—from bioinformatics to competition datasets issued by major conferences such as *Knowledge Discovery in Databases*.

New Zealand has several research centres dedicated to agriculture and horticulture, which provided the original impetus for our work, and many of our early applications. For example, we worked on predicting the internal bruising sustained by different varieties of apple as they make their way through a packing-house on a conveyor belt (Holmes et al., 1998); predicting, in real time, the quality of a mushroom from a photograph in order to provide automatic grading (Kusabs et al., 1998); and classifying kiwifruit vines into twelve classes, based on visible-NIR spectra, in order to determine which of twelve pre-harvest fruit management treatments has been applied to the vines (Holmes and Hall, 2002). The applicability of the workbench in agricultural domains was the subject of user studies (McQueen et al., 1998) that demonstrated a high level of satisfaction with the tool and gave some advice on improvements.

There are countless other applications, actual and potential. As just one example, Weka has been used extensively in the field of bioinformatics. Published studies include automated protein annotation (Bazzan et al., 2002), probe selection for gene expression arrays (Tobler et al., 2002), plant genotype discrimination (Taylor et al., 2002), and classifying gene expression profiles and extracting rules from them (Li et al., 2003). Text mining is another major field of application, and the workbench has been used to automatically extract key phrases from text (Frank et al., 1999), and for document categorization (Sauban and Pfahringer, 2003) and word sense disambiguation (Pedersen, 2002).

The workbench makes it very easy to perform interactive experiments, so it is not surprising that most work has been done with small to medium sized datasets. However, larger datasets have been successfully processed. Very large datasets are typically split into several training

sets, and a voting-committee structure is used for prediction. The recent development of the knowledge flow interface should see larger scale application development, including online learning from streamed data.

Many future applications will be developed in an online setting. Recent work on data streams (Holmes et al., 2003) has enabled machine learning algorithms to be used in situations where a potentially infinite source of data is available. These are common in manufacturing industries with 24/7 processing. The challenge is to develop models that constantly monitor data in order to detect changes from the steady state. Such changes may indicate failure in the process, providing operators with warning signals that equipment needs re-calibrating or replacing.

Summing up the workbench

Weka has three principal advantages over most other data mining software. First, it is open source, which not only means that it can be obtained free, but—more importantly—it is maintainable, and modifiable, without depending on the commitment, health, or longevity of any particular institution or company. Second, it provides a wealth of state-of-the-art machine learning algorithms that can be deployed on any given problem. Third, it is fully implemented in Java and runs on almost any platform—even a Personal Digital Assistant.

The main disadvantage is that most of the functionality is only applicable if all data is held in main memory. A few algorithms are included that are able to process data incrementally or in batches (Frank et al., 2002). However, for most of the methods the amount of available memory imposes a limit on the data size, which restricts application to small or medium-sized datasets. If larger datasets are to be processed, some form of subsampling is generally required. A second disadvantage is the flip side of portability: a Java implementation is generally somewhat slower than an equivalent in C/C++.

Acknowledgments

Many thanks to past and present members of the Waikato machine learning group and the many external contributors for all the work they have put into Weka.

References

- Bazzan, A. L., Engel, P. M., Schroeder, L. F., and da Silva, S. C. (2002). Automated annotation of keywords for proteins related to mycoplasma-tataceae using machine learning techniques. *Bioinformatics*, 18:35S–43S.

- Frank, E., Holmes, G., Kirkby, R., and Hall, M. (2002). Racing committees for large datasets. In *Proceedings of the International Conference on Discovery Science*, pages 153–164. Springer-Verlag.
- Frank, E., Paynter, G. W., Witten, I. H., Gutwin, C., and Nevill-Manning, C. G. (1999). Domain-specific keyphrase extraction. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 668–673. Morgan Kaufmann.
- Holmes, G., Cunningham, S. J., Rue, B. D., and Bollen, F. (1998). Predicting apple bruising using machine learning. *Acta Hort*, 476:289–296.
- Holmes, G. and Hall, M. (2002). A development environment for predictive modelling in foods. *International Journal of Food Microbiology*, 73:351–362.
- Holmes, G., Kirkby, R., and Pfahringer, B. (2003). Mining data streams using option trees. Technical Report 08/03, Department of Computer Science, University of Waikato.
- Kusabs, N., Bollen, F., Trigg, L., Holmes, G., and Inglis, S. (1998). Objective measurement of mushroom quality. In *Proc New Zealand Institute of Agricultural Science and the New Zealand Society for Horticultural Science Annual Convention*, page 51.
- Li, J., Liu, H., Downing, J. R., Yeoh, A. E.-J., and Wong, L. (2003). Simple rules underlying gene expression profiles of more than six subtypes of acute lymphoblastic leukemia (all) patients. *Bioinformatics*, 19:71–78.
- McQueen, R., Holmes, G., and Hunt, L. (1998). User satisfaction with machine learning as a data analysis method in agricultural research. *New Zealand Journal of Agricultural Research*, 41(4):577–584.
- Pedersen, T. (2002). Evaluating the effectiveness of ensembles of decision trees in disambiguating Senseval lexical samples. In *Proceedings of the ACL-02 Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*.
- Sauban, M. and Pfahringer, B. (2003). Text categorisation using document profiling. In *Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 411–422. Springer.
- Taylor, J., King, R. D., Altmann, T., and Fiehn, O. (2002). Application of metabolomics to plant genotype discrimination using statistics and machine learning. *Bioinformatics*, 18:241S–248S.
- Tobler, J. B., Molla, M., Nuwaysir, E., Green, R., and Shavlik, J. (2002). Evaluating machine learning approaches for aiding probe selection for gene-expression arrays. *Bioinformatics*, 18:164S–171S.