Laboratory Module 1
Description of WEKA (Java-implemented machine learning tool)

Purpose:

- Install and run WEKA
- Experiment environment in GUI version and in command line version

1. Theoretical Aspects

1.1. What is WEKA?

Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

Weka is used for research, education, and applications. The tool gathers a comprehensive set of data pre-processing tools, learning algorithms and evaluation methods, graphical user interfaces (incl. data visualization) and environment for comparing learning algorithms.

Weka is open source software issued under the GNU General Public License.

"WEKA" stands for the Waikato Environment for Knowledge Analysis, which was developed at the University of Waikato in New Zealand. WEKA is extensible and has become a collection of machine learning algorithms for solving real-world data mining problems. It is written in Java and runs on almost every platform.

WEKA is easy to use and to be applied at several different levels. You can access the WEKA class library from your own Java program, and implement new machine learning algorithms.

There are three major implemented schemes in WEKA. (1) Implemented schemes for classification. (2) Implemented schemes for numeric prediction. (3) Implemented "meta-schemes".

Besides actual learning schemes, WEKA also contains a large variety of tools that can be used for pre-processing datasets, so that you can focus on your algorithm without considering too much details as reading the data from files, implementing filtering algorithm and providing code to evaluate the results.

Some practical applications that use Weka:

Acronym identification

This addresses the task of finding acronym-definition pairs in text. Most of the previous work on the topic is about systems that involve manually generated rules or regular expressions. In this paper, we present a supervised learning approach to the acronym identification task. Our approach reduces the search space of the supervised learning system by putting some weak constraints on the kinds of acronym-definition pairs that can be identified. We obtain results comparable to hand-crafted systems that use stronger constraints. We describe our method for reducing the search space, the features used by our supervised learning system, and our experiments with various learning schemes.

Gene selection from microarray data for cancer classification

A DNA microarray can track the expression levels of thousands of genes simultaneously. Previous research has demonstrated that this technology can be useful in the classification of cancers. Cancer microarray data normally contains a small number of samples which have a large number of gene expression levels as features. To select relevant genes involved in different types of cancer remains a challenge. In order to extract useful gene information from cancer microarray data and reduce dimensionality, feature selection algorithms were systematically investigated in this study.

Using a correlation-based feature selector combined with machine learning algorithms such as decision trees, nave Bayes and support vector machines, we show that classification performance at least as good as published results can be obtained on acute leukemia and diffuse large B-cell lymphoma microarray data sets. We also demonstrate that a combined use of different classification and feature selection approaches makes it possible to select relevant genes with high confidence. This is also the first paper which discusses both computational and biological evidence for the involvement of zyxin in leukaemogenesis.

Benchmarking of Linear and Nonlinear Approaches for Quantitative Structure–Property Relationship Studies of Metal Complexation with Ionophores

A benchmark of several popular methods, Associative Neural Networks (ANN), Support Vector Machines (SVM), k Nearest Neighbors (kNN), Maximal Margin Linear Programming (MMLP), Radial Basis Function Neural Network (RBFNN), and Multiple Linear Regression (MLR), is reported for quantitative−structure property relationships (QSPR) of stability constants logK1 for the 1:1 (M:L) and logβ2 for 1:2 complexes of metal cations Ag+ and Eu3+ with diverse sets of organic molecules in water at 298 K and ionic strength 0.1 M. The methods were tested on three types of descriptors: molecular descriptors including E-state values, counts of atoms determined for E-state atom types, and substructural molecular fragments (SMF).

1.2.Installing and running WEKA

1.2.1. In lab (this assumes WEKA is already installed)

1.2.2. On your home computer

For installing WEKA on your home computer you must check the following:

There are two stable versions of WEKA. Either you can download the self-extraction executable version that includes the Java Virtual Machine 1.4 (weka-3-4jre.exe; 19,543,851 bytes), http://prdownloads.sourceforge.net/weka/weka-3-4jre.exe
or the self-extracting executable without Java VM (weka-3-4.exe; 6,467,165 bytes).
http://prdownloads.sourceforge.net/weka/weka-3-4.exe

This version comes with the GUI, which provides the user with more flexibility than the command line.

After extracting the files, you will need to set your *classpath* variable to a complete path to weka.jar (suppose you extracted WEKA to C:\Weka, then set your *classpath* variable to

C:\Weka\weka.jar, ie add "C:\Weka\weka.jar;" to the list of values that environment variable Path can take when working in Windows)

If you don't have administrator privileges, you can still install WEKA. For that, download the jar archive (weka-3-4.jar; 6,322,417 bytes).

http://prdownloads.sourceforge.net/weka/weka-3-4.jar

Make sure that the Java J2SE 1.4 (download from SUN) is installed on your system (which includes the jar utility). Then open a command line console, change into the directory containing weka-3-4.jar, and enter

jar -xvf weka-3-4.jar

This will create a new directory called weka-3-4. To un-jar (install) the source code, position yourself in the recently created weka-3-4 directory and type

jar -xvf weka-src.jar

Which will create a new directory weka containing the source code. Since WEKA is open source software issued under the GNU General Public License, you can use and modify the source code as you like.

NOTE: It seems that Windows will not set up your CLASSPATH properly if any of the WEKA directories contains spaces. Therefore, installing Weka in the Program Files folder is not a good idea.

1.3.Online documentation and further help

From your weka-3-4 directory, you will find:

* A jarfile containing the classes only
* A jarfile containing the complete source code
* The tutorial for the experiment environment in the GUI version of Weka (written by David Scuse), and the README file
* The API documentation
* Some example datasets

The most detailed and up-to-date information could be found in the online documentation on WEKA Web Site . This page has a lot of documentation and guides on installation/usage pages.

http://www.cs.waikato.ac.nz/~ml/weka/index_documentation.html

1.4. Launching WEKA

The Weka GUI Chooser (class weka.gui.GUIChooser) provides a starting point for launching Weka's main GUI applications and supporting tools. If one prefers a MDI ("multiple document interface") appearance, then this is provided by an alternative launcher called "Main" (class weka.gui.Main).

The GUI Chooser consists of four buttons—one for each of the four major Weka applications—and four menus.

The buttons can be used to start the following applications:

• **Explorer** An environment for exploring data with WEKA (the rest of this documentation deals with this application in more detail).

• **Experimenter** An environment for performing experiments and conducting statistical tests between learning schemes.

• **KnowledgeFlow** This environment supports essentially the same functions as the Explorer but with a drag-and-drop interface. One advantage is that it supports incremental learning.

• **SimpleCLI** Provides a simple command-line interface that allows direct execution of WEKA commands for operating systems that do not provide their own command line interface.

The menu consists of four sections:

1. **Program**
   - **LogWindow** Opens a log window that captures all that is printed to stdout or stderr. Useful for environments like MS Windows, where WEKA is normally not started from a terminal.
   - **Exit** Closes WEKA.
2. **Tools -** Other useful applications.
   - **ArffViewer** An MDI application for viewing ARFF files in spread-sheet format.
   - **SqlViewer** Represents an SQL worksheet, for querying databases via JDBC.
   - **Bayes net editor** An application for editing, visualizing and learning Bayes nets.
3. **Visualization** - Ways of visualizing data with WEKA.
   - **Plot** For plotting a 2D plot of a dataset.
   - **ROC** Displays a previously saved ROC curve.
   - **TreeVisualizer** For displaying directed graphs, e.g., a decision tree.
   - **GraphVisualizer** Visualizes XML BIF or DOT format graphs, e.g., for Bayesian networks.
   - **BoundaryVisualizer** Allows the visualization of classifier decision boundaries in two dimensions.
4. **Help -** Online resources for WEKA can be found here.
   - **Weka homepage** Opens a browser window with WEKA's home page.
   - **HOWTOs**, **code snippets**, **etc**. The general WekaWiki [2], containing lots of examples and HOWTOs around the development and use of WEKA.
   - **Weka on Sourceforge** WEKA's project homepage on Sourceforge.net.
   - **SystemInfo** Lists some internals about the Java/WEKA environment, e.g., the CLASSPATH.

1.5. Simple CLI

The Simple CLI provides full access to all Weka classes, i.e., classifiers, filters, clusterers, etc., but without the hassle of the CLASSPATH (it facilitates the one, with which Weka was started). It offers a simple Weka shell with separated command line and output.

1.5.1. Commands

The following commands are available in the Simple CLI:

- **java <classname> [<args>]**
  invokes a java class with the given arguments (if any)
- **break**
  stops the current thread, e.g., a running classifier, in a friendly manner
- **kill**
  stops the current thread in an unfriendly fashion
- **cls**
  clears the output area
- **exit**
  exits the Simple CLI
- **help [<command>]**
  provides an overview of the available commands if without a command name as argument, otherwise more help on the specified command

1.5.2. Invocation

In order to invoke a Weka class, one has only to prefix the class with ”java”. This command tells the Simple CLI to load a class and execute it with any given parameters. E.g., the J48 classifier can be invoked on the iris dataset with the following command:

java weka.classifiers.trees.J48 -t c:/temp/iris.arff

1.5.3. Command Redirection

Starting with this version of Weka one can perform a basic redirection:

java weka.classifiers.trees.J48 -t test.arff > j48.txt

Note: the > must be preceded and followed by a space, otherwise it is not recognized as redirection, but part of another parameter.

1.5.4. Command completion

Commands starting with java support completion for classnames and filenames via Tab (Alt+BackSpace deletes parts of the command again). In case that there are several matches, Weka lists all possible matches.

- **package name completion**

  java weka.cl<Tab>

  results in the following output of possible matches of package names:

  Possible matches:
  weka.classifiers
  weka.clusterers

- **classname completion**

java weka.classifiers.meta.A<Tab>

lists the following classes

Possible matches:
weka.classifiers.meta.AdaBoostM1
weka.classifiers.meta.AdditiveRegression
weka.classifiers.meta.AttributeSelectedClassifier

- **filename completion**

In order for Weka to determine whether a the string under the cursor is a classname or a filename, filenames need to be absolute (Unix/Linx: /some/path/file;Windows: C:\Some\Path\file) or relative and starting with a dot (Unix/Linux: ./some/other/path/file;Windows: .\Some\Other\Path\file).

1.5.5. The WEKA Explorer

1.5.5.1. Section Tabs

At the very top of the window, just below the title bar, is a row of tabs. When the Explorer is first started only the first tab is active; the others are greyed out. This is because it is necessary to open (and potentially pre-process) a data set before starting to explore the data.

The tabs are as follows:

1. **Preprocess.** Choose and modify the data being acted on.
2. **Classify.** Train and test learning schemes that classify or perform regression.
3. **Cluster.** Learn clusters for the data.
4. **Associate.** Learn association rules for the data.
5. **Select attributes.** Select the most relevant attributes in the data.
6. **Visualize.** View an interactive 2D plot of the data.

Once the tabs are active, clicking on them flicks between different screens, on which the respective actions can be performed. The bottom area of the window (including the status box, the log button, and the Weka bird) stays visible regardless of which section you are in.

1.5.5.2. Preprocessing

1.5.5.2.1. Opening Files

The first three buttons at the top of the preprocess section enable you to load data into WEKA:

1. Open file.... Brings up a dialog box allowing you to browse for the data file on the local filesystem.
2. Open URL.... Asks for a Uniform Resource Locator address for where the data is stored.
3. Open DB.... Reads data from a database.

1.5.5.2.2. The Current Relation

Once some data has been loaded, the Preprocess panel shows a variety of information. The Current relation box (the "current relation" is the currently loaded data, which can be interpreted as a single relational table in database terminology) has three entries:

1. Relation. The name of the relation, as given in the file it was loaded from. Filters (described below) modify the name of a relation.
2. Instances. The number of instances (data points/records) in the data.
3. Attributes. The number of attributes (features) in the data.

1.5.5.2.3. Working With Attributes

Below the Current relation box is a box titled Attributes. There are three buttons, and beneath them is a list of the attributes in the current relation.

The list has three columns:

1. No.. A number that identifies the attribute in the order they are specified in the data file.
2. Selection tick boxes. These allow you select which attributes are present in the relation.
3. Name. The name of the attribute, as it was declared in the data file.

When you click on different rows in the list of attributes, the fields change in the box to the right titled Selected attribute. This box displays the characteristics of the currently highlighted attribute in the list:

1. Name. The name of the attribute, the same as that given in the attribute list.
2. Type. The type of attribute, most commonly Nominal or Numeric.
3. Missing. The number (and percentage) of instances in the data for which this attribute is missing (unspecified).
4. Distinct. The number of different values that the data contains for this attribute.
5. Unique. The number (and percentage) of instances in the data having a value for this attribute that no other instances have.

Below these statistics is a list showing more information about the values stored in this attribute, which differ depending on its type. If the attribute is **nominal**, the list consists of each possible value for the attribute along with the number of instances that have that value. If the

attribute is numeric, the list gives four statistics describing the distribution of values in the data—the minimum, maximum, mean and standard deviation. And below these statistics there is a **colored histogram**, color-coded according to the attribute chosen as the **Class** using the box above the histogram. (This box will bring up a drop-down list of available selections when clicked.) Note that only nominal **Class** attributes will result in a color-coding. Finally, after pressing the **Visualize All** button, histograms for all the attributes in the data are shown in a separate witting.

Returning to the attribute list, to begin with all the tick boxes are **unticked**. They can be toggled **on/off** by clicking on them individually. The three buttons above can also be used to change the selection:

1. **All.** All boxes are ticked.
2. **None.** All boxes are cleared (unticked).
3. **Invert.** Boxes that are ticked become unticked and vice versa.

Once the desired attributes have been selected, they can be removed by clicking the **Remove** button below the list of attributes. Note that this can be undone by clicking the **Undo** button, which is located next to the **Edit** button in the top-right corner of the Preprocess panel.

1.5.5.2.4. Working With Filters

The preprocess section allows filters to be defined that transform the data in various ways. The **Filter** box is used to set up the filters that are required.
At the left of the **Filter** box is a **Choose** button. By clicking this button it is possible to select one of the filters in Weka. Once a filter has been selected, its name and options are shown in the field next to the **Choose** button. Clicking on this box brings up a **GenericObjectEditor** dialog box.

The GenericObjectEditor Dialog Box

The **GenericObjectEditor** dialog box lets you configure a filter. The same kind of dialog box is used to configure other objects, such as classifiers and clusterers (see below). The fields in the window reflect the available options.

Clicking on any of these gives an opportunity to alter **the filters settings**.
For example, the setting may take a text string, in which case you type the string into the text field provided. Or it may give a drop-down box listing several states to choose from. Or it may do something else, depending on the information required.

Information on the options is provided in a tool tip if you let the mouse pointer hover of the corresponding field. More information on the filter and its options can be obtained by clicking on the **More** button in the **About** panel at the top of the GenericObjectEditor window.

Some objects display a brief description of what they do in an **About** box, along with a **More** button. Clicking on the **More** button brings up a window describing what the different options do.

At the bottom of the **GenericObjectEditor** dialog are four buttons. The first two, **Open...** and **Save...** allow object configurations to be stored for future use. The **Cancel** button backs out without remembering any changes that have been made. Once you are happy with the object and settings you have chosen, click **OK** to return to the main Explorer window.

Applying Filters

Once you have selected and configured a filter, you can apply it to the data by pressing the **Apply** button at the right end of the **Filter** panel in the **Preprocess** panel. The **Preprocess** panel will then show the transformed data. The change can be undone by pressing the **Undo** button. You can also use the **Edit...** button to modify your data manually in a dataset editor.

Finally, the **Save...** button at the top right of the **Preprocess** panel saves the current version of the relation in the same formats available for loading data, allowing it to be kept for future use.

Note: Some of the filters behave differently depending on whether a class attribute has been set or not (using the box above the histogram, which willbring up a drop-down list of possible selections when clicked). In particular, the "supervised filters" require a class attribute to be set, and some of the "unsupervised attribute filters" will skip the class attribute if one is set. Note that it is also possible to set **Class** to **None**, in which case no class is set.

1.5.5.3. Classification

1.5.5.3.1. Selecting a Classifier

At the top of the classify section is the **Classifier** box. This box has a text field that gives the name of the currently selected classifier, and its options. Clicking on the text box brings up a **GenericObjectEditor** dialog box, just the same as for filters, that you can use to configure the options of the current classifier. The **Choose** button allows you to choose one of the classifiers that are available in **WEKA**.

1.5.5.3.2. Test Options

The result of applying the chosen classifier will be tested according to the options that are set by clicking in the Test options box. There are four test modes:

1. Use training set. The classifier is evaluated on how well it predicts the class of the instances it was trained on.
2. Supplied test set. The classifier is evaluated on how well it predicts the class of a set of instances loaded from a file. Clicking the Set... button brings up a dialog allowing you to choose the file to test on.
3. Cross-validation. The classifier is evaluated by cross-validation, using the number of folds that are entered in the Folds text field.

4. Percentage split. The classifier is evaluated on how well it predicts a certain percentage of the data which is held out for testing. The amount of data held out depends on the value entered in the % field.

1.5.5.3.3. The Class Attribute

The classifiers in WEKA are designed to be trained to predict a single 'class' attribute, which is the target for prediction. Some classifiers can only learn nominal classes; others can only learn numeric classes (regression problems); still others can learn both.
By default, the class is taken to be the last attribute in the data. If you want to train a classifier to predict a different attribute, click on the box below the **Test options** box to bring up a drop-down list of attributes to choose from.

1.5.5.3.4. Training a Classifier

Once the classifier, test options and class have all been set, the learning process is started by clicking on the **Start** button. While the classifier is busy being trained, the little bird moves around. You can stop the training process at any time by clicking on the **Stop** button.

When training is complete, several things happen. The **Classifier output** area to the right of the display is filled with text describing the results of training and testing.

A new entry appears in the **Result list** box. We look at the result list below; but first we investigate the text that has been output.

1.5.5.3.5. The Classifier Output Text

The text in the Classifier output area has scroll bars allowing you to browse the results. Of course, you can also resize the Explorer window to get a larger display area. The output is split into several sections:

1. Run information. A list of information giving the learning scheme options, relation name, instances, attributes and test mode that were involved in the process.
2. Classifier model (full training set). A textual representation of the classification model that was produced on the full training data.
3. The results of the chosen test mode are broken down thus:
4. Summary. A list of statistics summarizing how accurately the classifier was able to predict the true class of the instances under the chosen test mode.
5. Detailed Accuracy By Class. A more detailed per-class break down of the classifier's prediction accuracy.
6. Confusion Matrix. Shows how many instances have been assigned to each class. Elements show the number of test examples whose actual class is the row and whose predicted class is the column.

1.5.5.3.6. The Result List

After training several classifiers, the result list will contain several entries. Left-clicking the entries flicks back and forth between the various results that have been generated.

Right-clicking an entry invokes a menu containing these items:

1. View in main window. Shows the output in the main window (just like left-clicking the entry).
2. View in separate window. Opens a new independent window for viewing the results.
3. Save result buffer. Brings up a dialog allowing you to save a text file containing the textual output.
4. Load model. Loads a pre-trained model object from a binary file.
5. Save model. Saves a model object to a binary file. Objects are saved in Java 'serialized object' form.
6. Re-evaluate model on current test set. Takes the model that has been built and tests its performance on the data set that has been specified with the Set.. button under the Supplied test set option.
7. Visualize classifier errors. Brings up a visualization window that plots the results of classification. Correctly classified instances are represented by crosses, whereas incorrectly classified ones show up as squares.
8. Visualize tree or Visualize graph. Brings up a graphical representation of the structure of the classifier model, if possible (i.e. for decision trees or Bayesian networks). The graph visualization option only appears if a Bayesian network classifier has been built. In the tree visualizer, you can bring up a menu by right-clicking a blank area, pan around by dragging the mouse, and see the training instances at each node by clicking on it. CTRL-clicking zooms the view out, while SHIFT-dragging a box zooms the view in. The graph visualizer should be self-explanatory.
9. Visualize margin curve. Generates a plot illustrating the prediction margin. The margin is defined as the difference between the probability predicted for the actual class and the highest probability predicted for the other classes. For example, boosting algorithms may achieve better performance on test data by increasing the margins on the training data.
10. Visualize threshold curve. Generates a plot illustrating the tradeoffs in prediction that are obtained by varying the threshold value between classes. For example, with the default threshold value of 0.5, the predicted probability of 'positive' must be greater than 0.5 for the instance to be predicted as 'positive'. The plot can be used to visualize the precision/recall tradeoff, for ROC curve analysis (true positive rate vs false positive rate), and for other types of curves.
11. Visualize cost curve. Generates a plot that gives an explicit representation of the expected cost, as described by Drummond and Holte (2000).

Options are greyed out if they do not apply to the specific set of results.

1.5.5.4. Clustering

1.5.5.4.1. Selecting a Clusterer

By now you will be familiar with the process of selecting and configuring objects. Clicking on the clustering scheme listed in the **Clusterer** box at the top of the window brings up a **GenericObjectEditor** dialog with which to choose a new clustering scheme.

1.5.5.4.2. Cluster Modes

The **Cluster mode** box is used to choose what to cluster and how to evaluate the results. The first three options are the same as for classification: **Use training set**, **Supplied test set** and **Percentage split** — the data is assigned to clusters instead of trying to predict a specific class. The fourth mode, **Classes to clusters evaluation**, compares how well the chosen clusters match up with a pre-assigned class in the data. The drop-down box below this option selects the class, just as in the **Classify** panel.

An additional option in the **Cluster mode** box, the **Store clusters for visualization** tick box, determines whether or not it will be possible to visualize the clusters once training is complete. When dealing with datasets that are so large that memory becomes a problem it may be helpful to disable this option.

1.5.5.4.3. Ignoring Attributes

Often, some attributes in the data should be ignored when clustering. The **Ignore attributes** button brings up a small window that allows you to select which attributes are ignored. Clicking on an attribute in the window highlights it, holding down the **SHIFT** key selects a range of consecutive attributes, and holding down **CTRL** toggles individual attributes on and off. To cancel the selection, back out with the **Cancel** button. To activate it, click the Select button. The next time clustering is invoked, the selected attributes are ignored.

1.5.5.4.4. Learning Clusters

The **Cluster** section, like the **Classify** section, has **Start/Stop** buttons, a result text area and a result list. These all behave just like their classification counterparts. Right-clicking an entry in the result list brings up a similar menu, except that it shows only two visualization options: **Visualize cluster assignments** and **Visualize tree**. The latter is grayed out when it is not applicable.

1.5.5.5. Associating

1.5.5.5.1. Setting Up

This panel contains schemes for learning association rules, and the learners are chosen and configured in the same way as the clusterers, filters, and classifiers in the other panels.

1.5.5.5.2. Learning Associations

Once appropriate parameters for the association rule learner bave been set, click the **Start** button. When complete, right-clicking on an entry in the result list allows the results to be viewed or saved.

1.5.5.6. Selecting Attributes

1.5.5.6.1. Searching and Evaluating

Attribute selection involves searching through all possible combinations of attributes in the data to find which subset of attributes works best for prediction. To do this, two objects must be set up: an attribute evaluator and a search method. The evaluator determines what method is used to assign a worth to each subset of attributes. The search method determines what style of search is performed.

The Attribute Selection Mode box has two options:

1. Use full training set. The worth of the attribute subset is determined using the full set of training data.

2. Cross-validation. The worth of the attribute subset is determined by a process of cross-validation. The Fold and Seed fields set the number of folds to use and the random seed used when shuffling the data.There is a drop-down box that can be used to specify which attribute to treat as the class.

1.5.5.6.2. Performing Selection

Clicking **Start** starts running the attribute selection process. When it is finished, the results are output into the result area, and an entry is added to the result list. Right-clicking on the result list gives several options. The first three, (**View in main window**, **View in separate window** and **Save result buffer**), are the same as for the classify panel. It is also possible to **Visualize reduced data**, or if you have used an attribute transformer such as Principal-Components, **Visualize transformed data**.

1.5.5.7. Visualizing

WEKA's visualization section allows you to visualize 2D plots of the current relation.

1.5.5.7.1. The scatter plot matrix

When you select the Visualize panel, it shows a scatter plot matrix for all the attributes, color coded according to the currently selected class. It is possible to change the size of each individual 2D plot and the point size, and to randomly jitter the data (to uncover obscured points). It also possible to change the attribute used to color the plots, to select only a subset of attributes for inclusion in the scatter plot matrix, and to sub sample the data. Note that changes will only come into effect once the Update button has been pressed.

1.5.5.7.2. Selecting an individual 2D scatter plot

When you click on a cell in the scatter plot matrix, this will bring up a separate window with a visualization of the scatter plot you selected.

1.5.5.7.3. Selecting Instances

A group of data points can be selected in four ways:

1. Select Instance. Clicking on an individual data point brings up a window listing its attributes. If more than one point appears at the same location, more than one set of attributes is shown.
2. Rectangle. You can create a rectangle, by dragging, that selects the points inside it.
3. Polygon. You can build a free-form polygon that selects the points inside it. Left-click to add vertices to the polygon, right-click to complete it. The polygon will always be closed off by connecting the first point to the last.
4. Polyline. You can build a polyline that distinguishes the points on one side from those on the other. Left-click to add vertices to the polyline, right-click to finish. The resulting shape is open (as opposed to a polygon, which is always closed).

Once an area of the plot has been selected using Rectangle, Polygon or Polyline, it turns grey. At this point, clicking the Submit button removes all instances from the plot except those within the grey selection area. Clicking on the Clear button erases the selected area without affecting the graph.
Once any points have been removed from the graph, the Submit button changes to a Reset button. This button undoes all previous removals and returns you to the original graph with all points included. Finally, clicking the Save button allows you to save the currently visible instances to a new ARFF file.

2. Examples

2.1. Practice WEKA with the classification example about Play Golf

Data format: the Datasets for WEKA are formatted according to the *arff* format. For this example you will use the file weather.nominal.arff as a training file to construct a classification model. Save the file in your workspace for example (C:\WEKA_Tutorial), and open it in a text processor to see an example of the arff format; note that the last attribute corresponds to the class.

Run WEKA in the Windows environment:
Find the WEKA directory in your machine (C:\Program Files\Weka-3-4). Double click in the file"weka.jar"; Select the option "Simple CLI". Now you are ready to run WEKA using some commands in this window.

Probe the example with different classifiers, and compare the results obtained with each of the classifiers for example in terms of and number of examples correctly and incorrectly classified:

Decision Trees: In order to probe decision tree you will use the Id3 classifier. Type the following command:

java weka.classifiers.trees.Id3 -t PATH/weather.nominal.arff

(note that the option -t calls the training file according the PATH location of this file in your machine)

Support Vector Machines: In order to probe the SVM classifier, type the following command

java weka.classifiers.functions.SMO -t PATH/weather.nominal.arff

Neural Networks: In order to probe the NNs classifier, type the following command

java weka.classifiers.functions.VotedPerceptron -t PATH/weather.nominal.arff

Naive Bayes: In order to probe the NB classifier, type the following command

java weka.classifiers.bayes.NaiveBayes -t PATH/weather.nominal.arff

Save the classification model and then use it to classify new examples: You can save the classification model generated by each one of the above classifiers by using the option -d in the following way:
java weka.classifiers.TYPE.CLASSIFIER_NAME -t PATH/weather.nominal.arff -d PATH/modelname.model

You should generate a file that contains the model; this can be named for example in the form: weather_Id3.model
weather_SVM.model
weather_NN.model
weather_NB.model

e.g. by

java weka.classifiers.trees.Id3 -t PATH/weather.nominal.arff -d PATH/weather_Id3.model

In order to use the stored model to classify new examples, use the file "test_weather.arff" (save this file in the same folder than weather.nominal.arff and *.model files). In this file you have two examples without classification. Then classify these examples using the models previously generated in the following way:

java weka.classifiers.~.classifier_name -T PATH/test_weather.arff -l PATH/modelname.model -p 0

In this case you use the options: -T that calls a test file (test_weather.arff); and -l that call the model file to be used. Compare the results obtained using the four models generated.

2.2. Classification of breast cancer examples

Download the file Breast_Cancer.arff that include a set of 699 cases, 9 attributes and the class attribute related to the type of cancer cell (in this dataset class 4 is equivalent to malignant cells and class 2 is equivalent to benign cells). This dataset is from the Wisconsin Breast Cancer Database (January 8, 1991). You can look for this and others examples of dataset in this link

Classify the examples in the "Breast_Cancer.arff" dataset (benign and malignant cells) using the four classifiers mentioned in the exercise 1, and compare the results.

NOTE: This dataset contains numerical data, so you you can not use Id3 classifier (Id3 only support nominal attributes). In this case try decision trees with J48 classifier with the following command

java weka.classifiers.trees.J48 -t PATH/Breast_Cancer.arff

2.3. Classification of Gene expression data

Download the file ALLAML.arff ((Golub et al 1999)) gene expression data that include 72 examples, 7129 genes (attributes) and 2 clases "acute myeloid leukemia (AML)" and "acute lymphoblastic leukemia (ALL)". For more information you can read the gene list in the file ALLAML.gene_names.txt, and in the paper Golub et al 1999

Classify the examples in this dataset (ALL or AML class) using the four classifiers mentioned in the exercise 1, and compare the results.

Interpretation: Go to PubMed and search the selected genes, do they have any biological meaning? Can you identify the unknown gene function? (Try using other bioinformatics tools)

3. Assignments

3.1 Become familiar with the *vowel* data set and use it to perform the following experiments:

1. Remove the first three attributes as well as the class attribute.
2. Cluster the data using the simple k-Means algorithm, with values of k from 1 to 12. What do you see?
3. Now add the class attribute back in and repeat the clusterings, comparing the clusterings with the class. How well does the clustering appear to correlate with class? What might this mean?
4. Choose several different classifiers and use them to classify the data. How does their performance compare with the clustering's "performance"? Is this something you might expect?
5. Does adding back in any of the original first three attributes have any effect on either the clustering or the classification performance?