AutoUniv Version 1 User Guide

1 About AutoUniv

AutoUniv (AU) is a tool for creating classification models and generating classified examples.

AU was designed by

Ray J. Hickey

E: ray.j.hickey@gmail.com

W: http://sites.google.com/site/autouniv

AU was programmed in Win-Prolog, version 4.9 from Logic Programming Associates and makes use of the Prolog compiler and reasoning engine.

2 Installation

AU runs under Windows. Around 2GB of memory will be sufficient. Simply extract AU.zip. AU does not alter registry settings.

All files are in the top-level folder. AU consists of four files: *au.exe*, *au.dll*, *au.ini*, *au.ovl*. Two sub-folders exist containing examples of models and of data sets. In addition to this *User Guide* there is also a *Reference Manual*.

Memory settings are in *au.ini*. These have been set to cater for the largest models and should not need adjusting.

3 Running AU

Run *au.exe*. This will start the *Launcher*. From here you can run the two applications namely the model builder, *Create Model*, and the data generator, *Generate Data*.

The following sections provide a brief description of the use of the model builder and data generator. These should be read in conjunction with the more detailed account of AU in the *Reference Manual*.

4 The model builder tool

The *Build Classification Model* dialog is available from the *Launcher*. Settings are made for attributes, classes and rules. The order in which settings are entered is controlled. For example, the number of relevant attributes cannot be specified until the number of attributes overall is specified. Validity checks operate to prevent illegitimate entries. There is also auto completion.

The *Class Distribution Settings* dialog is available from the *Customise Distribution* button or these settings can be randomised from the *Randomise Distributions* button. A total weight of 1000 must be distributed across the classes to reflect the prominence of each class. Applying *Hold* to any weight will cause it to be retained when randomisation is applied. Minimum and maximum values for the majority class probability are set for each class. These are expressed as integers up to 1000 subject to a minimum necessary to secure a majority class (which is displayed). Randomisation is also available, with hold. Selecting *Zero Noise* followed by randomisation will set minimum and maximum to 1000 for unheld classes. Illegitimate values will be notified to the user when the *Apply* button is pressed.

When all build settings have been made the *Apply* button in the *Build Classification Model* dialog is enabled. When activated, this will enable the *Start* button for model building. Progress in the building of the model is echoed in the *Build Log* window. After completion or failure, the model may be rebuilt with or without attribute definitions and attribute factorisation being retained.

Once built, a model may be saved which creates its *.aupl*, *.aurules* and *.auprops* files. The *.aupl* file provides the Prolog source code definition of the model.

A new model may now be built or drift may be applied to the existing model by selecting *Keep attribute definitions* and, possibly, other keep options depending on the type of drift required.

When the user returns to the *Launcher*, the current build settings are retained.

5 The data generator tool

The *Generate Data Set from Model* dialog is available from the *Launcher*. Here a model is selected and will be compiled into the Prolog system. A file format for the examples is selected. The three available formats are CSV, ARFF (for WEKA) and C4.5. Examples may be generated with or without header attribute information. The latter is useful for appending examples to existing files.

6 Modification of the *.aupl* file

The user can edit the *.aupl* file for a model in order to manually alter the model definition. Caution should be exercised when doing this. It is the user's responsibility to ensure the logical integrity and syntactic correctness of any alterations. Logical flaws will cause the data generator to crash. In this situation, the system should be exited and restarted. Syntax errors will produce a Prolog compiler error message.