Weka: Decision Trees – J48

Lab 02 (in-class): 14 NOV 2016, 10:00-12:00, TURING

ACKNOWLEDGEMENTS: INFORMATION, EXAMPLES AND TASKS IN THIS LAB COME FROM MANY WEB SOURCES.

Learning objectives

In this assignment you are going to:

- using decision trees as implemented in weka (J48);
- playing with hyperparameters
- playing with filters

Preliminaries

J48

The C4.5 algorithm for building decision trees is implemented in Weka as a classifier called J48. Classifiers, like filters, are organized in a hierarchy: J48 has the full name weka.classifiers.trees.J48. The classifier is shown in the text box next to the Choose button: It reads J48 –C 0.25 –M 2. This text gives the default parameter settings for this classifier.

C4.5 has several parameters, by the default visualization (when you invoke the classifier) only shows –c ie. Confidence value (default 25%): lower values incur heavier pruning and -M ie. Minimum number of instances in the two most popular branches (default 2).
The full set of J48 parameter settings are explained here:
http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html

Valid for all classifiers: the parameter settings specified in the text box next to Choose are not editable directly. You must click on the box in order to open the weka.gui.GenericObjectEditor. For some obscure reason, the opions that refer to the parameter settings have unintuitive names and no explict reference exists between the switch (eg. -m) and the corresponding option name (eg. minNumObj). This mismatch is, in my opinion, quite unhandy, and hopfully it will be fixed in some future weka version. When you invoke a classifier the first time, not all the parameters are shown in the box. But if you open the weka.gui.GenericObjectEditor and start changing the default values, you will see that more swithes will appear in the box. This is, perhaps, another idiosyncrasy that might require some thought by weka's usability expert.

Parameters and Hyperparameters

In machine learning, we use the term hyperparameter to distinguish from standard model parameters.

Fitting a model to the data: Parameters are learned directly from data. A machine learning model is the definition of a mathematical formula with a number of parameters that need to be learned from the data. This is done through a process known as **model training**. In other words, by training a model with **existing data**, we are able to fit the model parameters. This means that if we train the same model (say a decision tree) first with, say, sales data and then with election data, we will have two different decision tree models with different parameters that are tailored to the training data. Oversimplifying, we could say that parameters are directly learned in the regular training process.

Hyperparameters express "higher-level" properties of the model, such as its complexity or how fast it should learn. Hyperparameters are usually fixed before the actual training process begins. So, how are hyperparameters decided? This is done by setting different values for those hyperparameters, training different models, and deciding which ones work best by testing them. Examples of hyperparametes: number of leaves or depth of a tree, or number of clusters in a k-means clustering. In weka, hyperpameters are the switches next to the classifir's name in the Choose box and the options in the weka.gui.GenericObjectEditor window. *Hyperparemeters are only called only "parameters" in weka*.

When the amount of data and computation time permits it, divide the data in three parts: training set, validation set (aka development set), and test set. We use the train and validation data to select the best model and the test data to assess the chosen model. In the first part, model selection, the validation model is treated as the test data. We train all competing models on the train data and define the best model as the one that predicts best in the validation set. We could re-split the train/validation data, do this many times, and select the method that, on average, performs best. Because we chose the best model among many competitors, the observed performance will be a bit biased. Therefore, to appropriately assess performance on independent data we look at the performance on the test set.

Since finding the optimal hyperparameters for a classifier is a tedious process, Weka offers some ways of automating this process a bit. Read here: https://weka.wikispaces.com/Optimizing+parameters (Search terms: Optimizing parameters in Weka).

In this lab will go for some manual explorations of hyperparameters.

Tasks 1 – The *Classify* Panel

We apply a classifier to the weather data. Load the weather data again. Load the *weather nominal* dataset:
http://stp.lingfil.uu.se/~santinim/ml/2016/Datasets/weather.nominal.arff

Then switch to the *Classify* panel by clicking the Classify tab at the top of the window. Select J48 by clicking the Choose button near the top of the Classify tab. A dialog window appears showing various types of classifier. Click the trees entry to reveal its subentries, and click J48 to choose that classifier.

We evaluate the performance using the training data, which has been loaded in the Preprocess panel—this is not generally a good idea because it leads to unrealistically optimistic performance estimates. Choose Use training set from the Test options part of the Classify panel. Once the test strategy has been set, the classifier is built and evaluated by pressing the Start button. This processes the training set using the currently selected learning algorithm, C4.5 in this case. Then it classifies all the instances in the training data and outputs performance statistics.

Interpreting the Output

The outcome of training and testing appears in the Classifier Output box on the right. Each time the Start button is pressed and a new classifier is built and evaluated, a new entry appears in the Result List panel. To see the tree, right-click on the entry trees.J48 that has just been added to the result list and choose Visualize tree. A window pops up that shows the decision tree. Right-click a blank spot in this window to bring up a new menu enabling you to auto-scale the view. You can pan around by dragging the mouse. Now look at the rest of the information in the Classifier Output area. The next two parts of the output report on the quality of the classification model based on the chosen test option. This text states how many and what proportion of test instances have been correctly classified:

Correctly Classified Instances 14 100%

This is the accuracy of the model on the data used for testing. In this case it is completely accurate (100%), which is often the case when the training set is used for testing.

Q1: what is the size of the tree? Explore the Result panel and find this information.

Task 2: Cross-Validation

Now evaluate J48 on the same dataset using 10-fold cross-validation. Click on the relevant radio button.

Q2: What is the estimated percentage of correct classifications for the training set (previous task) and for 10-fold cross-validation? Which one is more realistic? Why?

Task 3 – The Spam dataset: Exploring parameters

Download the *spambase* dataset:
http://stp.lingfil.uu.se/~santinim/ml/2016/Datasets/spambase.arff

Upload the dataset in weka.
Get familiar with the dataset and read the comments in the header of the dataset. Which are the classes? How many attributes, what types of attributes, etc.

As already mentioned in the Preliminaries, J48 algorithm has two important parameters, denoted by C (default value: 0.25) and M (default value: 2). Below is a table with

different combinations of values for these parameters:

	M=2	M=10	M=50	M=100
C=0.40				
C=0.25				
C=0.05				

Q3: Report the error rate on the training set of trees built with these different parameter settings. Explain your findings in relation to the meaning of the parameters.

Q4: Perform the same analysis, but now using 10-fold cross-validation. Compare with previous results. What conclusion can you draw from results in Q3 and Q4?

Q5: Are the default settings of these parameters (in red) reliable?

Task 4 – Filters: StringToWordVector

Make an ARFF file from the labeled mini-documents in the table below. Find out how to represent stings in an arff dataset.
Once you have finished, upload you dataset. In the Preprocess panel choose Filter→unsupervised→attribute and run StringToWordVector with default options on this data.

Some training data

Document text	Classification
The price of crude oil has increased significantly	yes
Demand of crude oil outstrips supply	yes
Some people do not like the flavor of olive oil	no
The food was very oily	no
Crude oil is in short supply	yes
Use a bit of cooking oil in the frying pan	no

Fig. Manually re-writing data: from tabular format to the ARFF format

Q6: How many attributes are generated?
Q7: Now change the value of the option minTermFreq to 2. What attributes are generated now?
Q8: Can you explain the behavior of this filter? Find out the correct information in the manual book, by using the online help, and other resources.

Build a J48 decision tree from the last version of the data you generated
Q9: Describe your results.

Additional practice

Witten et al. (2011) - Chapter 11: 410-414 ("Working with Models" excluded).