Hands-on Practice: Classifying Actual Documents

A standard collection of newswire articles is widely used for evaluating document classifiers. *ReutersCorn-train.arff* and *ReutersGrain-train.arff* are training sets derived from this collection; *ReutersCorn-test.arff* and *ReutersGrain-test.arff* are corresponding test sets. The actual documents in the corn and grain data are the same; only the labels differ. In the first dataset, articles concerning corn-related issues have a class value of 1 and the others have 0; the aim is to build a classifier that identifies "corny" articles. In the second, the labeling is performed with respect to grain-related issues; the aim is to identify "grainy" articles.

Exercise 1. Build classifiers for the two training sets by applying *FilteredClassifier* with *StringToWordVector* using (1) *J48* and (2) *NaiveBayesMultinomial*, evaluating them on the corresponding test set in each case. What percentage of correct classifications is obtained in the four scenarios? Based on the results, which classifier would you choose?

Exercise 2. Based on the formulas in Table 5.7 (page 176), what are the best possible values for each of the output statistics? Describe when these values are attained.

The Classifier Output also gives the ROC area (also known as AUC), which, as explained in Section 5.7 (page 177), is the probability that a randomly chosen positive instance in the test data is ranked above a randomly chosen negative instance, based on the ranking produced by the classifier. The best outcome is that all positive examples are ranked above all negative examples, in which case the AUC is 1. In the worst case it is 0. In the case where the ranking is essentially random, the AUC is 0.5, and if it is significantly less than this the classifier has performed anti-learning!

Exercise 3. Which of the two classifiers used above produces the best AUC for the two Reuters datasets? Compare this to the outcome for percent correct. What do the different outcomes mean?

The ROC curves discussed in Section 5.7 (page 172) can be generated by right-clicking on an entry in the result list and selecting *Visualize threshold curve*. This gives a plot with FP Rate on the *x*-axis and TP Rate on the *y*-axis. Depending on the classifier used, this plot can be quite smooth or it can be fairly irregular.

Exercise 4. For the Reuters dataset that produced the most extreme difference in Exercise 3, look at the ROC curves for class 1. Make a very rough estimate of the area under each curve, and explain it in words.
Exercise 5. What does the ideal ROC curve corresponding to perfect performance look like?

Other types of threshold curves can be plotted, such as a precision–recall curve with Recall on the *x*-axis and Precision on the *y*-axis.

Exercise 6. Change the axes to obtain a precision–recall curve. What is the shape of the ideal precision–recall curve, corresponding to perfect performance?

Exploring the *StringToWordVector* Filter

By default, the *StringToWordVector* filter simply makes the attribute value in the transformed dataset 1 or 0 for all single-word terms, depending on whether the word appears in the document or not. However, there are many options:

- *outputWordCounts* causes actual word counts to be output.
- *IDFTransform* and *TFTransform*: When both are set to *true*, term frequencies are transformed into TF × IDF values.

- *stemmer* gives a choice of different word-stemming algorithms.
- *useStopList* lets you determine whether or not stopwords are deleted.
- *tokenizer* allows different tokenizers for generating terms, such as one that produces word *n*-grams instead of single words.

There are several other useful options. For more information, click on *More* in the Generic Object Editor window.

> **Exercise 7.** Experiment with the options that are available. What options give a good AUC value for the two datasets above, using *NaiveBayesMulti- nomial* as the classifier?

Not all of the attributes (i.e., terms) are important when classifying documents. The reason is that many words are irrelevant for determining an article's topic. Weka's *AttributeSelectedClassifier*, using ranking with *InfoGainAttributeEval* and the *Ranker* search, can eliminate less useful attributes. As before, *FilteredClassifier* should be used to transform the data before passing it to *AttributeSelectedClassifier*.

> **Exercise 8.** Experiment with this, using default options for *StringToWordVector* and *NaiveBayesMultinomial* as the classifier. Vary the number of the most informative attributes that are selected from the information gain–based ranking by changing the value of the *numToSelect* field in the *Ranker*. Record the AUC values you obtain. How many attributes give the best AUC for the two datasets discussed before? What are the best AUC values you managed to obtain?