Lab 07 - Testing

Testing

In the previous labs we tried different learning methods for classification and looked at the outputs provided under “Confusion Matrix” sections in the “Classifier output” windowpane. In this lab we will consider different testing options. Percentage split (Holdout) and Cross-validation are common techniques for assessing accuracy based on randomly sampled partitions of the given data. The use of such techniques to estimate accuracy increases the overall computation time yet is useful for model selection.

K-fold Cross-validation

Cross-validation is the statistical practice of partitioning a sample of data into subsets and the subsets are combined in some way for both training and testing.

In K-fold cross-validation, the original sample is partitioned into K sub- samples. Of the K sub-samples, a single sub-sample is retained as the testing data, and the remaining K-1 sub-samples are used as training data. The cross- validation process is then repeated K times (the folds), with each of the K sub- samples used exactly once as the validation data. The K results from the folds can then be averaged (or otherwise combined) to produce a single estimation.

Leave-one-out cross-validation

As the name suggests, leave-one-out cross-validation involves using a single observation from the original sample as the testing data, and the remaining observations as the training data. This is repeated such that each observation in the sample is used once as the testing data. This is the same as K-fold cross- validation where K is equal to the number of instances in the original sample.

Holdout validation

Holdout (Percentage split in WEKA) validation is not strictly cross-validation because the data are never crossed over. Instances (examples) are chosen randomly from the

initial sample to form the testing data, and the remaining instances are retained as the training data. Normally, less than a third of the initial sample is used for testing data.

Exercises

1. Fire up the Weka software, launch the explorer window and select the “Pre-process” tab. Open the iris training dataset.
2. Select the “Classify” tab. Under the “Classifier” section select “Multi-layer Perceptron” leave the parameters as the defaults.
3. Run the model using each of the different “Test options” in turn.

    a. For “Supplied test set” use iris-testing.arff.
    b. For “Cross-validation” try different fold values e.g., 2, 3, 5, 10, 15, 30, 45, 90. What do these values mean? What is interesting about making “Folds” equal to 90?
    c. For “Percentage split” try different “%” values e.g. 20, 40, 60, 80. What do these values mean?
    d. How does each of the test options affect the performance of the trained model (look at the visualisations as well as the text output)? Which test option gives the best performance? Is it generally a good idea to use the same dataset to both test and train a model, i.e., option “Use training set” *(note: evaluating a model on training data might still be useful, because it generally represents an upper bound to the model’s performance on fresh data)*?

4. Try changing some of the classifier parameters *(read the “more” section for a brief explanation of the classifiers parameters)* and repeat exercise. How does changing the classifier parameters affect the performance?
5. Under the “Classifier” section change the classifier to “BayesNet”, leave the parameters as the defaults.
6. Repeat exercises 3 and 4 for the new classifier.
7. How does altering the classifier affect performance? How do you determine which is the best classifier, testing/training strategy and parameter set for a particular classification problem. Make a note of what you think is the best

classifier and its parameters and the best testing/training strategy and its parameters.

8. Open the diabetes.arff data set. Assess the performance of your selected classifier and testing/training strategy using different parameters. Are the parameters your noted in 7 are still the best parameters? What are your reasons?