184.702 Machine Learning – Exercise 3

The third exercise deals with advanced experiments on particular topics in machine learning. The exercise shall be done in groups of 2-3 students, which may be the same as in the first two exercises. Please arrange with your colleagues, and then apply to the same group in TUWEL.

You will have to present your findings in a presentation for all the course participants in January. You will have to be present during the whole duration of the time slot you will be presenting; participation in other slots is optional.

You have to select your topic by applying to one of the groups - the topic will be indicated in the group name. All group members should apply to the same group - so please, do not apply to a group where already a student from a different group has applied to... If you have an interesting idea of any other topic, please don't hesitate to contact us and discuss it with us. If we see it fitting to the topics of the course, you'll get our ok to work on your idea.

General comments for the exercise

This exercise is supposed to be done using an advanced machine learning environment that allows you to script your experiments. This can be for example the WEKA API (WEKA provides a vast number of different learning algorithms, and additional features such as a defined format for writing & reading results, for performing significance tests, etc.), or another toolkit that provides a similar wealth of algorithms and evaluation methods, such as Python or R. If you, for whatever reason, want to implement your solution in a different environment, then please contact us (mayer@ifs.tuwien.ac.at & musliu@dbai.tuwien.ac.at), and explain why the exercise can be performed in the same quality in the different environment.

Your submission should contain

- a zip file with all needed files (your source code, your code compiled, data sets used, any libraries you are using)
  - If applicable, provide a means to compile your classes, preferably with a Maven or ANT build file, and a short how-to explaining the way to start your program (which is the main class, which command-line options does it expect, ..)
- a report, describing
  - Your task
  - Your solutions (also describe failed approaches!)
  - Your results, evaluated on different datasets, parameters, ...
  - An analysis of the results
- If using WEKA, please use the most recent version from http://www.cs.waikato.ac.nz/ml/weka/ !
- Where applicably, your program shall be configurable via command-line options or a configuration file, to modify parameters, evaluation types, etc... i.e. it should not be needed to modify the code to change these options.
  For Java, a useful framework for command-line-option is Apache Commons CLI (http://commons.apache.org/cli/), which takes most of the burden to parse the supplied

arguments. JSAP (http://martiansoftware.com/jsap/) might be another solution, there are numerous others available as well.

You need to chose to do either Exercise 3.1, or one of the topics of Exercise 3.2; there are limited groups for each topic available.

Exercise 3.1.: Meta Learning

See separate set of slide set.

Exercise 3.2.: Large-Scale evaluation

In these exercises, you shall employ the API from an existing ML Toolkit (WEKA, Python scikit-learn, R, …) and perform a more automated evaluation than in the previous exercises, and an evaluation beyond what you can obtain from manually running single experiments/parameter settings.

The descriptions of the different topics are not to be read as complete specifications, they just summarise the general idea and goals. If there are certain unclarities, please do not hesitate to ask in the forum.

To have a common format for the results, you shall use the WEKA ARFF Format to persist your results (e.g. tested accuracy, precision, TP rate, …) and meta-data (name and parameters of classifiers, runtime (total, and each of training and testing), date of experiment, input data used (e.g. filename) about your experiments. An example of what you can include in your file will be provided in TUWEL.

APIs for manipulating ARFF files are available in many toolkits; if not, it is also a rather simple format to create yourself. In WEKA itself, you can utilise the WEKA implementations of ResultListener, such as InstancesResultListener or DatabaseResultListener.

As you shall do mass evaluation, you should generate tables and diagrams automatically from the result files, instead of manually compiling them.

You can re-use the data sets you used in your first exercise, as you should be familiar with them - which should help you in the evaluation. ***Mind however that you shall perform experiments with a large number of data sets, so you will need to choose additional data sets.***

1. Ensemble classifier learning & evaluation

Using algorithms available in your toolkit, develop an ensemble classifier that uses different algorithms, and different parameters of these classifiers. Implement two different strategies for combining the individual predictions of the classifiers, namely majority voting and weighted majority voting.

For the weighted majority voting, use a part of your data as validation set to estimate the classifiers accuracy (e.g. on the class it predicts), and use this value subsequently as weight. You should do this accuracy estimation not by doing a single split on the training set, but by doing cross-validation on the training set to achieve a more sound estimation of the expected performance.

Your program shall allow the user either to specify which algorithms (and parameters) to use (via command-line, or via a configuration file), or have some heuristics on building a "good" ensemble of classifiers.

Afterwards, run a suite of experiments on different data sets and different classifier ensembles, and compare the results of the ensemble with the results of each individual classifier used, and also compare requirements in runtime, etc.. Analyse the results in regard of whether you can always find better results, in whether one combination strategy is dominant over the other, and whether there is a set of (nearly) optimal combination of classifiers.

2. Automatic feature (attribute) selection & evaluation

There is a wealth of feature selection algorithms, both supervised and unsupervised ones; the WEKA API gives a good overview what is available and used, such as PCA, Information Gain, .. (see http://weka.sourceforge.net/doc.dev/weka/attributeSelection/AttributeEvaluator.html for a list)

In this task, you should evaluate the effect of the different techniques, and different parameters to it. For this exercise, we will provide you with a set of data sets from the music classification domain, which (contrary to e.g. text classification) does not yet widely employ feature selection.

Develop a small framework that on a set of different data sets automatically applies different feature selection techniques. There are two different topics to choose:

a) Evaluate the number of selected features

Evaluate the effect of the desired number of features after the selection (or thresholds used in the selection technique to decide how many features are kept). Vary the number of features from very few to very high, and record and analyse the results; specifically, for each feature selection technique and each target number of features selected, record the the runtime and classification accuracies. Then analyse the chart of performance and runtime for each method on several dataset, and try to make an evaluation of a potentially optimal number (or range) of features to select. The user should be able to specify the different numbers of features, or a step-size; if none of these is provided, think of a good strategy yourself (e.g. smaller step-size for smaller number of features, then increasing, depending on the total number of features in the dataset).

b) Compare feature selection methods

Analyse the differences in the rankings between the different methods, i.e. analyse how each individual feature is ranked by the different methods. Investigate whether there is a trend in that a certain subset of features is selected by all (or most) techniques - do they tend to select a specific subset of features, or does that vary a lot? Is there a trend in supervised and unsupervised methods?

Further, investigate whether a combination of feature selection methods is more beneficial, i.e. by taking those features that get selected by most algorithms, or by combining the top N from each algorithm, into a subset of features that you would then use for the actual training & testing.

For both topics, your program shall allow the user either to specify which feature selection techniques (and parameters) to use (via command-line, or via a configuration file), or otherwise simply use all available techniques.

3. Automatic comparison of k-nearest neighbour search optimisation & evaluation

k-nearest neighbours, even though a very simple approach, yields good results in many classification tasks. One major issues is however the classification time on larger data sets, mainly stemming from the expensive (pair-wise) comparisons needed to find the nearest neighbours. In this exercise, you shall thus evaluate different optimisation strategies for this neighbour search.

A set of such strategies is for example in implemented in WEKA in combination with the IBk and LWL classifiers, and might also be available in other ML packages.

Develop an evaluation framework that on different data sets automatically evaluates the effect of applying these strategies on the runtime and performance of the algorithms, and compares the different strategies against each other, and against the "base-line" of using the classic linear (brute-force) search. Record how long it takes to build the search optimisation structures, and how the optimisation changes the time needed for the classification process, and how the overall time needed compares to the base line. Try to estimate what the "break-even" point of time consumption is in regard to the size of the dataset and the dimensionality, i.e. at which combination of size and dimensionality building the search optimisation structure is equal in overall runtime than the linear search.

Further evaluate in detail the effects on the search optimisation strategies on the accuracy of the algorithm - does that differ? Also, even if the accuracy stays the same, are there data samples that get classified differently by the different strategies? Is there a trend that the optimisation strategies, when they predict a different class than the "base-line", all classify the pattern in the same (other) class, or is the prediction different for all strategies?

Subsequently, try to analyse which strategies lead to a similar nearest-neighbour selection.

Your program shall allow the user either to specify which search optimisation strategies (and parameters) to use (via command-line, or via a configuration file), or otherwise simply use all available strategies

4. Analysis of runtime behaviour

Different algorithms exhibit different runtime (efficiency) behaviour on datasets of different dimensionality or size. Your task is to pick a number of different algorithms (5), and analyse how these algorithms should theoretically behave in runtime regarding the size and dimensionality of the dataset, for both the training and classification steps.

Then, pick a number of datasets that differ in size and dimensionality, and analyse the actual runtime behaviour observable in experimentation. Report on whether these match the theoretical analysis.

Mind that for very small datasets resp. very short runtimes, these numbers might not be very precise, as there might be a certain startup overhead for starting the program (JVM startup (or comparable), memory allocation, etc..)

5. Analysis of predicted generalisation power / stability

When estimating the expected generalisation power of a classifier, this is normally done by testing it on a test set (sometimes called validation set), which contains labelled data NOT used in the training process. Your task in this exercise is an evaluation on how stable these predictions are, by dividing your data into a training set and two (or more) different testing sets. After evaluating your trained models on the first testing set, re-evaluate them on the second test set, and analyse the differences.

Evaluate how well the generalisation estimation performs - is the best model on the first test set always the one having the best generalisation power also on the second test set?How different is the generalisation performance on the two sets? Is there a trend in which classifiers are more reliable, i.e. have similar performance on the test sets, and which ones tend to fluctuate more?

Also, vary the size of the test sets, to analyse what a reasonable size for each of those sets is. Perform you evaluation on a number of different datasets, with different characteristics.

The user shall be able to specify which algorithms (and which parameters for these) he would like to use. If no models are provided, use a pre-defined list of algorithms (with different parameters).