F20/21DL. Data Mining and Machine Learning

Lab 8. Decision Trees

Covering Practical work to be done by students in Week 8

The purpose of this lab is:

1. to practice what we have learned so far:
   - Algorithms for building and using Decision trees
2. understand methods applied in Supervised learning; common pitfalls in supervised learning (such as overfitting, failure to generalise to new data, dealing with unbalanced data);
3. understand practical issues of running Decision trees (by means of experimenting with the $J4.8$ and $C4.5$ algorithm for Decision tree learning; its various learning parameters and options);
4. prepare for the electronic test;
5. to help you to make progress with Python tutorial and your DM & ML portfolio.

   *Note: the exercises marked by *** are harder, as they assume some maths background and do not necessarily rely on lecture slides only.*

1 Decision trees: Understanding the Algorithm

1. Read the lecture slides, make sure you understand them, ask questions.
2. **Preparing for Part 1 of the Test on the Decision tree algorithm.**

   Take the following toy data set for **Face Emotion Recognition**:

Picture	Cell 33	Cell 42	Cell 48	Cell 58	Face expression
P1	White	Black	White	White	Happy
P2	Black	Black	White	White	Happy
P3	White	White	White	Black	Sad
P4	White	White	Black	White	Happy
P5	White	White	White	White	Happy
P6	White	White	Black	Black	Sad
P7	Black	White	White	Black	Sad
P8	Black	White	Black	Black	Sad
P9	Black	Black	Black	White	Happy
P10	White	Black	White	Black	Happy

   Using the *Decision tree* algorithm from the lecture slides:

   (a) Construct a decision tree for this data set manually:
   - In your tree, non-leaf nodes should be given by input features and the leaves – by values of the output feature. The edges should be labelled with values of the input features, i.e. "White" and "Black".
   - The splitting criterion is: take each attribute in order of its appearance in the data set.
   - The terminating condition is: all examples are classified in the same class.

- Left branches should stand for "White", and right branches should stand for "Black".

Be ready to answer questions about the tree's architecture.

(b) Use this tree and the tests:

- Test 1: "a neutral (Happy?) face with noise":
  `Black, White, White, White, ???`
- Test 2: "a Happy face with a beard":
  `Black, Black, White, Black, ???`

Make predictions for classes of these examples.

(c) (*) Construct another decision tree for the data set (again manually). This time, use a **myopic** splitting criterion.

- There are many options for defining myopic criterion (see the lecture slides). But, to save our time, we will use its simplified version:
  - *always choose the attribute splitting of which allows to firmly classify instances to classes and thus avoid further splits.*[1]
    - If such does not exist, choose the leftmost available attribute.
    - If several such exist, choose the leftmost attribute of these.
  - The attribute that results in two leaves should be preferred to the one that only results in one leaf.
- All other settings are the same.

Be ready to answer questions about the tree's architecture.

(d) Perform the same tests on this new tree

- Test 1: `Black, White, White, White, ???`
- Test 2: `Black, Black, White, Black, ???`

(e) (***) *Make conclusions about accuracy, size, performance of these two trees; redundant and most correlating attributes. Do they **generalise** well to new data? Make conclusions about the issues of **over-fitting** and the relation between the training and testing data sets.*

2 DM & ML Portfolio

This part is to be completed in groups, and will be assessed during the labs. Marking scheme: this lab will bring you up to 2 points. 1 point for completing the task, 1 additional point for any non-trivial analytical work with the material.

2.1 Python Tutorial and Programming Practice (Prior to the lab)

This part is for your individual programming practice during the week.

- Watch recordings, and run the Python code accompanying tutorial **P5. Decision Trees, Linear Regression and Logistic Regression (week 8)**.
- Make sure you can run this code using **your chosen data set**. In case you have any issues, contact your lab tutor and ask for help.
- Note: Make sure that you obtained or created a test set.
- Use Decision trees (the J48/C4.5 algorithm) on a training set, measure the accuracy. Then measure the accuracy using 10-fold cross-validation on the training set[2]. Record all your findings and explain them. Use the major metrics: accuracy, TP rate, FP rate, precision, recall, F measure, the ROC area if needed.

[1] NB: this is related to the Gini index, see Geron's book "Hands-on Machine Learning" (and Python lab on Canvas) for Python exercises.

[2] If you are dealing with an unbalanced dataset, what else should you consider when doing the train/test split?

- *(optional for BSc but recommended for higher marks, mandatory for MSc)* Experiment with various decision tree parameters that control the size of the tree. For example: depth of the tree, confidence threshold for pruning, splitting criteria and the minimal number of instances permissible per leaf.

  Note you can try to tune your model hyperparameters manually, until you find a great combination of hyperparameter values. Alternatively, you can use Scikit-Learn's GridSearchCV to optimize hyperparameters for you. Check GridSearchCV Documentation: `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html` *GridSearchCV works well if you are exploring relatively few hyperparameter combinations. However when the hyperparameter search space is large, it is often preferable to use RandomizedSearchCV instead. Check RandomizedSearchCV() Documentation:* `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html` *Refer to last part of tutorial* ***P3. Classifiers and Advanced Features (week 5)****.*

- *(optional for BSc but recommended for higher marks, mandatory for MSc)* Put all your results in a suitable form: it can be a table or a series of graphs, that visualise the variations of performance between different settings of the decision tree algorithm. (Lab 5 gave an example of how machine learning experiments may be assembled into a comparative table. You can use it as a starting point. But let it not limit your creativity.)

Tree 1 Using the best working parameters, repeat the experiment, this time using training and testing data sets instead of the cross validation. That is, build the J48 classifier using the training data set, and test the classifier using the test data set. Note the accuracy. Answer the question: Does the decision tree generalize well to new data? How do you tell?

Tree 2 Make new training and testing sets, by moving 30% of the instances from the original training set into the testing set. Note the accuracies on the training and the testing sets.

Tree 3 Make new training and testing sets, by moving 60% of the instances from the original training set into the testing set. Note the accuracies on the training and the testing sets.

- *(optional for BSc but recommended for higher marks, mandatory for MSc)* Try some other decision tree algorithms (e.g. random forests). Repeat all of the above experiments and make conclusions.

2.2 During the lab:

- Firstly, make conclusions about our experiments with tuning parameters of the decision trees. Make conclusions: what was the influence of various parameters on the classifier's performance? Hypothesise why.

- Secondly, analyse **Tree 1**, **Tree 2** and **Tree 3** from the point of view of the problem of classifier over-fitting. Do you notice the effects of over-fitting? How? Note your conclusions in the Jupyter notebook.

  Note: to make conclusions about overfitting, you must compare accuracies of your three decision trees on training and on testing data sets.

- **The tutors will mark:** quality of your code, completeness of your tables/graphs that summarise the results of your group experiments and your analysis of the tables/graphs, i.e. what sort of conclusions you make, how well the conclusions reflect your understanding of the algorithms.

2.3 After the lab:

- *Group rep:* Make sure all group members have tasks for the week

- *Everyone:* Incorporate the discussion during the lab into your Python code

- *Everyone:* Incorporate all code used in the lab into your Portfolio repository.