DECISION TREES PART 2

All theory issues were discussed during the lecture, so feel free to go back to the slides and notes from the lecture or [1].

As we have already talked about the decision tree induction, we also need to remember about some potential problems that can occur. The first thing is dealing with missing values in the data.

Missing values

Question
Do you remember any possibilities how can we deal with missing data?

There are a few different ways of dealing with missing data:

- *ignore* - we can just ignore all the records that have missing data or the records that have a great number of missing data. Although it is pretty easy to apply we need to take into account the whole situation. Firstly, then can be a great amount of records with missing data, so ignoring records with any missing value causes removing a large amount of information from our dataset. Secondly, the missing values might not be random, so even if there are only a few of them, we can still lose some important information.
- *most common attribute value* - we need to find the most common value that appears in the dataset and fill all the empty places with this value. Comparing to the previous method, we save all the records, so we do not lose information that comes from them. However, there can potentially appear some bias in the data or problems in getting proper results from classification, especially when there are many missing values. Moreover, the missing values are filled with the value regardless the class that the example is classified.
- *most common attribute value in the class* - this method deals with one of the problems that method above does not. Taking into account class assignments of each example can improve the result and reduce the bias.
- *most common with additional info* - in this method we fill missing value with the most common attribute value (globally or within the class), but we also add a new column which has information if the value was missing or not at the beginning.
- *all possible attribute values* - this method assumes that we add to our dataset the example that has missing value(s) as many times as we have different values on this attribute. It creates a great amount of fictitious

examples and, depending on the amount of different values, can also made a bias.

- *all possible attribute values in the class* - as it was before, we can just focus on one specific class and reduce the number of artificial examples, so it can reduce the influence of this example.
- *null as value* - there is also possibility to consider empty value as one of values, so we do not add any information but only focus on the values that we have. However, as the missing value is treated equally to the ones that we have, such information can appear in the decision tree and may be treated as more discriminating than the values that we have.

Inconsistent data

Question
Do you remember any reasons of inconsistencies that can occur in the data?

First reason of inconsistencies is **noise**. We can distinguish two main types of noise [3]:

1. *label noise* which means that there are some problems with assigning label/class for some examples.
    - *contradictory examples* - there are at least two examples that have the same values for all attributes but have different label/class.
    - *misclassifications* - there are examples with wrong class assignment e.g. because small differences in boundaries between classes.
2. *attribute noise* is connected with the errors that appear on attribute values, such as:
    - incorrect attribute values,
    - missing values,
    - don't care values.

The other reasons of inconsistent data can be **missing attributes**. It happens when the final decision is made basing on the information that is not included in the attributes, for example the way of treatment that is made basing on your medical history, not only your current symptoms. Moreover, there can be also some **experts' inconsistencies** that can result in different decisions for similar examples.

Overfitting

As we induce a decision tree, we want it be able to generalize as good as it is possible, to achieve good results not only for train set but also for a test set.

However, is is often that the decision tree is to "focused" on training examples that it creates too accurate representation which is called **overfitting**.

Question
Do you remember the reasons of overfitting?

Question
Can you explain the difference between **training set**, **validation set** and **test set**?

Question
There are two terms that you should remember while talking about overfitting: **training error** and **generalization error**. Can you explain the difference between them? Can both of them be calculated precisely?

As you already know the difference between **training error** and **generalization error**, we can focus on the latter. We can use different ways of generalization error estimation such as:

1. We can assume that the training error is our *optimistic estimate* of the generalization error,
2. We should also take into account the *complexity* of the tree. While we have two trees with the same generalization error and different complexities, we should choose simpler one, as it is supposed to be higher possibilities of generalization,
3. We can use validation set to make the prediction of the generalization error,
4. As a *pessimistic estimate*, we can make an assumption that the generalization error will be worse than training error, depending on the number of leaves in the tree. We can use the following equation to calculate this error rate:

$$e'(T) = \frac{errors\_in\_training + \Omega * number\_of\_leaves}{training\_set\_size}$$

   Ω is a parameter that we can freely set. It gives us information about potential number of errors in each leaf.

Task
Given the training error $e(T) = 3/25$, $\Omega = 0.5$ and number of leaves equals to 5, calculate the pessimistic estimate.

Cross validation

We need to divide our training dataset to k folds and use one of the folds as validation set and the rest as training set. We repeat the whole procedure k

times, each time evaluating it on the validation set. Finally, we can summarize the quality of our model, taking into account the results from each run. The parameter k should be chosen experimentally, but mostly it is value $k = 5$ or $k = 10$.

What is worth remembering?

- while choosing k, remember to have sizes of each set representative, do you remember assumptions that were presented on introduction to statistics?
- it is a good practice to shuffle dataset before making a split on it, to avoid similar objects next to each other,
- a simple evaluation method for checking quality of the model is the ratio of properly classified examples to all that we have in the dataset,
- if we do not have a validation set, we can use cross validation to better adjust parameters for a classifier.

Extentions and variations

1. *leave-one-out cross validation* (LOOCV) - we have an extreme situation where k is set to the size of the set. It has really big computational cost, so you need to remember that this method cannot be used on large dataset. However, it can be used especially when estimated model performance is critical.
2. *stratified cross validation* - in each fold should be a similar number of observations for each class (or proportionally to class distribution).
3. *nested cross validation* - we run the program more than once on each fold, because we want to perform tuning on parameters basing on the first results.

Pruning

To cope with overfitting problem we can prune our decision tree. There are two main concepts of pruning.

Pre-pruning

The decision tree is no longer built from the current node after fulfilling the chosen criterion:

- all instances belong to the same class,
- all attribute values are the same,
- the number of instances in the node is less that it was predefined by the user,

- the split on the node does not change the purity/impurity, e.g. information gain does not change,
- estimated generalization error falls below our threshold.

It may require from us introducing some parameters e.g. the minimum number of instances in the node, but it needs less memory than the post-pruning approach.

Post-pruning

Post-pruning approach firstly lets the tree to grow to the end and then trim it from the bottom.

- If generalization error improves after trimming, replace subtree with leaf node.
- The leaf node gets the label from the majority class in the node.

Example
Let's remind the example with cars from the previous lesson. This time we have 30 examples and some of the attributes as before:

- *buying_price*: low, med, high, vhigh
- *safety*: low, med, high

Moreover, $\Omega = 0.5$. Having only one node in our decision tree, we get training error equals to $\frac{10}{30}$.

Task
Calculate pessimistic estimate for the given data.

Task
Then we make a split on *buying_price*. We get training error equals $\frac{9}{30}$. Calculate pessimistic estimate for this situation.

Task
Compare values of pessimistic estimate before and after split. What should we do?

Homework

1. The whole task is to say if a student pass or not the subject. Check the website `https://archive.ics.uci.edu/ml/datasets/student+performance` and analyze the list of attributes. Which of them would you consider as the most significant according to you? Choose 5-10 attributes, which will take part in the part of experiments.
2. Open *student-mat_train* and *student-mat_test*.
3. Determine which metrics will be proper for the given datasets. Report three the most accurate metrics.
4. Get the chosen 5-10 attributes and test a few different values of the parameters, at least: confidenceFactor, minNumObj and binarySplits. Show the results for each set of parameters (you can visualize it also). For which set do you have the best result for test set?
5. Repeat the previous step but this time get the whole set of attributes.
6. Load file *student-port* and run analysis for $k = 10$ in cross-validation. Present your results.
7. Both datasets (math and Portuguese) have the same set of attributes. Compare the structure of trees with the best results for each dataset. Are there any similarities between them? Basing on these results, can we say which attributes can say if the student is attentive?
8. Choose any other algorithm that you already know e.g. algorithm for rule induction that we used on first laboratories (PRISM) or Naive Bayes and run it on *student-mat* dataset. Compare the results from both algorithms. Which attributes had the biggest influence on the result? Are these attributes similar to those that you chose intuitively at the beginning of the task?

The whole task should be done in Weka.

The final report should be sent in pdf format (any other format will not be checked).

You need to remember to put in your report:

- for each set of parameters, confusion matrix and values on the metrics that you chose as the most accurate,
- charts that visualize how change of each parameter influences the results,
- show the decision tree which gives the best results for each situation (chosen attributes on math dataset, all attributes on math dataset, all attributes on Portuguese dataset).

Time: + 2 week

References

[1] KRAWIEC, Krzysztof; STEFANOWSKI, Jerzy. Uczenie maszynowe i sieci neuronowe. Wydaw. Politechniki Poznańskiej, 2003.

[2] GAVANKAR, Sachin; SAWARKAR, Sudhirkumar. Decision tree: Review of techniques for missing values at training, testing and compatibility. In: 2015 3rd international conference on artificial intelligence, modelling and simulation (AIMS). IEEE, 2015. p. 122-126.

[3] ZHU, Xingquan; WU, Xindong. Class noise vs. attribute noise: A quantitative study. Artificial intelligence review, 2004, 22.3: 177-210.