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].

We have already talked about the induction of decision trees, but we also need to be aware of some potential problems that can occur. The first issue is dealing with missing values in the data.

Missing values

Question
Do you remember how we can 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, it may not always be the best solution. First of all, there can be a great amount of records with missing data, so ignoring records with any missing value leads to removing a large amount of information from our dataset. Second, 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 of the attribute that appears in the dataset and fill all the empty places with this value. Compared to the previous method, we save all the records, so we do not lose information that comes from them. However, this way we may introduce some bias in the data which can lead to problems in getting proper results from classification, especially when there are many missing values. Moreover, in this approach the missing values are filled with the same value regardless of the class the example is classified as.

- *most common attribute value in the class* - this time, we fill the blanks with the value of the argument which occurs most often for a given class (as opposed to the previous method, which chooses the value which occurs most often in the entire data set). Taking class assignments of each example into account 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 an information whether the value was originally missing.

- *all possible attribute values* - this method assumes that we add to our data set a copy of the example with missing value(s), once for each of

the different values on this attribute. It creates a great amount of fictitious examples and, depending on the amount of different values, can also introduce 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 behind the inconsistencies that can occur in the data?

One of the reasons for 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 there are 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 inconsistency in our data may also be caused by **missing attributes**. This happens when the class depends on the information that is not included in the attributes (imagine a doctor diagnosing you based on your medical history, instead of 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 well as possible, so it can achieve good results not only on the train set but also on a test set. However, it is often the case that the decision tree is so "focused" on training examples that it creates a model which is too specific. This is what we call **overfitting**.

Question
Do you remember the causes 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 when 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 the **training error** and the **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 can also take into account the *complexity* of the tree. If we have two trees with the same generalization error and different complexities, we should choose the simpler one, as it is expected to generalize better,
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, with a penalty 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 can be interpreted as the expected number of additional errors per leaf, when testing data on a test set of the same size as the training set.

Task
Given the training error $e(T) = 3/25$, $\Omega = 0.5$ and number of leaves equal 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 a validation set and the rest as training set. We repeat the whole procedure k times, each time evaluating it on a different validation set (different fold). Finally, we can summarize the quality of our model, taking into account the results from each run. The quality of the models averaged over all k folds is a good estimation of a quality of a model trained on the full data set. The parameter k should be chosen experimentally, but usually $k = 5$ or $k = 10$ is used.

What is worth remembering?

- while choosing k, remember to have representative sizes of each fold. Do you remember the 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 being next to each other (and therefore placing them in the same fold),
- a simple evaluation method for checking quality of the model (*accuracy*) is the ratio of properly classified examples to all examples that we have in the dataset,
- if we do not have a seperate validation set, we can use a cross validation to better adjust parameters for a classifier.

Extensions and variations

1. *leave-one-out cross validation* (LOOCV) - we have an extreme situation where k is set to the size of the dataset. This approach has a high computational cost, so you need to remember that this method shouldn't be used on large datasets. However, it can be used in some situations, especially if the proper estimation of model's 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 combat overfitting we can prune our decision tree. There are two main approches to pruning.

Pre-pruning

We stop growing the decision tree 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, i.e. information gain equals zero,
- estimated generalization error falls below our threshold.

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

Post-pruning

In the post-pruning approach we let the tree to grow normally. Once we have a full tree, we prune it starting from the bottom (leaves).

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

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

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

Let's assume $\Omega = 0.5$. Having only one node in our decision tree, we get a training error equal to $\frac{10}{30}$.

Task
Calculate the pessimistic estimate for the given data.

Task
Let's make a split on *buying_price*. After the split, the training error equals $\frac{9}{30}$. Calculate the pessimistic estimate for this situation.

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

Exercise

1. The task should be done in Weka.
2. The classification task is to predict if a student will pass the subject or not. Check the website https://archive.ics.uci.edu/ml/datasets/student+performance and analyze the list of attributes. Which of them would you consider to be the most important? Choose 5-10 attributes, which you will use in the experiments.
3. Open *student-mat_train* and *student-mat_test*.
4. Determine which accuracy metrics will be the most useful for this problem.
5. Using J48 algorithm in Weka, restrict the list of attributes to the 5-10 attributes selected earlier and test a few different values of the parameters of the algorithm (for example, confidenceFactor, minNumObj and binarySplits).
6. Repeat the previous step, but this time use all of the attributes.
7. Load file *student-por* and run analysis for $k = 10$ in cross-validation.
8. 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? Based on these results, can we tell which attributes are the best predictors of whether the student is attentive?

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.