DD2421
Machine Learning

Decision Trees

Klas Lindgren

kllindg@kth.se

Arvid Holmäng

aholmang@kth.se

September 21, 2022

Contents

1 Assignment 0

Each dataset has properties that make them hard to learn. Motivate which of the three problems is most difficult for a decision tree algorithm to learn.

1.1 Explaining the datasets

Each MONK dataset is filled with six-dimensional samples.

$$MONK_n = \begin{bmatrix} s_1 \\ \vdots \\ s_k \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \\ & & \vdots & & & \\ a_1 & a_2 & a_3 & a_4 & a_5 & a_6 \end{bmatrix}$$

where each attribute $(a_1, a_2, a_3, a_4, a_5, a_6)$ may take the following values

$$\begin{array}{lll} a_1 \in \{1,2,3\} & a_2 \in \{1,2,3\} & a_3 \in \{1,2\} \\ a_4 \in \{1,2,3\} & a_5 \in \{1,2,3,4\} & a_6 \in \{1,2\} \end{array}$$

Each dataset then has properties and true concepts.

A `MONK1` sample can be classified by checking if $a_5 = 1$ or if $a_1 = a_2$. This can be explained with the true concept:

$$\text{MONK1: } (a_1 = a_2) \vee (a_5 = 1)$$

A `MONK2` sample can be classified by checking if the sample contains exactly two attributes that are 1. This can be explained with the true concept:

$$\text{MONK2: } a_i = 1 \text{ for exactly two } i \in \{1, 2, \ldots, 6\}$$

A `MONK3` sample can be classified by checking if $a_5 = 1$ and $a_4 = 1$ or if $a_5 \neq 1$ and $a_2 \neq 3$. This can be explained with the true concept:

$$\text{MONK3: } (a_5 = 1 \wedge a_4 = 1) \vee (a_5 \neq 4 \wedge a_2 \neq 3)$$

`MONK1` training set has 124 samples, `MONK2` training set has 169 samples, and `MONK3` training set has 122 samples. The test datasets for `MONK1-3` all have 432 samples.

1.2 Difficulties with each dataset

There should be little difficulty classifying samples from `MONK1`. One could first check if $a_5 = 1$. If this is not true then one should check if $a_1 = a_2 = 1$, $a_1 = a_2 = 2$, or $a_1 = a_2 = 3$. This would require a maximum of 5 questions. The samples could be split into two subsets.

Classifying samples from `MONK2` will require checking a lot of combinations. This will probably be the most time-consuming decision tree, by asking at least 20 questions. This can be explained by finding the number of combinations of "is equal" questions

$$C(n, r) = \left(\begin{array}{c} n \\ r \end{array} \right) = \frac{n!}{(r!(n-r)!)} \Rightarrow \frac{5!}{(2!(5-2)!)} = 10$$

and then asking if the combinations are equal to 1, which would sum up to a maximum of 20 questions. The data can not be split into subsets.

Classifying samples from `MONK3` is probably the easiest classification problem. This only requires checking specific attributes, unlike `MONK1` and `MONK3` which require comparing multiple combinations of attributes. One could first ask if $a_4 = a_5 = 1$, and if this is not true then ask if $a_5 \neq 4$ and $a_2 \neq 3$. This would require a maximum of 4 questions. The data can be split into subsets.

This can be summarized as a list of datasets with increasing difficulty to learn, from easiest to the most difficult:

1. `MONK3`
2. `MONK1`
3. `MONK2`

2 Assignment 1

> The file `dtree.py` defines a function entropy that calculates the entropy of a dataset. Import this file along with the `monks` datasets and use it to calculate the entropy of the training datasets.

Entropy can be calculated according to the following equation.

$$\text{Entropy} = \sum_i -p_i \log_2 p_i$$

The higher the entropy, the more unpredictable it is.

Using the built-in function `entropy()` results in the entropy presented in Table 1.

Dataset	Entropy
MONK-1	1.0000
MONK-2	0.9571
MONK-3	0.9998

Table 1: Entropy of the monk training datasets.

3 Assignment 2

> Explain entropy for a uniform distribution and a non-uniform distribution, present some example distributions with high and low entropy.

A uniform distribution could be explained with fair dice or fair coin toss, while a non-uniform distribution could be presented with an unfair coin toss or weighted dice.

3.1 Coin toss

The probability to toss a head or a tail is $p_{\text{head}} = 0.5$ and $p_{\text{tail}} = 0.5$. The entropy is then calculated as follows.

$$\text{Entropy} = \sum_i -p_i \log_2 p_i = -0.5 \log_2(0.5) - 0.5 \log_2(0.5) = 1$$

A 50/50 chance can therefore be represented with entropy equal to 1.

3.2 Weighted dice

Every side of the dice should have a probability of $\frac{1}{6}$. However, weighted dice have weighted probabilities. An example of this could be $p_1 = ... = p_5 = 0.1$, while the six has a probability of $p_6 = 0.5$. The entropy for this scenario can be calculated as follows.

$$\text{Entropy} = \sum_i -p_i \log_2 p_i = -5 \cdot 0.1 \log_2 2(0.1) - 0.5 \log_2(0.5) = 2.16$$

Similarly, the entropy for a fair dice is 2.58. This means that an unfair dice is more *predictable* than a fair dice, due to the lower entropy.

4 Assignment 3

> Use the function `averageGain` (defined in `dtree.py`) to calculate the expected information gain corresponding to each of the six attributes. Note that the attributes are represented as instances of the class Attribute (defined in `monkdata.py`) which you can access via `m.attributes[0], ..., m.attributes[5]`. Based on the results, which attribute should be used for splitting the examples at the root node?

Dataset	a_1	a_2	a_3	a_4	a_5	a_6
MONK-1	0.0753	0.0058	0.0047	0.0263	0.2870	0.0008
MONK-2	0.0038	0.0025	0.0011	0.0157	0.0173	0.0062
MONK-3	0.0071	0.2937	0.0008	0.0029	0.2559	0.0071

Table 2: Expected information gain corresponding to each of the six attributes.

From Table 2, it is clear that for MONK-1 the obvious choice is that attribute a_5 should be used for splitting the examples at the root node.

For MONK-2, the information gain by splitting the examples at the root node is overall very small compared to MONK-1 and MONK-3. However, using attribute a_5 is marginally better than the rest.

For MONK-3, the best attribute to be used for splitting the examples at the root node is attribute a_2, but attribute a_5 also being a good choice.

5 Assignment 4

For splitting, we choose the attribute that maximizes the information gain. Looking at Eq.3 how does the entropy of the subsets, S_k, look like when the information gain is maximized? How can we motivate using the information gain as a heuristic for picking an attribute for splitting? Think about the reduction in entropy after the split and what the entropy implies.

The information gain of an attribute A, relative to a collection of samples S is defined as

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{k \in \text{values}(A)} \frac{|S_k|}{S} \text{Entropy}(S_k)$$

where S_k is the subset of examples in S for the attribute A has the value k.

The goal when building a binary classifier is to classify data points in the most effective manner possible. We want to split our decision tree at the attribute that minimizes uncertainty.

Entropy is utilized to measure uncertainty. Reducing entropy through splitting the dataset, based on attributes that minimize entropy, results in a maximum gain.

The function measures information gain as a reduction of entropy as $\text{Entropy}(S)$ in the function remains constant. Iterating across the subsets using different attributes and analyzing the subset $\text{Entropy}(S_k)$ will lead to a maximum expected information gain. The maximum information gain will be by splitting at the attribute that minimizes the subset $\text{Entropy}(S_k)$ and thus maximizing information gain.

6 Assignment 5

Build the full decision trees for all three Monk datasets using `buildTree`. Then, use the function `check` to measure the performance of the decision tree on both the training and test datasets. Compute the train and test set errors for the three Monk datasets for the full trees. Were your assumptions about the datasets correct? Explain the results you get for the training and test datasets.

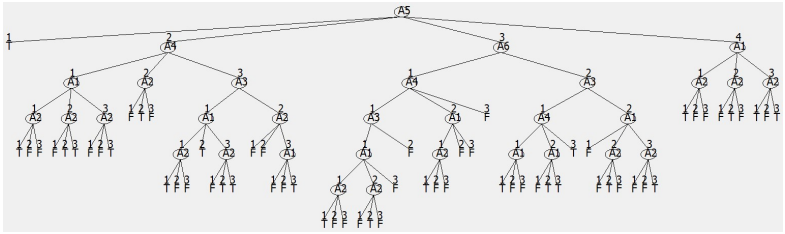
Decision trees are build with the function `buildTree`. The error for the classifier for both the train and the test datasets are then found using the `check` function. The `check` function gives the accuracy as the output, while the error is $E = 1 - accuracy$. The results are presented in Table 3.

	E_{train}	E_{test}
MONK-1	0.0000	0.1713
MONK-2	0.0000	0.3079
MONK-3	0.0000	0.0556

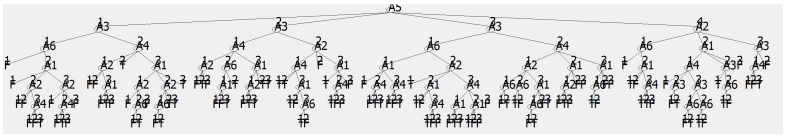
Table 3: Errors for the three Monk datasets for the full trees.

The decision trees are then displayed visually using `PyQt5`. These decision trees are displayed in Figure 1.

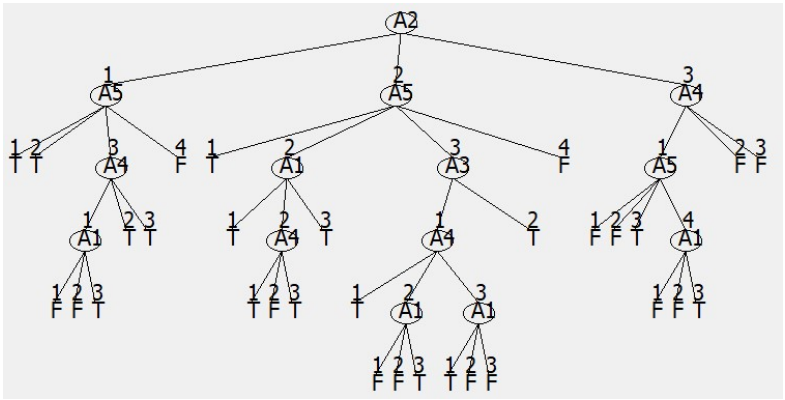
The assumptions made in the earlier question regarding the difficulty to learn the different datasets were correct. `MONK3` was the easiest and `MONK2` the hardest.



(a) `MONK1` decision tree



(b) `MONK2` decision tree



(c) `MONK3` decision tree

Figure 1: Decision trees for `MONK1`, `MONK2`, and `MONK3` classifier.

7 Assignment 6

> Explain pruning from a bias-variance trade-off perspective.

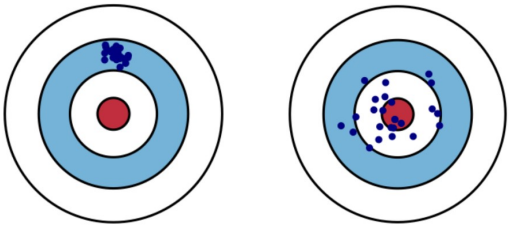
7.1 Bias-variance trade-off

By repeating the modeling many times and each time gathering a new set of training samples. The resulting models will have a range of predictions due to randomness in the underlying data set. These predictions can be described with error due to **bias** and error due to **variance**.

The error due to **bias** can be explained as the difference between the average expected prediction of our model and the correct value. In shorter terms, bias can be explained with the world **simpleness**.

The error due to **variance** can be explained as the variability of a model prediction for a given data point between different realizations of the model. As bias was compared with simpleness, variance can be explained with the word **tendency**.

A simple model would lead to high bias and low variance, while an advanced model would lead to low bias and high variance.



(a) High Bias - Low Variance

(b) Low Bias - High Variance

Figure 2: Bias-variance trade-off visualization.

By analyzing Figure 2 it is possible to visually understand the bias versus variance trade-off.

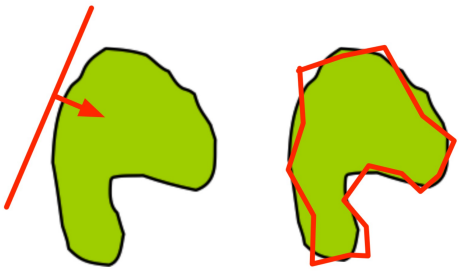
The presence of bias indicates that something is wrong with the model and algorithm. However, variance is also not desired in a model, but a model with high variance and low bias is at the very least able to predict well on average.

7.2 Bias-variance trade-off in decision trees

The **bias** of a classifier is the difference between its averaged estimated function and the true function.

The **variance** of a classifier is the expected divergence of the estimated prediction function from its average value. High-variance classifiers produce differing decision boundaries which are highly dependent on the training data.

By increasing the depth of the decision tree, the model is made more complex, which leads to higher variance and lower bias.



(a) High Bias Classifier

(b) High Variance Classifier

Figure 3: Bias-variance trade-off visualization.

By analyzing Figure 3 it is possible to visually understand the bias versus variance trade-off in a classifier.

7.3 Bias-variance trade-off in pruning

A decision tree that has not been pruned will be built perfectly for the training set. However, this will lead to a deep, and complex, decision tree with high variance. Such a decision tree will vary greatly in its classification depending on which training set was used.

With maximum variance, the bias is at its minimum, which leads to a model which captures all the features that were included in the training dataset.

By pruning the tree, and "cutting the branches", the variance is decreased. However, as explained earlier, this cannot be done without increasing the bias. This essentially minimizes the effect of over-fitting.

As a result, the decision tree will be more general and give a less varying classification. This comes at the cost of missing specific features included in the training dataset.

8 Assignment 7

> Evaluate the effect pruning has on the test error for the `monk1` and `monk3` datasets, in particular, determine the optimal partition into training and pruning by optimizing the parameter `fraction`. Plot the classification error on the test sets as a function of the parameter `fraction` $\in$ `{0.3, 0.4, 0.5, 0.6, 0.7, 0.8}`.
>
> Note that the split of the data is random. We, therefore, need to compute the statistics over several runs of the split to be able to draw any conclusions. Reasonable statistics include the mean and a measure of the spread. Do remember to print axes labels, legends, and data points as you will not pass without them.

In order to measure the effect pruning has on the datasets, the original datasets have to be split into a training and a validation set. This is because the validation should not be made on the actual dataset that is used for testing the model. The size of the training set is decided by the fraction, `fraction` $\in$ `{0.3, 0.4, 0.5, 0.6, 0.7, 0.8}`. Actually, this set of fractions is extended to include fractions `fraction` $\in$ `[0.01, 0.99]`.

When the fraction is set to 0.3, 30% of the original dataset is used for training, and 70% is utilized for validation. When the fraction is set to 0.8, 80% of the dataset is used for training, and 20% is used for validation.

8.1 How pruning is done

To evaluate the effect of pruning, a decision tree is built using a fraction of the original datasets (`monk1`, `monk2`, and `monk3`). Initially, the datasets are shuffled and then a fraction of this shuffled dataset is extracted to be used as a training dataset.

The created decision tree is then *pruned* using the `allPruned()` python function. This function returns a list of trees, each with one node replaced by the corresponding default class.

This new list of pruned trees is then looped through to find the most accurate one.

Due to the fact that the split of data is random, this pruning process is repeated a number of times to be able to extract statistical data. In this case, the process is repeated 200 times for all fractions.

8.2 The result of pruning

The mean errors of the decision trees after 200 repetitions of randomizing datasets and pruning is presented in Figure 4. The dashed lines represent the error achieved using the original decision trees, without any pruning. These values are presented earlier in Table 2.

Even though pruning of the `MONK2` decision tree was not requested, this is also presented in Figure 4.

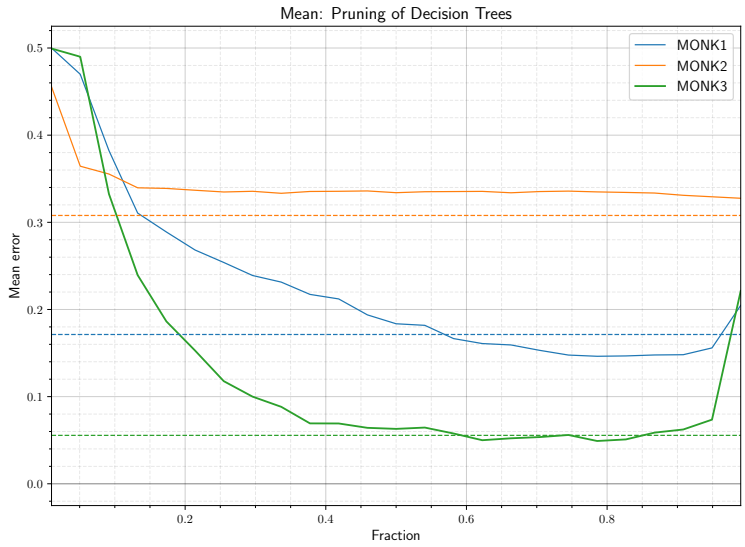


Figure 4: Evaluation of pruning by presenting the mean values of the error (solid lines), compared with the original decision tree error (dashed lines).

For both the `MONK1` and `MONK3` datasets, better performance is achievable with pruning. However, this is only possible when the fraction is sufficiently high. This can be seen in Figure 4 when the solid lines become smaller than the dashed lines.

Due to the fact that `MONK2` requires a complex model for classification, any pruning actually decreased performance.

The standard deviation of pruning the different decision trees is relatively stable. Standard deviation for each dataset and fraction is presented in Figure 5.

Finally, the values presented visually in Figures 4 and 5, is also presented in Table 4.

Fraction	0.3	0.4	0.5	0.6	0.7	0.8
MONK-1_{mean}	0.2345	0.2092	0.1883	0.1665	0.1561	0.1506
MONK-1_{std}	0.0399	0.0422	0.0442	0.043	0.0392	0.0414
MONK-2_{mean}	0.3368	0.3366	0.3363	0.3357	0.3355	0.3333
MONK-2_{std}	0.0226	0.0163	0.0155	0.0148	0.014	0.0113
MONK-3_{mean}	0.1019	0.0759	0.0627	0.0516	0.0484	0.0547
MONK-3_{std}	0.0581	0.0464	0.0404	0.0277	0.0308	0.033

Table 4: Evaluation of pruning by presenting the mean values of the error, as well as the standard deviation.

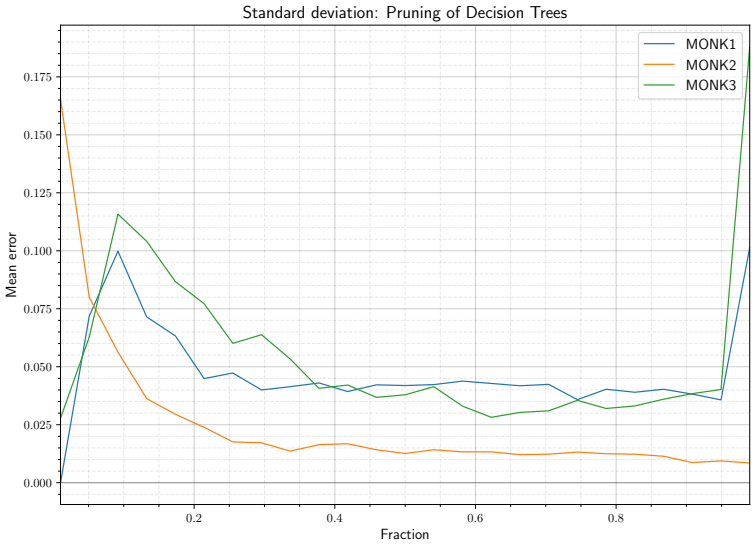


Figure 5: Evaluation of pruning by presenting the standard deviation of the error.