Machine Learning

Lab 1: Decision Trees

Linus Groß
Daniel Mensah

Assignment 0

Each one of the datasets has properties which makes them hard to learn. Motivate which of the three problems is most difficult for a decision tree algorithm to learn.

With the known true concept, the easiest to learn is probably the set MONK-1 because it just depends on 3 attributes and having a conjunction as a connection. However, the property $a_1 = a_2$ is more difficult to test, because there are 3 possibilities for this equation to be true. With the property $a_5 = 1$ there is a very easy equation to check which can terminate the tree quite early.

The set MONK-2 is the most difficult for a decision tree algorithm to learn because all six attributes have to be considered. Thus, the tree needs to have all six attributes for a decent decision.

The MONK-3 set depends like set MONK-1 on 3 attributes, but the tree will at least have a depth of 2 attributes. Furthermore, the attribute a_5 is considered in both equation and especially in the inequality, which means all 3 other branches, where the inequality is true, must be continued.

Assignment 1

The file dtree.py defines a function entropy which 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.

The following table shows the entropy of the monk training datasets. MONK-1 has the highest entropy, which makes it the most unpredictable dataset. However, the entropy of the other datasets is just slightly lower.

Dataset	Entropy
Monk-1	1
Monk-2	0,957117
Monk-3	0,999806

Assignment 2

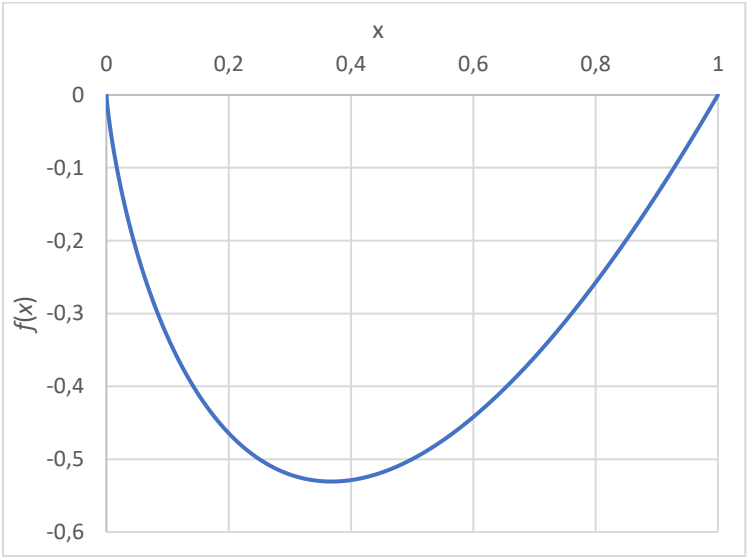
Explain the entropy for a uniform distribution and a non-uniform distribution, present some example distributions with high and low entropy.

Looking at the formula for the entropy

$$E(S) = -\sum_i p_i \cdot \log_2(p_i)$$

we have to consider the negative sum of the function $f(x) = x \cdot \log_2(x)$ with x equal to the different probabilities p_i.

The following graph show the function $f(x)$ for $0 \leq x \leq 1$. If the probability x is near to 0 or 1 the entropy tends to 0. This makes sense, since a high and respectively a low probability leads to a great predictability and thus to a low entropy.

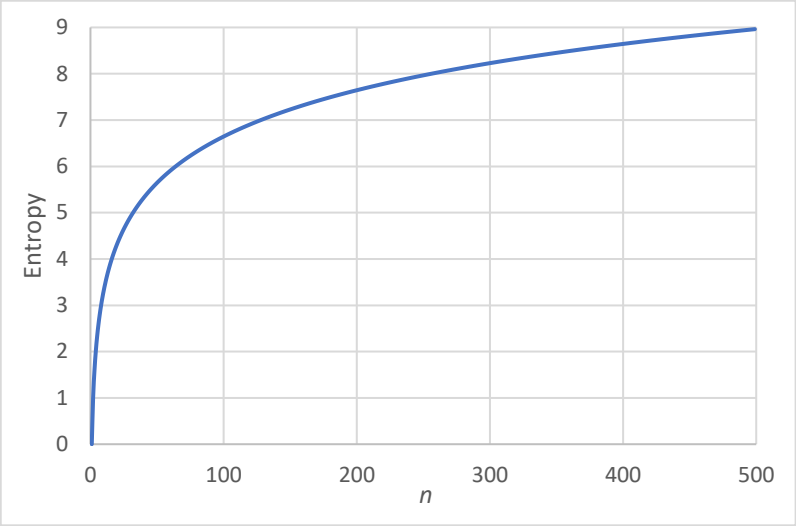


For a uniform distribution, every of the n events has the same probability $p_i = p = {}^1/_n$ and the formula for the entropy simplifies to:

$$E(S) = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i)$$

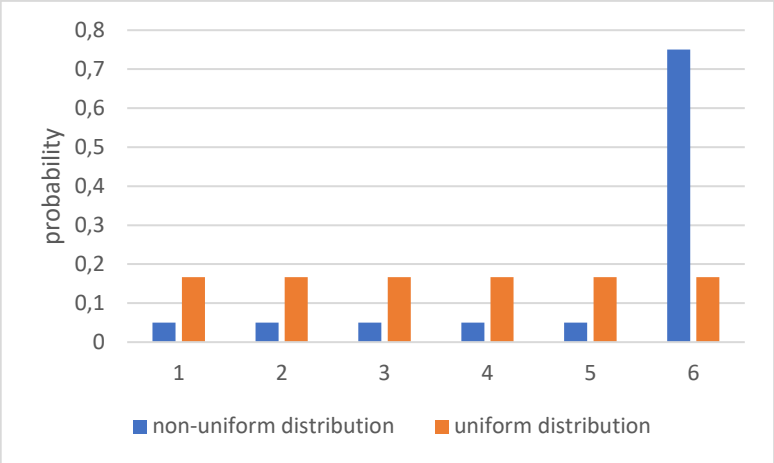
$$E(S) = -\sum_{i=1}^{n} \frac{\log_2\left(\frac{1}{n}\right)}{n} = -\log_2\left(\frac{1}{n}\right)$$

As the following graph shows, the entropy follows a logarithmical growth for a uniform distribution. The entropy reaches to quite high values, since a single event gets very unpredictable when the total amount of events n rises.
For example, rolling a dice has a uniform distribution with $n = 2$ and leads to an entropy of approximately 2,585.



On the other hand, non-uniform distributions can lead to much lower entropies. Considering, that some events are more likely than other events in these distributions, the predictability is much higher in contrast to a uniform distribution, which however leads to a lower entropy.
As a example we consider a manipulated dice with the following probability distribution:

Event	Non-uniform distribution	Uniform distribution
1	5%	16,67%
2	5%	16,67%
3	5%	16,67%
4	5%	16,67%
5	5%	16,67%
6	75%	16,67%
Entropy	1,3917	2,585



On the right graph we see, that the event 6 is much more likely in comparison to the uniform distribution. Using the entropy-formula we will get the entropy 1,39 for the manipulated non-uniform distribution, which is way lower than the entropy of the uniform dice (2,585).

So you can say that generally that uniform distributions maximize the entropy while non-uniform distributions on the other hand have always a lower entropy.

Assignment 3

Use the function averageGain to calculate the expected information gain corresponding to each of the six attributes. Based on the results, which attribute should be used for splitting the examples at the root node?

The following table shows the information gain for each attribute at the root node. A value corresponds to a high information gain, which means we lower the entropy as much as possible by asking for the give value.

Dataset	a_1	a_2	a_3	a_4	a_5	a_6
MONK-1	0.0752726	0.0058384	0.0047075	0.026311	0.2870307	0.00075786
MONK-2	0.003756177	0.002458499	0.00105614	0.015664247	0.01727717	0.006247622
MONK-3	0.007120868	0.2937361735	0.000831114	0.002891817	0.2559117246	0.00707702607

With the give true concepts, it makes sense to take a_5 as the first attribute in MONK-1. In MONK-2 all attributes need to be considered and it seems like a_5 is the best attribute. The true concept behind MONK-3 suggests that a_5 is the best attribute to ask for first, thus it is used in both equations. Yet the algorithms suggest a_2 as the best attribute to ask. The additional noise could be the reason for this decision.

Assignment 4

For splitting we choose the attribute that maximizes the information gain, Eq.3. 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 reduction in entropy after the split and what the entropy implies.

Maximizing the information gain means that we want to lower the entropy as much as possible. This means, that the term

$$\sum_{k \epsilon \text{values}(A)} \frac{|S_k|}{|S|} Entropy(S_k)$$

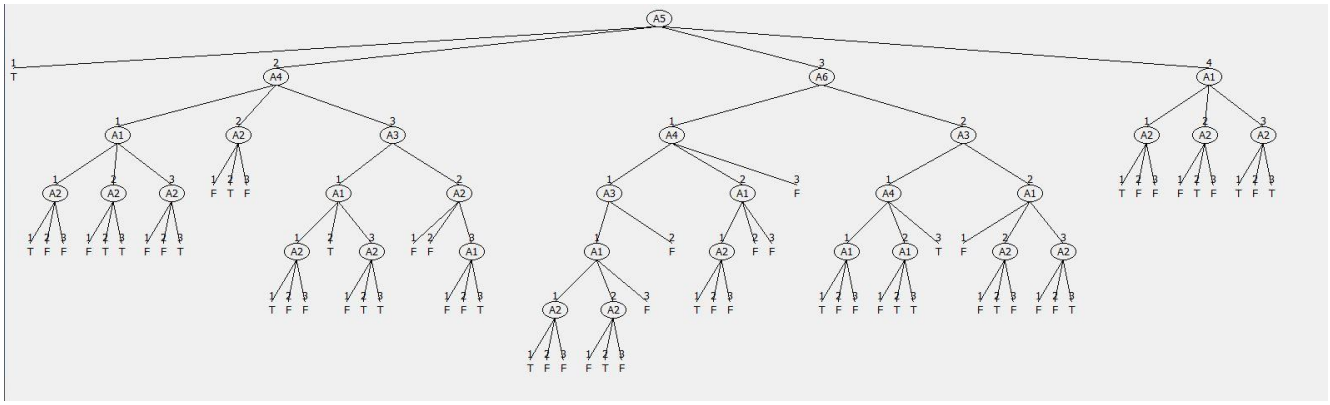
should be as low as possible. This implies that the sum of the weighted entropy of the given subset is as low as possible, which means it is very predictable. Consequently, the information gain is high, if the entropy of a subset is very high, we gain much information about the total set and are able to reduce the entropy of the remaining set as much as possible.

Assignment 5

Build the full decision trees for all three Monk datasets using buildTree.

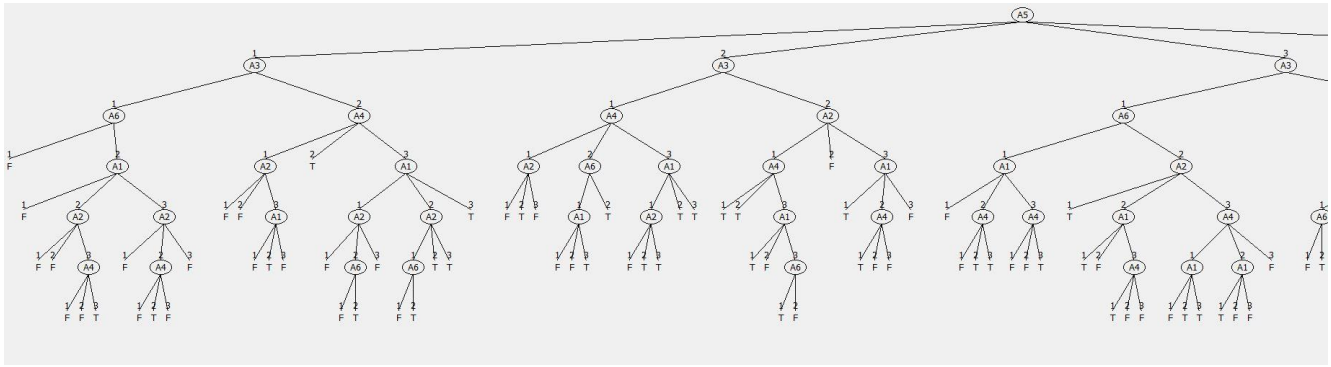
Monk-1:

As expected for the case of $a_5 = 1$ the tree terminates quickly. But however, the algorithm decides to check the important attributes a_1 and a_2 mostly on the end of the tree, which leads to more nodes than needed



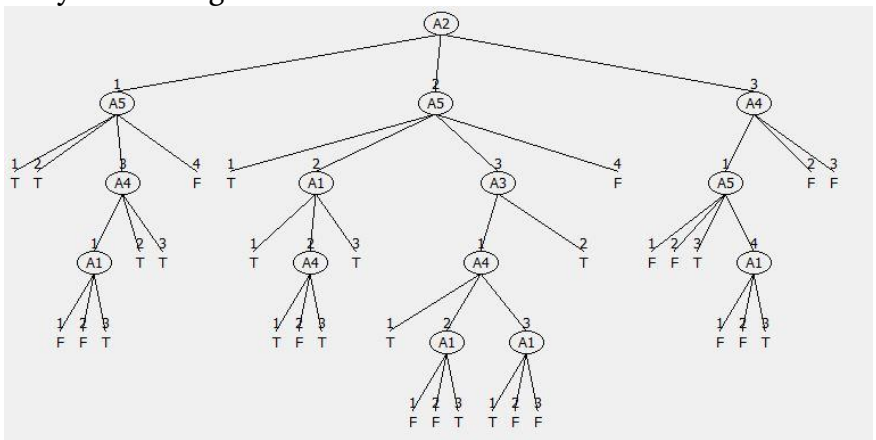
Monk-2:

Since all six attributes need to be checked the tree grows as expected very large and doesn't even fit the screen. Thus this is the hardest problem for the decision tree.



Monk-3:

For the Monk-3 dataset the algorithm grew the smallest tree. This is due to the true concept, which was quite easy for the algorithm to detect.



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.