DD2421 Machine Learning - Lab 1: Decision Trees Python version

Marina Herrera, Thi Thuy Nga Nguyen

1 MONK datasets

This lab uses the artificial MONK dataset from the UC Irvine repository. The MONK's problems are a collection of three binary classification problems MONK-1, MONK-2 and MONK-3 over a six-attribute discrete domain. The attributes $a_1, a_2, a_3, a_4, a_5, a_6$ may take the following values:

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

Consequently, there are 432 possible combinations of attribute values. The *true* concepts underlying each MONK's problem are given by table 1.

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.

The data consists of three separate datasets MONK-1, MONK-2 and MONK-3. Each dataset is further divided into a training and test set, where the first one is used for learning the decision tree, and the second one to evaluate its classification accuracy (see table 2). The datasets are available in the file `monkdata.py`. In particular, six variables are defined which contain the datasets: `monk1`, `monk1test`, `monk2`, `monk2test`, `monk3` and `monk3test`. Each dataset is a sequence (more precisely, a *tuple*) of instances of the class `Sample`, defined in the same file.

You can access the data in your own Python scripts by importing the `monkdata.py` file as a module like this:

```
import monkdata as m
```

This makes the variable `m` a shorthand for the module so that you can access the datasets by writing `m.monk1`, etc.

Table 1: True concepts behind the MONK datasets

MONK-1	$(a_1 = a_2) \vee (a_5 = 1)$
MONK-2	$a_i = 1$ for exacly two $i \in \{1, 2, \ldots, 6\}$
MONK-3	$(a_5 = 1 \wedge a_4 = 1) \vee (a_5 \neq 4 \wedge a_2 \neq 3)$

MONK-3 has 5% additional noise (misclassification) in the training set.

Table 2: Characteristics of the three MONK datasets

Name	# train	# test	# attributes	# classes
MONK-1	124	432	6	2
MONK-2	169	432	6	2
MONK-3	122	432	6	2

2 Entropy

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 entropies of all datasets are approximately 1 which is maximum value. That implies there is no information about the classification of the data.

Dataset	Entropy
MONK-1	1
MONK-2	0.95711743
MONK-3	0.99980613

Assignment 2: Explain entropy for a uniform distribution and a non-uniform distribution, present some example distributions with high and low entropy.

Consider discrete case of uniform distribution, let $X = \{x_1, x_2, \ldots, x_n\}$ be a random variable such that $p(x_i) = \mathbb{P}(X = x_i) = 1/n$, $(i = 1, 2, \ldots, n)$. The entropy of X is given by

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2(p(x_i)) = -\sum_{i=1}^{n} \frac{1}{n} \log_2\left(\frac{1}{n}\right) = \log_2(n)$$

In another case of non-uniform distribution, let $X = \{y_1, y_2, \ldots, y_n\}$ be random variable of a multinomial distribution with the number of trials n

and event probabilities $p_1, p_2, \ldots, p_n$ ($\sum_{i=1}^{n} p_i = 1$). We see that $p(y_i) = p_i$. The entropy of Y is given by

$$H(Y) = -\sum_{i=1}^{n} p(y_i) \log_2(p(y_i)) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

We have $H(Y) \leq H(X)$, the equals occurs when $p_i = 1/n$ for all $i = 1, 2, \ldots, n$. For example, take $n = 2$ and $p_1 = 0.8$, $p_2 = 0.2$ the entropies $H(X) = 1$ and $H(Y) = 0.72$. If $p_1 = 0.99$, $p_2 = 0.01$ the entropy $H(Y) = 0.08$.

3 Information Gain

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

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{k \in \text{values}(A)} \frac{|S_k|}{|S|} \text{Entropy}(S_k) \tag{1}$$

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

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?

Information Gain

Dataset	a_1	a_2	a_3	a_4	a_5	a_6
`MONK-1`	0.07527	0.00584	0.00471	0.02631	0.28703	0.00076
`MONK-2`	0.00376	0.00246	0.00106	0.01566	0.01728	0.00625
`MONK-3`	0.00712	0.29374	0.00083	0.00289	0.25591	0.00708

Based on the results, the attribute which maximizes the information gain should be used for splitting the examples at the root node. In particular, the attribute A5 in dataset MONK1, A5 in MONK2 and A2 in MONK3 should be roots.

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.

The below tables show the entropy of each subset S_k in Eq.3. We can see that at the attribute maximized the information gain (bold) there is at least one smaller entropy.

At each level of the tree, maximizing the information gain of the attributes minimized the expected entropy which is obtained after slipping examples S on the attribute A. The smaller the entropy at a node, the more information is known about the classification of data at this level of the tree. Therefore, by using the smallest entropy to slip data we get the best classification at this level of the tree.

However, this way does not guarantee an optimal solution for whole tree. It uses the locally best attribute to split the dataset. Sometime by selecting not the best but the good attribute implies better final tree.

Information Gain and Entropy of subsets of `MONK-1`

Attribute	Expected	S_1	S_2	S_3	S_4
a_1	0.92473	0.89445	0.99836	0.87796	
a_2	0.99416	0.98523	0.99836	0.99706	
a_3	0.99529	0.99573	0.99481		
a_4	0.97369	0.95871	0.96124	0.99961	
a_5	**0.71297**	0	0.93832	0.94808	0.90818
a_6	0.99924	0.99908	0.99938		

Information Gain and Entropy of subsets of `MONK-2`

Attribute	Expected	S_1	S_2	S_3	S_4
a_1	0.95336	0.9183	0.96215	0.9806	
a_2	0.95466	0.92994	0.9779	0.95261	
a_3	0.95606	0.96952	0.94307		
a_4	0.94145	0.85241	0.97512	0.99048	
a_5	**0.93984**	0.91035	1	0.96334	0.87796
a_6	0.95087	0.9183	0.98306		

Information Gain and Entropy of subsets of `MONK-3`

Attribute	Expected	S_1	S_2	S_3	S_4
a_1	0.99269	0.99498	0.98371	1	
a_2	**0.70607**	0.9183	0.82961	0.37765	
a_3	0.99898	0.99983	0.998		
a_4	0.99691	0.9982	0.99199	1	
a_5	0.74389	0.89604	0.90717	0.98523	0.20559
a_6	0.99273	0.98982	0.99545		

4 Building Decision Trees

Split the `monk1` data into subsets according to the selected attribute using the function `select` (again, defined in `dtree.py`) and compute the information gains for the nodes on the next level of the tree. The remaining attributes (A1, A2, A3, A4, A6) should be tested for these nodes.

The nodes at this level of the tree is found by maximizing the information gain of each subset on each value of root. In particular, in MONK1 the nodes corresponding to the values of A5 ($A5 = [1, 2, 3, 4]$) are null (leaf), A4, A5, A1; in MONK2 they are A3, A3, A3, A2; and in MONK3 with root A2 $= [1, 2, 3]$ they are A5, A5, A4. The leafs (or outcomes $= \{+, -\}$ are determined by using `mostCommon` function.

All results are the same as the one gotten from ID3 used in `buildTree(data, m.attributes)`.

Add figure .....

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.
For example to built a tree for `monk1` and compute the performance on the test data you could use

```
import monkdata as m
import dtree as d

t=d.buildTree(m.monk1, m.attributes);
print(d.check(t, m.monk1test))
```

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.

	E_{train}	E_{test}
MONK-1	0	0.1712963
MONK-2	0	0.30787037
MONK-3	0	0.05555556

In each data, since the classifier based on full tree which fitted all training sets the performance on training dataset is 1 and the error is thus 0. These errors on the test sets are quite large in MONK2 and small in MONK3 but all are larger than 0. This can be an overfitting of the full tree.

5 Pruning

Assignment 6: Explain pruning from a bias variance trade-off perspective.

The full tree decision classifier has low bias but high variance error. The variance is increasing as the number of the nodes in the tree. Pruning is to reduce the variance of the tree.

However, when the variance is decreasing the bias is increasing. Pruning should reduce the size of a learning tree without reducing predictive

accuracy. One technique is reduced error pruning in which the error of the sub-trees is minimized on a validation set not used in training. The error is used in the pruning is the same as MSE which is decomposed $MSE = Bias^2 + Variance$.

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 includes mean and a measure of the spread. Do remember to print axes labels, legends and data points as you will not pass without them.

Add figure .....