COMP9417 - Machine Learning
Tutorial: Tree Learning

Question 1. Expressiveness of Trees

Give decision trees to represent the following Boolean functions, where the variables `A`, `B`, `C` and `D` have values `t` or `f`, and the class value is either `True` or `False`. Can you observe any effect of the increasing complexity of the functions on the form of their expression as decision trees ?

(a) $A \wedge \neg B$

(b) $A \vee [B \wedge C]$

(c) A XOR B

(d) $[A \wedge B] \vee [C \wedge D]$

Question 2. Decision Tree Learning

(a) Assume we learn a decision tree to predict class Y given attributes A, B and C from the following training set, with no pruning.

A	B	C	Y
0	0	0	0
0	0	1	0
0	0	1	0
0	1	0	0
0	1	1	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	0	1
1	1	1	0
1	1	1	1

What would be the training set error for this dataset ? Express your answer as the number of examples out of twelve that would be misclassified.

(b) One nice feature of decision tree learners is that they can learn trees to do *multi-class* classification, i.e., where the problem is to learn to classify each instance into exactly one of $k > 2$ classes.

Suppose a decision tree is to be learned on an arbitrary set of data where each instance has a discrete class value in one of $k > 2$ classes. What is the maximum training set error, expressed as a fraction, that any dataset could have ?

Question 3. ID3 Algorithm

Here is small dataset for a two-class prediction task. There are 4 attributes, and the class is in the rightmost column (homeworld). Look at the examples. Can you guess which attribute(s) will be most predictive of the class ?

species	rebel	age	ability	homeworld
pearl	yes	6000	regeneration	no
bismuth	yes	8000	regeneration	no
pearl	no	6000	weapon-summoning	no
garnet	yes	5000	regeneration	no
amethyst	no	6000	shapeshifting	no
amethyst	yes	5000	shapeshifting	no
garnet	yes	6000	weapon-summoning	no
diamond	no	6000	regeneration	yes
diamond	no	8000	regeneration	yes
amethyst	no	5000	shapeshifting	yes
pearl	no	8000	shapeshifting	yes
jasper	no	6000	weapon-summoning	yes

You probably guessed that attributes 3 and 4 were not very predictive of the class, which is true. However, you might be surprised to learn that attribute "species" has higher information gain than attribute "rebel". Why is this ?

Suppose you are told the following: for attribute "species" the Information Gain is 0.52 and *Split Information* is 2.46, whereas for attribute "rebel" the Information Gain is 0.48 and *Split Information* is 0.98.

Which attribute would the decision-tree learning algorithm select as the split when using the *Gain Ratio* criterion instead of Information Gain ? Is Gain Ratio a better criterion than Information Gain in this case ?

Question 4. Working with Decision Trees

In `utils.py` you will find the implementation of the function `visualize_classifier` which allows us to visualise any classifier that has a `predict` method. You can use this function as a black box throughout. The `sklearn.datasets.make_blobs` function gives us a quick way to create toy data for classification. In the following we'll create a 3 class classification problem:

```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=120,                          # total number of samples
                  centers=[[0,0], [0,2], [-2,1]],         # cluster centers of the 3 classes
                  random_state=123,                       # reproducibility
                  cluster_std=0.6)                        # how spread out are the samples
    from their center

plt.scatter(X[:, 0], X[:, 1], c=y, s=50)                  # scatter with color=label
plt.show()
```

(a) Use `sklearn.neighbors.KNeighborsClassifier` and `sklearn.tree.DecisionTreeClassifier` objects to demonstrate the `visualize_classifier` function. Explain the differences between the decision boundaries of the two classifiers.

(b) Another way to visualise a tree can be done by running:

```
    from sklearn import tree

    fig, axes = plt.subplots(1, 1,figsize = (3,3), dpi=300)
    tree.plot_tree(model1,         # fitted decision tree
    feature_names=['f1', 'f2'],    # names for features
```

```
    class_names=['t1', 't2', 't3'], # names for class labels
    filled=True)
    plt.show()

```

Explain what is going on in the resulting plot. What do the colors represent? What does the `value` argument tells us? What about `entropy`?

(c) Generate data using the following code:

```
    X, y = make_blobs(n_samples=500,
                  centers=[[0,0], [0,2], [-2,1], [-2,2], [3,3], [1,-2]],
                  random_state=123,
                  cluster_std=0.6)

```

Then fit a decision tree (using information gain for splits) with `max_depth` set to $1, 2, \ldots, 12$ and visualize the classifier (use a 3×4 grid). What do you observe? Why do you think decision trees are described as performing 'recursive partitioning'?