UiT

THE ARCTIC UNIVERSITY OF NORWAY

4. Decision Trees

FYS-2021 Exercises

Department of Physics and Technology
Faculty of Science and Technology

Problem 1

In this problem, you will implement a Classification tree from scratch. For simplicity, you can assume continuous features, 2 classes and a binary split. We suggest adapting the pseudo code provided into an object-oriented python approach, where each node in the tree is represented as an object.

```
GenerateTree(X)
    If NodeEntropy(X)< θ_I /* equation 9.3 */
        Create leaf labelled by majority class in X
        Return
    i ← SplitAttribute(X)
    For each branch of x_i
        Find X_i falling in branch
        GenerateTree(X_i)

SplitAttribute(X)
    MinEnt← MAX
    For all attributes i = 1,...,d
        If x_i is discrete with n values
            Split X into X_1,...,X_n by x_i
            e ← SplitEntropy(X_1,...,X_n) /* equation 9.8 */
            If e<MinEnt MinEnt ← e; bestf ← i
        Else /* x_i is numeric */
            For all possible splits
                Split X into X_1,X_2 on x_i
                e←SplitEntropy(X_1,X_2)
                If e<MinEnt MinEnt ← e; bestf ← i
    Return bestf
```

Figure 1: Figure taken from the machine learning book (Fig. 9.3) showing pseudocode of a decision tree

(1a) Implement a function that computes the binary entropy:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \cdot \log_2 \big(p(x_i)\big). \tag{1}$$

The class should take an array of labels, and return the total entropy for the labels.

(1b) Implement a function that takes one given feature of a dataset, and finds the best split (the split that minimises the entropy) for the data. It is common to iterate over the data feature values, and calculate the entropy for the subsets that are larger or smaller (or equal) than the current value in the iteration. return the split that minimises the entropy and the corresponding entropy for the split.

(1c) Implement a function that takes a dataset ($n_{samples} \times n_{features}$) and finds the single best split (the split with the least entropy) across all features. That is the function `find_best_split`. Return the best split found.

(1d) Using the functions you implemented above, write your own version of the Classification Tree algorithm, using recursion. Remember to include a parameter specifying the maximum depth of the tree to prevent overfitting.

(1e) Test your implementation on the datasets in `blobs.csv` and `flame.csv`. Plot the data, and the regions found by the tree.

(1f) Draw or plot a diagram similar to Figure 9.6 in the book, illustrating the rules learned by the tree.

The file `tictac.csv` specifies the optimal move for a specified tic-tac-toe board layout. Each row corresponds to a different board. The first element is the label, which indicates the best move to make (0 = top left, …, 8 = bottom right). The rest of the row contains the (flattened) current board layout, with $1 = \times$, $-1 = \bigcirc$, and $0 =$ blank.

(1g) Split the data into training- and test-sets. Train your classification tree using the training set, and test it using the testing set. How is the generalization performance?

(1h) **Bonus:** Use your classification tree to create a program where the user can play tic-tac-toe against the computer.

Problem 2

(2a) Implement your own version of the Regression Tree algorithm. If you have implemented the Classification Tree, all you need to change is the impurity measure, and the creation of leaf-nodes. Remember to include a parameter specifying the maximum depth of the tree, to prevent overfitting.

(2b) Test your implementation using the dataset in `global-temperatures.csv`. Plot the predicted line together with the data for different values of the max-depth parameter. What do you observe? Plot/sketch a diagram of the resulting tree when the maximum depth equals 3.

(2c) Test your implementation using the dataset in `auto-mpg.csv`. Generate the tree for different maximum depths, and compute R^2 for the predictions made by each of the trees. Plot R^2 as a function of depth. The resulting plot should be monotonically increasing. Why is this the case?