

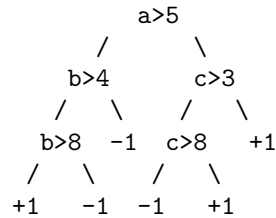
### Homework 3: Decision trees

1. [1pt] Given the dataset (where each sample has 3 feature values: a, b, and c) below, compute the gini impurity for the condition  $S1 : a > 10$ . Please show the estimations of  $Pr(class = +1|a > 10)$  and  $Pr(class = -1|a > 10)$ . If you do not show these two but a final result, you will get no point.

Sample Number	feature a	feature b	feature c	class
1	12	3	5	+1
2	4	7	6	+1
3	5	4	8	+1
4	6	6	7	+1
5	7	5	1	-1
6	8	2	2	-1
7	9	6	3	-1
8	11	8	1	-1

- 2 samples in total satisfy threshold  $S1 : a > 10$
  - $Pr(class = +1|a > 10) = 0.5$ . Since 1 sample (#1) belongs to class +1 given  $a > 10$
  - $Pr(class = -1|a > 10) = 0.5$ . Since 1 sample (#8) belongs to class -1 given  $a > 10$
  - $g(a > 10) = 1 - 0.5^2 - 0.5^2 = 0.5$
2. [1pt] Do the same for a condition  $S2 : a \leq 5$ . Again, intermediate steps need to be shown.
    - 2 samples in total satisfy threshold  $S2 : a \leq 5$
    - $Pr(class = +1|a \leq 5) = 1$ . Since 2 samples (#2,#3) belongs to class +1 given  $a \leq 5$
    - $Pr(class = -1|a \leq 5) = 0$ . Since 0 sample belongs to class -1 given  $a \leq 5$
    - $g(a \leq 5) = 1 - 1^2 - 0^2 = 0$
  3. [2pt] Based on the results from the two problems above, compute the expectation for gini impurity for the feature  $a$  and the threshold 5. Please show the estimations of the probabilities of both conditions, i.e.,  $P(a > 5)$  and  $P(a \leq 5)$ . If you just show a final result, no point.
    - $g(a > 5) = 1 - Pr^2(class = +1|a > 5) - Pr^2(class = -1|a > 5) = 1 - 0.33^2 - 0.67^2 = 0.44$
    - $g(a \leq 5) = 1 - 1^2 - 0^2 = 0$
    - $P(a > 5) = 6/8 = 0.75$
    - $P(a \leq 5) = 2/8 = 0.25$

- $E(a, 5) = P(a > 5)g(a > 5) + P(a \leq 5)g(a \leq 5) = 0.75 * 0.44 + 0.25 * 0 = 0.33$
4. [1pt] Using the decision tree below, decide the classification outcomes for all samples in Problem 1. Left branch is True and right branch is False. Present your result as a two-column table.



By two-column, this is it:

sample	prediction
1	-1
2	+1
3	+1
4	-1
5	-1
6	-1
7	-1
8	-1

## Programming [5pts plus 3 bonus points]

Please work from a provided file `hw3.py` and just upload this file.

### 5. [2pt] Estimating Gini impurity

Turn the computational steps you did for Problems 1 and 2 into a function `estimate_gini_impurity` that computes the Gini impurity given

- values of a feature on a set of samples,
- a threshold,
- labels corresponding to the samples,
- and, the polarity of comparison.

The variable `polarity` takes value from `operator.lt` and `operator.ge` in Python standard module `operator`. They are functions applied to two operands, e.g.,

```

>>> import operator
>>> operator.lt(5,4)
False
>>> operator.lt(4,5)
True
>>> operator.ge(4,5)
False
>>> operator.ge(5,4)
True
>>> operator.ge(5,5)
True

```

Note that there may be no sample satisfying a comparison expression at all. In such cases, when computing  $Pr(class = s|S)$ , it will result in division by zero. To avoid that, please set  $Pr(class = s|S)$  to a very small number `numpy.finfo(float).eps` when it is zero. AND, in such a case, the gini impurity should be 1 because  $Pr(class = c|S)$  is zero for either class.

6. [3pt] **The expectation of Gini impurity.** Turn the computational steps you did in Problem 3 into the function `estimate_gini_impurity_expectation` that computes the expectation of Gini impurity given
  - values of a feature on a set of samples,
  - a threshold,
  - and, labels corresponding to the samples.

7. [bonus, 3pt] **The split of a node** Each node of a decision tree is a comparison on a feature against a threshold. To determine the feature and the threshold, a common practice is a grid search. Search over every feature, against a set of thresholds. In the end, pick the combination of feature and threshold that minimize the expectation of Gini impurity. The function to implement in this problem is `grid_search_split_midpoint`.

There are many ways to establish the thresholds. Here we use a common practice that for each feature, the thresholds are the midpoints of any two consecutive values. For example, if the values on a feature is `[5,4,2,1,3]`, then the thresholds are `[1.5, 2.5, 3.5, 4.5]`. Note that because values on features differ (e.g., in feature 1, `[1,2,3,4,5]` and in feature 2, `[0.1, 0.2, 0.9, 0.1, -0.8]`), thresholds for features differ. To help you start, the thresholds for each feature are constructed in the function `grid_search_split_midpoint` in `hw3` already. So you can directly make use of the variable `thresholds`, each column of which corresponds to a feature dimension.

The function shall return 3 variables:

- a. **grid:** 2D numpy array, the expectations of Gini impurity. Each column corresponds to one feature and each row corresponds to a threshold. Note that each feature has its own thresholds.

- b. **best\_feature**: int, the index of the column containing the minimal value in **grid**.
- c. **best\_threshold**: float, the threshold of the **best\_feature** that minimizes the value in **grid**.

Given the grid, to find the corresponding threshold index and feature index, use Numpy's **argmin** and **unravel\_index**. Something like this:

```
(best_threshold_index, best_feature_index) = \
    numpy.unravel_index(numpy.argmin(grid, axis=None), grid.shape)
```

Then convert the threshold index into actual threshold value.

### How to test your submission

Test cases for all functions are provided in **doctest**. You can visually compare the outputs of your functions. Or you can run the **hw3.py** script as **python3 hw3.py** in a system command shell or **%run hw3.py** in a Jupyter hub cell. If you see no error, then all your functions are correct.

In addition to the doctest cases, an additional function **you\_rock** is provided to test your bonus problem solution. It generates many random datasets and compares the results from your **grid\_search\_split\_midpoint** function against those by scikit-learn's **tree** module. Because of the slight implementation difference, your **best\_feature** and **best\_threshold** may not be always the same as scikit-learn's'. But you should see they are the same for 70% of the cases or more, when calling it as **you\_rock(1000, 100, 3)** (1000 samples, feature values ranging from 1 to 100, 3 features).

### Hints for programming:

1. Instead of for-loops, take advantage of numpy's array-wise operations. For example,

```
numpy.array([1,2,3,4]) > 3
```

returns an array containing the comparison result between every element of the array against the threshold 3: **array([False, False, False, True])**

2. You can **logical\_and** two arrays to find elements satisfying both conditions (instead of using for-loops to check both conditions over elements):

```
numpy.logical_and(
    numpy.array([1,2,3,4]) > 2,
    numpy.array([+1,-1,+1,-1]) == +1
)
```

shall return **array([False, False, True, False])**

There are many other logic operations in Numpy. [Click here to see.](#)

3. You can sum over a Boolean array to count how many elements are True or False.

```
sum(numpy.array([True, False, True]))
```

shall return 2.

## How to submit

For hand computation part, upload one PDF file. For programming part, upload your edited `hw3.py`. Do NOT delete contents in the template for programming. Be sure that in your submitted `hw3.py`, you include only module importing and function definitions above the `if __name__ == "__main__":` line.