

DECISION TREE, INDUCTIVE BIAS AND HYPERPARAMETERS

Adhilsha A and Aritra Mukhopadhyay (Group 2)

CS460: Machine Learning, School of Computer Sciences,
National Institute of Science Education and Research, Bhubaneswar, India

January 23, 2023

PART I: DECISION TREE

| | | |
|----------|--------------------------------------|----------|
| 1 | Decision tree | 5 |
| 1.1 | Introduction | 5 |
| 1.2 | Features, Feature values, and Labels | 6 |
| 1.3 | Structure | 7 |
| 1.4 | Example | 8 |
| 1.5 | Algorithm related factors | 9 |
| 1.6 | Sample Pseudo-Code | 10 |
| 1.7 | Calculating the impact | 11 |

PART II: INDUCTIVE BIAS

- 1 Inductive Bias 13**
 - 1.1 Introduction and example 13
 - 1.2 Inductive bias of shallow decision tree 14

PART III: HYPERPARAMETER

- 1 Hyperparameter 16**
 - 1.1 Models, parameters and Hyperparameters 16
 - 1.2 Example 17
 - 1.3 Hyperparameter tuning 18

Part I

DECISION TREE

DECISION TREE

INTRODUCTION

- ▶ **Decision tree** is one of the classic models among the **supervised learning** algorithms in Machine Learning. As the name suggests, the decision tree helps to create a structure in tree form that helps to predict/reach an answer by taking a series of decisions modelled by the tree.
- ▶ This can be understood with an analogy. A decision tree is like an advisor who has a lot of experience. For knowing the future of your career (or some other query in the advisor's field), we consult the advisor. The advisor asks us a series of questions and analyzes our answer and gives the most likely outcome to our query. The better experienced the advisor is, the more accurate the answer will be.

DECISION TREE

FEATURES, FEATURE VALUES, AND LABELS

- ▶ **Features** are the input variables (*the questions we answer*)¹ used by the model before finally reaching an answer.
- ▶ **Feature values** are the different values (*the different options/decisions*) that a feature can have.
- ▶ **Labels** is the final result of the decision tree (*the possible outcomes predicted the advisor*).

Predicting the accurate² label based on the different feature values given by the user is the ultimate goal of the decision tree. The better the model is, the more accurate the result.

¹Refer to the analogy in the previous slide

²The desired or the true answer it should have given

DECISION TREE

STRUCTURE

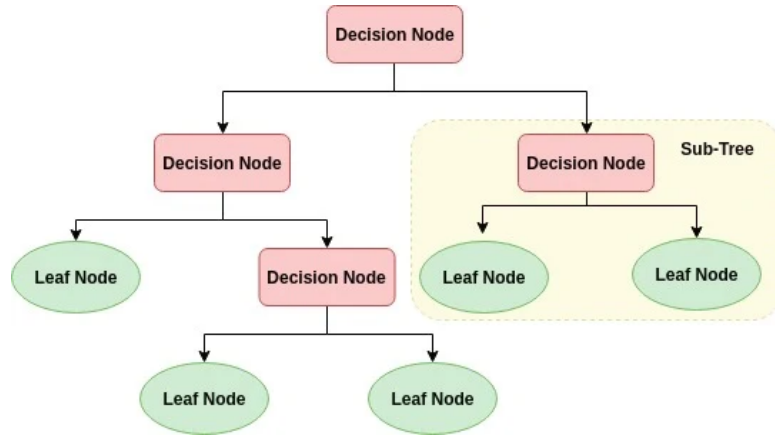


Figure. General structure of a Decision tree³

- ▶ From a **decision node** (*feature*), by selecting different decisions (*feature values*) you reach the next decision node. The **leaf nodes** are the final answer (*label*) it predicts.
- ▶ Note that there can be more than two options for a decision node.

³Taken from Datacamp (2018)

DECISION TREE

EXAMPLE

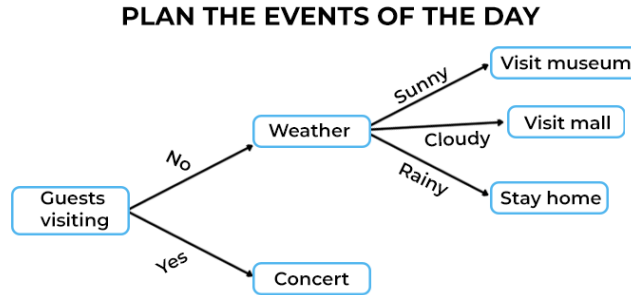


Figure. An example of simple decision tree⁴

- ▶ This decision tree decides what is the plan of the day after analyzing a series of answers given.
- ▶ The features here are 'Guest visiting' and 'Weather'.
- ▶ The feature values of weather are 'Sunny', 'Cloudy', and 'Rainy'.
- ▶ the labels are 'Visit museum', 'Visit mall', and 'Stay home'.

⁴Taken from Spiceworks (2022)

DECISION TREE

ALGORITHM RELATED FACTORS

As usual, this model is trained using a *training dataset* containing a set of features and their corresponding labels. In class we kept our discussion limited to the **ID3**⁵ (Iterative Dichotomiser 3) algorithm. Here, the basic factors or intuition⁶ to the decision tree algorithm are discussed.

- ▶ **Order of features:** This refers to the order in which the features are to be asked. By intuition, the questions whose answers/options help to reach a decision faster should be considered first. Let this ability of a question be termed '*impact*' for the time being. As such, the features have to be *ranked* according to their impact.
- ▶ **Factors in Ranking different features:** While measuring the impact of the features, the following factors have to be considered while measuring the impact:
 - Number of people answering that feature. (Bigger people imply more confidence)
 - Features favoring a certain label more than another. (More impact for a feature when it highly recommends certain labels)
 - Different feature values favor different labels. (if all feature values lead to the same label, then the feature has less impact)⁷

⁵For more details, see Wikipedia (2022)

⁶These are part of the class discussions like a discussion of intuitions and not specific instruction for the algorithm.

⁷These parameters are debated upon and different ranking systems exist based on that.

DECISION TREE

SAMPLE PSEUDO-CODE

A sample pseudo-code for the modeling of the binary decision tree is given below.

```
1 func make_binary_decision_tree(x, y): # x is a matrix of features, y is a vector of
  labels
2   if all y are the same:
3     return a leaf node with the label
4   if x is empty:
5     return a leaf node with the most common label
6   best_feature = find_best_split(x, y) # returns the feature with most 'impact'
7   x0, y0, x1, y1 = split(x, y, best_feature) # defined later
8   branch0 = make_decision_tree(x0, y0)
9   branch1 = make_decision_tree(x1, y1)
10  return a decision node with best_feature, branch0, branch1
11
12 func split(x, y, feature):
13   x0, x1 = splitting rows of x into two parts where feature value is 0 and 1 respectively
14   y0, y1 = splitting rows of y into two parts where feature value is 0 and 1 respectively
15   drop the feature column from x0 and x1
16   return x0, y0, x1, y1
```

* The `find_best_split(x, y)` function calculates the 'impact' for each feature and returns the feature with the highest 'impact'

DECISION TREE

CALCULATING THE IMPACT

Although the best way to calculate the impact was not discussed in class till now, we discussed some functions which can be used to calculate the impact:

1. **Average:** Calculate the fraction of times (a_i) the label is 1 for each feature value i (0/1 here). Then, calculate the average of these fractions: $\frac{\sum a_i}{\sum i}$
2. **Weighted Average:** Add the averages in proportions (p_i) of probabilities of occurrence of that feature value: $\frac{\sum p_i \times a_i}{\sum i}$
3. **Distance from 0.5:** Calculate the distance of the average from 0.5 for each feature value: $|0.5 - a_i|$ [$0 < a_i < 1$] because in this type of binary classification, of even random values give us 0.5% chance of predicting right.
4. **Entropy:** Calculate the entropy of the label for each feature value: $\sum -p_i \log(p_i)$ ⁸
5. **Gini Impurities:** Calculate the Gini impurities of the label for each feature value. *Details of this was taught in later classes.*

⁸ p_i is always less than 1. So, log of that will be negative. To counter this, we add another negative sign to make the whole thing positive.

Part II

INDUCTIVE BIAS

INDUCTIVE BIAS

INTRODUCTION AND EXAMPLE

- ▶ **Inductive Bias** is a set of assumptions taken by us about the model.
- ▶ These assumptions are taken **without looking at the data** . They are also called **prior knowledge** of the model.
- ▶ These biases prepare the model according to the user's needs. However, these assumptions, if made wrong, can lead to worse results. Hence, it is important to choose the assumptions carefully. It also tells us that no model is without bias as we have inherent assumptions about the algorithm we design to get useful results.
- ▶ For example: In case of a decision tree, we assume that the features are independent of each other. This assumption is made because the features are assumed to be asked in a random order. However, if the features are not independent of each other, then the model will not be able to give accurate results.

INDUCTIVE BIAS

INDUCTIVE BIAS OF SHALLOW DECISION TREE

- ▶ Consider a variant⁹ of decision tree where we can only ask d number of questions. This variant is called the '**shallow decision tree**'. Here, a label is predicted within asking d many questions and no more.
- ▶ In such a variant, the inductive bias is the assumption that an accurate label can be predicted by analyzing the answers of d many questions.
- ▶ If the d is set too low, we assume the model to predict with knowing very few features, which is closer to *God* level.
- ▶ If the d is set too high, the model is closer to *memorization* as it required too many features to predict.
- ▶ The d should set in between these in order to get the *generalization* right.

⁹This variant is mentioned in Daume (n.d.)

Part III

HYPERPARAMETER

HYPERPARAMETER

MODELS, PARAMETERS AND HYPERPARAMETERS

- ▶ **Model** is a function which takes in the data and returns the output.
- ▶ **Parameter** is a variable which is used by the model to make predictions. These are the variables which are learned by the model.
- ▶ **Hyperparameter** is a variable which are also used by the model to make predictions. Unlike parameters, these are **not learned** by the model but tuned by the user. These hyperparameters are used to **control/tune the inductive bias** of the model.

HYPERPARAMETER

EXAMPLE

- ▶ **Hyperparameter** of decision tree is the **maximum depth** of the tree. This hyperparameter is used to control the inductive bias of the model.
- ▶ If the maximum depth is set too low, the model will not be able to learn the complex patterns in the data. This will lead to higher training error.
- ▶ Also, if the maximum depth is set too high, the model will be able to memorize the training data. Although it will give a lower training error, it will not be able to generalize well on the test data. This will lead to higher test error.
- ▶ Thus we can tune the maximum depth to the optimum value to get a good balance between generalization and memorization.







HYPERPARAMETER

HYPERPARAMETER TUNING

- ▶ **Hyperparameter tuning** is the process of finding the best hyperparameter values for a model.
- ▶ While training the model, we are learning the parameters to minimize the training error. However, hyperparameters should not be adjusted only by looking into the training error. This will lead to memorization.
- ▶ We absolutely cannot ever use the test data to tune the hyperparameters. Seeing the test data will render it useless for testing the model.¹⁰
- ▶ So, we reserve a part of the training data to tune the hyperparameters. This part of the data is called the **validation set**. We use the validation set to tune the hyperparameters and then use the test set to evaluate the model.

¹⁰One should not even look at the test data. This is called **data snooping bias**. See Department of Mathematics (2011).

REFERENCES I

-  **Datacamp (2018)**. “Decision Tree Classification in Python Tutorial”. In: URL: https://res.cloudinary.com/dyd911kmh/image/upload/f_auto,q_auto:best/v1545934190/1_r5ikdb.png.
-  **Daume, Hal (n.d.)**. “Limits of Learning”. In: *A course in Machine Learning* (), p. 21. URL: http://ciml.info/dl/v0_99/ciml-v0_99-ch02.pdf.
-  **Department of Mathematics, University of Texas (Mar. 2011)**. URL: <https://web.ma.utexas.edu/users/mks/statmistakes/datasnooping.html>.
-  **Mishra, Subhankar (2023)**. “CS460: Machine Learning 2023 Lectures”. In: URL: <https://www.niser.ac.in/~smishra/teach/cs460/23cs460/>.
-  **Spiceworks (2022)**. “What Is a Decision Tree? Algorithms, Template, Examples, and Best Practices”. In: URL: [Spiceworks.com](https://www.spiceworks.com).
-  **Wikipedia (2022)**. “ID3 algorithm”. In: URL: https://en.wikipedia.org/wiki/ID3_algorithm#:~:text=In%20decision%20tree%20learning%2C%20ID3,and%20natural%20language%20processing%20domains..