Decision Tree & Random Forest
Summary

M. R. Hasan
Readings

• Bishop: 14.2, 14.4
• Murphy: 16, 16.1, 16.2.1, 16.2.2, 16.2.3, 16.2.4
• Geron: chapter 6, 7
Decision Tree: Facts

- Decision Tree is a **versatile** model: can perform both classification and regression.
- **Pro**: The most important benefit of Decision Tree: we can build an **explainable model**.
- **Con**: Decision Tree training algorithm is based on **greedy search**, thus it **doesn’t guarantee** a highly accurate solution.
- **Con**: Decision Tree has **high variance** (prone to overfitting).
- A powerful technique to combat this overfitting is to build an ensemble of decision trees, i.e., a **Random Forest**.
- How is the Decision Tree model **different** from all previous models including the Artificial Neural Network (ANN) model?
Concept Learning: ANN

• Previously we studied the ANN model.
• Let’s motivate the Decision Tree learning model from ANN.
• Can the ANN model do **common sense reasoning**?

• **No!**
Common sense reasoning is based on original information (raw features) combined by a sets of rules.

Example: If (lab score > 70 & final score > 60), Then pass

But, ANN combines pieces of information that may have never been seen together (not conforming with common sense).
Knowledge is Compositional

- Some knowledge or concepts are **compositional**.
- These concepts can be learned by **discovering the logical rules**.

<table>
<thead>
<tr>
<th>Tribe</th>
<th>Strength</th>
<th>Tools</th>
<th>Inspiration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbolists</td>
<td>Formal Systems</td>
<td>Decision Tree, Inductive Logic</td>
<td>Logic, Philosophy</td>
</tr>
<tr>
<td>Connectionists</td>
<td>Neural Networks</td>
<td>Backpropagation, Reinforcement Learning</td>
<td>Neuroscience</td>
</tr>
<tr>
<td>Bayesians</td>
<td>Probabilistic Inference</td>
<td>Hidden Markov Model (HMM), Graphical Models</td>
<td>Statistics</td>
</tr>
<tr>
<td>Evolutionaries</td>
<td>Structure Learning</td>
<td>Genetic Algorithms, Evolutionary Programming</td>
<td>Biology</td>
</tr>
<tr>
<td>Analogizers</td>
<td>Mapping to Novelty</td>
<td>K-NN, Support Vector Machine (SVM)</td>
<td>Psychology</td>
</tr>
</tbody>
</table>

The **Symbolists’** school in Machine Learning solves the concept learning problem by **learning sets of rules**.
Introduction to Decision Tree
Decision Tree

• Consider a **decision-making problem**.
• We want to distinguish **three different species** of Iris flowers: Iris-Setosa, Iris-Versicolor, and Iris-Virginica.
• We will use **4 features** for identification: Sepal width & length, Petal width & length.
• The Decision Tree model creates partitions in the data by randomly selecting a feature (e.g., petal length) and a split value (e.g., 2.45 cm).

• Leaf nodes isolate the target (flower varieties).

**Rule = Left-most branch:**
If (petal length $\leq 2.45$ cm)
Then flower type = *Setosa*
Decision Tree

- To build a decision tree, we need to determine **two things**.
  - What are the **key questions** (i.e., parameters)?
  - In which **order the questions** should be posed (i.e., the tree structure)?

**Selecting the parameters**

**Tree Structure**
How to Grow a Decision Tree?
Decision Tree

- Example: We will grow a decision tree to decide whether to wait for a table at a restaurant.

<table>
<thead>
<tr>
<th>Example</th>
<th>Input Attributes</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt</td>
<td>Bar</td>
<td>Fri</td>
</tr>
<tr>
<td>x1</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>x2</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>x3</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>x4</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>x5</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>x6</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>x7</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>x8</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>x9</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>x10</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>x11</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>x12</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The aim here is to learn a set of rules for the goal predicate WillWait.
Decision Tree

- Every variable (parameter) has a **small set of possible values**.

1. Alternate: whether there is a suitable alternative restaurant nearby.
2. Bar: whether the restaurant has a comfortable bar area to wait in.
3. Fri/Sat: true on Fridays and Saturdays.
4. Hungry: whether we are hungry.
5. Patrons: how many people are in the restaurant (values are None, Some, and Full).
6. Price: the restaurant’s price range ($, $$, $$$).
7. Raining: whether it is raining outside.
8. Reservation: whether we made a reservation.
9. Type: the kind of restaurant (French, Italian, Thai, or burger).
10. Wait Estimate: the wait estimated by the host (0–10 minutes, 10–30, 30–60, or >60).

We will learn how to choose these parameters to make a decision about the target **Will Wait**.
Let’s look at a specific choice of the parameters.

Example: **Patrons = Full and WaitEstimate = 0 ~ 10** will be classified as **positive** (i.e., yes, we will wait for a table).
1. **Alternate**: whether there is a suitable alternative restaurant nearby.
2. **Bar**: whether the restaurant has a comfortable bar area to wait in.
3. **Fri/Sat**: true on Fridays and Saturdays.
4. **Hungry**: whether we are hungry.
5. **Patrons**: how many people are in the restaurant (values are *None*, *Some*, and *Full*).
6. **Price**: the restaurant’s price range ($, $$, $$$).
7. **Raining**: whether it is raining outside.
8. **Reservation**: whether we made a reservation.
9. **Type**: the kind of restaurant (French, Italian, Thai, or burger).
10. **WaitEstimate**: the wait estimated by the host (0–10 minutes, 10–30, 30–60, or >60).

To make a decision at the leaf node (wait or not), one **hypothesis includes** two variables: **Patrons** & **WaitEstimate**

There are **other hypotheses** that includes combination of variables.
Growing a Decision Tree is an Intractable Problem!!

Why?
Decision Tree

- Each path of the tree represents a hypothesis/decision function.
- For $n$ attributes, there are $2^{2n}$ different functions.

To find the correct hypotheses, we need to search through $2^{2n}$ different functions!
Decision Tree

- For example, with just the **10 Boolean attributes** of the restaurant problem there are $2^{2^{10}} = 2^{1024}$ or $10^{308}$ different functions to choose from.
- For **20 attributes**, number of functions is $2^{2^{20}} = \text{over } 10^{300,000}$.
- We will need **some ingenious algorithms** to find good hypotheses in such a large space.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <strong>Alternate</strong>: whether there is a suitable alternative restaurant nearby.</td>
<td></td>
</tr>
<tr>
<td>2. <strong>Bar</strong>: whether the restaurant has a comfortable bar area to wait in.</td>
<td></td>
</tr>
<tr>
<td>3. <strong>Fri/Sat</strong>: true on Fridays and Saturdays.</td>
<td></td>
</tr>
<tr>
<td>4. <strong>Hungry</strong>: whether we are hungry.</td>
<td></td>
</tr>
<tr>
<td>5. <strong>Patrons</strong>: how many people are in the restaurant (values are None, Some, and Full).</td>
<td></td>
</tr>
<tr>
<td>6. <strong>Price</strong>: the restaurant’s price range ($, $$, $$$).</td>
<td></td>
</tr>
<tr>
<td>7. <strong>Raining</strong>: whether it is raining outside.</td>
<td></td>
</tr>
<tr>
<td>8. <strong>Reservation</strong>: whether we made a reservation.</td>
<td></td>
</tr>
<tr>
<td>9. <strong>Type</strong>: the kind of restaurant (French, Italian, Thai, or burger).</td>
<td></td>
</tr>
<tr>
<td>10. <strong>WaitEstimate</strong>: the wait estimated by the host (0–10 minutes, 10–30, 30–60, or &gt;60).</td>
<td></td>
</tr>
</tbody>
</table>
Decision Tree

- The time-complexity of training is: \( O(e^{\text{Samples}}) \) time.
- Thus, finding the optimal hypothesis (combination of features) is a NP-Complete problem.
- It’s an intractable problem!
**Decision Tree**

**Exhaustive search is intractable**

**Solution??**

**Perform Greedy Search!**

<table>
<thead>
<tr>
<th>Example</th>
<th>Alt</th>
<th>Bar</th>
<th>Fri</th>
<th>Hun</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$$$</td>
<td>No</td>
<td>Yes</td>
<td>French</td>
<td>0–10</td>
</tr>
<tr>
<td>$x_2$</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Thai</td>
<td>30–60</td>
</tr>
<tr>
<td>$x_3$</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Burger</td>
<td>0–10</td>
</tr>
<tr>
<td>$x_4$</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>Thai</td>
<td>10–30</td>
</tr>
<tr>
<td>$x_5$</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Full</td>
<td>$$$</td>
<td>No</td>
<td>Yes</td>
<td>French</td>
<td>&gt;60</td>
</tr>
<tr>
<td>$x_6$</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Some</td>
<td>$</td>
<td>Yes</td>
<td>Yes</td>
<td>Italian</td>
<td>0–10</td>
</tr>
<tr>
<td>$x_7$</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>Burger</td>
<td>0–10</td>
</tr>
<tr>
<td>$x_8$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Some</td>
<td>$$</td>
<td>Yes</td>
<td>Yes</td>
<td>Thai</td>
<td>0–10</td>
</tr>
<tr>
<td>$x_9$</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Full</td>
<td>$</td>
<td>Yes</td>
<td>No</td>
<td>Burger</td>
<td>&gt;60</td>
</tr>
<tr>
<td>$x_{10}$</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$$$</td>
<td>No</td>
<td>Yes</td>
<td>Italian</td>
<td>10–30</td>
</tr>
<tr>
<td>$x_{11}$</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Thai</td>
<td>0–10</td>
</tr>
<tr>
<td>$x_{12}$</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Full</td>
<td>$</td>
<td>No</td>
<td>No</td>
<td>Burger</td>
<td>30–60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Goal</th>
<th>WillWait</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_1$</td>
<td>Yes</td>
</tr>
<tr>
<td>$y_2$</td>
<td>No</td>
</tr>
<tr>
<td>$y_3$</td>
<td>Yes</td>
</tr>
<tr>
<td>$y_4$</td>
<td>Yes</td>
</tr>
<tr>
<td>$y_5$</td>
<td>No</td>
</tr>
<tr>
<td>$y_6$</td>
<td>Yes</td>
</tr>
<tr>
<td>$y_7$</td>
<td>No</td>
</tr>
<tr>
<td>$y_8$</td>
<td>Yes</td>
</tr>
<tr>
<td>$y_9$</td>
<td>No</td>
</tr>
<tr>
<td>$y_{10}$</td>
<td>No</td>
</tr>
<tr>
<td>$y_{11}$</td>
<td>No</td>
</tr>
<tr>
<td>$y_{12}$</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Tractable Solution for Growing a Decision Tree
Decision Tree

• We will use **greedy divide-and-conquer** strategy.
• What heuristic (local strategy) should we use to perform greedy search?
• Heuristic: **test the most important attribute first**.
• This test divides the problem up into smaller subproblems that can then be **solved recursively**.

1. *Alternate*: whether there is a suitable alternative restaurant nearby.
2. *Bar*: whether the restaurant has a comfortable bar area to wait in.
3. *Fri/Sat*: true on Fridays and Saturdays.
4. *Hungry*: whether we are hungry.
5. *Patrons*: how many people are in the restaurant (values are *None*, *Some*, and *Full*).
6. *Price*: the restaurant’s price range ($, $$, $$$).
7. *Raining*: whether it is raining outside.
8. *Reservation*: whether we made a reservation.
9. *Type*: the kind of restaurant (French, Italian, Thai, or burger).
10. *Wait Estimate*: the wait estimated by the host (0–10 minutes, 10–30, 30–60, or >60).
Decision Tree

- What is the "most important attribute": compare two attributes: "Type" & "Patrons".
- Here "Patrons" is the most important attribute as it creates splits that are less impure.

Light boxes: WillWait is true
Dark boxes: WillWait is false
Algorithm for Growing (Training) a Decision Tree
Training Decision Tree

• Resort to a **greedy heuristic**:  
  - Start from an empty decision tree.  
  - Split on **next best attribute** (feature).  
  - Recurse.

---

**Classification and Regression Trees (CART) Algorithm**

- Iterative Dichotomizer (ID3)
- C4.5 (ID3 + improvements)
Learning Decision Tree: CART

• The **Classification and Regression Trees** (CART) algorithm was proposed by *Breiman* et al. in 1984.
Learning Decision Tree: CART

• In CART, the sequential decision making process corresponds to the traversal of a **binary tree**.

ID3 generalizes CART by producing **more than two children**.
Training Decision Tree

• At each step of the CART, we need to use the **best** attribute/feature to make the split.

• How do we choose a **good attribute/feature**?

• What’s the best feature?

**Features:**
- Sepal width
- Sepal length
- Petal width
- Petal length

**Features:**
- Iris Versicolor
- Iris Setosa
- Iris Virginica
Training Decision Tree

- Best feature *reduces impurity/uncertainty*!
- In the classification setting, there are several ways to measure the quality of a split.
  - Entropy and/or Information Gain
  - Gini Index

Features:
- Sepal width
- Sepal length
- Petal width
- Petal length

![Iris Varieties](image)
Decision Tree: Hyperparameters to Handle Overfitting
Training Decision Tree: Overfitting

- Decision tree is based on a greedy algorithm.
- It can easily overfit the data (has high variance).
- How do we prevent overfitting?
- We need to tune the hyperparameters.
Training Decision Tree: Hyperparameters

• There are **several regularization hyperparameters** that can be used to control overfitting.
• Maximum depth of the tree
  - Reducing the depth will **reduce the overfitting**.
• Minimum number of samples a node must have before split
• Minimum number of samples a leaf must have
• Maximum number of leaf nodes
• Maximum number of features that are evaluated for splitting at each node
Decision Tree: Summary

- Decision trees are one of the **most popular** data mining tools.
- Easy to understand
- Easy to implement and use
- Computationally cheap (to solve heuristically)

*Interpretable by humans*

Main problem: Suffers from **overfitting** (high variance)!
Decision Tree

• For an empirical understanding of the Decision Tree model, see the Jupyter notebooks in the GitHub repository:
Random Forest
Random Forest

- The Decision Tree model suffers from high variance (overfitting).
- One technique to reduce the high variance of a Decision Tree model is to train a group of Decision Trees, each on a different random subset of the training set.
Random Forest

• To make predictions, we obtain the predictions of all individual trees.
• Then, predict the class that gets the **most votes**.
• Such an ensemble of Decision Trees is called a **Random Forest**.
Random Forest

- How does the Random Forest model improve generalizability?
- It achieves better generalizability by reducing the variance of individual Decision Trees.
Random Forest

• To reduce variance, the component trees are all designed to be **randomly different** from one another.

• This leads to **de-correlation** between the individual tree predictions and, in turn, to improved generalization.
Random Forest

- Forest randomness also helps achieve **high robustness** with respect to *noisy data.*
Random Forest: Randomization Techniques

• Randomness is injected into the trees during the training phase.

• Two of the most popular ways of doing so are:
  - Random training dataset sampling: samples from data
  - Randomized node optimization: uses all data, but samples from features
Random Forest Classifier: Advantages
Random Forest: Classification

- Classification forests enjoy a number of **useful properties**.
- They naturally handle problems with **more than two classes** (unlike SVM and boosting methods).
- They provide **probabilistic output**.
- They **generalize well** to previously unseen data & high-dimensional data.
- They are efficient due to their **parallelism** and reduced set of tests per data point.
Random Forest: Scikit-Learn

• For an empirical understanding of the Random Forest model, see the Jupyter notebooks in the GitHub repository:
  • https://github.com/rhasanbd/Random-Forest-Wisdom-of-a-Diverse-Crowd