DECISION TREES

Based upon *Pattern Classification*, chapter 8
Tutorial outline

• How decision trees work

• Building a decision tree
  • Single- and multi-value
  • Impurity metrics
    • Entropy
    • Gini
    • Pruning

• Example – Iris Dataset
*Akinator’s implementation is probably more involved than a decision tree. It’s writers didn’t reveal how it’s implemented.
“20 questions” as a decision tree

- **Features** – different properties of a character we can ask about

- **Label** – The identity of the character

- **Training Set** – The characters you’ve encountered in your life

- **Test/Inference** – Play “20 questions”

- Note: 20 question is a special case of classification where each sample is its own class.
Decision tree visualization

- **color = Green?**
  - yes
    - **size = big?**
      - no
        - **Watermelon**
  - no

- **color = yellow?**
  - yes
    - **shape = round?**
      - no
        - **shape = round?**
          - yes
            - **size = big?**
              - no
                - **Breadfruit**
          - no
            - **shape = round?**
              - yes
                - **taste = sweet?**
                  - no
                    - **Apple**
    - no
      - **size = small?**
        - yes
          - **Grape**
        - no
          - **Apple**

- **size = medium?**
  - no
    - **Grape**
  - yes
    - **Apple**
Good strategy for “20 questions”

• Eliminate as many options as possible

• Simple questions

• Consider more likely characters (Leibniz isn’t who most people think of…)
Consideration for building a tree

1. Binary or multi-valued?
2. Which property to test?
3. When should a node be declared a leaf?
4. If the tree becomes “too large”, how can it be made smaller and simpler (pruned)?
Binary or multi-valued

Every polythetic tree can be represented by an equivalent monothetic tree
Decision boundaries

• How do the decision boundaries look like for binary decision trees?
Which **property** to test?

- **Simple** questions: each branch should consider **few features** (1 if it’s a binary tree).

- The ones that “eliminate” the most options.
- = increase our certainty in the decision
- = lead us to nodes that are homogenous (pure)
Class conditional probability

- We estimate the class-conditional probabilities $P(c|\mathcal{D})$ by fitting a categorical MLE:

\[
p_i \triangleq P(c|D) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \mathbb{I}(y_i = c)
\]

- $c$ – class
- $\mathcal{D}$ – Data
- Reminder:
  - Categorical distribution – generalization of Bernoulli to multiple classes
Impurity

• Leading principle in tree creation: simplicity (Occam’s razor)
• Therefore we seek an attribute $A$ at each node $S$ that makes data reaching the immediate descendent nodes as “pure” as possible.
• At each node, $S$, the vector $P = (p_1, p_2, \ldots, p_k)$, represents the probability (ratios) of samples belonging to class $i$.

$\phi(P) \geq 0$ – impurity of node $S$.
• $\phi(P) = 0$ if all samples have the same label
• $\phi(P)$ is maximized for uniform distribution
• $\phi(P)$ is symmetric w.r.t. $P$
• $\phi(P)$ is smooth (differentiable everywhere)
Entropy

• The most common measure for impurity:

\[ H(P) = \sum_i p_i \log \left( \frac{1}{p_i} \right) = -\sum_i p_i \log(p_i) \]

• Given a system whose exact description is unknown, its entropy is defined as the amount of information needed to exactly specify the state of the system
  • If the system has k states, we need \( \log(k) \) bits to represent them
  • \( p_i \) is the ratio of elements belonging to class \( i \).

* A friendly introduction to information theory
Gini Index

- Another common measure for impurity:

\[ GiniIndex(P) = \sum_{i=1}^{c} p_i (1 - p_i) = 1 - \sum_{i=1}^{c} p_i^2 \]

- This can be interpreted as a *variance impurity*.

- Reminder: Given \( X \sim Bernoulli(p) \), \( Var(X) = p \cdot (1 - p) \)

- This is the expected error rate at node \( N \) if the category label is selected randomly from the class distribution present at \( N \).
Which split to choose

• The one that most decreases impurity (increases purity).
• Given node $S$, with probabilities $P$ and discrete attribute $A$, the impurity drop is defined as:

$$\Delta \phi(S, A) = \phi(S) - \sum_{v \in Values(A)} p_v \phi(S_v)$$

• For the binary case:

$$\Delta \phi(S, A) = \phi(S) - P_L \phi(S_L) - (1 - P_L) \phi(S_R)$$

• For the entropy measure, this measure is called **information gain (or Mutual Information)**:

$$IG(P) = H(P) - \sum_{v \in Values(A)} p_a H(p_a)$$

• For the Gini index, the measure is called **Gini Gain**
Highly-branching attributes

- Problem: attributes with a large number of values
  - extreme case: each example has its own value
    e.g. example ID; Day attribute in weather data

- Information gain is biased towards choosing attributes with a large number of values

- This may cause several problems:
  - Overfitting:
    Selection of an attribute that is non-optimal for prediction
  - Fragmentation:
    Data are fragmented into (too) many small sets
Split Information

• Intrinsic information of a split:
  • Entropy of distribution of instances into branches
  • i.e. how much information do we need to tell which branch an instance belongs to

\[
SplitInformation(S, A) = - \sum_{v \in Values(A)} p_v \log(p_v)
\]

• Observation:
  attributes with higher intrinsic information are less useful
Gain Ratio

• Modification of the information gain that reduces its bias towards multi-valued attributes
• Takes number and size of branches into account when choosing an attribute
  • corrects the information gain by taking the intrinsic information of a split into account
• Definition of Gain Ratio:

\[
GR(S, A) = \frac{IG(S, A)}{SplitInformation(S, A)}
\]
Under- and over-fitting

- If we allow our trees to grow until absolute purity we will reach overfitting
Stop splitting Pruning methods

• Stop splitting: stop growing a tree once a stopping criteria is reached
  • May suffer from the horizon effect - Such a method will be biased towards trees in which greatest impurity reduction is near the root

• Pruning: grow a full tree, than merge nodes until a stopping criteria is reached
  • Higher computational cost

• Common stopping criteria:
  • Minimum number of samples in leaf nodes
  • Reach purity threshold
  • Limit the depth of the tree or the number of leaves
  • Use hypothesis testing (chi-squared) to determine if the splitting reduces the impurity significantly
  • No improvement in validation accuracy
Scikit-learn’s DecisionTree

```python
class sklearn.tree.DecisionTreeClassifier (criterion='gini', splitter='best', max_depth=None, 
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, 
max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False)
```

A decision tree classifier.

Read more in the User Guide.

**Parameters:**

- **criterion** : string, optional (default="gini")
  
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

- **splitter** : string, optional (default="best")
  
The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

- **max_depth** : int or None, optional (default=None)
  

**IPython demo – iris dataset**

- Iris flower data set
- The labels - 3 different types of irises’ (Setosa, Versicolour, and Virginica)
- The features - their petal and sepal length and width (עלי כותרת, עלי גביש)
- 50 samples from each class
- A standard ML benchmark