ASSIGNMENT 4: DECISION TREES

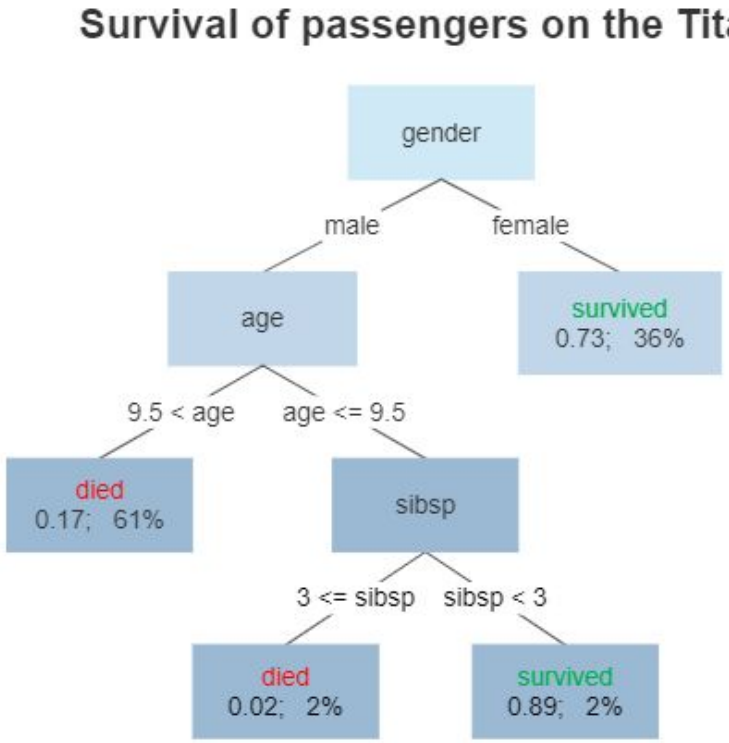
Johannes Kofler

JOHANNES KEPLER
UNIVERSITY LINZ

Institute for
Machine Learning

Copyright statement:

Introductory Example (taken from Wikipedia)



- "sibsp“ is the number of spouses or siblings aboard.
- The figures under the leaves show the probability of survival and the percentage of observations in the leaf.
- Summarizing: Chances of survival were good if you were (i) a female or (ii) a male younger than 9.5 years with strictly less than 3 siblings.

Decision tree learning: Part 1

- All decision tree learning algorithms are recursive, depth-first search algorithms that perform hierarchical splits.
- There are three main design issues:
  - Splitting criterion: which splits to choose?
  - Stopping criterion: when to stop further growing of the tree?
  - Pruning: whether/how to collapse unnecessarily deep sub-trees?
- The two latter: relevant for adjusting the complexity of decision trees (underfitting vs. overfitting).

Decision tree learning: Part 2: Recursive procedure

- Given:
  - Training set $\mathbf{Z} = \{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, l\}$
  - Stopping criterion
  - Splitting criterion
- Call DecTree($\mathbf{Z}$, Root node, $S = \{$all possible splits$\}$)
- DecTree($\mathbf{Z}$, N, S)
  - If stopping criterion is fulfilled, exit.
  - Determine split $s \in S$ such that splitting criterion is maximal.
  - Divide $\mathbf{Z}$ into disjoint subsets $\mathbf{Z}_{s,t}$ according to split s and result t.
  - For all t such that $\mathbf{Z}_{s,t} \neq \emptyset$
    - Generate new node N_t
    - Call DecTree($\mathbf{Z}_{s,t}$, N_t, $S \backslash \{s\}$)

Splitting: categorical vs. numerical features

- Categorical features:
  - Binary split:
    $\mathbf{Z}_L = \{(\mathbf{x}, y) \in \mathbf{Z} \mid x_i = c\}, \mathbf{Z}_R = \{(\mathbf{x}, y) \in \mathbf{Z} \mid x_i \neq c\}$
  - Split according to entire feature: $\mathbf{Z}_j = \{(\mathbf{x}, y) \in \mathbf{Z} \mid x_i = c_j\}$
  - Typically, all possible (binary or entire feature) splits w.r.t. all features are considered.
- Numerical features:
  - Apply threshold c to i-th feature, i.e.
    $$\mathbf{Z}_L = \{(\mathbf{x}, y) \mid x_i < c\}, \mathbf{Z}_R = \{(\mathbf{x}, y) \mid x_i \geq c\}.$$
  - Typically, all possible splits w.r.t. all features are considered
  - Thresholds are chosen as mean values of “neighboring” values occurring in the data.

Common splitting criteria:

- Classification:
  - Information gain
  - Gini impurity (gain)
- Regression:
  - Variance reduction
- Nowadays mostly Gini impurity and variance reduction are used.

Information gain

- For any (sub)set of data $\mathbf{Z}$, the relative proportions of samples belonging to the k-th class (of classes $1, \ldots, M$) are defined as:

$$p_k(\mathbf{Z}) = \frac{|\{(\mathbf{x},y) \in \mathbf{Z} | y = k\}|}{|\mathbf{Z}|}.$$

- The entropy of $\mathbf{Z}$ w.r.t. the target is defined as

$$H(\mathbf{Z}) = -\sum_{k=1}^{M} p_k(\mathbf{Z}) \ln p_k(\mathbf{Z}).$$

  - Maximal ($\ln M$) if classes are uniformly distrib. in the set $\mathbf{Z}$.
  - Minimal (0) if all samples of $\mathbf{Z}$ belong to one single class.
  - The smaller/larger the entropy the larger/smaller the information.

- Information gain (Kullback-Leibler divergence) of employing the s-th split for partitioning $\mathbf{Z}$ into sets $\mathbf{Z}_{s,t=1}, \ldots, \mathbf{Z}_{s,t=K_s}$ is then defined as

$$g_E(\mathbf{Z}, s) = H(\mathbf{Z}) - \sum_{t=1}^{K_s} \frac{|\mathbf{Z}_{s,t}|}{|\mathbf{Z}|} H(\mathbf{Z}_{s,t}).$$

- g_E is maximized if subset entropies $H(\mathbf{Z}_{s,t})$ are minimized, i.e. subsets should be as homogeneous as possible (limiting case: only one class).
- Typically, in each step, all possible splits are considered and the one with highest information gain is selected.

Gini impurity (gain)

- With the notation from above: Gini impurity of $\mathbf{Z}$ is defined as

$$I_G(\mathbf{Z}) = \sum_{k=1}^{M} p_k(\mathbf{Z})\big(1 - p_k(\mathbf{Z})\big) = 1 - \sum_{k=1}^{M} p_k^2(\mathbf{Z})$$

- Interpretation: gives probability of incorrectly classifying randomly chosen element in dataset if it were randomly labeled according to the class distribution in the dataset.
- Value is:
  - Maximal $(1 - 1/M)$ if classes are uniformly distributed in the set $\mathbf{Z}$.
  - Minimal (0) if all samples of $\mathbf{Z}$ belong to one single class.
- Gini impurity (gain) of employing the s-th split for partitioning $\mathbf{Z}$ into sets $\mathbf{Z}_{s,t=1}, \ldots, \mathbf{Z}_{s,t=K_s}$ is then defined as:

$$g_G(\mathbf{Z}, s) = I_G(\mathbf{Z}) - \sum_{t=1}^{K_s} \frac{|\mathbf{Z}_{s,t}|}{|\mathbf{Z}|} I_G(\mathbf{Z}_{s,t}).$$

- g_V is maximized if subset Gini impurities $I_G(\mathbf{Z}_{s,t})$ are minimized, i.e. subsets should be as homogeneous as possible.
- Standard splitting criterion employed by the decision tree algorithm CART for classification.

Computing predictions

- Tree recursively partitions/splits training set into subsets, each of which is associated with leaf node.
- "Assignment" of samples to leaf nodes is the basis for making predictions with decision trees.
- For new input $\mathbf{x}$ traverse through tree by answering questions associated with each node until leaf node is reached to which sample $\mathbf{x}$ is assigned.
- Classification:
  - Assign class to leaf node that appears most prominently among training samples associated with this leaf node
  - Alternatively: compute relative frequencies of classes in leaf node and use frequencies as estimates of conditional probabilities $p(y = k \mid \mathbf{x})$.
- Regression: use mean target value of samples associated with leaf node.

Pros and Cons of decision trees

- Pros:
  - Simple and computationally efficient
  - Built-in feature selection
  - Interpretable models
  - Can be applied to categorical and numerical attributes
  - Scaling-invariant for numerical features
  - Can be applied both to classification and regression
- Cons:
  - Greedy splitting may lead to sub-optimal solutions.
  - Only axis-parallel splits of numerical features
  - Shallow trees are not accurate (high bias), deep trees overfit (high variance). Number of parameters not fixed before training.