1 Decision Trees

Decision trees (DTs) are one of the most widely used and practical methods for supervised machine learning. They are successfully used in many classification and regression problems.

Given an object (or instance), a decision tree is either:

- a leaf node that assigns a label (in the case of classification) or numeric value (in the case of regression) to the object; or
- a condition node that performs a test on the object; has as many children as the number of possible outcomes of the test, and each child is a decision tree itself.

Tests performed at condition nodes can be:

- univariate tests:
  - for a categorical feature: possible outcomes are all the categories (or levels)
  - for a numeric feature and a threshold: possible outcomes are the value of the feature being less than or greater than the threshold.
- multivariate tests: the condition involves more than one variable (or feature).

An object is assigned a label or value by starting at the root of the tree and following the path determined by the conditions until a leaf node is reached.

1.1 The space of decision trees

1.1.1 Boolean functions

In a logical setting, where the leaf nodes of the DT are either 'True' or 'False', the tree can be seen as a disjunction of conjunctions:

- any path from the root to a 'True' leaf node is a conjunction of constraints;
- the whole tree is a disjunction of all the conjunctions.

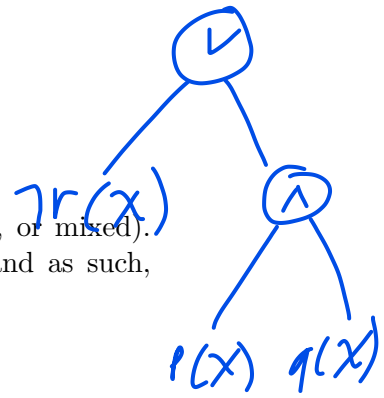
A DT can therefore be seen as a *disjunctive normal form* which can be used to represent any logical formula (and thus any Boolean function). Therefore the cardinality of the space of hypotheses defined by decision trees is equal to the number of possible functions that can be defined over the input space:

in trouble to not learn anything H=F, will fit data perfectly, but unseen data will have equal decision tree - no generalization

$$|\mathcal{H}_{DT}| = |\mathcal{F}|$$

Any logical formula can be converted into a DT. For each formula, there may be more than one (logically equivalent) DT depending on the order of predicates (or tests). This implies that $|\mathcal{R}_{DT}| > |\mathcal{H}_{DT}|$.

Example give a decision tree for $(p(x) \wedge q(x)) \vee \neg r(x)$.



1.1.2 Other functions

DTs can operate in any type of space (numerical, categorical, or mixed). They can partition the space into arbitrarily small regions, and as such, they can represent any deterministic function.

1.2 Bias

Since DTs can represent any function, there is no representation bias. This creates a risk of overfitting. A hypothesis is said to *overfit* the training data if there exists some other hypothesis that has larger error over the training data but smaller error over unseen instances.

This issue is addressed by introducing a search bias. The search bias also helps with the exponentially large space of decision trees which cannot be exhaustively searched.

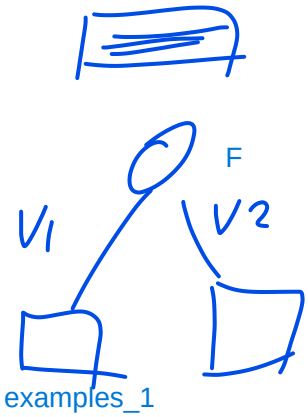
2 DT Construction

2.1 A basic DT construction algorithm for classification problems

todo formula understa

used features on top, in some situation cannot be used in features down

```
DTree(examples, features) # returns a tree
    if all examples are in one class:
        return a leaf node with that class label;
    elif the set of features is empty:
        return a leaf node with the most common class label in examples;
    else:
        create a new decision (condition) node R;
        pick a categorical feature F (or a numeric feature and a threshold);
        for each possible outcome v_i of R:
            add an out-going edge E_v_i to node R;
            let examples_i be the subset of examples that result in outcome v_i;
            if examples_i is empty:
                attach to E_v_i a leaf node (label) that is the most common in examples;
            else:
                attach to E_v_i the result of DTree(examples_i, features \ {F});
        return the subtree rooted at R.
```



Remarks on types of features:

- For a categorical feature, the outgoing branches are possible levels or groupings of them.
- For a numerical feature, a threshold is set and there are two outgoing branches (for less than or greater than the threshold). Numerical features can be reused again. Therefore when R uses a numeric feature and a threshold as the test, the recursive call is `DTree(examples_i , features)`.

intervals in numeric features that can be reused

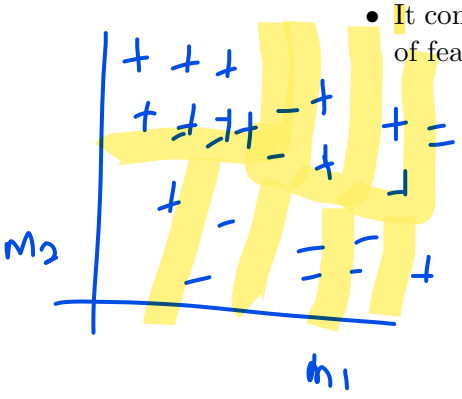
- For ordinal features (a subtype of categorical), either of the above can be applied.

categorical label instead of a number, but there is still an order. eg large, medium,

:

cannot call it greedy yet, more uninformed, as pick A categorical

- The algorithm is greedy and does not backtrack to consider other options.
- It continues until each leaf node is completely pure or until it runs out of features. The former can lead to overfitting.



assume finite examples. will we get complete purity with numeric examples. will keep going until completely pure? but will stop coz go deeper and deeper tree.

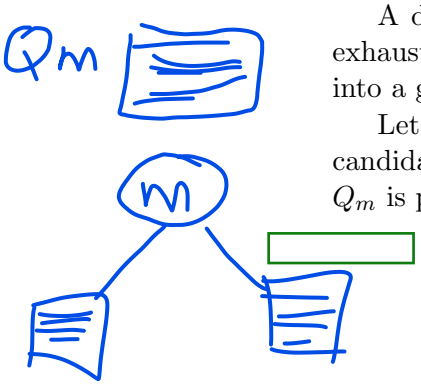
carving out examples that could overfit.

we can have overlapping data. + and - point can be overlapping each other. eg polynomial. input x, and x^2 could have both positive and negative value. overlapping data when features not informative.

- How to pick a feature (and thresholds) is an important decision. You can pick features randomly and still have a perfect tree on training data. However picking the right feature, generates (usually) smaller trees that are less likely to overfit the data and are easier to understand.

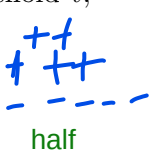
2.2 Finding the best local feature (and threshold)

The training data consists of n instances: $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)$, where $\boldsymbol{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,l})$ is the i-th (l-dimensional) input vector (or feature values), and y_i is the corresponding output (which in the case of classification is categorical, and in regression it is numerical).



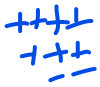
A decision tree recursively partitions the input space into disjoint and exhaustive regions, and makes the same prediction for all instances that fall into a given region.

Let Q_m denote the training data available at node m. Then, for each candidate split $\theta = (j, t)$ consisting of the j-th feature and a threshold t, Q_m is partitioned into subsets $Q_{m,\text{left}}(\theta)$ and $Q_{m,\text{right}}(\theta)$:

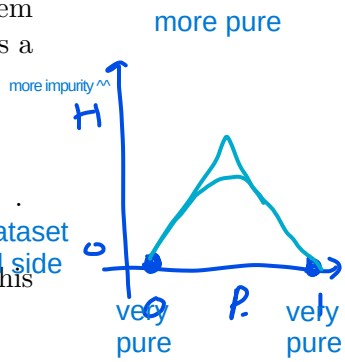


$$Q_{m,\text{left}}(\theta) = \{(\boldsymbol{x}, y) \in Q_m \mid x_j \leq t\} \ ;$$

$$Q_{m,\text{right}}(\theta) = \{(\boldsymbol{x}, y) \in Q_m \mid x_j > t\} = Q_m \setminus Q_{m,\text{left}}(\theta) \ .$$



The *impurity* at node m is computed using an impurity function denoted by $H(.)$. The choice of this function depends on the type of learning problem being solved: classification or regression. The objective function, which is a function of $H(.)$, is $\frac{X_L}{X} H(X_L) + \frac{X_R}{X} H(X_R)$



$$G(Q_m, \theta) = \frac{|Q_{m,\text{ left}}(\theta)|}{|Q_m|} H(Q_{m,\text{left}}(\theta)) + \frac{|Q_{m,\text{right}}(\theta)|}{|Q_m|} H(Q_{m,\text{right}}(\theta)) \ .$$

XL/X, XR/X are the weighted fraction proportions.

weight * impurity

relative size of data on left — impurity on left — relative size of data on right — impurity of dataset on right hand side

For a test (or condition) that has b branches (or possible outcomes), this objective function becomes

$$G(Q_m, \theta) = \sum_{i=1}^{b} \frac{|Q_{m,i}(\theta)|}{|Q_m|} H(Q_{m,i}(\theta)) \ .$$

The best split at node m is then determined by the parameter that minimizes the objective function:

$$\theta^* = \text{argmin}_\theta \, G(Q_m, \theta) \ .$$

objective function want to optimize it, eg this theta * want to minimize it

This process is continued, recursing through subsets of Q_m, until a stopping condition is met: for instance, until the number of observations in a subset (corresponding to a node) is less than some given threshold.

2.2.1 Classification criteria

For a classification problem, where the target can take any of the K possible values $0, 1, \ldots, K-1$, the proportion of training instances in the m-th node that have class label k is

ratio of examples that have class k at node m

probability, node m, up to k possible values

$$p_{mk} = \frac{1}{|Q_m|} \sum_{i:(\boldsymbol{x}_i, y_i) \in Q_m} [y_i = k]$$

where the bracket are Iverson brackets.

Impurity measures are defined as functions of this proportion. Three commonly-used measures are:

- misclassification error:

$$H(Q_m) = 1 - \max_k \ p_{mk} \ ;$$

- Gini index:

$$H(Q_m) = \sum_{k=0}^{K-1} p_{mk} \ (1 - p_{mk}) \ ;$$

- Entropy:

$$H(Q_m) = - \sum_{k=0}^{K-1} p_{mk} \ \log p_{mk} \ ,$$

  where by definition $0 \ \log 0 = 0$.

Example Draw the plot of $H(Q)$ for a binary classification problem.

todo answer

2.2.2 Regression criteria

For a regression problem, where the target can take a continuous value, common measures used to determine the best split at each node are:

- Mean Squared Error:

$$H(Q_m) = \frac{1}{|Q_m|} \sum_{i:(\boldsymbol{x}_i, y_i) \in Q_m} (y_i - \bar{y}_m)^2 \ ;$$

- Mean Absolute Error:

$$H(Q_m) = \frac{1}{|Q_m|} \sum_{i:(\boldsymbol{x}_i, y_i) \in Q_m} \text{abs}(y_i - \bar{y}_m) \ ,$$

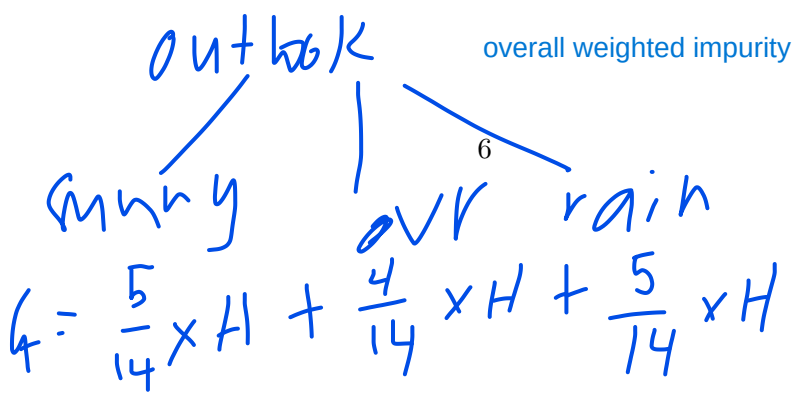
  where $\bar{y}_m$ is the mean of the output values of the instances in region Q_m:

$$\bar{y}_m = \frac{1}{|Q_m|} \sum_{i:(\boldsymbol{x}_i, y_i) \in Q_m} y_i \ .$$

Example compute the impurity at the root of the tree for the following problem. Determine what condition node should be used at the root of the tree.

day	outlook	temp	humidity	wind	play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

todo type up this example



3 Time complexity

In this section, we use n to denote the number of training instances, and m to denote the number of features.

3.1 Prediction

- The time is proportional to the depth of the tree (the length of the longest path from the root).
- The maximum depth can be specified by the user before learning.
- For problems with numeric features the depth is $O(\log n)$ for balanced trees, or $O(n)$ for degenerate trees.
- For problems with categorical features, where features are used only once along each path, the depth is $O(m)$.

3.2 Learning

todo add type notes later

look at inbetween the values of the features as candidates

3.2.1 Problems with only numerical features

For each internal node in the tree, when considering which feature to use, determining the best threshold for a numeric feature takes $O(n \log n)$. This is because the values must be sorted and each in-between point examined. This process must be repeated for each feature. Thus each node takes $O(mn \log n)$ time. The total learning time is

$$O(mn^2 \log n) \ .$$

3.2.2 Problems with only categorical features

Because the features are not reused, at each level i, all $m - i$ features must be considered. The total time is

$$\sum_{i=0}^{m} iO(n) = O(m^2 n)$$

3.2.3 Time complexity in practice

In most practical settings, trees are much shallower than the worst case discussed above, and various preprocessing techniques can be used to optimize the learning process. In practice, the complexity of decision tree learning is roughly linear in m and n.

4 Controlling overfitting

Two basic approaches for decision trees:

- Early stopping: stop growing tree at some point during top-down construction when there is no longer sufficient data to make reliable decisions, or a maximum depth is reached, or increase in purity is negligible, ...
- Post-pruning: grow the full tree, then remove subtrees that do not contribute to good classification.

4.1 Post-pruning using cross-validation todo understand this algorithm

```
Partition training data into ‘grow‘ and ‘validation‘ sets.
Build a complete tree based on the ‘grow‘ data.
Until accuracy on validation set decreases do:
    For each non-leaf node, n, in the tree do:
        Temporarily prune the subtree below n; and
        Replace it with a leaf labeled with the majority class at n.
        Measure and record the accuracy of the pruned tree on the validation set.
    Permanently prune the node that results in the greatest increase in accuracy on the valid
```

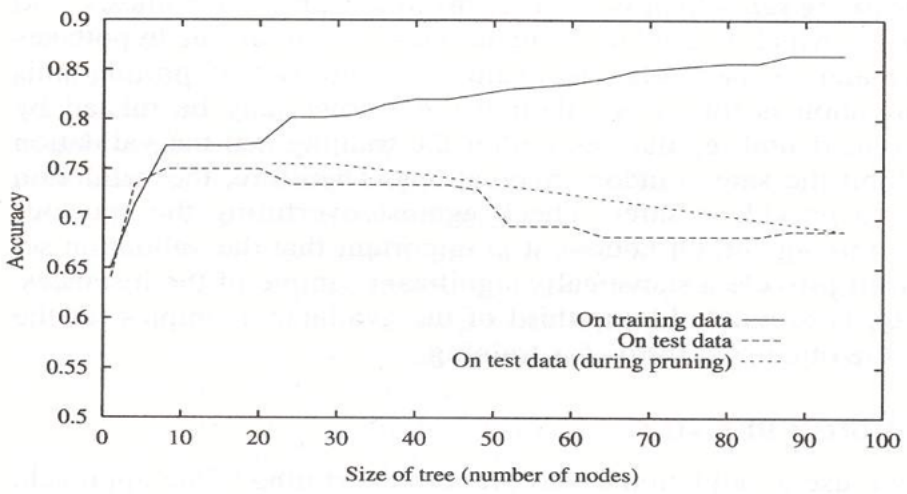


Figure 1: Effect of overfitting and pruning on classification performance in a sample problem. The image is from Tom Mitchell's ML textbook.