I2ML :: CHEAT SHEET

The **I2ML**: Introduction to Machine Learning course offers an introductory and applied overview of "supervised" Machine Learning. It is organized as a digital lecture.

Introduction to CART

CART refers to Classification And Regression Tree

Basic Idea:

1. Split the data into two parts according to one of the features.
2. Binary Splits are constructed top-down.
3. Constant prediction in each leaf (numerical, class label, probability vector)
4. An observation will end up in exactly one leaf node

Trees as an additive model:

trees divide the feature space $\mathcal{X}$ into **rectangular regions**:

$$f(\mathbf{x}) = \sum_{m=1}^{M} c_m \mathbb{I}(x \in Q_m),$$

a tree with M leaf nodes defines M "rectangles" Q_m.
c_m is the predicted numerical response, class label or class distribution in the respective leaf node.

Splitting Criteria

Use **empirical risk minimization** for finding good splitting rules to define the tree. Splitting criteria for trees are usually defined in terms of "impurity reduction".

Formalization:

The risk $\mathcal{R}(\mathcal{N})$ for a leaf is simply the average loss for the data assigned to that leaf under a given loss function L:

$$\mathcal{R}(\mathcal{N}) = \frac{1}{|\mathcal{N}|} \sum_{(x,y) \in \mathcal{N}} L(y, c)$$

Let $\mathcal{N} \subseteq \mathcal{D}$ be the data that is assigned to a terminal node $\mathcal{N}$ of a tree.
Let c be the predicted constant value for the data assigned to $\mathcal{N}$: $\hat{y} \equiv c$ for all $(x, y) \in \mathcal{N}$.
The prediction is given by the optimal constant $c = \arg\min_c \mathcal{R}(\mathcal{N})$

Splitting criteria for regression:

For regression trees usually L_2 loss is used:

$$\mathcal{R}(\mathcal{N}) = \frac{1}{|\mathcal{N}|} \sum_{(x,y) \in \mathcal{N}} (y - c)^2$$

The best constant prediction under L_2 is the mean

Splitting criteria for classification:

Typically uses Brier score or Bernoulli loss as loss functions Predicted probabilities in node $\mathcal{N}$ are the class proportions in the node:

$$\hat{\pi}_k^{(\mathcal{N})} = \frac{1}{|\mathcal{N}|} \sum_{(x,y) \in \mathcal{N}} \mathbb{I}(y = k)$$

Gini Impurity:

Minimizing the Brier score is equivalent to minimizing the Gini impurity

$$I(\mathcal{N}) = \sum_{k=1}^{g} \hat{\pi}_k^{(\mathcal{N})} (1 - \hat{\pi}_k^{(\mathcal{N})})$$

Entropy Impurity:

Minimizing the Bernoulli loss is equivalent to minimizing entropy impurity

$$I(\mathcal{N}) = -\sum_{k=1}^{g} \hat{\pi}_k^{(\mathcal{N})} \log \hat{\pi}_k^{(\mathcal{N})}$$

Brier score and Bernoulli loss are more sensitive to changes in the node probabilities, and therefore often preferred than misclassification loss.

Growing a Tree

We start with an empty tree, a root node that contains all the data. Trees are then grown by recursively applying *greedy* optimization to each node $\mathcal{N}$. Greedy means **exhaustive search**

1. All possible splits of $\mathcal{N}$ on all possible points t for all features x_j are compared in terms of their empirical risk $\mathcal{R}(\mathcal{N}, j, t)$.
2. Training data is then distributed to child nodes according to the optimal split and the procedure is repeated in the child nodes.

Splits are usually placed at the mid-point of the observations they split

Split computation

Monotone feature transformations:

Monotone transformations of one or several features will only change the numerical value of the split point.

CART: Nominal Features

1. A split on a nominal feature partitions the feature levels:
$$x_j \in \{a, c, e\} \leftarrow \mathcal{N} \rightarrow x_j \in \{b, d\}$$
2. For a feature with m levels, there are about 2^m different possible partitions of the m values into two groups ($2^{m-1} - 1$ because of symmetry and empty groups).
3. Searching over all these becomes prohibitive for larger values of m.

For $0-1$ responses, in each node:

1. Calculate the proportion of 1-outcomes for each category.
2. Sort the categories according to these proportions.

For continuous responses, in each node:

1. Calculate the mean of the outcome in each category
2. Sort the categories by increasing mean of the outcome

Missing Feature values:

CART often uses the so-called **surrogate split** principle to automatically deal with missing values in features used for splits during prediction.

Stopping Criteria & Pruning

Overfitting Trees:

1. Problem 1: longer time is needed and this can be solved by using stopping criteria.
2. Problem 2: very complex trees with lots of branches and leaves will *overfit the training data*. Stopping criteria is used to solve this.
3. Problem 3: it is very hard to tell where to stop while growing the tree. That is why pruning is used to resolve this problem.

Note:

1. **Advantages:** CART performs automatic feature selection. It is fast and scales well larger data. Not much preprocessing is required. It can also model discontinuities and non-linearities
2. **Disadvantages:** CART has linear dependencies. Prediction functions of trees are never smooth as they are always step functions. It is empirically not the best predictor. There is high instability (variance) of the trees.