Supervised Learning

Viktor

January 4, 2022

In a supervised learning problem we have a learning set $\mathcal{S}$ of N observations, $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{N}$. Where y_i are realizations of the so called target Y, which is a random variable (RV) with sample space (i.e., the space of all possible outcomes) Ω_t, $\Omega_t \subseteq \mathbb{R}$. The x_i are called feature vectors. They are realizations of a p-dimensional RV $X = (X_1, \dots, X_p)^T$ with sampel space Ω and associated sample spaces of the scalar RVs X_j, $\Omega_j \subseteq \mathbb{R}$. The learning problem is then finding a function

$$f : \Omega \to \Omega_t \tag{1}$$

such that $f(X)$ approximates Y as good as possible. f is also called model and is sometimes written as $f_{\mathcal{S}}$ to stress the dependence on the learning set.

1 Loss function

Assessing a model if often done with by means of a loss function,

$$L : \Omega_t \times \Omega_t \to \mathbb{R} \tag{2}$$

For classification the zero-one loss is often used $L(y', y) = \mathbb{1}(y' \neq y)$, where $\mathbb{1}$ is the indicator function. For regression problems the squared error loss is often used $L(y', y) = (y' - y)^2$ The expectation value of the loss function,

$$E_{XY}\left(L(Y, f(X))\right) = \iint L(y, f(x)) p(x, y)\, dx\, dy \tag{3}$$

is also called expected *prediction error* or *generalization error*. In practice the generalization error is used for model selection and assessment. However, the joint distribution p_{XY} is hardly ever known. The generalization error is therefore estimated by e.g., cross validation [7].

Bayes model. A model is called Bayes model [10] f_B if for any model f and learning Set $\mathcal{S}$ $E_{XY}\left(L(Y, f_B(X))\right) \leq E_{XY}\left(L(Y, f_{\mathcal{S}}(X))\right)$. That is the model with the smallest error any supervised model can attain. The generalization error of the Bayes model as also called *residual error.* Rewrite the expected loss 3 in terms of conditional probabilities

$$E_{XY}\left(L(Y, f(X))\right) = \int p(x) \int L(y', f(x)) p(y' \mid x)\, dy'\, dx,$$

and observe that the inner integrand is a function in x. Defining it's point wise minimum according to

$$f_B : \; x \mapsto \underset{y \in \Omega_t}{\operatorname{argmin}} \int L(y', y) p(y' \mid x)\, dy', \tag{4}$$

gives the Bayes model (by construction). Expression 4 may be also written in terms of conditional expectations,

$$f_B(x) = \underset{y \in \Omega_t}{\operatorname{argmin}}\; E_{Y \mid X=x}\left(L(Y, y)\right). \tag{5}$$

Example. In a regression problem with squared error loss the Bayes model is the conditional expectation value[1],

$$f_B(x) = \int y' p(y' \mid x)\, dy'. \tag{6}$$

In a classification problem with zero-one loss the Bayes model is the most probable class [2],

$$f_B(x) = \underset{y \in \Omega_t}{\operatorname{argmax}}\; P(y \mid x) \tag{7}$$

Remark. In the setup for the Bayes model we assumed to know the joint pdf $p(x, y)$. But, once we knew the exact pdf we know everything. In particular, we could directly use the canonical model by taking the most probable state, $f(x) = \operatorname{argmax}_y p(y|x)$. In the above examples this was only true for a classification problem with zero-one loss but not true for a regression problem with squared error loss. The reason is that the error function is still there and encodes additional information. In particular, in the regression problem the error penalty is quadratic, thus leading to an other Bayes model than the most probable state.

[1]Because $\partial y \int (y' - y)^2 p(y'|x)\, dy' \overset{!}{=} 0 \Rightarrow y = \int y' p(y' \mid x)\, dy' = E_{Y|X=x}(Y)$, where the fact that $\int p(y' \mid x)\, dy' = 1$ was used because $p(y'|x)$ is a pdf in y'.

[2]$\underset{y \in \Omega_t}{\operatorname{argmin}} \sum_{y' \in \Omega_t} \mathbb{1}(y' \neq y) p(y' \mid x) = \underset{y \in \Omega_t}{\operatorname{argmin}} \sum_{y' \in \Omega_t} (1 - \mathbb{1}(y' = y)) p(y' \mid x) = \underset{y \in \Omega_t}{\operatorname{argmin}}(1 - \sum_{y' \in \Omega_t} \mathbb{1}(y' = y)) p(y' \mid x) = \underset{y \in \Omega_t}{\operatorname{argmin}}(1 - p(y|x)) = \underset{y \in \Omega_t}{\operatorname{argmax}}\, p(y \mid x)$

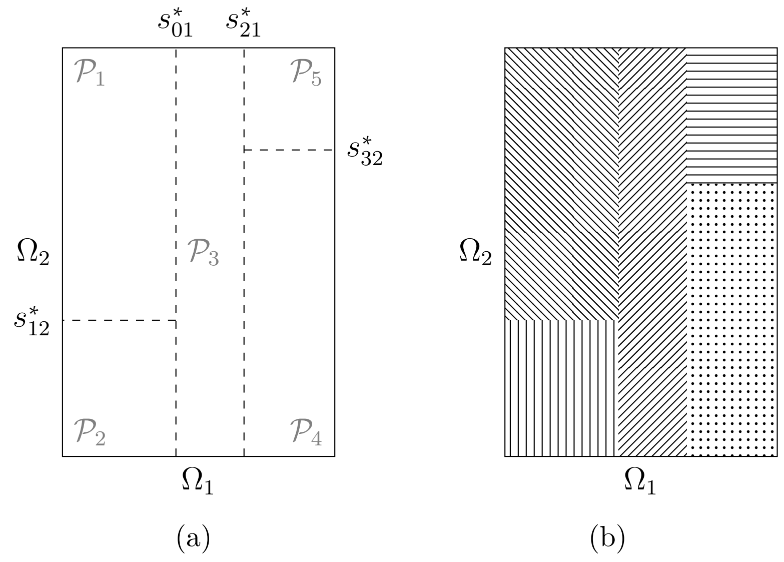

Figure 1: Illustration of sample space partitioning. (a) Partition of a two dimensional sample space $\Omega = \Omega_1 \times \Omega_2$ into $\{\mathcal{P}_1, \ldots \mathcal{P}_5\}$ regions according to splitting points $\{s^*_0, \ldots s^*_3\}$. Once the partition is obtained each each $\mathcal{P}_i$ predicts a constant output value (b).

Two learning paradigms. One can distinguish two supervised learning paradigms: the generative and the discriminative approach [1, Chapter 13]. The generative approach tries to model the whole pdf. Classification is then considered as a decision problem (based on the loss function), which is carried out *a posteriori*. The discriminative approach tries to directly learn the class labels using the loss function *a priori* in the learning step. Most standard supervised learning techniques fall into the latter category.

2 Decision Trees

Decision trees [6, 11, 10, 12] are supervised learning methods and and are applicable to both, classification and regression. In essence, they partition sample space and predict by applying assignment rules, i.e., the majority vote (classification) or the mean (regression) over the training set within that partition (Figure 1). Recall,

Definition 1 (Partition). *A partition $\mathcal{P}$ is a family of sets $\mathcal{P} = \{\mathcal{P}_i \mid i \in I\}$ such that*

$$\Omega = \dot{\bigcup_{i \in I}} \mathcal{P}_i, \ \ \mathcal{P}_i \neq \varnothing$$

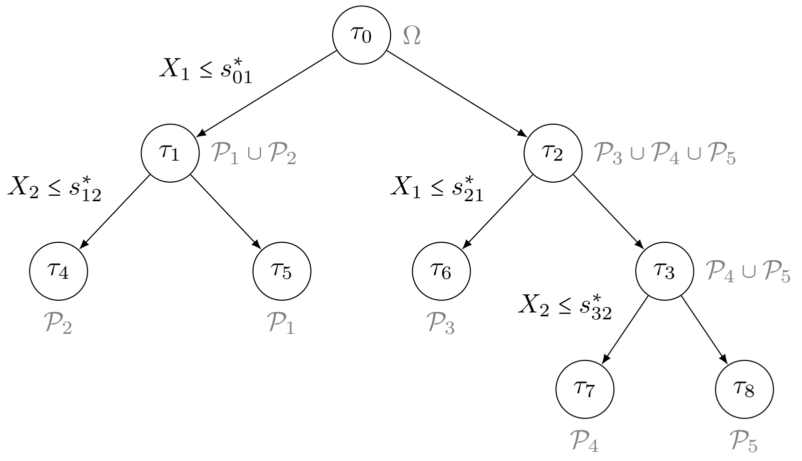

Figure 2: Illustration of a binary decision tree for a single sample with features (X_1, X_2). At each node the associated subspace of sample space is shown in gray. The sample is passed through the decision tree until it gets assigned to one of the leaf nodes $\{\tau_3, \tau_4, \tau_5, \tau_7, \tau_8\}$ that are associated with the sample space partition $\{\mathcal{P}_1, \dots \mathcal{P}_5\}$.

In practice, the partition is determined by a sequence of splitting points $(s^*_{\tau i_\tau})_{\tau=0,\dots,k}$ *in parallel* feature $i_\tau \in \{1, \dots, p\}$ in a *non-commutative* way. Learning corresponds to finding the partition in sample space.

The partitioning process can be also understood as a (decision) tree structure. Start with a root node τ_0 which corresponds to Ω. Each node τ then splits sample space according to the splitting point $s^*_{\tau i}$ (Figure 2). Note that this is a *binary* decision tree, i.e., there are two child nodes at each split node. These are determined if the splitting criterion (e.g. $X_1 \leq s^*_{01}$ for the 0-th split) is fulfilled (left child node) or not (right child node). At the terminal (leaf) nodes there is no further split. The set of all terminal leaf nodes correspond to the partition of sample space.

We are aiming to build a tree where each node further partitions sample space. To this end, we consider nodes of the form $\tau = (\Omega_\tau, \mathcal{S}_\tau, \mathcal{P}_\tau)$. Where $\Omega_\tau \subset \Omega$ is a subspace, which is further partitioned by τ according to $\mathcal{P}_\tau$. We call Ω_τ *active sample space* at node τ. Furthermore we call the set $\mathcal{S}_\tau = \{(x, y) \mid (x, y) \in \mathcal{S},\ x \in \Omega_\tau\}$ *active training set* at node τ. Following the idea of sequentially partitioning sample space gives raise to

Definition 2 (Decision tree)**.** *A decision tree is a directed acyclic, singly connected graph with nodes* $\{\tau \mid \tau = (\Omega_\tau, \mathcal{S}_\tau, \mathcal{P}_\tau)\}$*, where* Ω_τ *is the active sample set,* $\mathcal{S}_\tau$ *is the active training set and* $\mathcal{P}_\tau = \{\mathcal{P}_{\tau i} \mid i \in I_\tau\}$ *is the partition of* Ω_τ *such that:*

1. *The root node is of the form* $\tau_0 = (\Omega, \mathcal{S}, \mathcal{P}_{\tau_0})$

2. τ *is leaf node* $\Leftrightarrow |\mathcal{P}_\tau| = 1$

3. τ *has* $|\mathcal{P}_\tau|$ *child nodes for* $|\mathcal{P}_\tau| > 1$*. Each child node* τ_c *has active sample set* $\Omega_{\tau_c} = \mathcal{P}_{\tau c},\ c \in I_\tau$*.*

Apparently, this definition assures that for a decision tree the active sets of all leaf nodes $\{\Omega_\tau \mid \tau \text{ is leaf node}\}$ is indeed a partition of Ω.

Examples

- A binary decision tree is a decision tree where each non-terminal node has $|\mathcal{P}_\tau| = 2$. That is, each non-terminal node has exactly two child nodes.

- Consider a node $\tau = (\Omega_\tau, \mathcal{S}_\tau)$ where all features in S_τ are ordered. Define a splitting point $s^*_{\tau i}$ along feature i. This induces a partition $\mathcal{P}_\tau = \{\mathcal{P}_l, \mathcal{P}_r\}$ with $\mathcal{P}_l = \{x \mid x \in \Omega_\tau,\ x_i \le s^*_{\tau i}\}$ and $\mathcal{P}_r = \{x \mid x \in \Omega_\tau,\ x_i > s^*_{\tau i}\}$. Hence there are two child τ_l, τ_r of node τ with active sample sets $\Omega_{\tau_l} = \mathcal{P}_l$ and $\Omega_{\tau_r} = \mathcal{P}_r$, respectively.

Within this point of view learning consists of 1) finding the decision tree structure (i.e., the node partitions $\mathcal{P}_\tau$) and 2) applying assignment rules at the leaf nodes.

2.1 Assigment rules on leaf nodes.

Assume that we have learned in some way the decision tree and thus the partition of sample space (vie the leaf nodes). At the leaf nodes decision trees assign a constant value using the Bayes model. To this end we introduce a latent variable for the leaf node membership.

$$p(y, x, \tau) = p(y \mid x, \tau)\, p(\tau, x) \tag{8}$$
$$\approx p(y \mid \tau)\, p(\tau, x) \tag{9}$$

where the approximation accounts for the basic idea that the prediction is only based on the partition x falls into and not on its specific value. Plugging this factorization into the expected loss function gives,

$$E_{XY\mathrm{T}}\left(L(Y, f(\mathrm{T}))\right) = \sum_\tau \int_\Omega dx\, p(\tau, x) \int dy\, L(y, f(\tau))\, p(y \mid \tau) \tag{10}$$
$$= \sum_\tau p(\tau) \int dy\, L(y, f(\tau))\, p(y \mid \tau')\,, \tag{11}$$

where the model depends only on the node, $f(\tau)$, which is a consequence of factorization (9). Expression (11) is the expected loss over the whole model (i.e, including all nodes via the expectation value). By factorization (9) it does not depend on x.

Since we are interested in the assignment rule at a given leaf node, we would like to compute the expected loss function conditioned on leaf node τ,

$$E_{XY|\mathrm{T}}\left(L(Y, f(\mathrm{T}))\right) = \int_{\Omega} dx\, p(x \mid \tau) \int dy\, L(y, f(\tau))\, p(y \mid \tau) \tag{12}$$

$$= \int dy\, L(y, f(\tau))\, p(y \mid \tau')\,, \tag{13}$$

From (13) or (11) and following the derivations for the Bayes model for classification (7) and regression (6) we therefore get as predictions at leaf node τ:

- **Classification with zero-one loss**

  $$\hat{y}_\tau = \operatorname*{argmax}_{y \in \Omega_t} P(y \mid \tau) \tag{14}$$

  $$\approx \operatorname*{argmax}_{y \in \Omega_t} \frac{N_{y\tau}}{N_\tau} \tag{15}$$

  where $N_{y\tau} = |\{(x', y') \mid (x', y') \in \mathcal{S}_\tau,\ y' = y\}|$ is the count of class labels in the active training set at node τ and $N_\tau = |\mathcal{S}_\tau|$ is the size of the active training set.

- **Regression with squared loss**

  $$\hat{y}_\tau = \int y' p(y' \mid \tau)\, dy' \tag{16}$$

  $$\approx \frac{1}{N_\tau} \sum_{i=1}^{N_\tau} y_i. \tag{17}$$

So in classification problem, the class label is predicted by the maximum frequentist probability and in a regression problem the node value is given by the empirical mean. The approximations refer to the frequentist estimates of the conditional probability (classification) and expectation value (regression) based on the active training sets at node τ.

In summary, for a prediction at a leaf node we used approximation (9) and frequentist approximations (15) and (17). The first approximation neglects the specific value of x and rather predicts globally on the corresponding node. The second set of approximations has two implications [10, Chapter 3.4]: First the frequentist approximation leads to unreliable estimates for nodes with small active training sets (in particular, for nodes, which carry only one active training example). Second the estimates are based on the training data. Therefore we are taking the optimal decision not over the generalization error (as the Bayes model would do) but rather over the training set error. This leads to the tendency of deep decision trees to severely overfit.

2.2 Learning the tree structure.

Among all possible decision trees for a learning task, training corresponds to finding that decision tree that minimizes the training set error. If there are several such candidates then Occam's Razor dictates to choose the smallest decision tree [3], which is also simpler to interpret. However, finding the smallest tree that minimizes the training set error is an NP-complete problem [8]. Therefore, one needs to resort to heuristics. In essence, the tree is greedily/iteratively grown employing node splitting (partitioning) rules until a stopping criterion is met. To this end the following concepts are going to be introduced:

- **Node partitioning rules** Instruction for finding the partitions $\mathcal{P}_\tau$ and a criterion for finding the "best" partition (goodness of split $\Delta i(\tau)$).
- **Greedy assumption:** grow the tree such that the goodness of split gets locally maximized.
- **Canonical stopping criteria:** This is used to determine whether a given node is split further or is a terminal (leaf) node. Consider a node τ with active training set $\mathcal{S}_\tau$. There are two cases that make τ a leaf node. First, if the target values of $\mathcal{S}_\tau$ are locally constant,

  $$y = y' \ \forall (x, y), (x', y') \in \mathcal{S}_\tau. \tag{18}$$

  Second, if the features of $\mathcal{S}_\tau$ are locally constant,

  $$x = x' \ \forall (x, y), (x', y') \in \mathcal{S}_\tau. \tag{19}$$

  As we are aiming to minimize the training set error, in the first case it does not make sense to further refine sample space, whereas in the second case sample space cannot be further partitioned.
- **Early stopping** aims prevent to grow too deep trees and thus overfitting. This approach introduces additional stopping criteria (maximum depth of terminal node, minimum number of training examples in a leaf node, threshold for the total decrease in impurity, minimum number of training example in each child node upon node split)

- **Pruning:** this is an alternative to early stopping. First the tree is fully developed and then nodes are sequentially removed that degrade the generalization error. While pruning typically works better for decision trees, pruning in general is no longer required for tree based ensemble methods.

This allows to formulate the general form of a binary decision tree algorithm as given in algorithm 1.

Algorithm 1 Greedy induction of a binary decision tree with early stopping

```
 1: procedure BUILD DECISION TREE ($\mathcal{S}$)
 2:     Create a decision tree $f$ with root node $\tau_0$
 3:     Create an empty stack $\mathcal{N}$ of *open* nodes $(\tau, \mathcal{S}_\tau)$
 4:     $\mathcal{N}$.PUSH$(\tau_0, \mathcal{S})$
 5:     while $\mathcal{N}$ is not empty do
 6:         $\tau, \mathcal{S}_\tau = \mathcal{N}$.POP()
 7:         if the stopping criterion is met for $\tau$ then
 8:             $\hat{y}_\tau$ = some constant value
 9:         else
10:             Find split in $\mathcal{S}_\tau$:
                    $s^* = \operatorname{argmax}_{s \in Q} \Delta i(s, \tau)$
11:             Partition $\mathcal{S}_\tau = \mathcal{S}_{\tau_l} \cup \mathcal{S}_{\tau_r}$ according to $s^*$
12:             Create child nodes $\tau_l, \tau_r$ of $\tau$
13:             $\mathcal{N}$.PUSH$(\tau_l, \mathcal{S}_{\tau_l})$
14:             $\mathcal{N}$.PUSH$(\tau_r, \mathcal{S}_{\tau_r})$
        return $f$
```

2.3 Impurity measures for classification

Intuitively a ”good” split is a split that makes the child nodes purer. That is, the probability of a sample falling into a child node should be sharply peaked at one class label. Rather than working with the concept of purity, the complementary concept is used[4]:

Definition 3 (Impurity function). *An impurity function is a function $\phi : M \to \mathbb{R}$, where $M = \left\{p \mid p \in [0,1]^J, \ \sum_{j=1}^J p_j = 1\right\}$ such that*

- *ϕ is a maximum only the point $\left(\frac{1}{J}, \frac{1}{J}, \ldots, \frac{1}{J}\right)$*

- *ϕ is a minimum at the point* $(1, 0, \ldots, 0)$
- *ϕ is symmetric*

By symmetry also all permutations of $(1, 0, \ldots, 0)$ are minimizers. This concept is applied to the decision tree

Definition 4 (Node impurity measure)**.** *Given an impurity function and decision tree with node τ, the impurity measure at node τ is defined as,*

$$i(\tau) = \phi(p(c_1 \mid \tau), p(c_2 \mid \tau), \ldots, p(c_J \mid \tau)),$$

where $p(c_1 \mid \tau),\ c_i \in \Omega_t$ are the conditional class probabilities.

Having defined the impurity measure at a given node we would like to obtain a measure on how beneficial it is to add further child nodes (i.e., further partition sample space).

Definition 5 (Goodness of split)**.** *Let $\mathcal{C}_\tau = \{\tau' \mid \tau'$ is child of $\tau\}$. Then the goodness of split at node τ is defined as*

$$\Delta i(\tau, \mathcal{C}_\tau) = i(\tau) - \sum_{\tau' \in \mathcal{C}_\tau} p(\tau' \mid \tau) i(\tau')$$

Note the difference between $p(\tau' \mid \tau)$ and $p(\tau' \mid \tau, x)$. While in the latter expression there is exactly one child node that has probability one (by construction of a decision tree), the former approach is less trivial and is typically approximated by the proportions of the active training sets of the child nodes with respect to the parent node.

So the basic idea of Definition 5 is to take as measure the expected reduction of the node impurity measure upon adding further child nodes. Intuitively we would expect $\Delta i(\tau) \geq 0$. That is any addition of child nodes leads to a decrease in the goodness of split and in the worst case it does not decrease the goodness of split. The class of impurity functions that fulfill this intuition may be characterized by [4, Appendix of Chap. 4], [12]:

Theorem 1. *If the impurity function is strictly concave, then the goodness of split*

$$\Delta i(\tau, \mathcal{C}_\tau) \geq 0,$$

with equality if, and only if, $p(c \mid \tau') = p(c \mid \tau),\ \forall \tau' \in \mathcal{C}_\tau$ and $\forall c \in \Omega_\tau$.

Proof. First show that $\sum_{\tau' \in \mathcal{C}_\tau} p(\tau' \mid \tau) p(c \mid \tau') = p(c \mid \tau)$. Let τ' be a child node of τ. Then, $p(\tau' \mid \tau) p(c \mid \tau') = \frac{p(\tau', \tau)}{p(\tau)} p(c \mid \tau') = \frac{p(\tau \mid \tau') p(\tau')}{p(\tau)} p(c \mid \tau') =$

$p(c, \tau' \mid \tau)$, because $p(\tau \mid \tau') = 1$ as τ' is child of τ. Thus for the sum $\sum_{\tau' \in \mathcal{C}_\tau} p(c, \tau' \mid \tau) = p(c \mid \tau)$. Apparently $p(\tau' \mid \tau) \geq 0$ and $\sum_{\tau' \in \mathcal{C}_\tau} (\tau' \mid \tau) = 1$. So we can use Jensens inequality to show the inequality statement:

$$
\begin{aligned}
\sum_{\tau' \in \mathcal{C}_t} p(\tau' \mid \tau) i(\tau') &= \sum_{\tau' \in \mathcal{C}_t} p(\tau' \mid \tau) \phi(p(c_1 \mid \tau'), \ldots, p(c_J \mid \tau')) \\
&\leq \phi \left(\sum_{\tau' \in \mathcal{C}_t} p(\tau' \mid \tau) p(c_1 \mid \tau), \ldots, \sum_{\tau' \in \mathcal{C}_t} p(\tau' \mid \tau) p(c_J \mid \tau) \right) \\
&= \phi\left(p(c_1 \mid \tau), \ldots, p(c_J \mid \tau)\right) \\
&= i(\tau).
\end{aligned}
$$

Now assume that $p(c \mid \tau') = p(c \mid \tau)$, $\forall \tau' \in \mathcal{C}_\tau$ and $\forall c \in \Omega_\tau$. Then $\sum_{\tau' \in \mathcal{C}_t} p(\tau' \mid \tau) i(\tau') = i(\tau) \sum_{\tau' \in \mathcal{C}_t} p(\tau' \mid \tau) = i(\tau)$. Therefore $\Delta i(\tau, \mathcal{C}_\tau) = 0$.

Todo:opposite direction □

The most common strictly concave impurity functions encountered in practice are:

$$i(\tau) = -\sum_{c \in \Omega_t} p(c \mid t) \log_2 p(c \mid t) \quad \text{(Shannon Entropy)}, \tag{20}$$

$$i(\tau) = \sum_{c \in \Omega_t} p(c \mid \tau)(1 - p(c \mid \tau)) \quad \text{(Gini Index)}. \tag{21}$$

For practical calculations the empirical probabilities based on the training data are taken. Note that the goodness of split of the Shannon impurity is just the information gain.

Comparing the Gini and Shannon impurity [5], Gini prefers splits that put the largest class in one pure node while Shannon prefers to balance the size of the child nodes. For problems with small number of classes they should give similar results. However for many classes, Gini may produce trees that are too unbalanced high up in the tree, while the entropy lacks uniqueness. Note that there are many other impurity measures proposed in the literature [9, 13, 2].

Empirically, as long as the impurity function is reasonable, it rather effects the structure of the decision tree (and thus the interpretation) rather than the accuracy.

Remark. There is another strictly convex impurity function that suggests itself. Taking the conditional expected loss (13) for a decision tree at node

τ with the corresponding Bayes model for the zero-one loss (14),

$$\begin{aligned} i(\tau) &= \sum_{c\in\Omega_t} L(c,\hat{c}_\tau)p(c \mid \tau) \\ &= \sum_{y\in\Omega_t} \mathbb{1}(c \neq \hat{c}_\tau)p(c \mid \tau) \\ &= 1 - p(\hat{c}_\tau \mid \tau) \\ &= 1 - \max_{c\in\Omega_t} p(c \mid \tau) \quad \text{(Misclassification error).} \end{aligned} \tag{22}$$

The corresponding goodness of split is then the reduction of the misclassification error upon adding child nodes. Despite the natural attractiveness of this impurity function it has two serious flaws: 1) it could lead to scenarios where all possible node splits have zero goodness of split and 2) it does not account for beneficial shifts in the probability distributions of the child nodes. The first point may happen when all child nodes have the same majority class [4, 10] and thus, making this impurity function useless. The second point is problematic within the greedy tree growing algorithm. Thus, the misclassification error seems to short-sighted upon growing a tree [4, 7]. On the other hand the misclassification rate is useful in pruning.

2.4 Impurity measures for regression

For the regression problem impurity functions seem to be less well characterized in the literature. The reason is that the regression pendant of (22) is less problematic (in classification the main problems are rooted in the few number of classes). Computing the expected loss (13) for a decision tree at node τ with the corresponding Bayes model for the squared loss gives the *impurity function for a regression tree*: (16)

$$\begin{aligned} i(\tau) &= \int dy\, L(y, f(x))p(y \mid \tau) \\ &= \int dy\,(y - \hat{y}_\tau)^2\, p(y \mid \tau), \quad \hat{y}_\tau = \int y' p(y' \mid \tau)\, dy'. \end{aligned} \tag{23}$$

Remark. There is an interesting relation between the expected loss (23) and the Gini index (21) for binary classification tasks. Consider a two class problem, $c_1 = 0$ and $c_2 = 1$. The Gini index is given by $i_G(\tau) = 2\,p(c_1|\tau)\,p(c_2|\tau)$. On the other hand, if we were treating the classification problem as a regression, the expected loss (23) translates into $i_R(\tau) = \sum_i (c_i - \hat{c})^2 p(c_i \mid \tau)$ with $\hat{c} = \sum_i c_i\, p(c_i \mid \tau) = p(c_2 \mid \tau)$. Therefore, $i_R = p(c_1|\tau)\,p(c_2|\tau)$. Therefore,

$$i_R(\tau) = i_G(\tau)/2.$$

So the Gini index for a two class problem is up to a factor of two equivalent to the expected loss in the regression problem. In this sense the impurity function (23) is applicable to both, a regression problem and a binary classification problem.

2.5 Generating bianary node splits

Once the impurity function is defined the optimal node splits may be found. Node splits are just partitions of the active sample space. While partitions are quite generic in practice often binary node splits are used.

Categorical features For a categorical feature with L levels $\{l_1, \ldots, l_L\}$ there are in principle $2^{L-1}-1$ partitions. For the binary classification problem this may be reduced by

Theorem 2. *Suppose a binary classification problem and given a stricly concave impurity function. For a categorical feature order the levels in increasing $p(c = 1 \mid l = l_i)$. Then a partition of the form $\{l_1, \ldots, l_k\} \cup \{l_k + 1, \ldots, l_L\}$ maximises the goodness of split.*

Proof. A proof may be found in Refs. [12, 4]. □

Intuitively, all level leading to high probabilities of one class should be put together. Unfortunately theorem 2 does not extend to multi class problems. In practice exhaustive searches are applied for features with $L \leq 10$ and random sampling for features with more categories [10].

2.6 Pruning

Early stopping is one heuristics to avoid over fitting by growing too large trees. Pruning tackles this problem by first fully growing a tree and then removing branches in a second step. While there are several approaches in the literature to pruning we consider cost-complexity pruning [4].

Definition 6 (Pruning)**.** *Given a tree T*

- *A subtree of T is a subgraph of T that is also a tree.*
- *A rooted subtree of T is a subtree of T with the same root as T.*
- *A pruned subtree is a rooted subtree.*

3 Hierarchical mixture of experts

Very similar to decision trees but with different notation [7]. Non terminal nodes are called gates and terminal nodes are called experts. The experts are classiers or regressors on the whole dataset (in contrast to decision trees). Gates make a soft probability split whereas decision tree make a hard one (based on splitting criteria).

References

[1] David Barber. *Bayesian Reasoning and Machine Learning.* Cambridge University Press, New York, NY, USA, 2012.

[2] Fernando Berzal, Juan-Carlos Cubero, Fernando Cuenca, and María J. Martín-Bautista. On the quest for easy-to-understand splitting rules. *Data and Knowledge Engineering*, 44(1):31 – 48, 2003.

[3] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Information Processing Letters*, 24(6):377 – 380, 1987.

[4] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth and Brooks, Monterey, CA, 1984. new edition [**?**]?

[5] Leo Breiman. Technical note: Some properties of splitting criteria. *Machine Learning*, 24(1):41–47, Jul 1996.

[6] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[7] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning.* Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[8] Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15 – 17, 1976.

[9] Murat Kayri and İsmail Kayri. The comparison of gini and twoing algorithms in terms of predictive ability and misclassification cost in data mining: An empirical study. *International Journal of Computer Trends and Technology*, 27:21–30, 09 2015.

[10] Gilles Louppe. Understanding random forests: From theory to practice, 2014.

[11] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[12] Brian D. Ripley and N. L. Hjort. *Pattern Recognition and Neural Networks.* Cambridge University Press, New York, NY, USA, 1st edition, 1995.

[13] Carolin Strobl, Anne-Laure Boulesteix, and Thomas Augustin. Unbiased split selection for classification trees based on the gini index, 2005.