Supervised Learning

December 23, 2018

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 $X = (X_1, \ldots, X_p)^T$ 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 [4].

Bayes model. A model is called Bayes model [6] 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

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

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

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

Example. In a regression problem with squared error loss Bayes model is the 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],

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

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

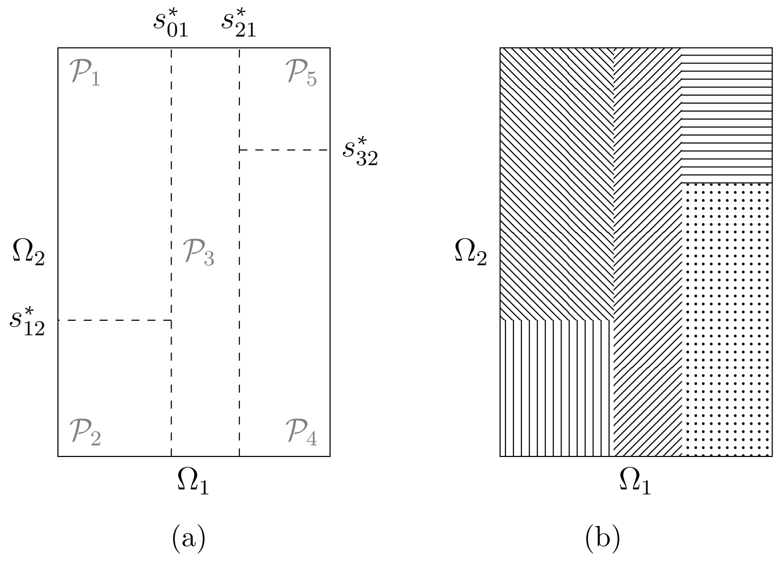

Figure 1: Illustration of sample space partitioning. (a) Partition of a two dimensional sample space $\Omega = \Omega_1 \times \Omega_2$ into $\{\mathcal{P}_1, \dots \mathcal{P}_5\}$ regions according to splitting points $\{s^*_0, \dots s^*_3\}$.

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 [3, 7, 6] 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*

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

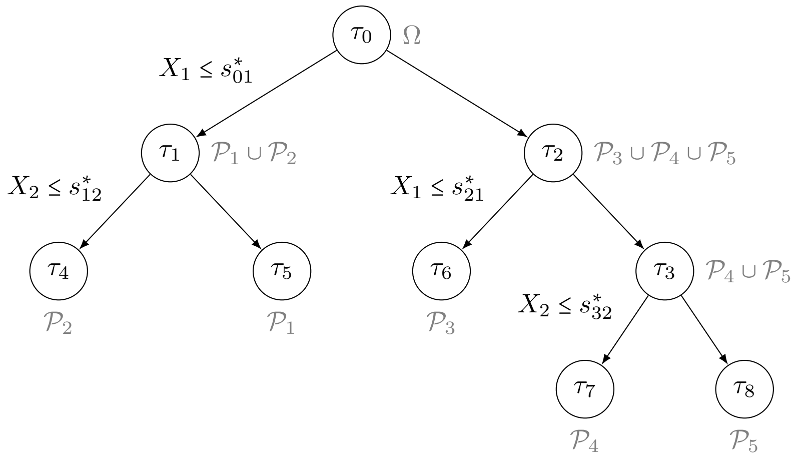

Figure 2: Illustration of a binary decision tree for a single sample with features (X_1, X_2). At each node the associated subspace of sample space is shown in gray. The sample is passed through the decision tree until it gets assigned to one of the leaf nodes $\{\tau_3, \tau_4, \tau_5, \tau_7, \tau_8\}$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

In practice, the partition is determined by a sequence of splitting points $(s^*_{\tau i_\tau})_{\tau=0,\ldots,k}$ *in parallel* feature $i_\tau \in \{1, \ldots, p\}$ 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 sample 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* $\Omega_{\tau_c} = \mathcal{P}_{\tau c},\ c \in \mathcal{I}_\tau$*.*

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 $\mathcal{P}_l = \{x \mid x \in \Omega_\tau,\ x_i \leq s^*_{\tau i}\}$ 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.

Assigment rules on leaf nodes. At the leaf nodes decision trees assign a constant value. Warning: using the frequentists approximation to the pdf leads to unreliable estimates for small nodes. And in particular, for nodes, which carry only one active training example. Indeed, this is one reason why deep decision trees tend to severely overfit.

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 [2], which is also simpler to interpret. However, finding the smallest tree that minimizes the training set error is an NP-complete problem [5]. 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** Criterion for finding the partitions $\mathcal{P}_\tau$.

- **Greedy assumption:** grow the tree such that the decrease in the impurity measure 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,

  $$y = y' \ \forall (x, y), (x', y') \in \mathcal{S}_\tau. \tag{8}$$

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

  $$x = x' \ \forall (x, y), (x', y') \in \mathcal{S}_\tau. \tag{9}$$

  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.
- **Pre-pruning** 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)
- **Post-pruning:** this is an alternative to pre-pruning. First the tree is fully developed and then nodes are sequentially removed that degrade the generalization error. While post-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.

Impurity measures

- Cross entropy
- Gini index
- Misclassification

Decison Tree Algorithms

- CART
- C4.5

Algorithm 1 Greedy induction of a binary decision tree with pre-pruning

```
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 $f$
```

- Patient rule induction method (PRIM)
- Multivariate adaptive regression splines (MARS)
- Hierarchical mixture of experts (HME)
- Bayesian approaches (Denison et al., 1998; Chipman et al., 1998, 2002, 2010)

Disadvantages: split only parallel to features.

Categorical predictors

Missing values Instead of the standard procedures with missing values (throwing away datapoints or fetures with missing values or imputing) decision trees offer additional possibilities:

- ”Missing” as a category. This is applicable to categorical predictors
- Surrogate variables.

3 Ensemble methods

- Bagging
- Boosting
- Random Forest
- AdaBoos
- XGboost

4 Outlook

- Additive models
- Support Vector machines
- Logistic regression

5 Bias-Variance Decomposition

6 Model selection

7 Feature selection

References

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

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

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

[4] 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.

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

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

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