

Machine Learning for Industry

Lecture 2 - Regression Trees and Beyond

Mattias Villani

Department of Computer and Information Science
Linköping University

Department of Statistics
Stockholm University



Outline

- Additive models
- Regression trees
- Random forest
- Tree ensembles and boosting

Additive models

General regression models

$$y = f(\mathbf{x}; \mathbf{w}) + \varepsilon,$$

where $f(\mathbf{x}; \mathbf{w}) : \mathbb{R}^p \rightarrow \mathbb{R}$ is a function of all p features and weights/parameters \mathbf{w} .

Example 1: Polynomial regression

$$f(\mathbf{x}; \mathbf{w}) = w_0 + xw_1 + \dots + x^p w_p$$

Example 2: Regression with interactions

$$f(\mathbf{x}; \mathbf{w}) = w_0 + \underbrace{x_1 w_1 + x_2 w_2}_{\text{main effects}} + \underbrace{x_1 x_2 w_3}_{\text{interaction}}$$

Additive models

$$y = \sum_{k=1}^p f_k(x_k; \mathbf{w}_k) + \varepsilon.$$

General relations between y and each x_k , but **no interactions**.

Fitting additive models

Algorithm 1: Backfitting algorithm

Input: $n \times 1$ response vector \mathbf{y} , feature vectors $\mathbf{x}_1, \dots, \mathbf{x}_p$, loss function $\ell(y, \hat{y})$

set $\hat{\mathbf{w}}_1 = \dots = \hat{\mathbf{w}}_p = 0$

set $\hat{\mathbf{w}}_0 = \text{mean}(\mathbf{y})$

repeat

 for $j = 1$ to p do

$\hat{\mathbf{w}}_j = \text{argmin}_{\mathbf{w}_k} \sum_{i=1}^n \ell(y_i, \hat{\mathbf{w}}_0 + \sum_{k \neq j} f_k(\mathbf{x}_i; \hat{\mathbf{w}}_k))$

 end

until *tolerance met*;

Output: $\hat{\mathbf{w}}_0, \hat{\mathbf{w}}_1 = \dots = \hat{\mathbf{w}}_p = 0$.

- **Greedy forward version:** no repeats, but enter the best fitting $f_k(\mathbf{x}_k; \mathbf{w}_k)$ at each step.

Regression trees

- Partition the feature space into M rectangles, R_1, \dots, R_M .
- Fit a constant in each rectangle.

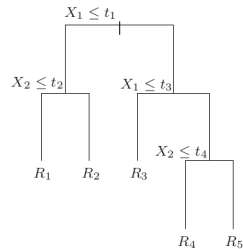
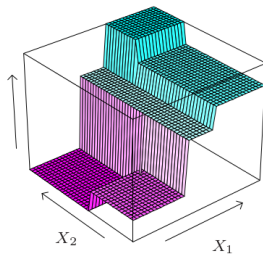
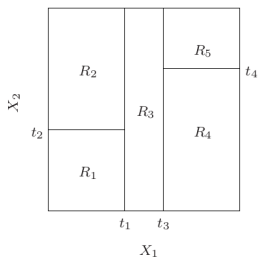
$$\hat{f}(\mathbf{x}) = \sum_{m=1}^M w_m I\{\mathbf{x} \in R_m\}$$

- ▶ Rectangle indicator functions:

$$I\{\mathbf{x} \in R_m\} = \begin{cases} 1 & \text{if } \mathbf{x} \in R_m \\ 0 & \text{if } \mathbf{x} \notin R_m \end{cases}$$

- ▶ the **Level** in rectangle R_m is w_m .
- Regression trees uses **binary splits**, one feature at the time.
- Computationally efficient and nice interpretation.

Regression trees



Fitting regression trees

■ Parameters:

- ▶ **split variable** at each stage: j_1, j_2, \dots
- ▶ **splitting point** at each stage: s_1, s_2, \dots
- ▶ **constants/weights** w_1, w_2, \dots

■ **Weights.** For a given tree, LS estimates

$$\hat{w}_m = \text{mean}(y_i | \mathbf{x}_i \in R_m).$$

■ Greedy for **split variable** j and **splitting point** j

$$\min_{j,s} \left[\min_{w_L} \sum_{\mathbf{x}_i \in R_L(j,s)} (y_i - w_L)^2 + \min_{w_H} \sum_{\mathbf{x}_i \in R_H(j,s)} (y_i - w_H)^2 \right]$$

where $R_L(j, s) = \{\mathbf{x} | x_j \leq s\}$ and $R_H(j, s) = \{\mathbf{x} | x_j > s\}$.

■ For any choice of (j, s) the **weights** are estimated by

$$\hat{w}_L = \text{mean}(y_i | \mathbf{x}_i \in R_L(j, s)) \text{ and } \hat{w}_H = \text{mean}(y_i | \mathbf{x}_i \in R_H(j, s)).$$

Cost-complexity pruning

- **How big** tree?
- **Bias** (small tree) vs **Variance** (large tree)
- **Cost-complexity pruning**:
 - ▶ grow a large tree (few observation at each leave)
 - ▶ **prune** the tree by collapsing non-terminal nodes to minimize

$$\sum_{i=1}^n \ell(y_i, \hat{f}_T(\mathbf{x}_i)) + \eta |T| + \lambda \|\mathbf{w}\|_2^2$$

where $\ell(y_i, \hat{f}_T(\mathbf{x}_i))$ is a **loss function** (SSE) and $|T|$ is the **number of leaves** in the subtree T .

- ▶ Hyperparameters η and λ can be set with cross-validation.

- **Weakest link pruning**, see the ESL book, page 308.

Tree ensemble

- Regression trees suffer from large variance.
- **Tree ensembles** combine many trees additively

$$\hat{f}(\mathbf{x}) = \sum_{k=1}^K \hat{f}_k(\mathbf{x}), \hat{f}_k \in \mathcal{F}$$

where \mathcal{F} is the collection of all trees

$$\mathcal{F} = \left\{ f(\mathbf{x}) = \mathbf{w}_{q(\mathbf{x})} \right\}$$

- ▶ $q(\mathbf{x}) : \mathbb{R}^p \rightarrow T$ is the **tree structure** (split variables and split points)
- ▶ $\mathbf{w}_{q(\mathbf{x})}$ are the **leave weights**.

Random forest

- **Random forest** is a tree ensemble with trees grown on **random data subsets**
 - ▶ **random** choice of allowed **splitting variables**
 - ▶ **random subset of observations** sampled with replacement (**bagging**).
- Measures of **variable importance** by comparing with predictions from permuted feature observations.

Boosted tree ensembles

- **Boosting**: iterative fitting. **Poorly predicted observations** at previous iteration are **upweighted** (**boosted**).
- **Boosted tree ensembles**: add tree that fits boosted errors.
- Boosting \approx Greedy forward selection (with special loss).

Algorithm 2: Greedy forward algorithm for tree ensembles.

Input: Data $\{y_i, \mathbf{x}_i\}_{i=1}^n$, tree generator $f(\mathbf{x}; \gamma)$ parametrized by split variables, split points and leave values.

set $\phi_0(\mathbf{x}) = 0$

for $m = 1$ **to** M **do**

 Compute $\gamma_m = \arg \min_{\gamma} \sum_{i=1}^n \ell(y_i, \phi_{m-1}(\mathbf{x}_i; \gamma) + f(\mathbf{x}_i; \gamma))$
 Set $\phi_m(\mathbf{x}; \gamma) = \phi_{m-1}(\mathbf{x}; \gamma) + f(\mathbf{x}; \gamma_m)$

end

Output: Ensemble $\phi_M(\mathbf{x}; \gamma)$ and tree parameters $\gamma_1, \dots, \gamma_M$.

XGBoost - Extreme Gradient Boosting

- **Computationally efficient** boosted tree ensemble:
 - ▶ gradient boosting with smooth penalty $\eta |T| + \lambda \|\mathbf{w}\|_2^2$.
 - ▶ efficient data structure for large datasets.
- Good performance in competitions.
- **Gradient boosting**: approximate objective at iteration t

$$\sum_{i=1}^n \ell \left(y_i, \hat{y}_i^{(t-1)} + f_t(\mathbf{x}_i) \right) \approx \sum_{i=1}^n \ell \left(y_i, \hat{y}_i^{(t-1)} \right) + g_i f_t(\mathbf{x}_i) + h_i f_t^2(\mathbf{x}_i)$$

- ▶ $\hat{y}_i^{(t-1)}$ the fit from ensemble at previous iteration
 - ▶ $g_i = \frac{\partial \ell(y_i, \hat{y})}{\partial \hat{y}} \Big|_{\hat{y} = \hat{y}_i^{(t-1)}}$ is the gradient
 - ▶ $h_i = \frac{\partial^2 \ell(y_i, \hat{y})}{\partial^2 \hat{y}} \Big|_{\hat{y} = \hat{y}_i^{(t-1)}}$ is the Hessian.
- For a given tree structure $q_t(\mathbf{x})$, we can solve for $\hat{\mathbf{w}}_{q(\mathbf{x})}$.
- Tree structure $q_t(\mathbf{x})$ can be optimized in a greedy fashion.

RuleFit and HorseRule

- To be written.