# 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
- RuleFit, HorseRule and BART

# Additive models

- General regression models

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

where $f(\boldsymbol{x}; \boldsymbol{w}) : \mathbb{R}^p \to \mathbb{R}$ is a function of all $p$ features.

- Example 1: Polynomial regression

$$f(\boldsymbol{x}; \boldsymbol{w}) = w_0 + x w_1 + \ldots + x^p w_p$$

- Example 2: **Regression with interactions**

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

- **Additive models**

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

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 $\mathbf{x}_1, \ldots, \mathbf{x}_p$, loss function $\ell(y, \hat{y})$

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

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

set $\mathbf{y} = \mathbf{y} - \hat{w}_0$

**repeat**

  **for** $j = 1$ **to** $p$ **do**

    $\hat{\mathbf{w}}_j = \arg\min_{\mathbf{w}_k} \sum_{i=1}^{n} \ell\left(y_i - \sum_{k \neq j} f_k(x_i; \hat{\mathbf{w}}_k), f_j(x_i; \mathbf{w}_k)\right)$

  **end**

**until** *tolerance met*;

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

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

# Regression trees

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

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

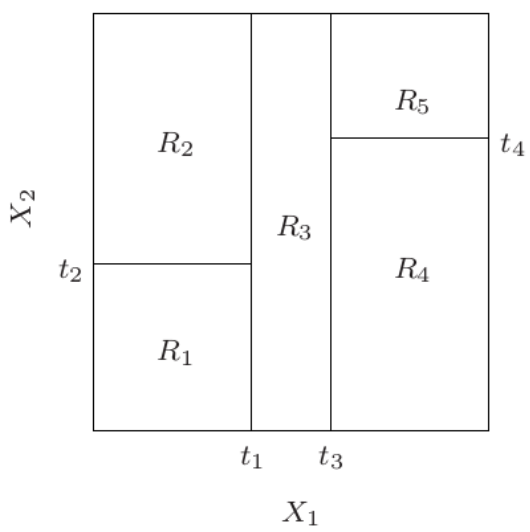
  - **Rectangle indicator functions**:

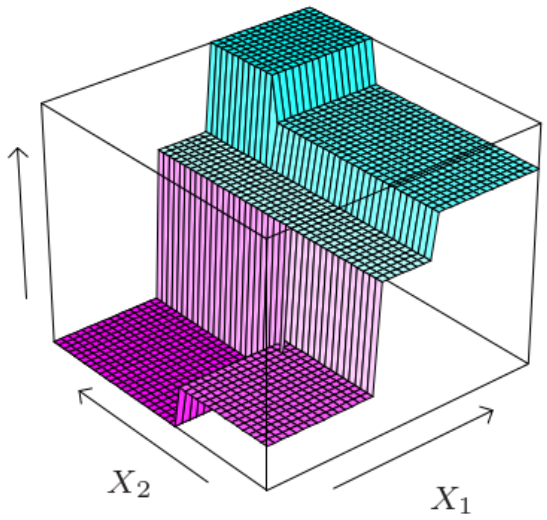
$$I\{\boldsymbol{x} \in R_m\} = \begin{cases} 1 & \text{if } \boldsymbol{x} \in R_m \\ 0 & \text{if } \boldsymbol{x} \notin R_m \end{cases}$$

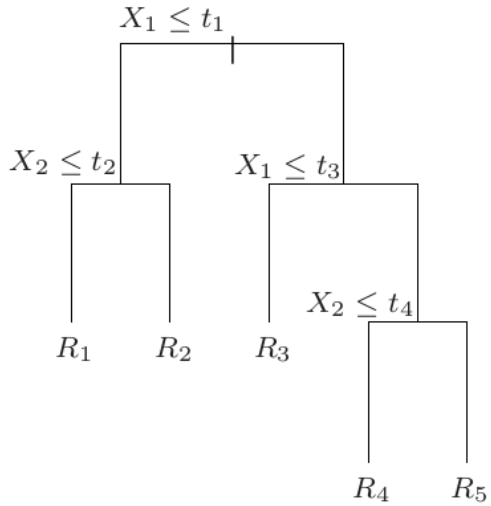
  - **Level** in rectangle $R_m$ is $w_m$.

- **Regression trees** use **binary splits**, one feature at the time.
- Computationally efficient and nice interpretation.

# Regression trees



# Fitting regression trees

- **Parameters**:
  - **split variable** at each stage: $j_1, j_2, \ldots$
  - **splitting point** at each stage: $s_1, s_2, \ldots$
  - **constants/weights** $w_1, w_2, \ldots$
- **Weights**. For a given tree, LS estimates

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

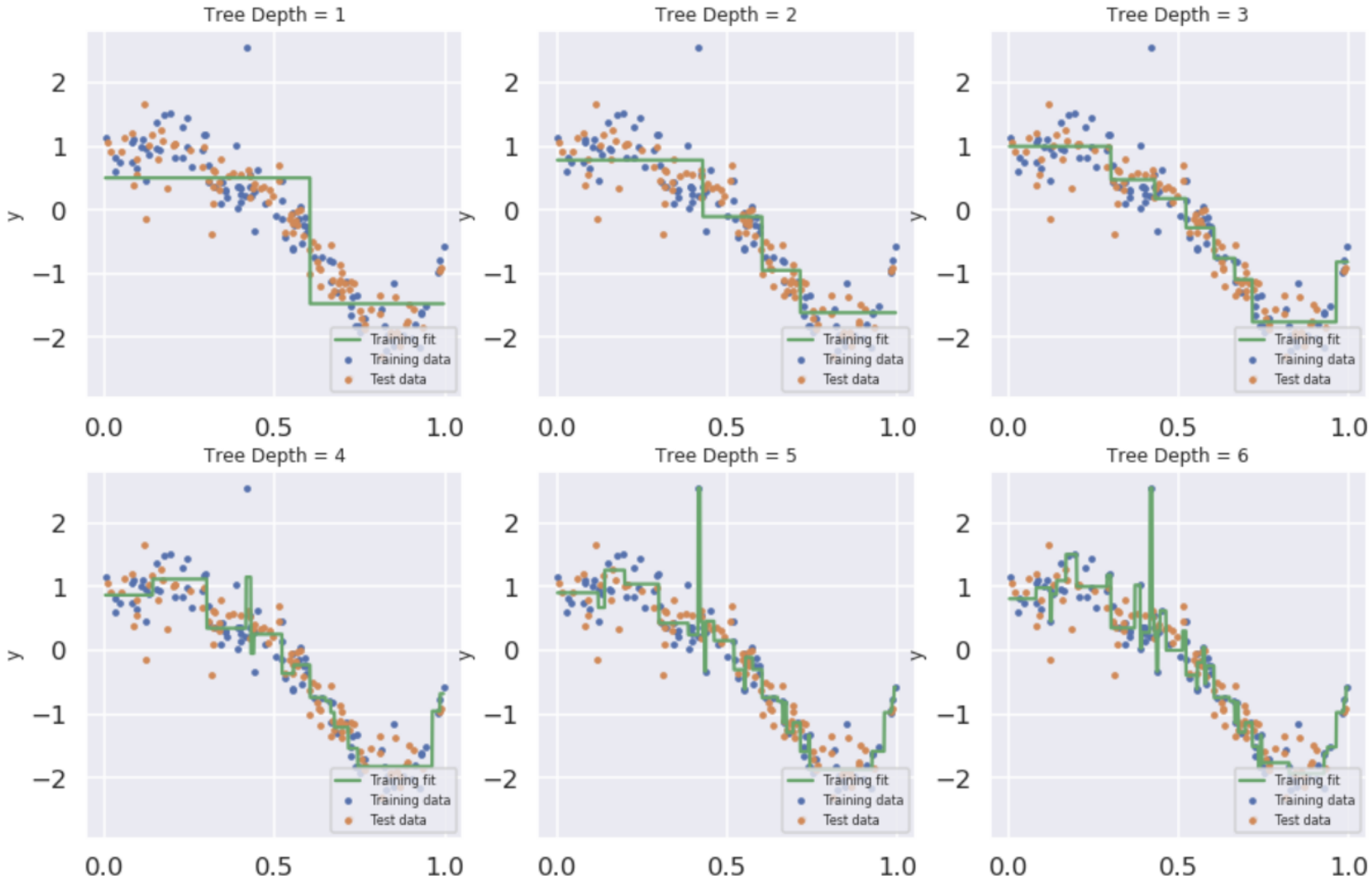
- Greedy for **split variable** $j$ and **splitting point** $j$

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

  where $R_L(j,s) = \{\boldsymbol{x} | x_j \leq s\}$ and $R_H(j,s) = \{\boldsymbol{x} | x_j > s\}$.
- For any choice of $(j,s)$ the **weights** are estimated by

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

# Fitting a regression tree



# 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

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

    - $\ell\left(y_i, \hat{f}_T(\boldsymbol{x}_i)\right)$ is a **loss function** (SSE)
    - $|T|$ is the **number of leaves** in the subtree $T$.
  - Hyperparameters $\eta$ and $\lambda$ can be set with cross-validation.
- **Weakest link pruning**, see the ESL book, page 308.
- **Variable importance** for $x_j$: summing the improvement in fit at each node that is split by $x_j$.

# Tree ensemble

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

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

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

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

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

# Random forest

- **Random forest** is a tree ensemble with trees grown by **bagging**.
- **Bagging features**: random choice of allowed splitting variables at each node.
- **Bagging observations**: tree grown on subsets of observations sampled with replacement.
- Measures of **variable importance** by either:
  - Averaging the variable importance over all trees in forest
  - 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 $\gamma_m = \arg\min_\gamma \sum_{i=1}^n \ell\big(y_i, \phi_{m-1}(\mathbf{x}_i; \gamma) + f(\mathbf{x}_i; \gamma)\big)$

  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 $\gamma_1, \ldots, \gamma_M$.

---

# XGBoost - Extreme Gradient Boosting

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

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

  - $\hat{y}_i^{(t-1)}$ the fit from ensemble at previous iteration
  - $g_i = \frac{\partial \ell(y_i, \hat{y})}{\partial \hat{y}}\Big|_{\hat{y}=\hat{y}_i^{(t-1)}}$ and $h_i = \frac{\partial^2 \ell(y_i, \hat{y})}{\partial^2 \hat{y}}\Big|_{\hat{y}=\hat{y}_i^{(t-1)}}$.
- For a given tree structure solve for $\hat{\boldsymbol{w}}_{q(\boldsymbol{x})}$ to get the objective

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{|T|} \frac{\left(\sum_{i \in I_j} g_i\right)^2}{\sum_{i \in I_j} h_i + \lambda} + \eta |T|, \text{ where } I_j = \{i | q(\boldsymbol{x}_i) = j\}$$

- $\tilde{\mathcal{L}}^{(t)}(q)$ can be optimized w.r.t. tree structure $q_t(\boldsymbol{x})$ in a greedy fashion, starting with a single leave and adding splits.

# More tree ensembles with funny names

- **RuleFit**:
  - Run XGBoost.
  - Construct **binary covariates from tree rules**.
  - Run **Lasso regression** on these tree-based covariates.
  - May also add other features in the Lasso (e.g. polynomials).
- **HorseRule**. Like RuleFit, but uses tree-structured horseshoe regularization.
  - **Horseshoe** regularization. Shrinks noise rules, keeps signal.
  - More shrinkage on deep trees branches with little data at leaf.
  - Use also rules from Random forest to get more diversity.
  - Uses **Markov Chain Monte Carlo**, so much slower to train.
- **Bayesian Additive Regression Trees** (**BART**). Bayesian version of tree ensembles.