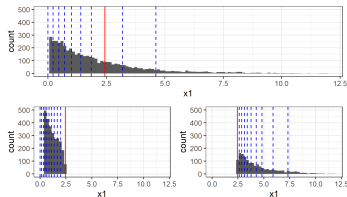


# Introduction to Machine Learning

## Boosting

## Gradient Boosting: XGBoost

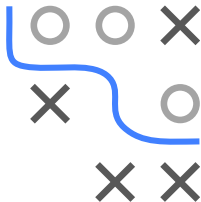


### Learning goals

- Overview over XGB
- Regularization in XGB
- Approximate split finding

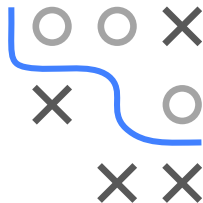
## XBG - EXTREME GRADIENT BOOSTING

- Open-source and scalable tree boosting system
- Efficient implementation in *C++* with interfaces to many other programming languages
- Parallel approximate split finding
- Additional regularization techniques
- Feature and data subsampling
- Cluster and GPU support
- Highly optimized and often achieves top performance in benchmarks – if properly tuned



### 3 EXTRA REGULARIZATION TERMS

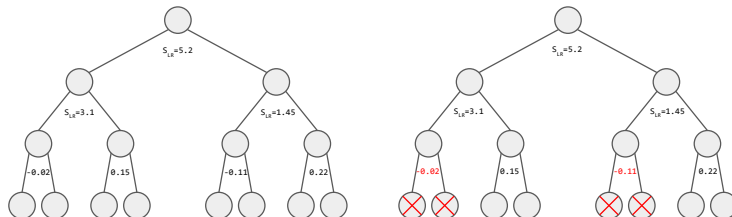
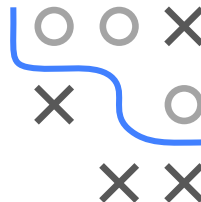
$$\mathcal{R}_{\text{reg}}^{[m]} = \sum_{i=1}^n L\left(y^{(i)}, f^{[m-1]}(\mathbf{x}^{(i)}) + b^{[m]}(\mathbf{x}^{(i)})\right) \\ + \lambda_1 J_1(b^{[m]}) + \lambda_2 J_2(b^{[m]}) + \lambda_3 J_3(b^{[m]}),$$



- $J_1(b^{[m]}) = T^{[m]}$ : Nr of leaves to penalize tree depth
- $J_2(b^{[m]}) = \|\mathbf{c}^{[m]}\|_2^2$ : L2 penalty over leaf values
- $J_3(b^{[m]}) = \|\mathbf{c}^{[m]}\|_1$ : L1 penalty over leaf values

# TREE GROWING

- Grown to max depth
- Fully expanded and leaves split even if no improvement
- At the end, each split that did not improve risk is pruned



# SUBSAMPLING

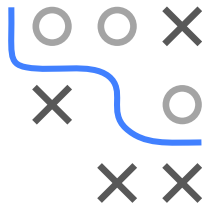
**Data Subsampling:** XGB uses stochastic GB.

**Feature Subsampling:** Similar to `mtry` in a random forest only a random subset of features is used for split finding.

The fraction of features for a split can be randomly sampled for each

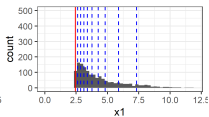
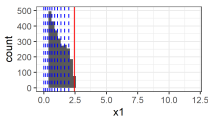
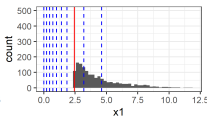
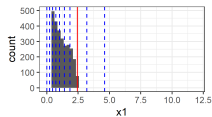
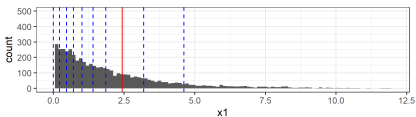
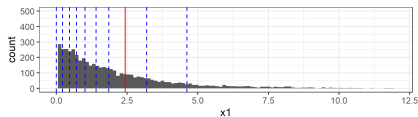
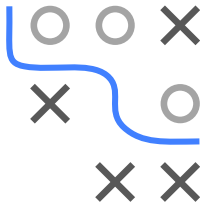
- 1 tree
- 2 level of a tree
- 3 split

Feature subsampling speeds up training even further and can create a more diverse ensemble that often performs better.



# APPROXIMATE SPLIT-FINDING ALGORITHMS

- Speeds up tree building for large data
- Considers not all, but only  $I$  splits per feature
- Usually percentiles of the empirical distribution of each feature
- Computed once (global) or recomputed after each split (local)
- Called **Histogram-based Gradient Boosting**



Global

Local

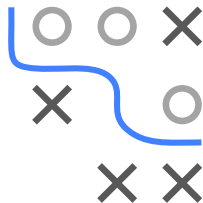
Blue lines are percentiles and red = selected split

# DROPOUT ADDITIVE-REGRESSION TREES

DART introduces idea of *dropout* regularization used in DL to boosting

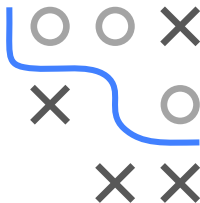
- In iteration  $m$  we construct  $\hat{b}^{[m]}$
- To compute PRs we need  $\hat{f}^{[m-1]}$
- We compute this differently, by using random subset  $D \subset \hat{b}^{[1]}, \dots, \hat{b}^{[m-1]}$  of size  $(m-1) \cdot p_{\text{drop}}$  is ignored
- To avoid *overshot predictions* in ensemble, we scale the BLs at the end of the iteration, by  $\frac{1}{|D|+1} \hat{b}^{[m]}$  and  $\frac{|D|}{|D|+1} \hat{b} \quad \forall \hat{b} \in D$ .
- $p_{\text{drop}} = 0$ : Ordinary GB
- $p_{\text{drop}} = 1$ : All BLs are trained independently, and equally weighted. Model is very similar to random forest.

$\Rightarrow p_{\text{drop}}$  is smooth transition from GB to RF



# PARALLELISM AND GPU COMPUTATION

- GB is inherently sequential, not easy to parallelize
- **But:** Building of BLs can be parallelized
- Data sort and split eval in different branches of tree BLs can be computed in parallel by using efficient block data structures
- Can also gain huge speed-up by moving from CPU to GPU





# OVERVIEW OF IMPORTANT HYPERPARAMETERS

HP (as named in software)	Type	Typical Range	Trafo	Default	Description
<code>eta</code>	R	$[-4, 0]$	$10^x$	0.3	learning rate (also called $\nu$ ) shrinks contribution of each boosting update
<code>nrounds</code>	I	$\{1, \dots, 5000\}$	–	–	number of boosting iterations. Can also be optimized with early stopping.
<code>gamma</code>	R	$[-7, 6]$	$2^x$	0	minimum loss reduction required to make a further partition on a leaf node of the tree
<code>max_depth</code>	I	$\{1, \dots, 20\}$	–	6	maximum depth of a tree
<code>colsample_bytree</code>	R	$[0.1, 1]$	–	1	subsample ratio of columns for each tree
<code>colsample_bylevel</code>	R	$[0.1, 1]$	–	1	subsample ratio of columns for each depth level
<code>lambda</code>	R	$[-10, 10]$	$2^x$	1	$L2$ regularization term on weights
<code>alpha</code>	R	$[-10, 10]$	$2^x$	0	$L1$ regularization term on weights
<code>subsample</code>	R	$[0.1, 1]$	–	1	subsample ratio of the training instances

