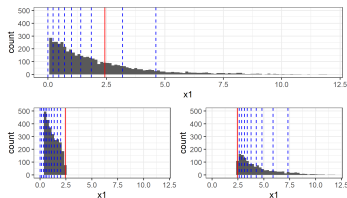
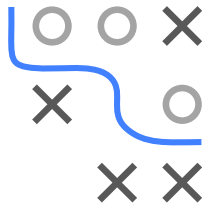


# Introduction to Machine Learning

## Boosting

## Gradient Boosting: Modern Techniques



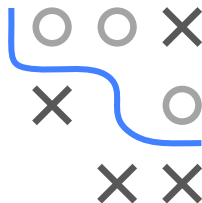
### Learning goals

- Know extensions of XGBoost and how they differ
- Understand areas upon which extensions of XGBoost improve

# BEYOND XGBOOST

Next to **XGBoost** two other important modern boosting libraries exist:

- **LightGBM** by Ke et al. (2017)
- **CatBoost** by Prokhorenkova et al. (2017)



Both libraries extend the ideas of **XGBoost** in several areas:

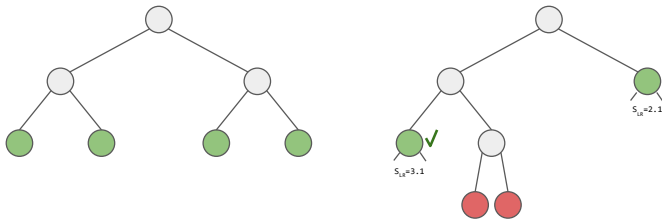
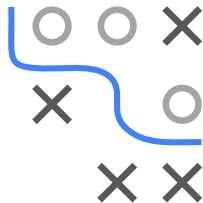
- 1 Tree growing efficiency
- 2 Data sampling
- 3 Feature compression
- 4 Categorical feature handling

Many of the the proposed ideas have later been implemented in **XGBoost** as well.

# TREE GROWING EFFICIENCY

Recall: **XGBoost** grows a balanced tree of `max_depth` and prunes leaves that do not improve the risk.

**Leaf-wise (Best-first) Tree Growth** allows the growing of unbalanced trees by comparing improvements between all possible leaves.

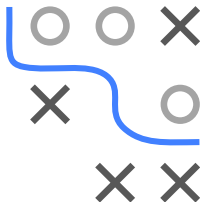


Balanced tree (left) of `max_depth=3`: All 4 leaves (colored green) will be split (in order from left to right). Leaf-wise growth (right) of `max_depth=3`: From the valid leaves (green), the leaf with largest improvement will be split next (marked). Invalid leaves (red) are not considered.

# DATA SAMPLING: GRADIENT-BASED ONE-SIDE SAMPLING (GOSS)

Recall: **XGBoost** use random data subsampling, i.e. stochastic gradient boosting.

Stochastic gradient boosting can be improved by *smarter* sampling strategies based on the values of the gradients.

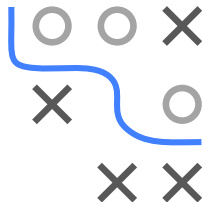


## GOSS:

- To evaluate a split GOSS only uses the  $a \cdot n$  observations with largest (absolute) gradients and samples  $b \cdot n$  observations from the remaining.
- The randomly sampled observations with smaller gradients are weighted by  $\frac{1-a}{b}$ .
- Default values are  $a = 0.2$  and  $b = 0.1$ .
- GOSS is only used after  $\frac{1}{\nu}$  iterations of regular boosting steps.

# DATA SAMPLING: MINIMAL VARIANCE SAMPLING (MVS)

- MVS computes weights and selection probabilities of observations for a tree.
- The weighting is computed from the regularized absolute value  $\hat{g}^{[m]}(\mathbf{x}^{(i)}) = \sqrt{g^{[m]}(\mathbf{x}^{(i)})^2 + \lambda h^{[m]}(\mathbf{x}^{(i)})^2}$ .
- Observations with a value of  $\hat{g}^{[m]}(\mathbf{x}^{(i)}) > \mu$  are always used and other observations are selected with probability  $\frac{\hat{g}^{[m]}(\mathbf{x}^{(i)})}{\mu}$ .
- $\mu$  has a closed-form nearly optimal solution for minimizing the risk of a tree base learner (**Ibragimov et al. 2019**).
- For the tree fit each observation is weighted inversely proportional to its selection probability.



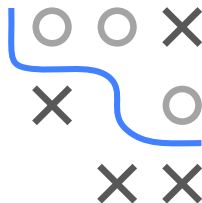
**Note:**  $g^{[m]}(\mathbf{x}) = \frac{\partial L(y, f^{[m-1]}(\mathbf{x}))}{\partial f^{[m-1]}(\mathbf{x})}$  and  $h^{[m]}(\mathbf{x}) = \frac{\partial^2 L(y, f^{[m-1]}(\mathbf{x}))}{\partial f^{[m-1]}(\mathbf{x})^2}$ .

# FEATURE COMPRESSION

For high dimensional (sparse) data it can be helpful to bundle similar features together to speed up split computations.

**Exclusive feature bundling** looks for mutually exclusive features, i.e. features that never take nonzero values simultaneously.

- A single histogram for approximate split finding in boosting can be built from multiple mutually exclusive features nearly without loss of information.
- Mutually exclusive features only occur in sparse data.
- This approach speeds up the histogram building from  $\mathcal{O}(np)$  to  $\mathcal{O}(nb)$  where  $b$  is the number of feature bundles.
- While finding the optimal bundling is  $np$ -hard, greedy approximations give good results empirically.



# CATEGORICAL FEATURES

Even though **XGBoost** uses trees it does not support categorical features.

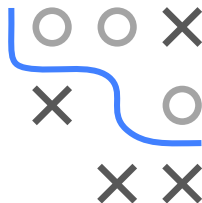
Both **LightGBM** and **CatBoost** provide *target* encoding strategies for categorical features:

$$\tilde{\mathbf{x}}_j = \frac{\sum_{i:\mathbf{x}_j=l} y^{(i)}}{N_l}, \quad l = 1, \dots, k$$

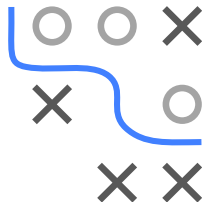
where  $N_l$  is the number of observations of the  $l$ 'th level of categorical feature  $\mathbf{x}_j$ .

Additional noise can be added to the encoding to avoid overfitting for level with few observations.

Features with relatively few levels  $k \leq \tau_{\text{max\_cat\_to\_onehot}}$  (default 4) are one-hot encoded.



# FEATURE COMPARISON OF BOOSTING FRAMEWORKS



	Parallel	GPU Support	Approx. splits	Categ. feats
XGBoost	x	x	x	
LightGBM	x	x	x	x
CatBoost	x	x	x	x
GBM				x
H2O	x	x	x	x
sklearn	x		x	x

	Tree growing		Subsampling		Feats
	Depth-wise	Leaf-wise	Observations		
			Regular	Gradient-based	
XGBoost	x	x	x	x	x
LightGBM	x	x	x	x	x
CatBoost	x	x	x	x	x
GBM		x	x		
H2O	x		x		x
sklearn	x		x		x