



Ensemble Learning: CatBoost

Pilsung Kang

School of Industrial Management Engineering

Korea University

CatBoost: Backgrounds

Prokhorenkova et al. (2018)

- Background in brief

- ✓ Assume we observe a dataset of examples $\mathcal{D} = \{(\mathbf{x}_k, y_k)\}_{k=1, \dots, n}$
 - where $\mathbf{x}_k = (x_k^1, \dots, x_k^m)$ is a random vector of m features and $y_k \in \mathbb{R}$ is a target
- ✓ Gradient boosting builds iteratively a sequence of approximations $F^t : \mathbb{R}^m \rightarrow \mathbb{R}$ in a greedy fashion
 - $F^t : F^{t-1} + \alpha h^t$ (alpha is a step size and h is a base predictor)
 - h^t is chosen from a family of functions H in order to minimize the expected loss

$$h^t = \arg \min_{h \in H} \mathcal{L}(F^{t-1} + h) = \arg \min_{h \in H} \mathbb{E} L(y, F^{t-1}(\mathbf{x}) + h(\mathbf{x}))$$

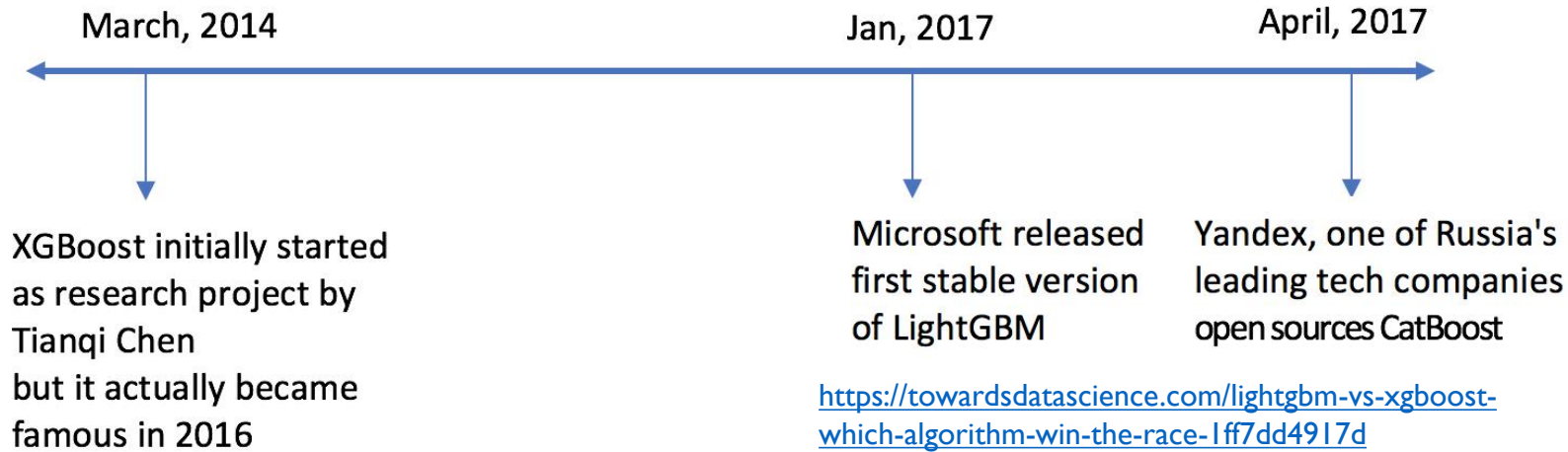
- The gradient step h^t is chosen in such a way that $h^t(\mathbf{x})$ approximates $-g^t(\mathbf{x}, y)$, where

$$g^t(\mathbf{x}, y) := \left. \frac{\partial L(y, s)}{\partial s} \right|_{s=F^{t-1}(\mathbf{x})}$$






- Usually the least squares approximation is used

$$h^t = \arg \min_{h \in H} \mathbb{E} (-g^t(\mathbf{x}, y) - h(\mathbf{x}))^2$$

CatBoost: Backgrounds



Features

-  **Great quality without parameter tuning**
Reduce time spent on parameter tuning, because CatBoost provides great results with default parameters
-  **Categorical features support**
Improve your training results with CatBoost that allows you to use non-numeric factors, instead of having to pre-process your data or spend time and effort turning it to numbers.
-  **Fast and scalable GPU version**
Train your model on a fast implementation of gradient-boosting algorithm for GPU. Use a multi-card configuration for large datasets.
-  **Improved accuracy**
Reduce overfitting when constructing your models with a novel gradient-boosting scheme.
-  **Fast prediction**
Apply your trained model quickly and efficiently even to latency-critical tasks using CatBoost's model applier

<https://catboost.ai/>

CatBoost: Backgrounds

- Gradient Boosting

- ✓ Statistical Issue I: Prediction Shift

$$\arg \min_{h \in H} \mathbb{E} \left(-g^t(\mathbf{x}, y) - h(\mathbf{x}) \right)^2 \approx \arg \min_{h \in H} \frac{1}{n} \sum_{k=1}^n \left(-g^t(\mathbf{x}_k, y_k) - h(\mathbf{x}_k) \right)^2$$

Training Dataset : $\mathcal{D} = (\mathbf{x}_k, y_k)_{k=1, \dots, n}$

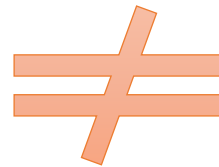
where $\mathbf{x}_k = (x_k^1, \dots, x_k^m)$, $y_k \in \mathbb{R}$

Conditional distribution

$$F^{t-1}(\mathbf{x}_k | \mathbf{x}_k)$$

for training example \mathbf{x}_k

is SHFTED from



Conditional distribution

$$F^{t-1}(\mathbf{x} | \mathbf{x})$$

for test example \mathbf{x}

CatBoost: Backgrounds

- Gradient Boosting

- ✓ Statistical Issue 2: Target Leakage

- Target Statistic (TS): Replace the categorical feature with it mean target value in the training dataset
 - (Problem) Target value of an instance is used to compute its feature value (target information is leaked)

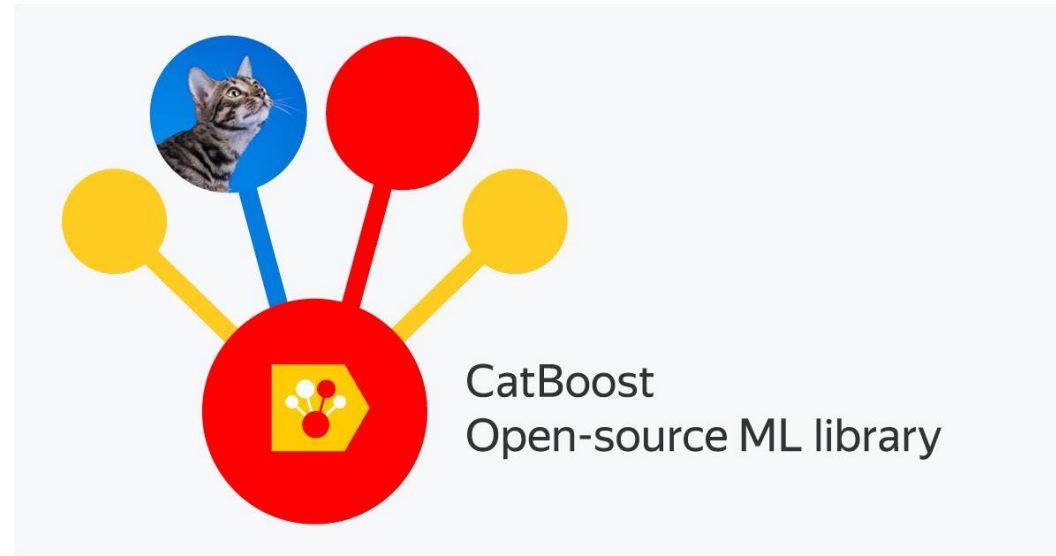
	...	x^i	...	y
I_1	...	A	...	1
I_2	...	B	...	1
I_3	...	C	...	1
I_4	...	A	...	0
I_5	...	B	...	1
I_6	...	C	...	1
I_7	...	B	...	0
I_8	...	C	...	1
I_9	...	C	...	1
I_{10}	...	C	...	0



	...	$x^i(\text{TS})$...	y
I_1	...	0.50	...	1
I_2	...	0.67	...	1
I_3	...	0.80	...	1
I_4	...	0.50	...	0
I_5	...	0.67	...	1
I_6	...	0.80	...	1
I_7	...	0.67	...	0
I_8	...	0.80	...	1
I_9	...	0.80	...	1
I_{10}	...	0.80	...	0

CatBoost: Backgrounds

- Solution




Target Leakage → Ordered Target Statistics

Prediction Shift → Ordered Boosting

CatBoost: Ordered TS

- Categorical features in boosted tree
 - ✓ Basic approach: one-hot encoding

	...	x^i	...
l_1	...	A	...
l_2	...	B	...
l_3	...	C	...
l_4	...	A	...
l_5	...	B	...
l_6	...	C	...
l_7	...	B	...
l_8	...	C	...
l_9	...	C	...
l_{10}	...	C	...



	...	$x^i(A)$	$x^i(B)$	$x^i(C)$...
l_1	...	1	0	0	...
l_2	...	0	1	0	...
l_3	...	0	0	1	...
l_4	...	1	0	0	...
l_5	...	0	1	0	...
l_6	...	0	0	1	...
l_7	...	0	1	0	...
l_8	...	0	0	1	...
l_9	...	0	0	1	...
l_{10}	...	0	0	1	...

CatBoost: Ordered TS

- Categorical features in boosted tree
 - ✓ Another popular method: to group categories by **target statistics (TS)**
 - an estimate expected target value in each category
 - Greedy TS without smoothing $\hat{x}_k^i \approx \mathbb{E}(y \mid x^i = x_k^i)$
 - i is a categorical feature while k is training example index

	...	x^i	...	y
l_1	...	A	...	1
l_2	...	B	...	1
l_3	...	C	...	1
l_4	...	A	...	0
l_5	...	B	...	1
l_6	...	C	...	1
l_7	...	B	...	0
l_8	...	C	...	1
l_9	...	C	...	1
l_{10}	...	C	...	0



	...	$x^i(\text{TS})$...	y
l_1	...	0.50	...	1
l_2	...	0.67	...	1
l_3	...	0.80	...	1
l_4	...	0.50	...	0
l_5	...	0.67	...	1
l_6	...	0.80	...	1
l_7	...	0.67	...	0
l_8	...	0.80	...	1
l_9	...	0.80	...	1
l_{10}	...	0.80	...	0

CatBoost: Ordered TS

- Categorical features in boosted tree
 - ✓ Another popular method: to group categories by **target statistics (TS)**
 - Greedy TS with smoothing

$$\hat{x}_k^i = \frac{\sum_{j=1}^n \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + ap}{\sum_{j=1}^n \mathbb{1}_{\{x_j^i = x_k^i\}} + a}$$

- $a > 0$ is a parameter
- A common setting for p is the average target value in the dataset
- Used to remove the negative effect of low-frequency noisy categories

CatBoost: Ordered TS

- Greedy TS with smoothing example
 - ✓ $a = 0.1$ (parameter) , $p = 0.7$ (computed from the training dataset)
 - ✓ For category A

	...	x^i	...	y
I_1	...	A	...	1
I_2	...	B	...	1
I_3	...	C	...	1
I_4	...	A	...	0
I_5	...	B	...	1
I_6	...	C	...	1
I_7	...	B	...	0
I_8	...	C	...	1
I_9	...	C	...	0
I_{10}	...	C	...	1

$$\hat{x}_k^A = \frac{\sum_{j=1}^n \mathbb{1}_{\{x_j^A = x_k^A\}} \cdot y_j + ap}{\sum_{j=1}^n \mathbb{1}_{\{x_j^A = x_k^A\}} + a}$$

$$= \frac{1 + 0.1 \times 0.7}{2 + 0.1} = 0.5095$$

CatBoost: Ordered TS

- Greedy TS with smoothing example

- ✓ $a = 0.1$ (parameter) , $p = 0.7$ (computed from the training dataset)

- ✓ For category B

	...	x^i	...	y
I_1	...	A	...	1
I_2	...	B	...	1
I_3	...	C	...	1
I_4	...	A	...	0
I_5	...	B	...	1
I_6	...	C	...	1
I_7	...	B	...	0
I_8	...	C	...	1
I_9	...	C	...	0
I_{10}	...	C	...	1

$$\hat{x}_k^B = \frac{\sum_{j=1}^n \mathbb{1}_{\{x_j^B = x_k^B\}} \cdot y_j + ap}{\sum_{j=1}^n \mathbb{1}_{\{x_j^B = x_k^B\}} + a}$$

$$= \frac{2 + 0.1 \times 0.7}{3 + 0.1} = 0.6677$$

CatBoost: Ordered TS

- Greedy TS with smoothing example

- ✓ $a = 0.1$ (parameter) , $p = 0.7$ (computed from the training dataset)

- ✓ For category C

	...	x^i	...	y
I_1	...	A	...	1
I_2	...	B	...	1
I_3	...	C	...	1
I_4	...	A	...	0
I_5	...	B	...	1
I_6	...	C	...	1
I_7	...	B	...	0
I_8	...	C	...	1
I_9	...	C	...	0
I_{10}	...	C	...	1

$$\hat{x}_k^C = \frac{\sum_{j=1}^n \mathbb{1}_{\{x_j^C = x_k^C\}} \cdot y_j + ap}{\sum_{j=1}^n \mathbb{1}_{\{x_j^C = x_k^C\}} + a}$$

$$= \frac{4 + 0.1 \times 0.7}{5 + 0.1} = 0.7980$$

CatBoost: Ordered TS

- Greedy TS with smoothing example

✓ $a = 0.1$ (parameter) , $p = 0.7$ (computed from the training dataset)

	...	x^i	...	y
I_1	...	A	...	1
I_2	...	B	...	1
I_3	...	C	...	1
I_4	...	A	...	0
I_5	...	B	...	1
I_6	...	C	...	1
I_7	...	B	...	0
I_8	...	C	...	1
I_9	...	C	...	0
I_{10}	...	C	...	1

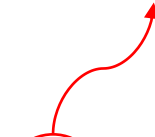
	...	$x^i(\text{TS})$...	y
I_1	...	0.5095	...	1
I_2	...	0.6677	...	1
I_3	...	0.7980	...	1
I_4	...	0.5095	...	0
I_5	...	0.6677	...	1
I_6	...	0.7980	...	1
I_7	...	0.6677	...	0
I_8	...	0.7980	...	1
I_9	...	0.7980	...	0
I_{10}	...	0.7980	...	1

CatBoost: Ordered TS

- Target leakage

- ✓ feature \hat{x}_k^i is computed using y_k , the target of \mathbf{x}_k
- ✓ leads to a **conditional shift**: the distribution of $\hat{x}^i|y$ differs for training and test examples

We should the use this information when we predict the target of current example

$$\hat{x}_k^i = \frac{\sum_{j=1}^n \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + ap}{\sum_{j=1}^n \mathbb{1}_{\{x_j^i = x_k^i\}} + a}$$


CatBoost: Ordered TS


- Target leakage

- ✓ feature \hat{x}_k^i is computed using y_k , the target of \mathbf{x}_k

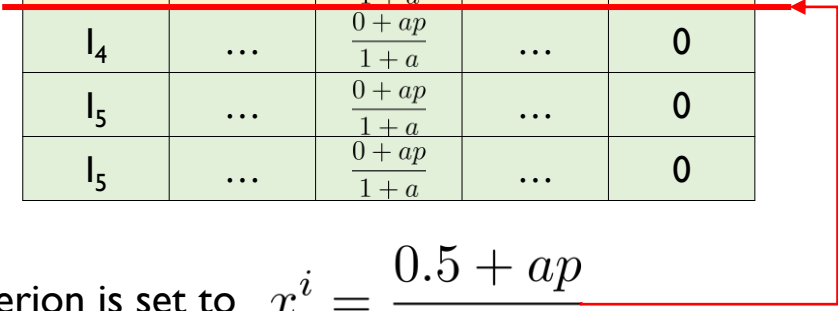
- ✓ leads to a **conditional shift**: the distribution of $\hat{x}^i|y$ differs for training and test examples

- Assume that i^{th} feature is categorical, all its values are unique, $P(y = 1|x^i) = 0.5, x^i \in \{A, B, C, D, E, F, \dots\}$

	...	x^i	...	y
l_1	...	A	...	1
l_2	...	B	...	1
l_3	...	C	...	1
l_4	...	D	...	0
l_5	...	E	...	0
l_6	...	F	...	0



	...	x^i	...	y
l_1	...	$\frac{1+ap}{1+a}$...	1
l_2	...	$\frac{1+ap}{1+a}$...	1
l_3	...	$\frac{1+ap}{1+a}$...	1
l_4	...	$\frac{0+ap}{1+a}$...	0
l_5	...	$\frac{0+ap}{1+a}$...	0
l_5	...	$\frac{0+ap}{1+a}$...	0



- Training data can be perfectly classified when the split criterion is set to $x^i = \frac{0.5 + ap}{1 + a}$

CatBoost: Ordered TS

- Target leakage

Training Set		...	x^i	...	y			...	x^i	...	y
	l_1	...	A	...	1		l_1	...	$\frac{1+ap}{1+a}$...	1
	l_2	...	B	...	1		l_2	...	$\frac{1+ap}{1+a}$...	1
	l_3	...	C	...	1		l_3	...	$\frac{1+ap}{1+a}$...	1
	l_4	...	D	...	0		l_4	...	$\frac{0+ap}{1+a}$...	0
	l_5	...	E	...	0		l_5	...	$\frac{0+ap}{1+a}$...	0
Test Set	l_6	...	F	...	0		l_6	...	$\frac{0+ap}{1+a}$...	0
	l_7	...	G	...	1		l_7	...	p	...	1
	l_8	...	H	...	1		l_8	...	p	...	1
	l_9	...	I	...	0		l_9	...	p	...	0
	l_{10}	...	J	...	0		l_{10}	...	p	...	0

Perfect classification by setting the cut off value

$$x^i = \frac{0.5 + ap}{1 + a}$$

$$\hat{x}_k^i = \frac{\sum_{j=1}^n \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + ap}{\sum_{j=1}^n \mathbb{1}_{\{x_j^i = x_k^i\}} + a}$$

All feature values of test instances become p

No discrimination ability

CatBoost: Ordered TS

- Desired property for TS

- ✓ Property 1:

$$\mathbb{E}(\hat{x}^i \mid y = v) = \mathbb{E}(\hat{x}_k^i \mid y_k = v)$$

where (\mathbf{x}_k, y_k) is the k – th training example

- ✓ Property 2:

- Effective usage of all training data for calculating TS features and for learning a model

CatBoost: Ordered TS

- Ways to avoid conditional shift

- ✓ General idea: compute the TS for \mathbf{x}_k on a subset of examples $\mathcal{D}_k \subset \mathcal{D} \setminus \{\mathbf{x}_k\}$ excluding \mathbf{x}_k

$$\hat{x}_k^i = \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + ap}{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a}$$

- ✓ Holdout TS

- To partition the training dataset into two part $\mathcal{D} = \hat{\mathcal{D}}_0 \cup \hat{\mathcal{D}}_1$
 - Use the former to compute the TS
 - Use the latter for training
 - Violates the Property 2

CatBoost: Ordered TS

- Ways to avoid conditional shift

- ✓ Leave-one-out TS


- Take $\mathcal{D}_k = \mathcal{D} \setminus \mathbf{x}_k$ for training examples

- Take $\mathcal{D}_k = \mathcal{D}$ for test ones

- Does not prevent target leakage

$$\hat{x}_k^i = \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + ap}{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a}$$

	...	\mathbf{x}^i	...	y
l_1	...	A	...	1
l_2	...	A	...	0
l_3	...	A	...	1
l_4	...	A	...	0
l_5	...	A	...	1



	...	\mathbf{x}^i	...	y
l_1	...	$\frac{2 + ap}{4 + a}$...	1
l_2	...	$\frac{3 + ap}{4 + a}$...	0
l_3	...	$\frac{2 + ap}{4 + a}$...	1
l_4	...	$\frac{3 + ap}{4 + a}$...	0
l_5	...	$\frac{2 + ap}{4 + a}$...	1

- We can perfectly classify this training dataset by setting the cut-off value $\frac{2.5 + ap}{4 + a}$

CatBoost: Ordered TS

- Ways to avoid conditional shift

- ✓ **Ordered TS**: introduce an **artificial time**

- a random permutation σ of the training examples

$$\mathcal{D}_k = \{\mathbf{x}_j : \sigma(j) < \sigma(k)\}$$

- $a = 0.1$ (parameter) , $p = 0.7$ (computed from the training dataset)

	...	\mathbf{x}^i	...	TS	y
I_1	...	A	...	0.700	1
I_2	...	B	...		1
I_3	...	C	...		1
I_4	...	A	...		0
I_5	...	B	...		1
I_6	...	C	...		1
I_7	...	B	...		0
I_8	...	C	...		1
I_9	...	C	...		0
I_{10}	...	C	...		1

$$\begin{aligned}\hat{x}_k^i &= \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + ap}{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a} \\ &= \frac{0 + 0.1 \times 0.7}{0 + 0.1} = 0.700\end{aligned}$$

CatBoost: Ordered TS

- Ways to avoid conditional shift

- ✓ Ordered TS: introduce an artificial time

- a random permutation σ of the training examples

$$\mathcal{D}_k = \{\mathbf{x}_j : \sigma(j) < \sigma(k)\}$$

- $a = 0.1$ (parameter) , $p = 0.7$ (computed from the training dataset)

	...	\mathbf{x}^i	...	TS	y
I_1	...	A	...	0.700	1
I_2	...	B	...	0.700	1
I_3	...	C	...		1
I_4	...	A	...		0
I_5	...	B	...		1
I_6	...	C	...		1
I_7	...	B	...		0
I_8	...	C	...		1
I_9	...	C	...		0
I_{10}	...	C	...		1

$$\begin{aligned}\hat{x}_k^i &= \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + ap}{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a} \\ &= \frac{0 + 0.1 \times 0.7}{0 + 0.1} = 0.700\end{aligned}$$

CatBoost: Ordered TS

- Ways to avoid conditional shift

✓ **Ordered TS**: introduce an **artificial time**

- a random permutation σ of the training examples

$$\mathcal{D}_k = \{\mathbf{x}_j : \sigma(j) < \sigma(k)\}$$

- $a = 0.1$ (parameter) , $p = 0.7$ (computed from the training dataset)

	...	\mathbf{x}^i	...	TS	y
I_1	...	A	...	0.700	1
I_2	...	B	...	0.700	1
I_3	...	C	...	0.700	1
I_4	...	A	...		0
I_5	...	B	...		1
I_6	...	C	...		1
I_7	...	B	...		0
I_8	...	C	...		1
I_9	...	C	...		0
I_{10}	...	C	...		1

$$\begin{aligned}\hat{x}_k^i &= \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + ap}{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a} \\ &= \frac{0 + 0.1 \times 0.7}{0 + 0.1} = 0.700\end{aligned}$$

CatBoost: Ordered TS

- Ways to avoid conditional shift

✓ **Ordered TS**: introduce an **artificial time**

- a random permutation σ of the training examples

$$\mathcal{D}_k = \{\mathbf{x}_j : \sigma(j) < \sigma(k)\}$$

- $a = 0.1$ (parameter) , $p = 0.7$ (computed from the training dataset)

	...	\mathbf{x}^i	...	TS	y
I_1	...	A	...	0.700	1
I_2	...	B	...	0.700	1
I_3	...	C	...	0.700	1
I_4	...	A	...	0.973	0
I_5	...	B	...		1
I_6	...	C	...		1
I_7	...	B	...		0
I_8	...	C	...		1
I_9	...	C	...		0
I_{10}	...	C	...		1

$$\begin{aligned}\hat{x}_k^i &= \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + ap}{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a} \\ &= \frac{1 + 0.1 \times 0.7}{1 + 0.1} = 0.973\end{aligned}$$

CatBoost: Ordered TS

- Ways to avoid conditional shift

- ✓ **Ordered TS**: introduce an **artificial time**

- a random permutation σ of the training examples

$$\mathcal{D}_k = \{\mathbf{x}_j : \sigma(j) < \sigma(k)\}$$

- $a = 0.1$ (parameter) , $p = 0.7$ (computed from the training dataset)

	...	\mathbf{x}^i	...	TS	y
I_1	...	A	...	0.700	1
I_2	...	B	...	0.700	1
I_3	...	C	...	0.700	1
I_4	...	A	...	0.973	0
I_5	...	B	...	0.973	1
I_6	...	C	...		1
I_7	...	B	...		0
I_8	...	C	...		1
I_9	...	C	...		0
I_{10}	...	C	...		1

$$\begin{aligned}\hat{x}_k^i &= \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + ap}{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a} \\ &= \frac{1 + 0.1 \times 0.7}{1 + 0.1} = 0.973\end{aligned}$$

CatBoost: Ordered TS

- Ways to avoid conditional shift

- ✓ **Ordered TS**: introduce an **artificial time**

- a random permutation σ of the training examples

$$\mathcal{D}_k = \{\mathbf{x}_j : \sigma(j) < \sigma(k)\}$$

- $a = 0.1$ (parameter) , $p = 0.7$ (computed from the training dataset)

	...	\mathbf{x}^i	...	TS	y
I_1	...	A	...	0.700	1
I_2	...	B	...	0.700	1
I_3	...	C	...	0.700	1
I_4	...	A	...	0.973	0
I_5	...	B	...	0.973	1
I_6	...	C	...	0.973	1
I_7	...	B	...		0
I_8	...	C	...		1
I_9	...	C	...		0
I_{10}	...	C	...		1

$$\begin{aligned}\hat{x}_k^i &= \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + ap}{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a} \\ &= \frac{1 + 0.1 \times 0.7}{1 + 0.1} = 0.973\end{aligned}$$

CatBoost: Ordered TS

- Ways to avoid conditional shift

- ✓ Ordered TS: introduce an artificial time

- a random permutation σ of the training examples

$$\mathcal{D}_k = \{\mathbf{x}_j : \sigma(j) < \sigma(k)\}$$

- $a = 0.1$ (parameter) , $p = 0.7$ (computed from the training dataset)

	...	\mathbf{x}^i	...	TS	y
I_1	...	A	...	0.700	1
I_2	...	B	...	0.700	1
I_3	...	C	...	0.700	1
I_4	...	A	...	0.973	0
I_5	...	B	...	0.973	1
I_6	...	C	...	0.973	1
I_7	...	B	...	0.986	0
I_8	...	C	...		1
I_9	...	C	...		0
I_{10}	...	C	...		1

$$\begin{aligned}\hat{x}_k^i &= \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + ap}{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a} \\ &= \frac{2 + 0.1 \times 0.7}{2 + 0.1} = 0.986\end{aligned}$$

CatBoost: Ordered TS

- Ways to avoid conditional shift

- ✓ **Ordered TS**: introduce an **artificial time**

- a random permutation σ of the training examples

$$\mathcal{D}_k = \{\mathbf{x}_j : \sigma(j) < \sigma(k)\}$$

- $a = 0.1$ (parameter) , $p = 0.7$ (computed from the training dataset)

	...	\mathbf{x}^i	...	TS	y
I_1	...	A	...	0.700	1
I_2	...	B	...	0.700	1
I_3	...	C	...	0.700	1
I_4	...	A	...	0.973	0
I_5	...	B	...	0.973	1
I_6	...	C	...	0.973	1
I_7	...	B	...	0.986	0
I_8	...	C	...	0.986	1
I_9	...	C	...		0
I_{10}	...	C	...		1

$$\begin{aligned}\hat{x}_k^i &= \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + ap}{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a} \\ &= \frac{2 + 0.1 \times 0.7}{2 + 0.1} = 0.986\end{aligned}$$

CatBoost: Ordered TS

- Ways to avoid conditional shift

- ✓ Ordered TS: introduce an artificial time

- a random permutation σ of the training examples

$$\mathcal{D}_k = \{\mathbf{x}_j : \sigma(j) < \sigma(k)\}$$

- $a = 0.1$ (parameter) , $p = 0.7$ (computed from the training dataset)

	...	\mathbf{x}^i	...	TS	y
I_1	...	A	...	0.700	1
I_2	...	B	...	0.700	1
I_3	...	C	...	0.700	1
I_4	...	A	...	0.973	0
I_5	...	B	...	0.973	1
I_6	...	C	...	0.973	1
I_7	...	B	...	0.986	0
I_8	...	C	...	0.986	1
I_9	...	C	...	0.990	0
I_{10}	...	C	...		1

$$\begin{aligned}\hat{x}_k^i &= \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + ap}{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a} \\ &= \frac{3 + 0.1 \times 0.7}{3 + 0.1} = 0.990\end{aligned}$$

CatBoost: Ordered TS

- Ways to avoid conditional shift

- ✓ Ordered TS: introduce an artificial time

- a random permutation σ of the training examples

$$\mathcal{D}_k = \{\mathbf{x}_j : \sigma(j) < \sigma(k)\}$$

- $a = 0.1$ (parameter) , $p = 0.7$ (computed from the training dataset)

	...	\mathbf{x}^i	...	TS	y
I_1	...	A	...	0.700	1
I_2	...	B	...	0.700	1
I_3	...	C	...	0.700	1
I_4	...	A	...	0.973	0
I_5	...	B	...	0.973	1
I_6	...	C	...	0.973	1
I_7	...	B	...	0.986	0
I_8	...	C	...	0.986	1
I_9	...	C	...	0.990	0
I_{10}	...	C	...	0.749	1

$$\begin{aligned}\hat{x}_k^i &= \frac{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} \cdot y_j + ap}{\sum_{\mathbf{x}_j \in \mathcal{D}_k} \mathbb{1}_{\{x_j^i = x_k^i\}} + a} \\ &= \frac{3 + 0.1 \times 0.7}{4 + 0.1} = 0.749\end{aligned}$$

CatBoost: Ordered Boosting

- Prediction Shift

$$h^t = \arg \min_{h \in H} \mathbb{E} \left(-g^t(\mathbf{x}, y) - h(\mathbf{x}) \right)^2$$

✓ The expectation is unknown and is usually approximated using the same dataset \mathcal{D}

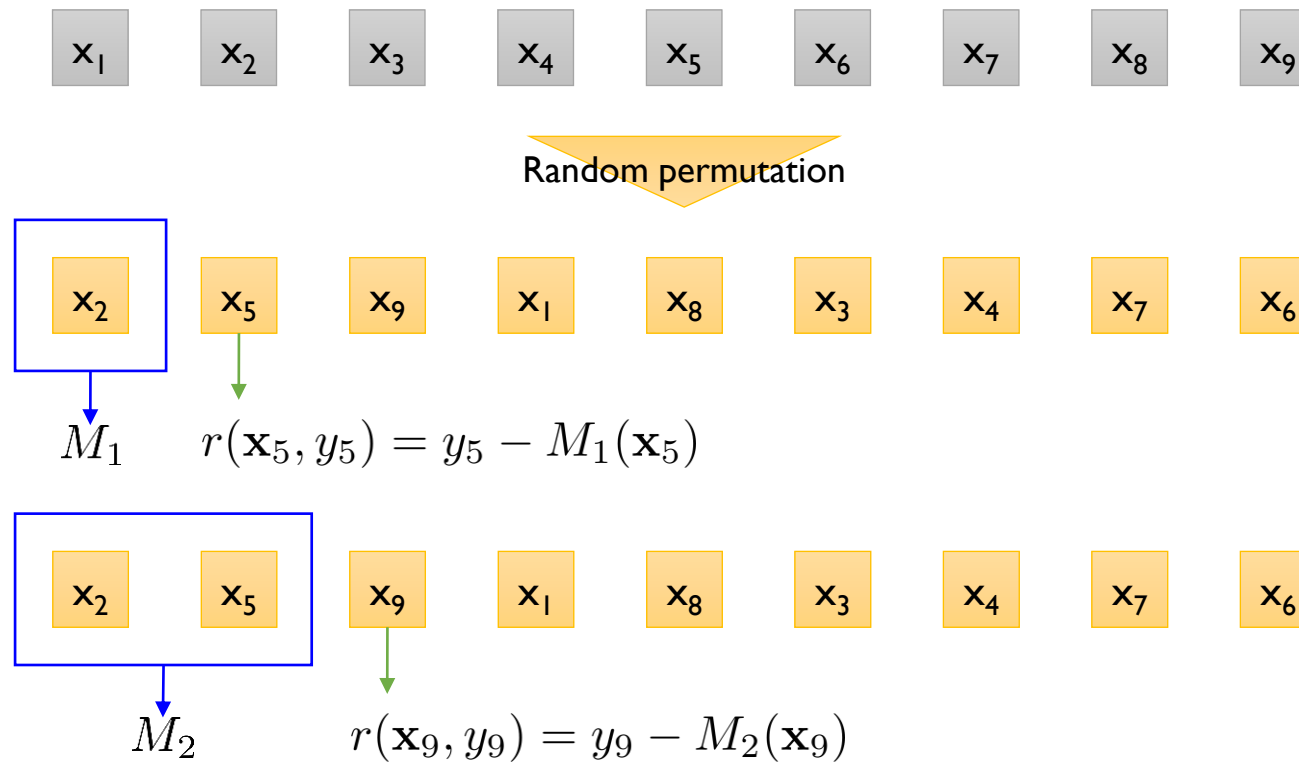
$$h^t = \arg \min_{h \in H} \frac{1}{n} \sum_{k=1}^n \left(-g^t(\mathbf{x}_k, y_k) - h(\mathbf{x}_k) \right)^2$$

- The conditional distribution of the gradient $g^t(\mathbf{x}_k, y_k) | \mathbf{x}_k$ is shifted from that distribution on a test example $g^t(\mathbf{x}, y) | \mathbf{x}$
 - In turn, base predictor h^t is biased from the solution
 - Finally it affects the generalization ability of the trained model F^t
- ✓ When we learn a model with I trees, we need to have F^{I-1} trained without the example \mathbf{x}_k to make the residual $r^{I-1}(\mathbf{x}_k, y_k)$ unshifted

CatBoost: Ordered Boosting

- Ordered Boosting

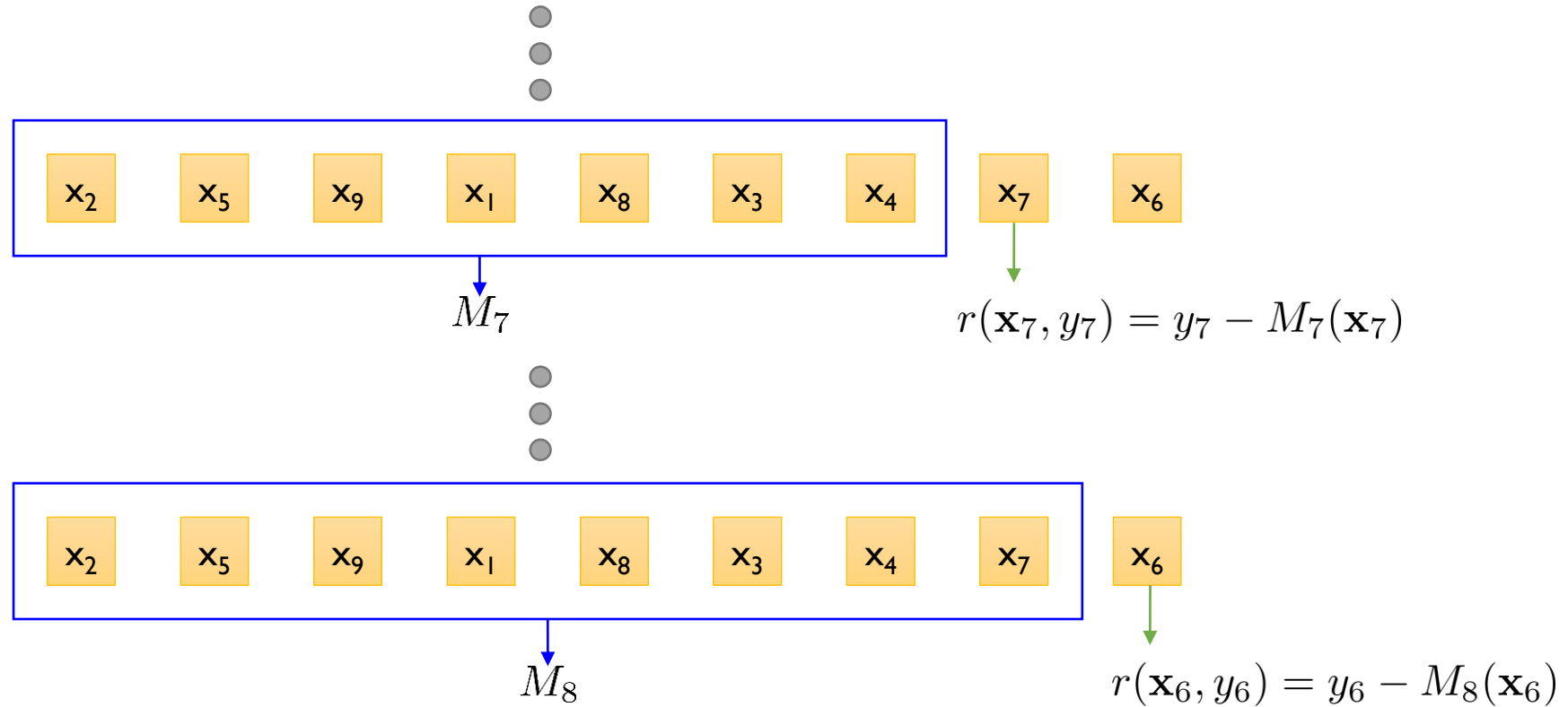
✓ A boosting algorithm not suffering from the prediction shift problem



CatBoost: Ordered Boosting

- Ordered Boosting

- ✓ A boosting algorithm not suffering from the prediction shift problem



- ✓ The same permutation is used for both TS computing and ordered boosting

CatBoost: Ordered Boosting

- Ordered Boosting and Building a Tree in CatBoost

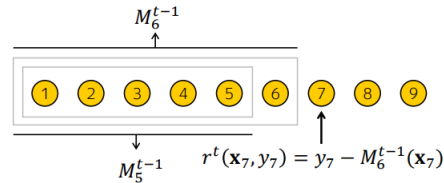


Figure 1: Ordered boosting principle, examples are ordered according to σ .

Algorithm 1: Ordered boosting

input : $\{(\mathbf{x}_k, y_k)\}_{k=1}^n, I$;
 $\sigma \leftarrow$ random permutation of $[1, n]$;
 $M_i \leftarrow 0$ for $i = 1..n$;
for $t \leftarrow 1$ **to** I **do**
 for $i \leftarrow 1$ **to** n **do**
 $r_i \leftarrow y_i - M_{\sigma(i)-1}(\mathbf{x}_i)$;
 for $j \leftarrow 1$ **to** n **do**
 $\Delta M \leftarrow$
 $\text{LearnModel}((\mathbf{x}_j, r_j) :$
 $\sigma(j) \leq i)$;
 $M_i \leftarrow M_i + \Delta M$;
return M_n

Algorithm 2: Building a tree in CatBoost

input : $M, \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \alpha, L, \{\sigma_i\}_{i=1}^s, Mode$
 $grad \leftarrow \text{CalcGradient}(L, M, y)$;
 $r \leftarrow \text{random}(1, s)$;
if $Mode = Plain$ **then**
 $G \leftarrow (grad_r(i) \text{ for } i = 1..n)$;
if $Mode = Ordered$ **then**
 $G \leftarrow (grad_{r, \sigma_r(i)-1}(i) \text{ for } i = 1..n)$;
 $T \leftarrow$ empty tree;
foreach *step of top-down procedure* **do**
 foreach *candidate split* c **do**
 $T_c \leftarrow$ add split c to T ;
 if $Mode = Plain$ **then**
 $\Delta(i) \leftarrow \text{avg}(grad_r(p) \text{ for } p : leaf_r(p) = leaf_r(i))$ for $i = 1..n$;
 if $Mode = Ordered$ **then**
 $\Delta(i) \leftarrow \text{avg}(grad_{r, \sigma_r(i)-1}(p) \text{ for } p : leaf_r(p) = leaf_r(i), \sigma_r(p) < \sigma_r(i))$ for $i = 1..n$;
 $loss(T_c) \leftarrow \text{cos}(\Delta, G)$
 $T \leftarrow \arg \min_{T_c} (loss(T_c))$
if $Mode = Plain$ **then**
 $M_{r'}(i) \leftarrow M_{r'}(i) - \alpha \text{avg}(grad_{r'}(p) \text{ for } p : leaf_{r'}(p) = leaf_{r'}(i))$ for $r' = 1..s, i = 1..n$;
if $Mode = Ordered$ **then**
 $M_{r', j}(i) \leftarrow M_{r', j}(i) - \alpha \text{avg}(grad_{r', j}(p) \text{ for } p : leaf_{r'}(p) = leaf_{r'}(i), \sigma_{r'}(p) \leq j)$ for $r' = 1..s, i = 1..n, j \geq \sigma_{r'}(i) - 1$;
return T, M

CatBoost: Procedure

- CatBoost Procedure

- ✓ **Initialization**: generate $(s+1)$ independent random permutations of the training dataset
 - s permutations to evaluate the split
 - 1 permutation to compute the leaf value of the obtained trees
- ✓ Both TS and predictions used by ordered boosting have a high variance
- ✓ Using only one permutation may increase the variance of the final model predictions, while several permutations can reduce it

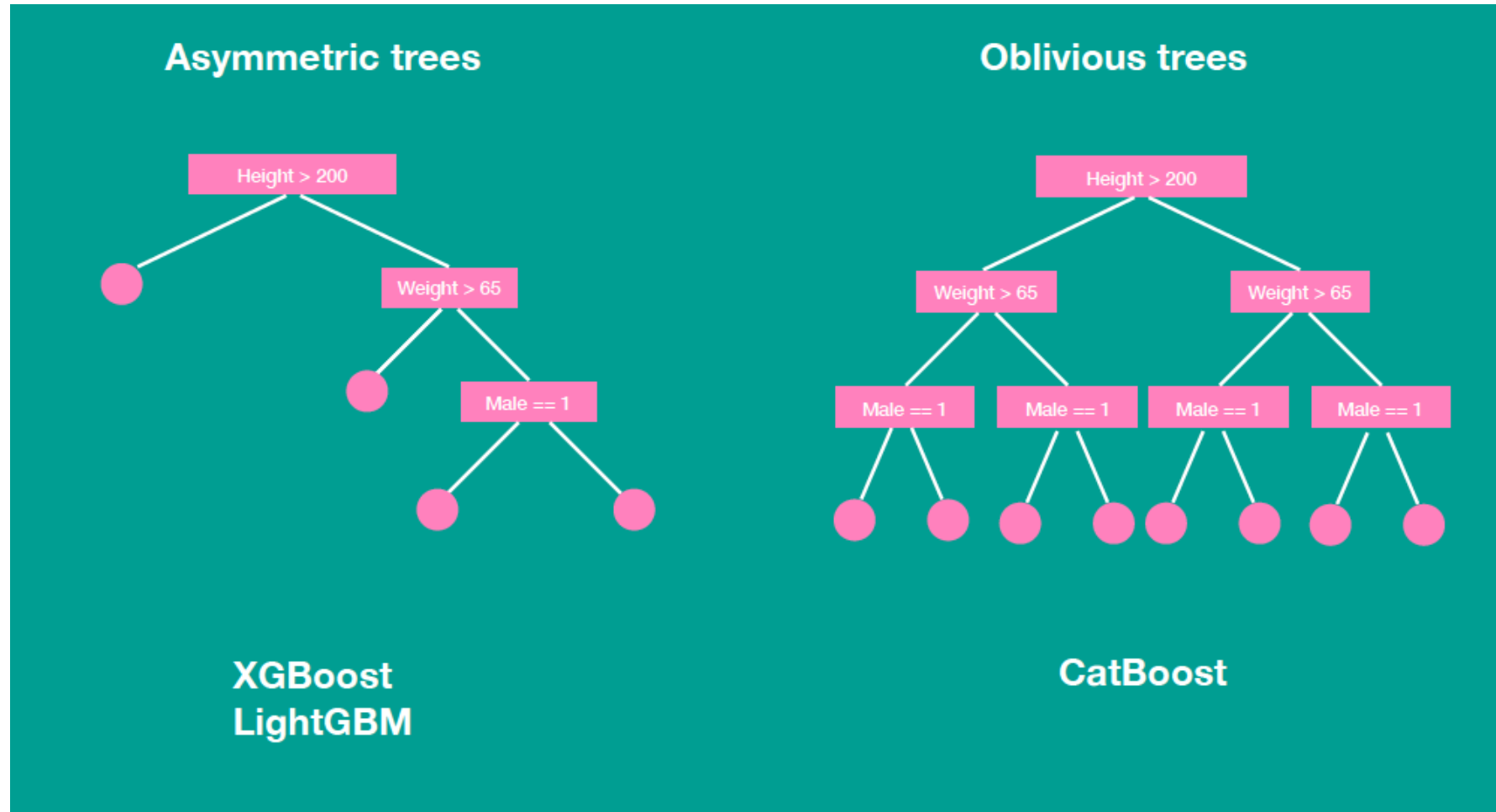
CatBoost: Procedure

- CatBoost Procedure

- ✓ Building an **oblivious tree** the same splitting criterion is used across an entire level of the tree in the ordered boosting mode
 - Maintain the supporting models $M_{r,j}$
 - $M_{r,j}(i)$: the current prediction for the i^{th} example based on the first j examples in the permutation r
 - At each iteration t , sample a random permutation σ_r and construct a tree T_t
 - TS are computed according to this permutation σ_r
 - The permutation affects the tree learning procedure
 - For each example i , take the gradient $grad_{r,\sigma_r(i)-1}(i)$
 - While constructing a tree, the loss of the candidate split c is computed by the cosine similarity between the leaf value and the gradient of each example
 - At the candidate splits evaluation step, the leaf value $\Delta(i)$ for example i is obtained individually by averaging the gradients $grad_{r,\sigma_r(i)-1}$ of the preceding example p lying in the same leaf node of i

CatBoost: Procedure

- Oblivious tree



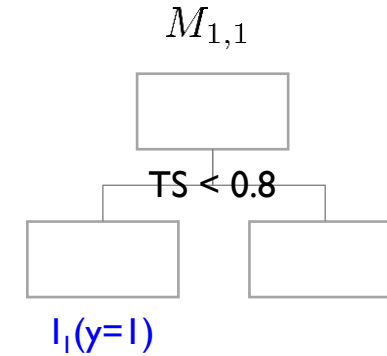
<https://towardsdatascience.com/introduction-to-gradient-boosting-on-decision-trees-with-catboost-d511a9ccbd14>

CatBoost: Procedure

- Ordered Boosting Example

- ✓ Assumption: squared loss (gradient: $f(x) - y$)
- ✓ random permutation is conducted

	...	x^i	...	TS	y	G	Δ
I_1	...	A	...	0.700	1	0	0
I_2	...	B	...	0.700	1		
I_3	...	C	...	0.700	1		
I_4	...	A	...	0.973	0		
I_5	...	B	...	0.973	1		
I_6	...	C	...	0.973	1		
I_7	...	B	...	0.986	0		
I_8	...	C	...	0.986	1		
I_9	...	C	...	0.990	0		
I_{10}	...	C	...	0.749	1		



$$G(I_1) = 0$$

$$\Delta(I_1) = 0$$

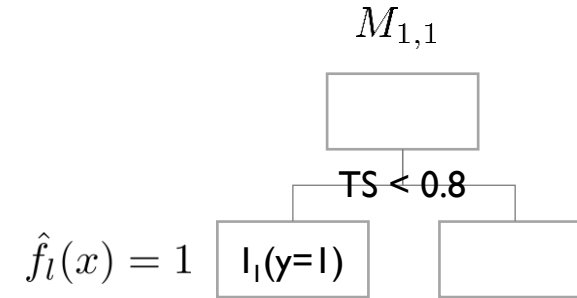
- Lecturer's guess**
- Initialize both the first gradient and leaf value to 0

CatBoost: Procedure

- Ordered Boosting Example

- ✓ Assumption: squared loss (gradient: $f(x) - y$)
- ✓ random permutation is conducted

	...	x^i	...	TS	y	G	Δ
I_1	...	A	...	0.700	1	0	0
I_2	...	B	...	0.700	1	0	0
I_3	...	C	...	0.700	1		
I_4	...	A	...	0.973	0		
I_5	...	B	...	0.973	1		
I_6	...	C	...	0.973	1		
I_7	...	B	...	0.986	0		
I_8	...	C	...	0.986	1		
I_9	...	C	...	0.990	0		
I_{10}	...	C	...	0.749	1		



$$G(I_2) = grad_{1, \sigma_1(2)-1}(I_2)$$

$$= \hat{f}_l(x) - y_2 = 1 - 1 = 0$$

$$\Delta(I_2) = grad_{1, \sigma_1(2)-1}(I_1)$$

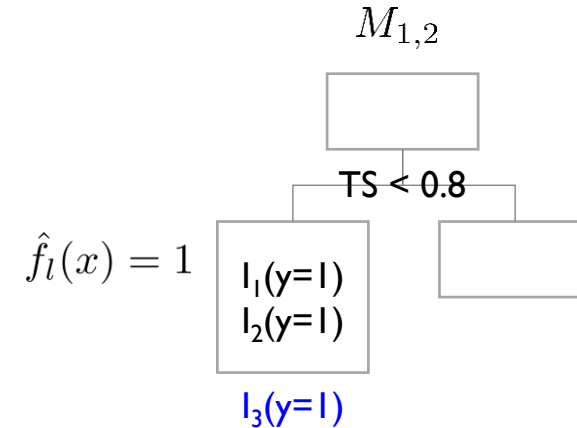
$$= 0$$

CatBoost: Procedure

- Ordered Boosting Example

- ✓ Assumption: squared loss (gradient: $f(x) - y$)
- ✓ random permutation is conducted

	...	x^i	...	TS	y	G	Δ
I_1	...	A	...	0.700	1	0	0
I_2	...	B	...	0.700	1	0	0
I_3	...	C	...	0.700	1	0	0
I_4	...	A	...	0.973	0		
I_5	...	B	...	0.973	1		
I_6	...	C	...	0.973	1		
I_7	...	B	...	0.986	0		
I_8	...	C	...	0.986	1		
I_9	...	C	...	0.990	0		
I_{10}	...	C	...	0.749	1		



$$G(I_3) = grad_{1, \sigma_1(3)-1}(I_3)$$

$$= \hat{f}_l(x) - y_3 = 1 - 1 = 0$$

$$\Delta(I_3) = \frac{1}{2}(grad_{1, \sigma_1(3)-1}(I_1)$$

$$+ grad_{1, \sigma_1(3)-1}(I_2)) = 0$$

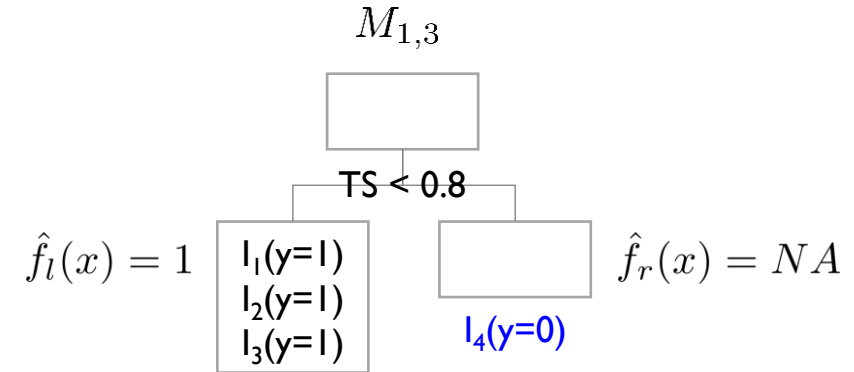
$$= \frac{1}{2}(0 + 0) = 0$$

CatBoost: Procedure

- Ordered Boosting Example

- ✓ Assumption: squared loss (gradient: $f(x) - y$)
- ✓ random permutation is conducted

	...	x^i	...	TS	y	G	Δ
I_1	...	A	...	0.700	1	0	0
I_2	...	B	...	0.700	1	0	0
I_3	...	C	...	0.700	1	0	0
I_4	...	A	...	0.973	0	0	0
I_5	...	B	...	0.973	1		
I_6	...	C	...	0.973	1		
I_7	...	B	...	0.986	0		
I_8	...	C	...	0.986	1		
I_9	...	C	...	0.990	0		
I_{10}	...	C	...	0.749	1		



$$G(I_4) = grad_{1, \sigma_1(4)-1}(I_4) = 0$$

$$\Delta(I_4) = 0$$

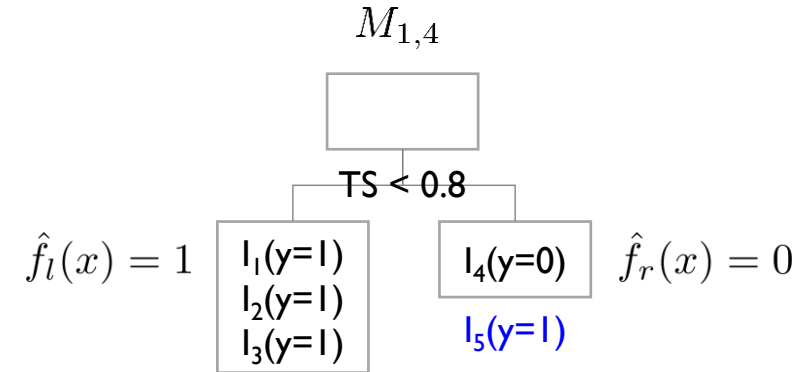
Lecturer's guess

CatBoost: Procedure

- Ordered Boosting Example

- ✓ Assumption: squared loss (gradient: $f(x) - y$)
- ✓ random permutation is conducted

	...	x^i	...	TS	y	G	Δ
I_1	...	A	...	0.700	1	0	0
I_2	...	B	...	0.700	1	0	0
I_3	...	C	...	0.700	1	0	0
I_4	...	A	...	0.973	0	0	0
I_5	...	B	...	0.973	1	-1	0
I_6	...	C	...	0.973	1		
I_7	...	B	...	0.986	0		
I_8	...	C	...	0.986	1		
I_9	...	C	...	0.990	0		
I_{10}	...	C	...	0.749	1		



$$G(I_5) = grad_{1, \sigma_1(5)-1}(I_5)$$

$$= \hat{f}_r(x) - y_5 = -1$$

$$\Delta(I_5) = grad_{1, \sigma_1(5)-1}(I_4)$$

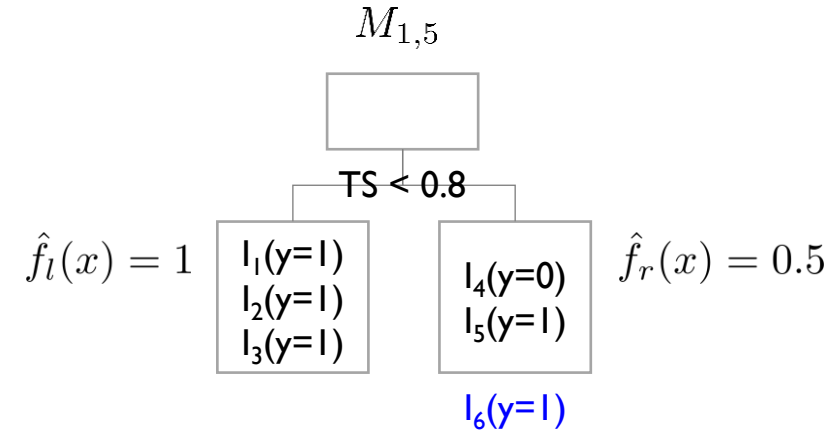
$$= 0$$

CatBoost: Procedure

- Ordered Boosting Example

- ✓ Assumption: squared loss (gradient: $f(x) - y$)
- ✓ random permutation is conducted

	...	x^i	...	TS	y	G	Δ
I_1	...	A	...	0.700	1	0	0
I_2	...	B	...	0.700	1	0	0
I_3	...	C	...	0.700	1	0	0
I_4	...	A	...	0.973	0	0	0
I_5	...	B	...	0.973	1	-1	0
I_6	...	C	...	0.973	1	-0.5	-0.5
I_7	...	B	...	0.986	0		
I_8	...	C	...	0.986	1		
I_9	...	C	...	0.990	0		
I_{10}	...	C	...	0.749	1		



$$G(I_6) = grad_{1, \sigma_1(6)-1}(I_6)$$

$$= \hat{f}_r(x) - y_6 = 0.5 - 1 = -0.5$$

$$\Delta(I_6) = \frac{1}{2} (grad_{1, \sigma_1(6)-1}(I_4)$$

$$+ grad_{1, \sigma_1(6)-1}(I_5))$$

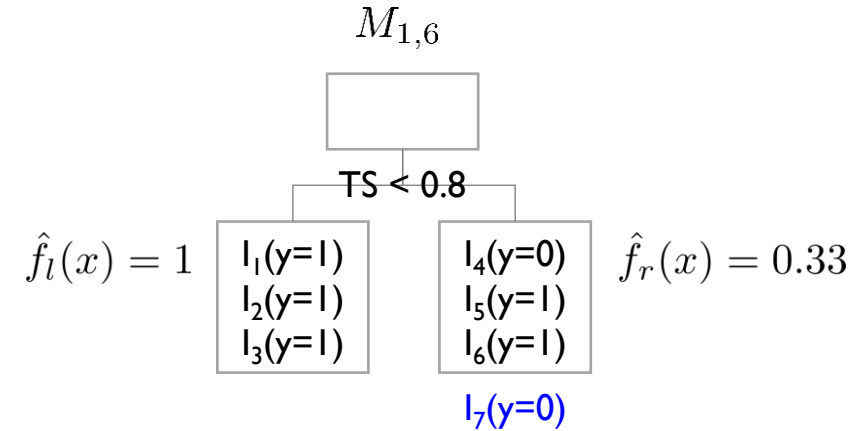
$$= \frac{1}{2} (0 - 1) = -0.5$$

CatBoost: Procedure

- Ordered Boosting Example

- ✓ Assumption: squared loss (gradient: $f(x) - y$)
- ✓ random permutation is conducted

	...	x^i	...	TS	y	G	Δ
I_1	...	A	...	0.700	1	0	0
I_2	...	B	...	0.700	1	0	0
I_3	...	C	...	0.700	1	0	0
I_4	...	A	...	0.973	0	0	0
I_5	...	B	...	0.973	1	-1	0
I_6	...	C	...	0.973	1	-0.5	-0.5
I_7	...	B	...	0.986	0	0.33	-0.5
I_8	...	C	...	0.986	1		
I_9	...	C	...	0.990	0		
I_{10}	...	C	...	0.749	1		



$$G(I_7) = grad_{1, \sigma_1(7)-1}(I_7)$$

$$= \hat{f}_r(x) - y_7 = 0.33 - 0 = 0.33$$

$$\Delta(I_7) = \frac{1}{3} (grad_{1, \sigma_1(7)-1}(I_4)$$

$$+ grad_{1, \sigma_1(7)-1}(I_5)$$

$$+ grad_{1, \sigma_1(7)-1}(I_6))$$

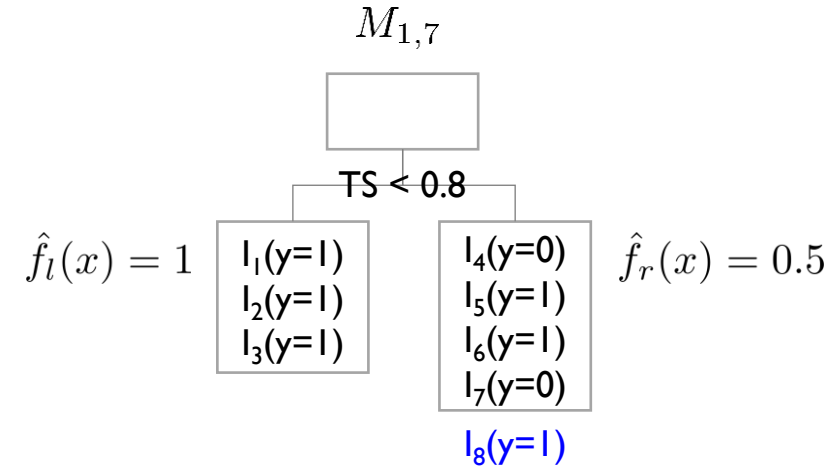
$$= \frac{1}{3} (0 - 1 - 0.5) = -0.5$$

CatBoost: Procedure

- Ordered Boosting Example

- ✓ Assumption: squared loss (gradient: $f(x) - y$)
- ✓ random permutation is conducted

	...	x^i	...	TS	y	G	Δ
I_1	...	A	...	0.700	1	0	0
I_2	...	B	...	0.700	1	0	0
I_3	...	C	...	0.700	1	0	0
I_4	...	A	...	0.973	0	0	0
I_5	...	B	...	0.973	1	-1	0
I_6	...	C	...	0.973	1	-0.5	-0.5
I_7	...	B	...	0.986	0	0.33	-0.5
I_8	...	C	...	0.986	1	-0.5	-0.29
I_9	...	C	...	0.990	0		
I_{10}	...	C	...	0.749	1		



$$G(I_8) = grad_{1, \sigma_1(8)-1}(I_8)$$

$$= \hat{f}_r(x) - y_8 = 0.5 - 1 = -0.5$$

$$\Delta(I_8) = \frac{1}{4} (grad_{1, \sigma_1(8)-1}(I_4) + grad_{1, \sigma_1(8)-1}(I_5)$$

$$+ grad_{1, \sigma_1(8)-1}(I_6) + grad_{1, \sigma_1(8)-1}(I_7))$$

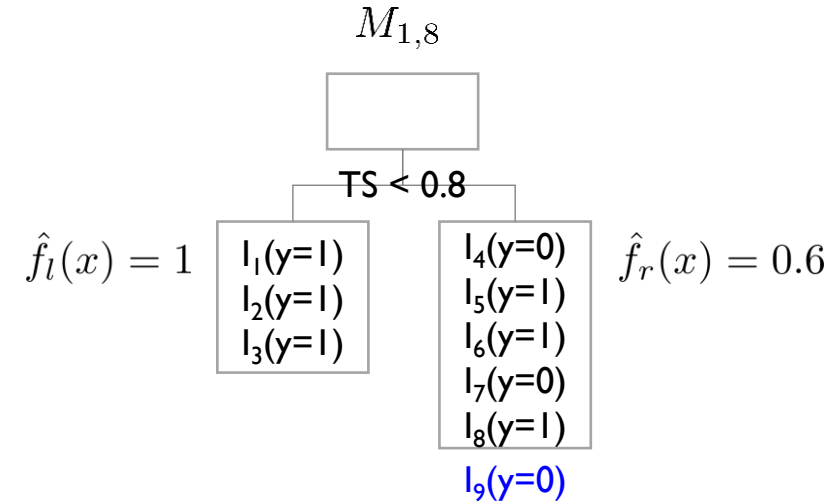
$$= \frac{1}{4} (0 - 1 - 0.5 + 0.33) = -0.29$$

CatBoost: Procedure

- Ordered Boosting Example

- ✓ Assumption: squared loss (gradient: $f(x) - y$)
- ✓ random permutation is conducted

	...	x^i	...	TS	y	G	Δ
I_1	...	A	...	0.700	1	0	0
I_2	...	B	...	0.700	1	0	0
I_3	...	C	...	0.700	1	0	0
I_4	...	A	...	0.973	0	0	0
I_5	...	B	...	0.973	1	-1	0
I_6	...	C	...	0.973	1	-0.5	-0.5
I_7	...	B	...	0.986	0	0.33	-0.5
I_8	...	C	...	0.986	1	-0.5	-0.29
I_9	...	C	...	0.990	0	0.6	-0.39
I_{10}	...	C	...	0.749	1		



$$G(I_9) = \text{grad}_{1, \sigma_1(9)-1}(I_9)$$

$$= \hat{f}_r(x) - y_9 = 0.6 - 0 = 0.6$$

$$\Delta(I_9) = \frac{1}{5} (\text{grad}_{1, \sigma_1(9)-1}(I_4) + \text{grad}_{1, \sigma_1(9)-1}(I_5)$$

$$+ \text{grad}_{1, \sigma_1(9)-1}(I_6) + \text{grad}_{1, \sigma_1(9)-1}(I_7)$$

$$+ \text{grad}_{1, \sigma_1(9)-1}(I_8))$$

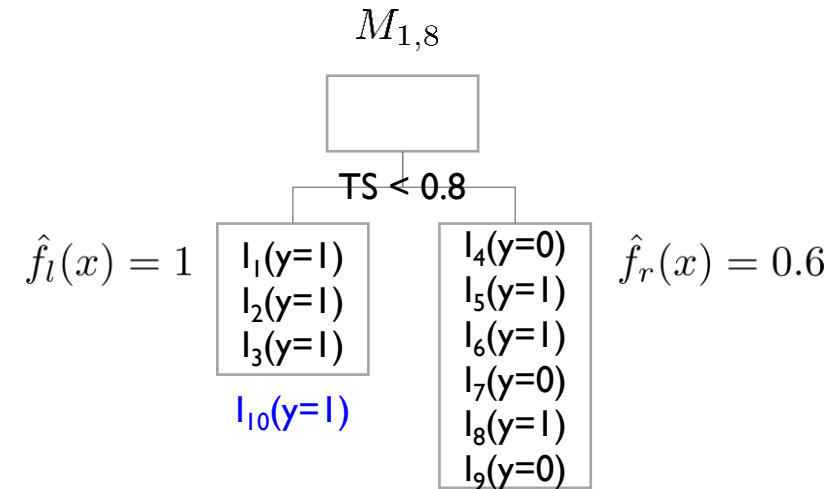
$$= \frac{1}{5} (0 - 1 - 0.5 + 0.33 - 0.5) = -0.39$$

CatBoost: Procedure

- Ordered Boosting Example

- ✓ Assumption: squared loss (gradient: $f(x) - y$)
- ✓ random permutation is conducted

	...	x^i	...	TS	y	G	Δ
I_1	...	A	...	0.700	1	0	0
I_2	...	B	...	0.700	1	0	0
I_3	...	C	...	0.700	1	0	0
I_4	...	A	...	0.973	0	0	0
I_5	...	B	...	0.973	1	-1	0
I_6	...	C	...	0.973	1	-0.5	-0.5
I_7	...	B	...	0.986	0	0.33	-0.5
I_8	...	C	...	0.986	1	-0.5	-0.29
I_9	...	C	...	0.990	0	0.6	-0.39
I_{10}	...	C	...	0.749	1	0	0



$$G(I_10) = grad_{1,\sigma_1(10)-1}(I_{10})$$

$$= \hat{f}_l(x) - y_{10} = 1 - 1 = 0$$

$$\Delta(I_10) = \frac{1}{3} (grad_{1,\sigma_1(10)-1}(I_1) + grad_{1,\sigma_1(10)-1}(I_2)$$

$$+ grad_{1,\sigma_1(10)-1}(I_3))$$

$$= \frac{1}{3} (0 + 0 + 0) = 0$$

CatBoost: Procedure

- Ordered Boosting Example

- ✓ Assumption: squared loss (gradient: $f(x) - y$)
- ✓ random permutation is conducted

	...	x^i	...	TS	y	G	Δ
I_1	...	A	...	0.700	1	0	0
I_2	...	B	...	0.700	1	0	0
I_3	...	C	...	0.700	1	0	0
I_4	...	A	...	0.973	0	0	0
I_5	...	B	...	0.973	1	-1	0
I_6	...	C	...	0.973	1	-0.5	-0.5
I_7	...	B	...	0.986	0	0.33	-0.5
I_8	...	C	...	0.986	1	-0.5	-0.29
I_9	...	C	...	0.990	0	0.6	-0.39
I_{10}	...	C	...	0.749	1	0	0

```

foreach step of top-down procedure do
  foreach candidate split c do
     $T_c \leftarrow$  add split  $c$  to  $T$ ;
    if  $Mode = Plain$  then
       $\Delta(i) \leftarrow \text{avg}(\text{grad}_r(p)$  for
         $p : \text{leaf}_r(p) = \text{leaf}_r(i)$  for  $i = 1..n$ ;
    if  $Mode = Ordered$  then
       $\Delta(i) \leftarrow \text{avg}(\text{grad}_{r, \sigma_r(i)-1}(p)$  for
         $p : \text{leaf}_r(p) = \text{leaf}_r(i), \sigma_r(p) < \sigma_r(i)$ 
        for  $i = 1..n$ ;
     $\text{loss}(T_c) \leftarrow \cos(\Delta, G)$ 
   $T \leftarrow \arg \min_{T_c} (\text{loss}(T_c))$ 
  
```

$$\text{loss}(T_c) = \cos(\Delta, G) = -0.2003$$

(TS < 0.8)

CatBoost: Procedure

- Choosing leaf values
 - ✓ Given all the trees constructed, the leaf values of the final model F are calculated by the standard gradient boosting procedure
 - ✓ Training examples i are matched to leaves $leaf_0(i)$ using permutation σ_0 to calculate TS
 - ✓ When the final model F is applied to a new example at testing time, TS is calculated on the whole training data

CatBoost: Performance

- Experimental Results

Table 8: Comparison with baselines: logloss / zero-one loss, relative increase is presented in the brackets.

	CatBoost	LightGBM	XGBoost
Adult	0.2695 / 0.1267	0.2760 (+2.4%) / 0.1291 (+1.9%)	0.2754 (+2.2%) / 0.1280 (+1.0%)
Amazon	0.1394 / 0.0442	0.1636 (+17%) / 0.0533 (+21%)	0.1633 (+17%) / 0.0532 (+21%)
Click	0.3917 / 0.1561	0.3963 (+1.2%) / 0.1580 (+1.2%)	0.3962 (+1.2%) / 0.1581 (+1.2%)
Epsilon	0.2647 / 0.1086	0.2703 (+1.5%) / 0.114 (+4.1%)	0.2993 (+11%) / 0.1276 (+12%)
Appetency	0.0715 / 0.01768	0.0718 (+0.4%) / 0.01772 (+0.2%)	0.0718 (+0.4%) / 0.01780 (+0.7%)
Churn	0.2319 / 0.0719	0.2320 (+0.1%) / 0.0723 (+0.6%)	0.2331 (+0.5%) / 0.0730 (+1.6%)
Internet	0.2089 / 0.0937	0.2231 (+6.8%) / 0.1017 (+8.6%)	0.2253 (+7.9%) / 0.1012 (+8.0%)
Upselling	0.1662 / 0.0490	0.1668 (+0.3%) / 0.0491 (+0.1%)	0.1663 (+0.04%) / 0.0492 (+0.3%)
Kick	0.2855 / 0.0949	0.2957 (+3.5%) / 0.0991 (+4.4%)	0.2946 (+3.2%) / 0.0988 (+4.1%)

CatBoost: Performance

- XGBoost vs. LightGBM vs. CatBoost

Function	XGBoost	CatBoost	Light GBM
Important parameters which control overfitting	<ol style="list-style-type: none"> 1. learning_rate or eta – optimal values lie between 0.01-0.2 2. max_depth 3. min_child_weight: similar to min_child leaf; default is 1 	<ol style="list-style-type: none"> 1. Learning_rate 2. Depth - value can be any integer up to 16. Recommended - [1 to 10] 3. No such feature like min_child_weight 4. l2-leaf-reg: L2 regularization coefficient. Used for leaf value calculation (any positive integer allowed) 	<ol style="list-style-type: none"> 1. learning_rate 2. max_depth: default is 20. Important to note that tree still grows leaf-wise. Hence it is important to tune num_leaves (number of leaves in a tree) which should be smaller than $2^{(\text{max_depth})}$. It is a very important parameter for LGBM 3. min_data_in_leaf: default=20, alias= min_data, min_child_samples
Parameters for categorical values	Not Available	<ol style="list-style-type: none"> 1. cat_features: It denotes the index of categorical features 2. one_hot_max_size: Use one-hot encoding for all features with number of different values less than or equal to the given parameter value (max – 255) 	<ol style="list-style-type: none"> 1. categorical_feature: specify the categorical features we want to use for training our model
Parameters for controlling speed	<ol style="list-style-type: none"> 1. colsample_bytree: subsample ratio of columns 2. subsample: subsample ratio of the training instance 3. n_estimators: maximum number of decision trees; high value can lead to overfitting 	<ol style="list-style-type: none"> 1. rsm: Random subspace method. The percentage of features to use at each split selection 2. No such parameter to subset data 3. iterations: maximum number of trees that can be built; high value can lead to overfitting 	<ol style="list-style-type: none"> 1. feature_fraction: fraction of features to be taken for each iteration 2. bagging_fraction: data to be used for each iteration and is generally used to speed up the training and avoid overfitting 3. num_iterations: number of boosting iterations to be performed; default=100

CatBoost: Performance

Anghel et al. (2019)

- XGBoost vs. LightGBM vs. CatBoost

Table 3: Best test scores across algorithms and datasets.

Dataset	Baseline		XGBoost		LightGBM		Catboost	
	Test	Val	Test	Val	Test	Val	Test	Val
Higgs	0.4996	0.5005	0.8353	0.8512	<i>0.8573</i>	0.8577	0.8498	0.8496
Epsilon	0.4976	0.5008	-	-	0.9513	0.9518	<i>0.9537</i>	0.9538
Microsoft	0.2251	0.3974	<i>0.4917</i>	0.6443	0.4871	0.6473	0.3782	0.5492
Yahoo	0.5802	0.8106	<i>0.7983</i>	0.9146	0.7965	0.9142	0.7351	0.8849

CatBoost: Performance

Anghel et al. (2019)

- XGBoost vs. LightGBM vs. CatBoost

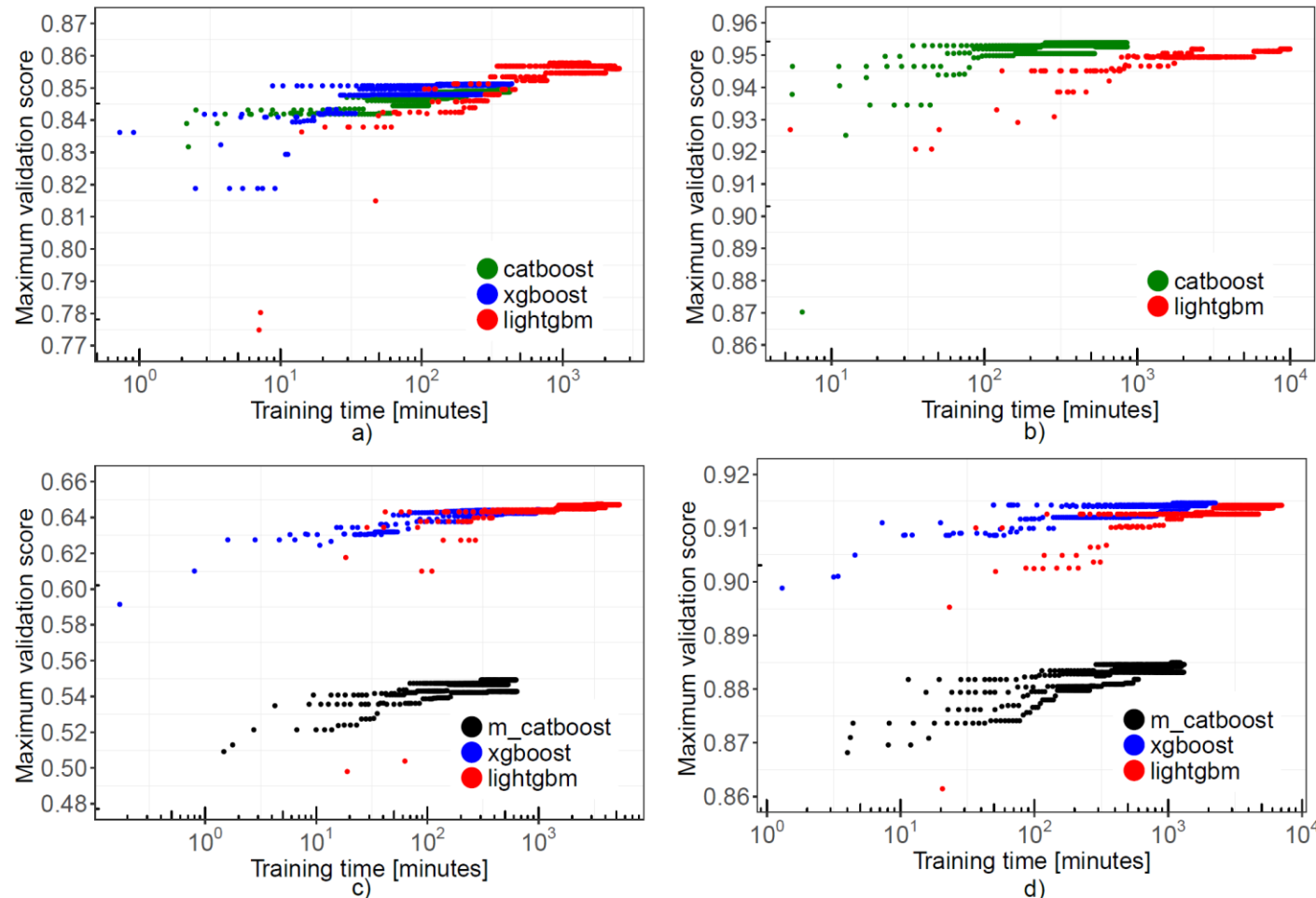


Figure 2: Max. validation score vs. total HPO runtime (a) Higgs (b) Epsilon (c) Microsoft (d) Yahoo.

