Minimal Variance Sampling in Stochastic Gradient Boosting

Bulat Ibragimov, Gleb Gusev

Yandex Sberbank

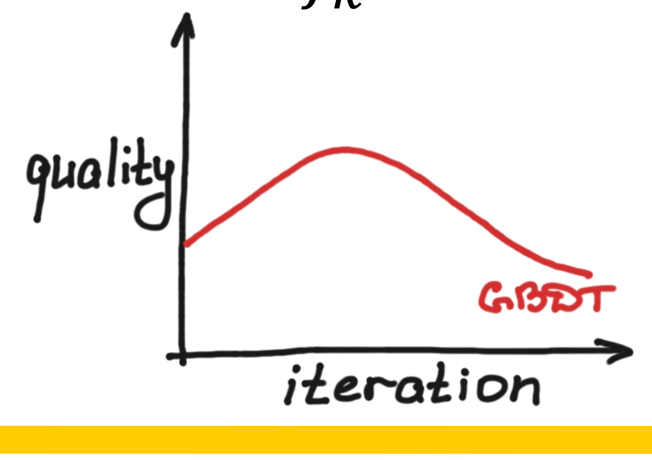
Gradient Boosted Decision Trees (GBDT)

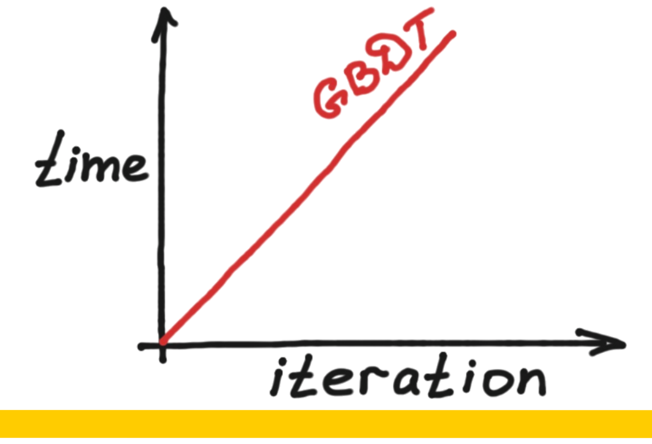
Iterative process of negative gradient steps

$$g_i^k = \frac{\partial L(y_i, \hat{y}_i)}{\partial \hat{y}_i}\bigg|_{\hat{y}_i = F_{k-1}(x_i)}$$

$$f_k(x_i) \approx g_i^k, \qquad F_N(x_i) = -\sum_{k=1}^{N} \alpha f_k(x_i)$$

f_k - decision tree function



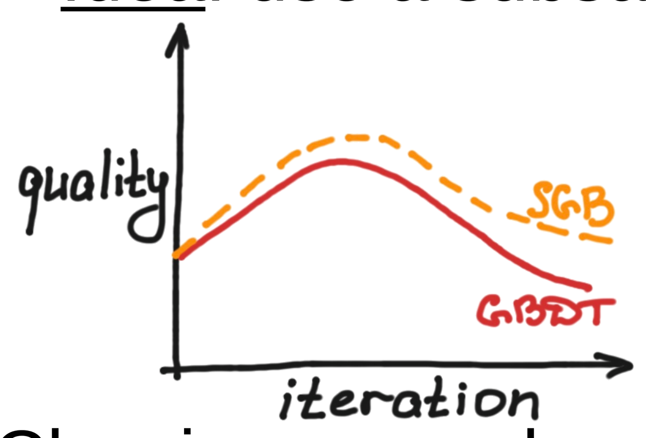


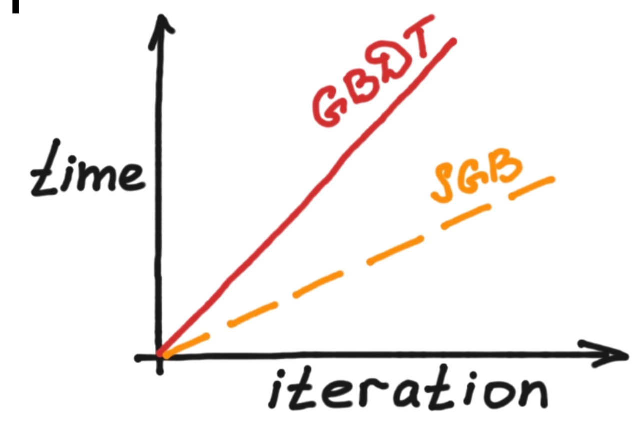
1

Stochastic Gradient Boosting (SGB)

Speed-up & Regularization

Idea: use a subsample at each iteration





Classic approach: uniform sampling (SGB[0])

Advanced approach: non-uniform sampling (e.g. GOSS[1])

☺ : it works

☹ : understudied, choice is heuristic

[0]J.H.Friedman "Stochastic Gradient Boosting"

[1]Guolin Ke, et.al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree"

2

Problem formulation

Unknown split score:

$$S = \sum_{l \in L} \frac{\sum_{i \in l} g_i^2}{|l|}, \textit{where } L - \textit{set of leaves}$$

Perform random sampling:

$$\xi_1, \dots, \xi_n \ - independent\ \ s.t.\ \ \xi_i \sim Bern(p_i)$$

$$i_{th}\ object\ is\ sampled\ iff\ \xi_i = 1$$

Construct an estimator: $\hat{S} = \sum_{l \in L} \frac{\sum_{i \in l} \frac{1}{p_i} \xi_i g_i^2}{\sum_{i \in l} \frac{1}{p_i} \xi_i}$

Such that: $E\Delta^2 = E\left(\hat{S} - S\right)^2 \rightarrow \min_{p_1, \dots, p_n}$

3

Main result

$$E\Delta^2 \approx \sum_{l \in L} c_l^2 \left(4Var(x_l) - 4c_l Cov(x_l, y_l) + c_l^2 Var(y_l)\right)$$

$$E\Delta^2 \lesssim 2 \sum_{l \in L} c_l^2 \left(4Var(x_l) + c_l^2 Var(y_l)\right),$$

$$where\ x_l = \sum_{i \in l} \frac{1}{p_i} \xi_i g_i, y_l = \sum_{i \in l} \frac{1}{p_i} \xi_i,\ and\ c_l = \frac{Ex_l}{Ey_l}$$

Upper bound minimization problem:

$$\sum_{i=1}^{n} \frac{1}{p_i} g_i^2 + \lambda \sum_{i=1}^{n} \frac{1}{p_i} \rightarrow \min\ w.r.t. \sum p_i = ns,$$

$$s\ - sampling\ ratio, \lambda - hyperparameter$$

4

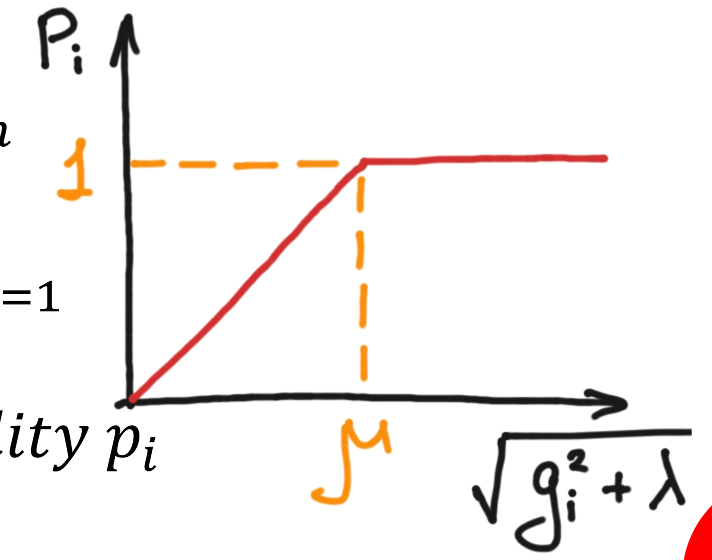
Minimal Variance Sampling (MVS)

$$p_i = min\left(\frac{1}{\mu}\sqrt{g_i^2 + \lambda}, 1\right),$$

$$\mu\ is\ set\ such\ that\ \sum p_i = ns$$

Algorithm:

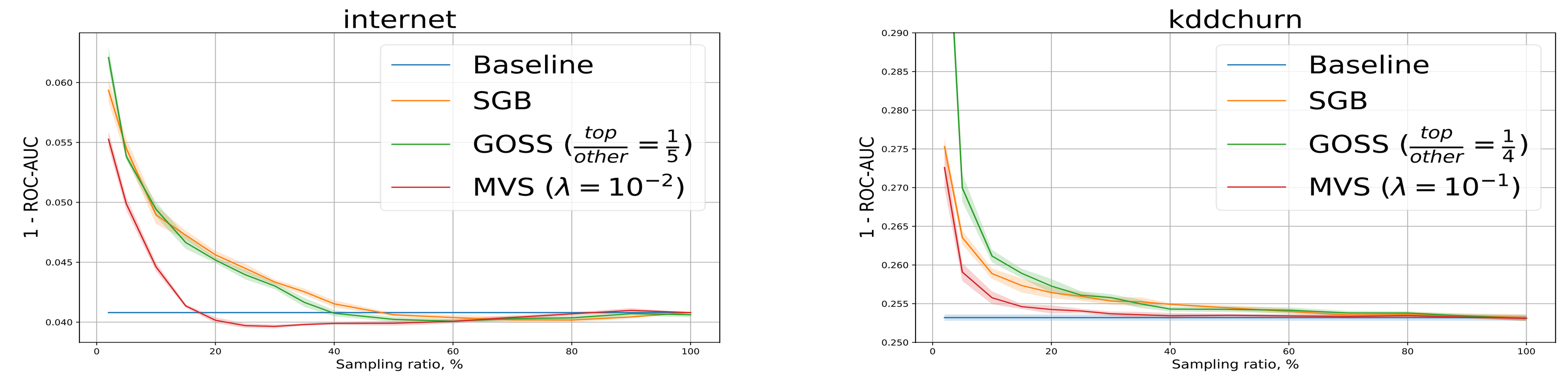
1. Calculate $\{\hat{g}_i\}_{i=1}^n = \left\{\sqrt{g_i^2 + \lambda}\right\}_{i=1}^n$
2. Calculate threshold μ
3. Sample object x_i with probability p_i
4. Assign weight $w_i = \frac{1}{p_i}$



*It is possible to generalize to second-order methods

5

Experiments



Relative error change, average over datasets:

Sample rate	0.02	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.5
SGB	+19.92%	+11.35%	+6.83%	+4.99%	+3.84%	+3.03%	+2.17%	+1.57%	+1.10%	+0.42%
GOSS	+22.37%	+12.75%	+8.00%	+5.32%	+3.39%	+2.25%	+1.41%	+0.75%	+0.23%	-0.16%
MVS	+13.93%	+7.76%	**+3.69%**	+1.91%	+0.74%	+0.14%	**-0.21%**	**-0.43%**	**-0.41%**	-0.45%
MVS Adaptive	**+13.72%**	**+7.47%**	+3.71%	**+1.70%**	**+0.55%**	**-0.03%**	-0.07%	-0.28%	-0.32%	**-0.51%**

Relative learning time gain:

	SGB	GOSS	MVS
time difference	-20.7%	-20.4%	-27.7%

*used datasets: KDD Internet, Adult, Amazon, KDD Upselling, Kick, KDD Churn,Click, Higgs, Recsys

6