Several custom losses implementation for the median regression task in Pyboost

Stas Pyatkin [1] **Dmitriy Kornilov** [1] **Danil Ivanov** [1] **Danil Gusak** [1] **Bari Khairullin** [1]

Abstract

Modern gradient boosting toolkits are very complex and are written in low-level programming languages. As a result,

- It is hard to customize them to suit one's needs
- New ideas and methods are not easy to implement
- It is difficult to understand how they work.

Py-Boost is a Python-based gradient boosting library which aims at overcoming the aforementioned problems. Py-boost can be easily customized. We have implemented several custom losses for the median regression task and compare results on the several datasets. Each loss was hard-coded as first- and second-order derivatives.

Github repo: `Py-Boost-custom-losses`

1. Introduction

The main contributions of this report are as follows:

- Several loss functions have been implemented for Py-boost library.
- Regressors with implemented loss functions have been compared by MAE metric.

2. Preliminaries

2.1. Gradient boosting on the Decision Trees

$$R^{(t)} \approx \sum_{i=1}^{m} \left[g_i h_i(x_i) + \frac{1}{2} d_i h_i^2(x_i) \right] + \Omega(h_t) \qquad (1)$$

$$g_i = \delta_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}),$$

$$d_i = \delta^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$$

The decision:

$$\omega_j^* = -\frac{G_j}{D_j + \lambda}, G_j = \sum_{i \in I_j} g_i, D_j = \sum_{i \in I_j} d_i$$

2.2. Median regression

$$MAE = \sum |y_{pred,i} - y_i|$$
$$\nabla MAE = \operatorname{sign}(y_{pred} - y)$$
$$\nabla^2 MAE = \operatorname{diag}(0, ..., 0)$$

What is a good alternative loss function for median regression?

2.3. Py-Boost

Py-Boost is GB library, supports all common features and hyperparameters. Computationally efficient, works only on GPU, uses Cupy and Numba. Sampling strategy, training control, losses and metrics are easily customizable.

3. Related work

Tree boosting is a highly effective and widely used machine learning method. In (Chen & Guestrin, 2016), authors describe a scalable end-to-end tree boosting system called XGBoost, which is used widely by data scientists to achieve state-of-the-art results on many machine learning challenges. A novel sparsity-aware algorithm for sparse data and weighted quantile sketch for approximate tree learning have been proposed. Insights on cache access patterns, data compression and sharding to build a scalable tree boosting system were provided. By combining these insights, XGBoost scales beyond billions of examples using far fewer resources than existing systems.

Gradient Boosting Decision Tree (GBDT) is a popular machine learning algorithm, and has quite a few effective implementations such as XGBoost and pGBRT. Although many engineering optimizations have been adopted in these implementations, the efficiency and scalability are still unsatisfactory when the feature dimension is high and data size is large. A major reason is that for each feature, they need

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Danil Ivanov <danil.ivanov@skoltech.ru>.

Final Projects of the Machine Learning 2020 Course, Skoltech, Moscow, Russian Federation, 2020.

to scan all the data instances to estimate the information gain of all possible split points, which is very time consuming. To tackle this problem, in (Ke et al., 2017) authors proposed two novel techniques: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). With GOSS, authors excluded a significant proportion of data instances with small gradients, and only use the rest to estimate the information gain. Authors proved that, since the data instances with larger gradients play a more important role in the computation of information gain, GOSS can obtain quite accurate estimation of the information gain with a much smaller data size. With EFB, authors bundled mutually exclusive features (i.e., they rarely take nonzero values simultaneously), to reduce the number of features. Authors proved that finding the optimal bundling of exclusive features is NP-hard, but a greedy algorithm can achieve quite good approximation ratio (and thus can effectively reduce the number of features without hurting the accuracy of split point determination by much). Authors called our new GBDT implementation with GOSS and EFB LightGBM. Our experiments on multiple public datasets show that, LightGBM speeds up the training process of conventional GBDT by up to over 20 times while achieving almost the same accuracy.

Paper (Prokhorenkova et al., 2018) presents the key algorithmic techniques behind CatBoost, a new gradient boosting toolkit. Their combination leads to CatBoost outperforming other publicly available boosting implementations in terms of quality on a variety of datasets. Two critical algorithmic advances introduced in CatBoost are the implementation of ordered boosting, a permutation-driven alternative to the classic algorithm, and an innovative algorithm for processing categorical features. Both techniques were created to fight a prediction shift caused by a special kind of target leakage present in all currently existing implementations of gradient boosting algorithms. In this paper, we provide a detailed analysis of this problem and demonstrate that proposed algorithms solve it effectively, leading to excellent empirical results.

Gradient boosting constructs additive regression models by sequentially fitting a simple parameterized function (base learner) to current "pseudo"-residuals by least squares at each iteration. The pseudo-residuals are the gradient of the loss functional being minimized, with respect to the model values at each training data point evaluated at the current step. It is shown in (Friedman, 2002) that both the approximation accuracy and execution speed of gradient boosting can be substantially improved by incorporating randomization into the procedure. Specifically, at each iteration a subsample of the training data is drawn at random (without replacement) from the full training data set. This randomly selected subsample is then used in place of the full sample to fit the base learner and compute the model update for the current iteration. This randomized approach also increases robustness against overcapacity of the base learner.

In (Iosipoi & Vakhrushev, 2022) authors propose novel methods aiming to accelerate the training process of GBDT in the multioutput scenario. The idea behind these methods lies in the approximate computation of a scoring function used to find the best split of decision trees. These methods are implemented in SketchBoost, which itself is integrated into our easily customizable Python-based GPU implementation of GBDT called Py-Boost. Numerical study demonstrates that SketchBoost speeds up the training process of GBDT by up to over 40 times while achieving comparable or even better performance.

4. Algorithms and Models

We tried several lossess:

- $MSLE = \sum(\log(1 + y_{pred,i}) - \log(1 + y_i))^2$
- $LogCosh = \sum \log(\text{cohs}(y_{pred,i} - y_i))$
  $\nabla LogCosh = \tanh(y_{pred} - y)$
  $\nabla^2 LogCosh = \text{diag}(\text{sech}^2(y_{pred,i} - y_i))$
- $\text{Huber}_\delta(x) = \sum\{|x| < \delta : \frac{1}{2}x^2, |x| > \delta : \delta(|x| - \frac{1}{2}\delta)\}$
- $\text{PseudoHuber}_\delta(x) = \delta^2(\sqrt{1 + (\frac{x}{\delta})^2} - 1)$

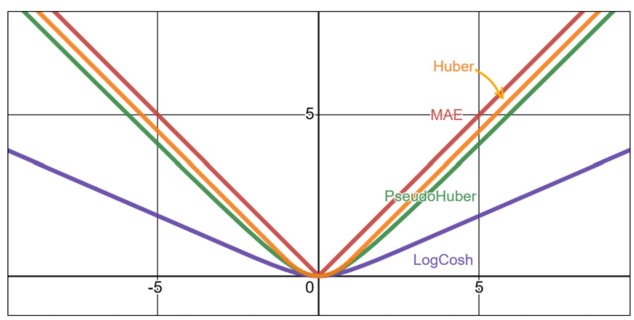

Figure 1. Customs losses investigated in the study

5. Experiments and Results

Github repo: `Py-Boost-custom-losses`

5.1. Hyperparameter tuning

```
def objective(trial, params=params):
    params = {x: trial.suggest_float(x,
    *params[x]) for x in params}

    model = GradientBoosting(
```

Table 1. Losses comparison on Allstate dataset

Loss	Pseudo Huber	Huber	MSLE	MAE	LogC
Test loss	1139	1139	1140	1915	1152

Table 2. Losses comparison on House Prices dataset

Loss	Pseudo Huber	Huber	MSLE	MAE	LogC
Test loss	5476	5889	5800	12687	7356

```
        loss=loss(**params),
        metric=CustomMAEMetric(),
        ntrees=NUM_TREES,
        lr=LR_TUNE,
        es=ES,
        lambda_l2 = trial
        .suggest_float(
            "lambda_l2", 1., 20.
            ),
        max_depth=trial
        .suggest_int(
            "max_depth", 4, 8
            ),
        subsample=trial
        .suggest_float(
            "subsample", 0.5, 1
            ),
        colsample=trial
        .suggest_float(
            "colsample", 0.5, 1
            ),
        verbose=100,

    )
    model.fit(
        X_train,
        y_train,
        eval_sets=
            [{'X': X_val,
            'y': y_val},]
    )
    y_pred = model.predict(
        X_val
    )[:, 0]

    return (np.abs(
        y_val - y_pred
    )).mean()
```

5.2. Results

5.2.1. Losses comparison

Alternative losses exhibit better results than MAE.

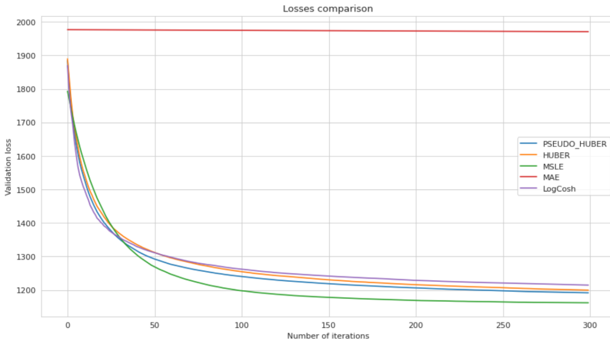

Figure 2. Losses comparison on Allstate dataset

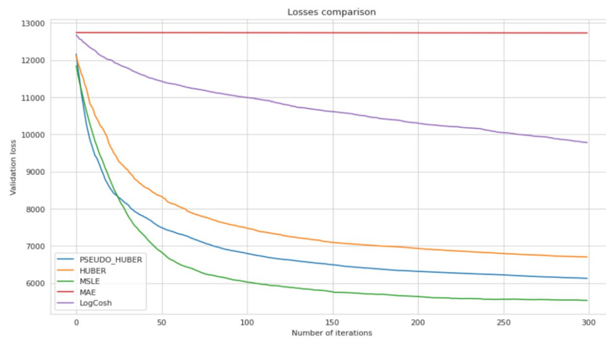

Figure 3. Losses comparison on House Prices dataset

5.2.2. Dependence on delta for Huber and Pseudo Huber losses

The losses for Huber/Pseudo Huber mostly tend to decrease with the larger values of delta.

6. Conclusion

- As expected, MAE is always a bad choice, maybe with the exception of the Life Expectancy dataset (there the scale of the target is suitable for this loss)
- MSLE always performs the best - astonishing results on the House Prices dataset illustrate this unequivocally
- The losses for Huber/Pseudo Huber mostly tend to decrease with the larger values of delta - although the plots for House Prices and Life Expectancy show some nontrivial dependencies

Table 3. Losses comparison on Life Expectancy dataset

Loss	Pseudo Huber	Huber	MSLE	MAE	LogCosh
Test loss	1.17	1.16	1.13	1.20	1.34

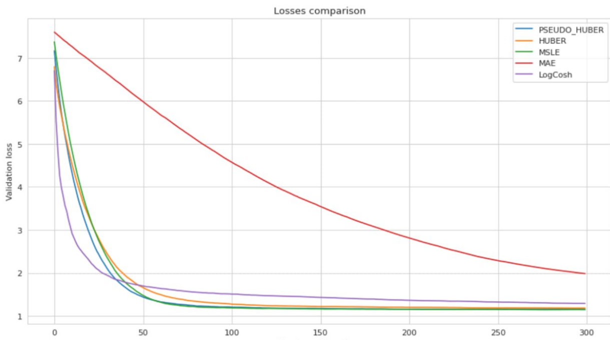

Figure 4. Losses comparison on Life Expectancy dataset

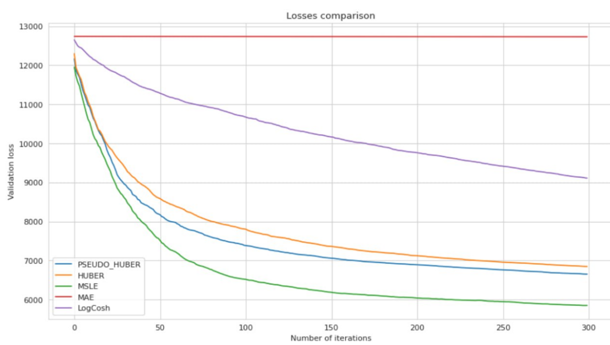

Figure 5. Losses comparison on Car Price Prediction Challenge dataset

References

Chen, T. and Guestrin, C. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2016. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145%2F2939672.2939785.

Friedman, J. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38:367–378, 02 2002. doi: 10.1016/S0167-9473(01)00065-2.

Iosipoi, L. and Vakhrushev, A. Sketchboost: Fast gradient boosted decision tree for multioutput problems. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=WSxarC8t-T.

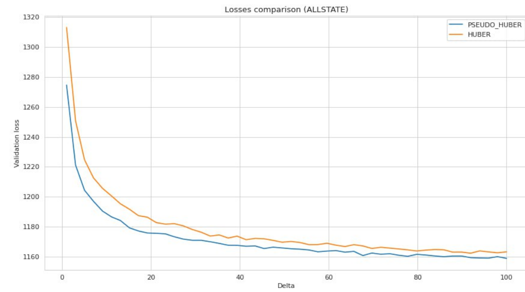

Figure 6. Dependence of losses on delta on the Allstate dataset

Figure 7. Dependence of losses on delta on the Allstate dataset

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 3149–3157, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Prokhorenkova, L. O., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. Catboost: unbiased boosting with categorical features. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *NeurIPS*, pp. 6639–6649, 2018. URL http://dblp.uni-trier.de/db/conf/nips/nips2018.html#ProkhorenkovaGV18.

Table 4. Losses comparison on Car Price Prediction Challenge dataset

Loss	Pseudo Huber	Huber	MSLE	MAE	LogCosh
Test loss	5872	5941	5868	12691	6892

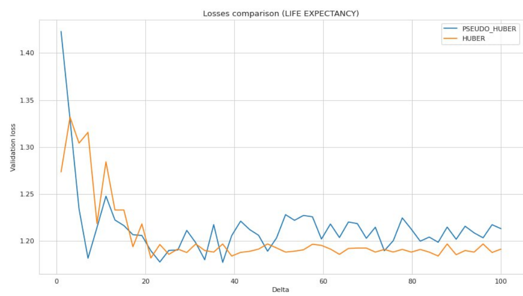

Figure 8. Dependence of losses on delta on the Allstate dataset

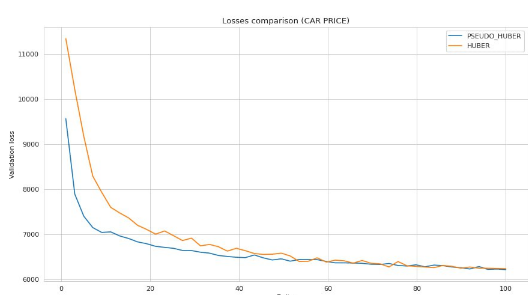

Figure 9. Dependence of losses on delta on the Allstate dataset

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Name 1 (20% of work)

- Reviewing literate on the topic (3 papers)
- Coding the main algorithm
- Experimenting with model parameters on MNIST dataset
- Preparing the GitHub Repo
- Preparing the Section N of this report
- ...

Name 2 (25% of work)

- ...

Name 3 (55% of work)

- ...

B. Reproducibility checklist

Answer the questions of following reproducibility checklist. If necessary, you may leave a comment.

1. A ready code was used in this project, e.g. for replication project the code from the corresponding paper was used.

   - [x] Yes.
   - [ ] No.
   - [ ] Not applicable.

   General comment: If the answer is **yes**, students must explicitly clarify to which extent (e.g. which percentage of your code did you write on your own?) and which code was used.

   Students' comment: None

2. A clear description of the mathematical setting, algorithm, and/or model is included in the report.

   - [ ] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

3. A link to a downloadable source code, with specification of all dependencies, including external libraries is included in the report.

   - [ ] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

4. A complete description of the data collection process, including sample size, is included in the report.

   - [ ] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

5. A link to a downloadable version of the dataset or simulation environment is included in the report.

   - [ ] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

6. An explanation of any data that were excluded, description of any pre-processing step are included in the report.

   - [ ] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

7. An explanation of how samples were allocated for training, validation and testing is included in the report.

   - [ ] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

8. The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results are included in the report.

   - [ ] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

9. The exact number of evaluation runs is included.

   - [ ] Yes.
   - [ ] No.
   - [ ] Not applicable.

   Students' comment: None

10. A description of how experiments have been conducted is included.

    - [ ] Yes.
    - [ ] No.
    - [ ] Not applicable.

    Students' comment: None

11. A clear definition of the specific measure or statistics used to report results is included in the report.

    - [ ] Yes.
    - [ ] No.
    - [ ] Not applicable.

    Students' comment: None

12. Clearly defined error bars are included in the report.

    - [ ] Yes.
    - [ ] No.
    - [ ] Not applicable.

    Students' comment: None

13. A description of the computing infrastructure used is included in the report.
    - ▢ Yes.
    - ▢ No.
    - ▢ Not applicable.

    Students' comment: None