Mastering The New Generation of Gradient Boosting

Tal Peretz Follow
Oct 18, 2018 · 7 min read

Catboost

Gradient Boosted Decision Trees and Random Forest are my favorite ML models for tabular heterogeneous datasets. These models are the top performers on Kaggle competitions and in widespread use in the industry.

Catboost, the new kid on the block, has been around for a little more than a year now, and it is already threatening *XGBoost*, *LightGBM* and *H2O*.

Why Catboost?

Better Results

Catboost achieves the best results on the benchmark, and that's great, yet I don't know if I would replace a working production model for only a fraction of a log-loss improvement alone (especially when the company who conducted the benchmark has a clear interest in the favor of Catboost 😅).
Though, when you look at datasets where **categorical features play a large role**, such as *Amazon* and the *Internet* datasets, this improvement becomes significant and undeniable.

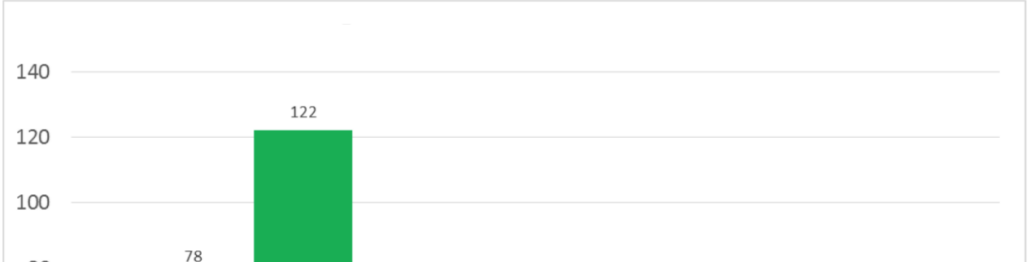
	CatBoost	LightGBM		XGBoost		H2O	
Adult	0.269741	0.276018	+2.33 %	0.275423	+2.11%	0.275104	+1.99%
Amazon	0.137720	0.163600	+18.79 %	0.163271	+18.55%	0.162641	+18.09%
Appet	0.071511	0.071795	+0.40 %	0.071760	+0.35%	0.072457	+1.32%
Click	0.390902	0.396328	+1.39 %	0.396242	+1.37%	0.397595	+1.71%
Internet	0.208748	0.223154	+6.90 %	0.225323	+7.94%	0.222091	+6.39%
Kdd98	0.194668	0.195759	+0.56 %	0.195677	+0.52%	0.195395	+0.37%
Kddchurn	0.231289	0.232049	+0.33 %	0.233123	+0.79%	0.232752	+0.63%
Kick	0.284793	0.295660	+3.82 %	0.294647	+3.46%	0.294814	+3.52%

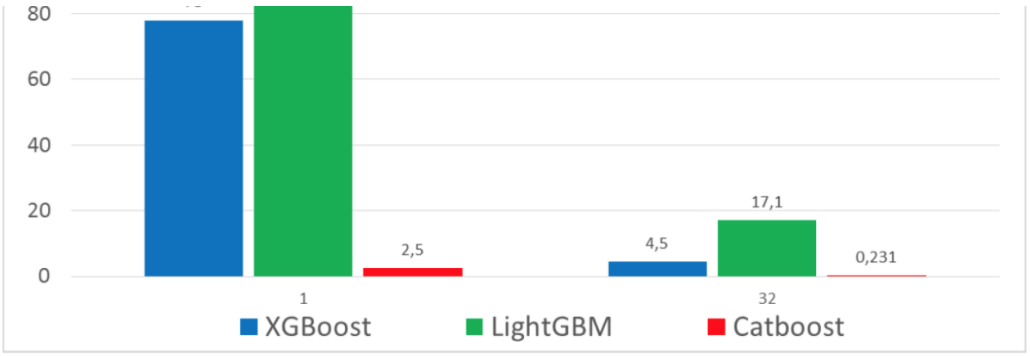
Logloss

GBDT Algorithms Benchmark

Faster Predictions

While training time can take up longer than other GBDT implementations, prediction time is 13–16 times faster than the other libraries according to the Yandex benchmark.





Left: CPU, Right: GPU

Batteries Included

Catboost's default parameters are a better starting point than in other GBDT algorithms. And this is good news for beginners who want a plug and play model to start experience tree ensembles or Kaggle competitions.

Yet, there are some very important parameters which we must address and we'll talk about those in a moment.

	CatBoost	LightGBM	XGBoost	H2O
	Default	Default	Default	Default
Adult	**0.27298**	0.28716 +5.20%	0.28009 +2.61%	0.27607 +1.14%
Amazon	**0.13811**	0.16716 +21.04%	0.16536 +19.74%	0.16950 +22.73%
Click prediction	**0.39112**	0.39749 +1.63%	0.39764 +1.67%	0.39785 +1.73%
KDD appetency	**0.07138**	0.07482 +4.82%	0.07466 +4.60%	0.07355 +3.05%
KDD churn	**0.23193**	0.23565 +1.61%	0.23369 +0.76%	0.23287 +0.41%
KDD internet	**0.22021**	0.23627 +7.30%	0.23468 +6.58%	0.24023 +9.10%
KDD upselling	**0.16674**	0.17107 +2.60%	0.16873 +1.20%	0.16981 +1.85%

KDD 98	**0.19479**	0.19837 +1.84%	0.19795 +1.63%	0.19607 +0.66%
Kick prediction	**0.28491**	0.29877 +4.87%	0.29816 +4.66%	0.29635 +4.02%

GBDT Algorithms with default parameters Benchmark

Some more noteworthy advancements by Catboost are the features interactions, object importance and the snapshot support.

In addition to classification and regression, Catboost supports **ranking** out of the box.

Battle Tested

Yandex is relying heavily on Catboost for ranking, forecasting and recommendations. This model is serving more than 70 million users each month.

> *CatBoost is an algorithm for **gradient boosting on decision trees**. Developed by Yandex researchers and engineers, it is the successor of the **MatrixNet algorithm** that is widely used within the company for ranking tasks, forecasting and making recommendations. It is universal and can be applied across a wide range of areas and to a variety of problems.*

Yandex

The Algorithm

Classic Gradient Boosting

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
    1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

    2. Fit a base learner (e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
    3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

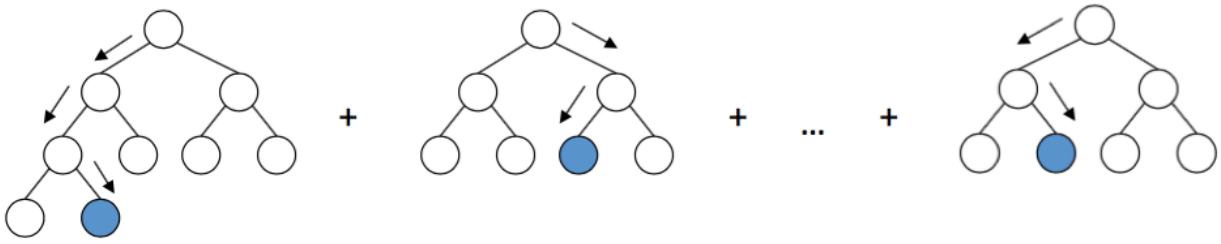
$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

    4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

Gradient Boosting on Wikipedia



Catboost Secret Sauce

Catboost introduces two critical algorithmic advances - the implementation of **ordered boosting**, a permutation-driven alternative to the classic algorithm, and an innovative algorithm for **processing categorical features**.

Both techniques are using random permutations of the training examples to fight the *prediction shift* caused by a special kind of *target leakage* present in all existing implementations of gradient boosting algorithms.

Categorical Feature Handling

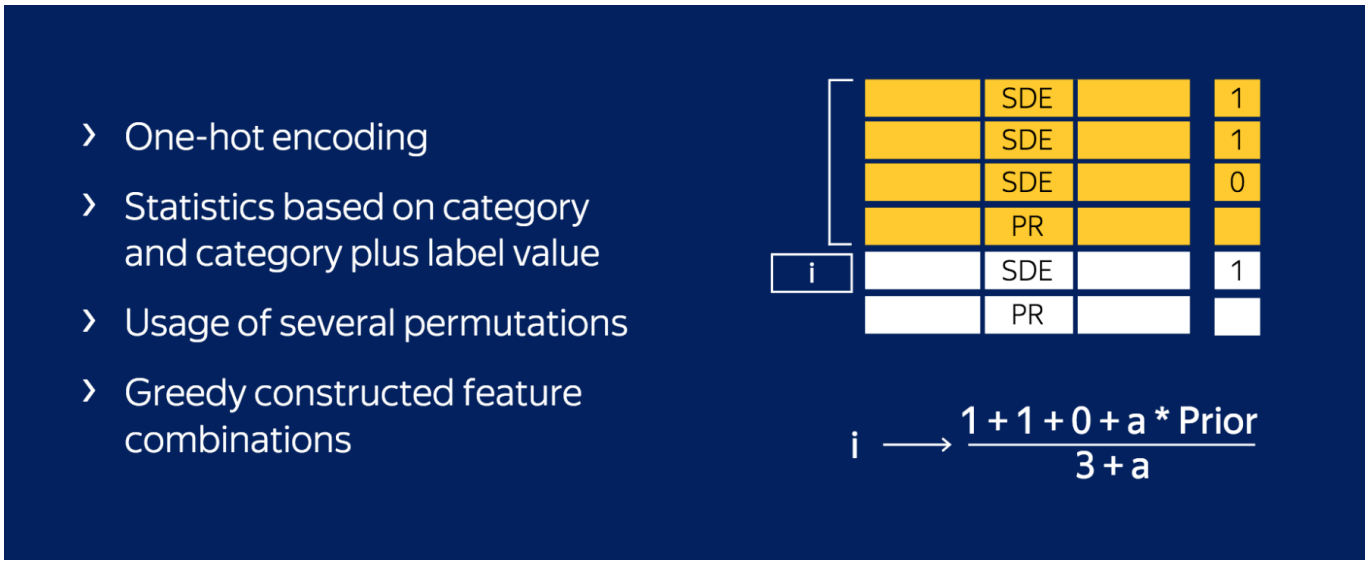
Ordered Target Statistic

Most of the GBDT algorithms and Kaggle competitors are already familiar with the use of Target Statistic (or target mean encoding).
It's a simple yet effective approach in which we encode each categorical feature with the estimate of the expected target y conditioned by the category.
Well, it turns out that applying this encoding carelessly (average value of y over the training examples with the same category) results in a target leakage.

To fight this *prediction shift* CatBoost uses a more effective strategy. It relies on the ordering principle and is inspired by online learning algorithms which get training examples sequentially in time. In this setting, the values of TS for each example rely only on the observed history.
To adapt this idea to a standard offline setting, Catboost introduces an artificial "time"—a random permutation $\sigma 1$ of the training examples.
Then, for each example, it uses all the available "history" to compute its Target Statistic.
Note that, using only one random permutation, results in preceding examples with higher variance in Target Statistic than subsequent ones. To this end, CatBoost uses different permutations for different steps of gradient boosting.

One Hot Encoding

Catboost uses a one-hot encoding for all the features with at most *one_hot_max_size* unique values. The default value is 2.



Catboost's Secret Sauce

Orederd Boosting

CatBoost has two modes for choosing the tree structure, Ordered and Plain. **Plain mode** corresponds to a combination of the standard GBDT algorithm with an ordered Target Statistic.

In **Ordered mode** boosting we perform a random permutation of the training examples - *σ2,* and maintain n different supporting models - *M1, . . . , Mn* such that the model *Mi* is trained using only the first *i* samples in the permutation.

At each step, in order to obtain the residual for *j*-th sample, we use the model *Mj−1*. Unfortunately, this algorithm is not feasible in most practical tasks due to the need of maintaining n different models, which increase the complexity and memory requirements by n times. Catboost implements a modification of this algorithm, on the basis of the gradient boosting algorithm, using one tree structure shared by all the models to be built.

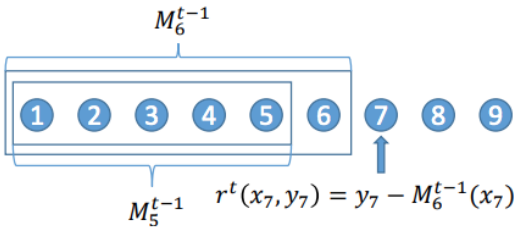


Figure 1: Ordered boosting principle.

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**
    $r_i \leftarrow y_i - M_{\sigma(i)-1}(i)$;
  for $i \leftarrow 1$ **to** n **do**
    $\Delta M \leftarrow$ $LearnModel((\mathbf{x}_j, r_j) : \sigma(j) \leq i)$;
    $M_i \leftarrow M_i + \Delta M$;
return M_n

Algorithm 2: Building a tree in CatBoost

input : $M, \{y_i\}_{i=1}^n$, α, L, $\{\sigma_i\}_{i=1}^s$, $Mode$

$grad \leftarrow CaclGradient(L, M, y)$;
$r \leftarrow random(1, s)$;
$G \leftarrow (grad_r(1), \ldots, grad_r(n))$ for $Plain$;
$G \leftarrow (grad_{r,\sigma_r(1)-1}(i)$ for $i = 1$ to $n)$ for $Ordered$;
$T \leftarrow$ 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**
      $\Delta(i) \leftarrow \text{avg}(grad_r(p)$ for $p : leaf(p) = leaf(i))$ for all i;
    if $Mode == Ordered$ **then**
      $\Delta(i) \leftarrow \text{avg}(grad_{r,\sigma_r(i)-1}(p)$ for $p : leaf(p) = leaf(i), \sigma_r(p) < \sigma_r(i))$ $\forall i$;
    $loss(T_c) \leftarrow ||\Delta - G||_2$
  $T \leftarrow \text{argmin}_{T_c}(loss(T_c))$
if $Mode == Plain$ **then**
  $M_{r'}(i) \leftarrow M_{r'}(i) - \alpha\,\text{avg}(grad_{r'}(p)$ for $p : leaf(p) = leaf(i))$ for all r', i;
if $Mode == Ordered$ **then**
  $M_{r',j}(i) \leftarrow M_{r',j}(i) - \alpha\,\text{avg}(grad_{r',j}(p)$ for $p : leaf(p) = leaf(i), \sigma_{r'}(p) \leq j$ for all r', j, i;
return T, M

In order to avoid *prediction shift*, Catboost uses permutations such that $\sigma 1 = \sigma 2$. This guarantees that the target-*yi* is not used for training *Mi* neither for the Target Statistic calculation nor for the gradient estimation.

Hands On

For this section, we'll use the *Amazon Dataset*, since it's clean and has a strong emphasize on categorical features.

Column Name	Description	dtype	None Null Rows Train	None Null Rows Test	Unique Values Train
ACTION	Target - 1 if the resource was approved, 0 if the resource was not	int64	32769	-	2
RESOURCE	An ID for each resource	int64	32769	58921	7518
MGR_ID	The EMPLOYEE ID of the manager of the current EMPLOYEE ID record; an employee may have only one manager at a time	int64	32769	58921	4243
ROLE_ROLLUP_1	Company role grouping category id 1 (e.g. US Engineering)	int64	32769	58921	128
ROLE_ROLLUP_2	Company role grouping category id 2 (e.g. US Retail)	int64	32769	58921	177
ROLE_DEPTNAME	Company role department description (e.g. Retail)	int64	32769	58921	449
ROLE_TITLE	Company role business title description (e.g. Senior Engineering Retail Manager)	int64	32769	58921	343
ROLE_FAMILY_DESC	Company role family extended description (e.g. Retail Manager, Software Engineering)	int64	32769	58921	2358
ROLE_FAMILY	Company role family description (e.g. Retail Manager)	int64	32769	58921	67
ROLE_CODE	Company role code; this code is unique to each role (e.g. Manager)	int64	32769	58921	343

Dataset in a brief

Tuning Catboost

Important Parameters

`cat_features` — This parameter is a must in order to leverage Catboost preprocessing of categorical features, if you encode the categorical features yourself and don't pass the columns indices as *cat_features* you are missing the essence of Catboost.

`one_hot_max_size` — As mentioned before, Catboost uses a one-hot encoding for all features with at most *one_hot_max_size* unique values. In our case, the categorical

features have a lot of unique values, so we won't use one hot encoding, but depending on the dataset it may be a good idea to adjust this parameter.

`learning_rate` & `n_estimators` — The smaller the learning_rate, the more n_estimators needed to utilize the model. Usually, the approach is to start with a relative high *learning_rate*, tune other parameters and then decrease the *learning_rate* while increasing *n_estimators*.

`max_depth` — Depth of the base trees, this parameter has an high impact on training time.

`subsample` — Sample rate of rows, can't be used in a *Bayesian* boosting type setting.

`colsample_bylevel, colsample_bytree, colsample_bynode` — Sample rate of columns.

`l2_leaf_reg` — L2 regularization coefficient

`random_strength` — Every split gets a score and random_strength is adding some randomness to the score, it helps to reduce overfitting.

```
from skopt.space import Real, Integer
catboost_params_space = [Real(1e-7, 1, prior='log-uniform', name='learning_rate'),
                Integer(2, 10, name='max_depth'),
                Real(0.5, 1.0, name='subsample'),
                Real(0.5, 1.0, name='colsample_bylevel'),
                Integer(1, 10, name='gradient_iterations'),
                Real(1.0, 16.0, name='scale_pos_weight'),
                Real(0.0, 1.0, name='bagging_temperature'),
                Integer(1, 20, name='random_strength'),
                Integer(2, 25, name='one_hot_max_size'),
                Real(1.0, 100, name='reg_lambda')]
```

Check out the recommended spaces for tuning here

Model Exploration with Catboost

In addition to feature importance, which is quite popular for GBDT models to share, Catboost provides **feature interactions** and **object (row) importance**

Catboost’s Feature Importance

Catboost’s Feature Interactions

Catboost's Object Importance

SHAP values can be used for other ensembles as well

The Full Notebook

Check it out for some useful Catboost code snippets

Catboost Playground

This notebook goal is to show off Catboost functions and attributes.

Amazon Employee Access Challenge

The objective of this competition is to build a model, learned using historical data, that will determine an employee's access needs, such that manual access transactions (grants and revokes) are minimized as the employee's attributes change over time. The model will take an employee's role information and a resource code and will return whether or not access should be granted.

Environment Setup

Imports

```
In [2]: import time
        import pandas as pd
        from matplotlib import pyplot as plt
```

catboost_playground.ipynb hosted with ♡ by GitHub view raw

Catboost Playground Notebook

Bottom Line

Catboost vs. XGBoost (default, greedy and exhaustive parameter search)

Take Away

- Catboost is built with a similar approach and attributes as with the “older” generation of GBDT models.
- Catboost’s power lies in its **categorical features preprocessing**, **prediction time** and **model analysis**.
- Catboost’s weaknesses are its **training and optimization times**.
- Don’t forget to pass ***cat_features*** argument to the classifier object. You aren’t really utilizing the power of Catboost without it.
- Though Catboost performs well with default parameters, there are several parameters that drive a significant improvement in results when tuned.

Further Reading

- Catboost Documentation
- Catboost Github
- Catboost official website
- I highly recommend you to dig into the CatBoost: unbiased boosting with categorical features paper on arXiv.
- Catboost playground notebook
- SHAP Values

Huge thanks to Catboost Team Lead Anna Veronika Dorogush.

If you enjoyed this post, feel free to hold down the clap button 👏 and if you're interested in posts to come, make sure to follow me on

Medium: https://medium.com/@talperetz24
Twitter: https://twitter.com/talperetz24
LinkedIn: https://www.linkedin.com/in/tal-per/

Like every year, I want to mention DataHack- the best data driven hackathon out there. This year דור פרץ and I used Catboost for our project and won the 1st place 🏆.

Machine Learning Catboost Xgboost Data Science Towards Data Science