LightGBM

Co to?

LightGBM - Light Gradient Boosting Machine

Framework do uczenia maszynowego z wykorzystaniem modeli drzewiastych i gradient boostingu. Cechuje się:

- Lekkością: Szybkością i mniejszym wykorzystaniem pamięci
- Podobnymi wynikami do innych frameworków
- Lepszym przystosowaniem do działania na klastrach obliczeniowych

Kto zaproponował

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. **Lightgbm: A highly efficient gradient boosting decision tree.**
In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, page 3149–3157, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Microsoft

LightGBM

Tak jak sugeruje logo, prace nad programem były w dużej części ufundowane przez Microsoft.

Składowe

Bazuje się na drzewach decyzyjnych, połączonych w komitet, który jest następnie boostowany za pomocą gradientu. Innowacje LightGBM to:

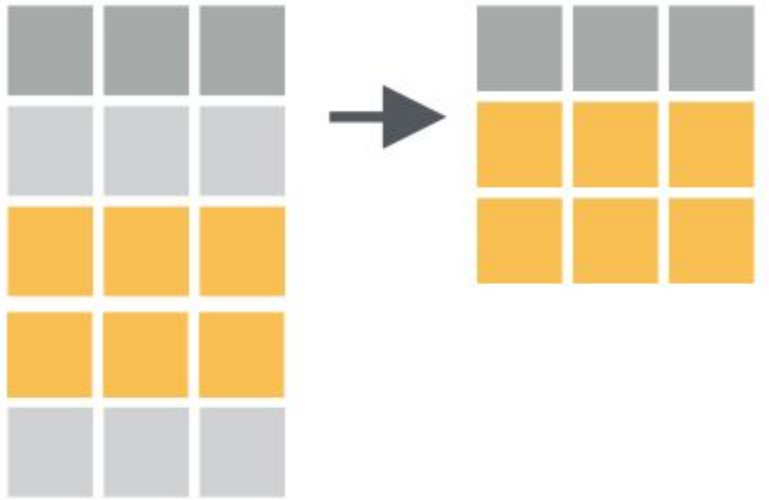
- **GOSS** - *Gradient One-Side Sampling*
- **EFB** - *Exclusive Feature Bundling*

Ale to nie są jedyne różnice:

- Dzielenie przy pomocy histogramów
- Dzielenie po liściach
- Lepsze możliwości uczenia rozproszonego

GOSS - Gradient One-Side Sampling

- Metoda na redukcję liczby potrzebnych do przetworzenia obserwacji.
- Intuicja: *Nie ma sensu się uczyć czegoś, co już umiemy - skupmy się na tym czego nie potrafimy.*
- Pomysł zapożyczony z AdaBoost, ale w nim są wagi, po których można wykonać selekcję.
- Więc wymyślamy wagi…



GOSS

- Heurystyka: Wielkość gradientu dla obserwacji jest indykatorem, że model jeszcze się jej nie nauczył.
- Więc możemy pominąć obserwacje, w których gradient jest mały, oszczędzając czas.

Uwaga - taka operacja zmienia dystrybucję zbioru. Dlatego:

- Z pominiętych obserwacji losujemy frakcję i obliczamy gradienty
- Mnożymy gradienty tak, żeby przybliżyć oryginalny rozkład.

Zostało pokazane, że takie podejście jest lepsze od losowego wyboru podzbioru.

EFB - Exclusive Feature Bundling

- Redukujemy liczbę cech
- Korzystamy z faktu, że wysoko wymiarowe dane często są rzadkie i praktycznie nigdy są niezerowe naraz.
  - *One Hot Encoding*
- EFB wykorzystuje te własności, wybierając odpowiednie grupy cech (*bundles*) i sprowadza je do jednej gęstej (niezerowej) cechy.

EFB - ale chwila…

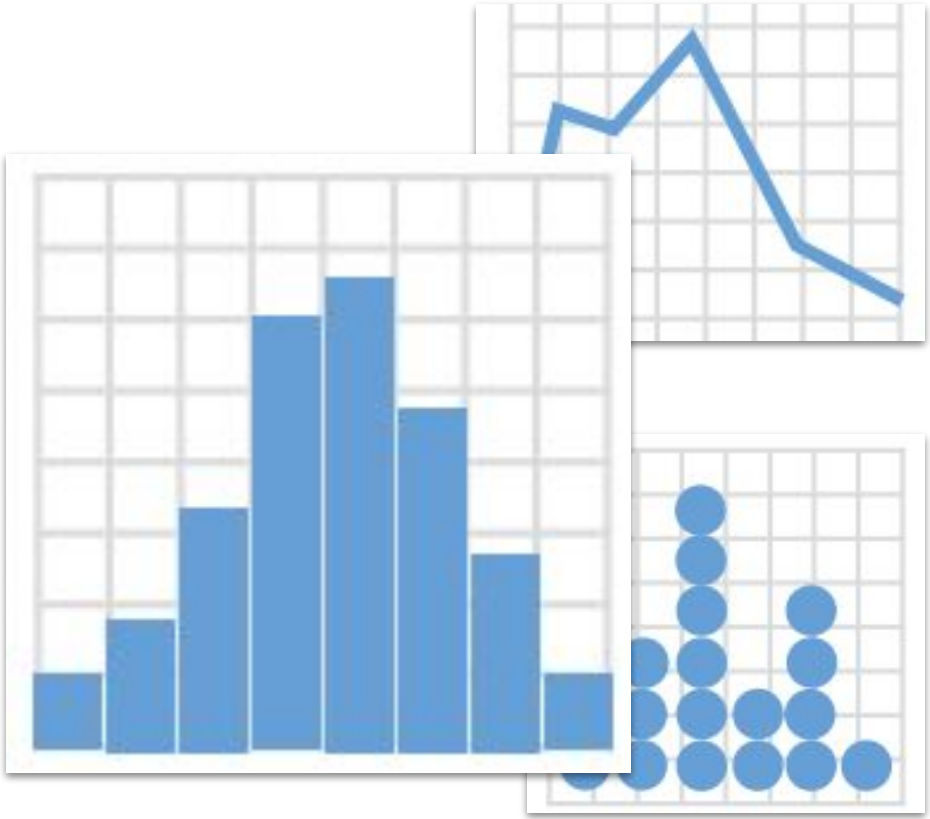
Czemu nie możemy tak zrobić na etapie preprocessingu?

- EFB buduje na podstawie swoich bundli histogramy, na podstawie których później przeprowadza dzielenia zbioru potrzebne do uczenia modeli drzewiastych.
- LightGBM ogólnie korzysta jedynie z podejścia histogramowego
  - W xgboost jest to jedną z opcji budowania drzew: `exact` / `approx` / **`hist`**
- Bez wiedzy o tym jak histogramy są zbudowane nie jesteśmy w stanie przygotować danych - więc musimy przetworzyć dane w obie strony: *One Hot Encoding -> Exclusive Feature Bundling*

Histogramy

Podziały gradientów możemy lepiej obliczać przy pomocy histogramów. Zamiast trzymać w pamięci ile wyszedł każdy gradient, to trzymamy ile obserwacji było na pewnym przedziale.

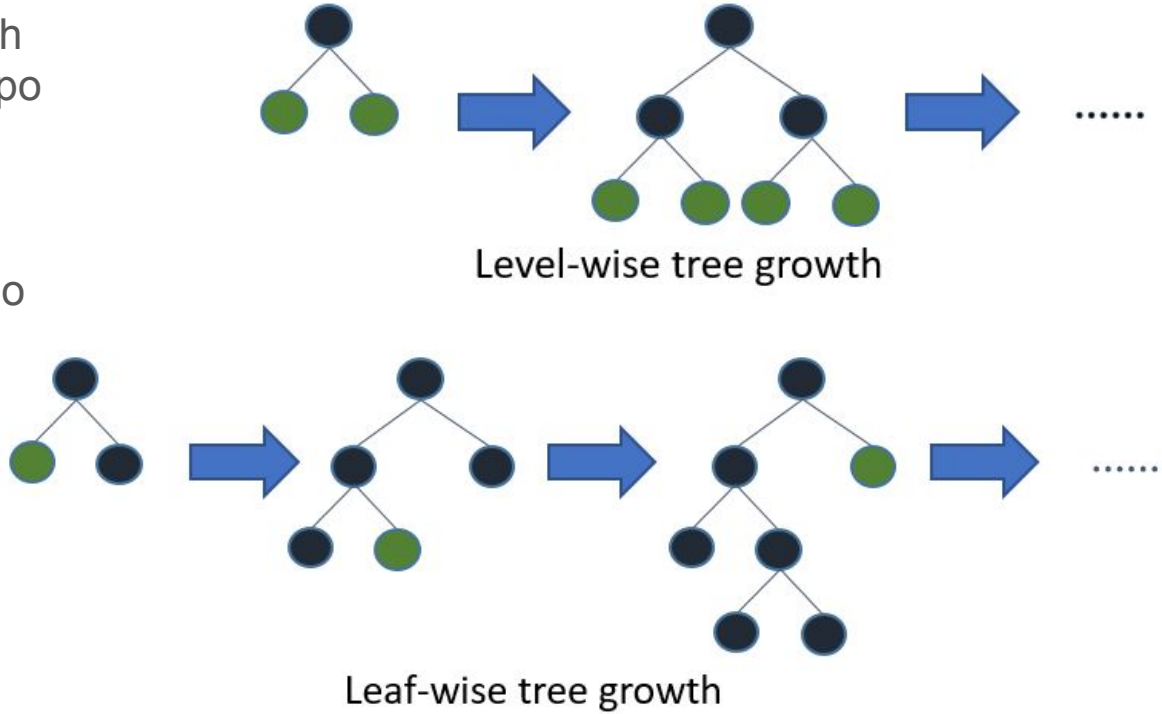
Nie jest innowacją LightGBM, ale wiele optymalizacji w LightGBM bazuje na histogramach.



Dzielenie po liściach

LightGBM dzieli zbiór w liściach drzewa decyzyjnego, zamiast po poziomach drzewa.

To generalnie zwiększa jakość modelu, ale może prowadzić do przeuczania.



LightGBM documentation

Feature parallel, data parallel

LightGBM nie rozdziela cech pomiędzy maszynami w klastrze.

Każda maszyna wie jak wyliczyć EFB jak ma pełen zestaw, więc nie ma sensu dzielić i uzgadniać po sieci co wyszło.

Używając metody histogramów dla też znacznie zmniejsza potrzebę na komunikację - wysyłamy po sieci tylko zagregowane dane.

Najważniejsze hiperparametry LightGBM

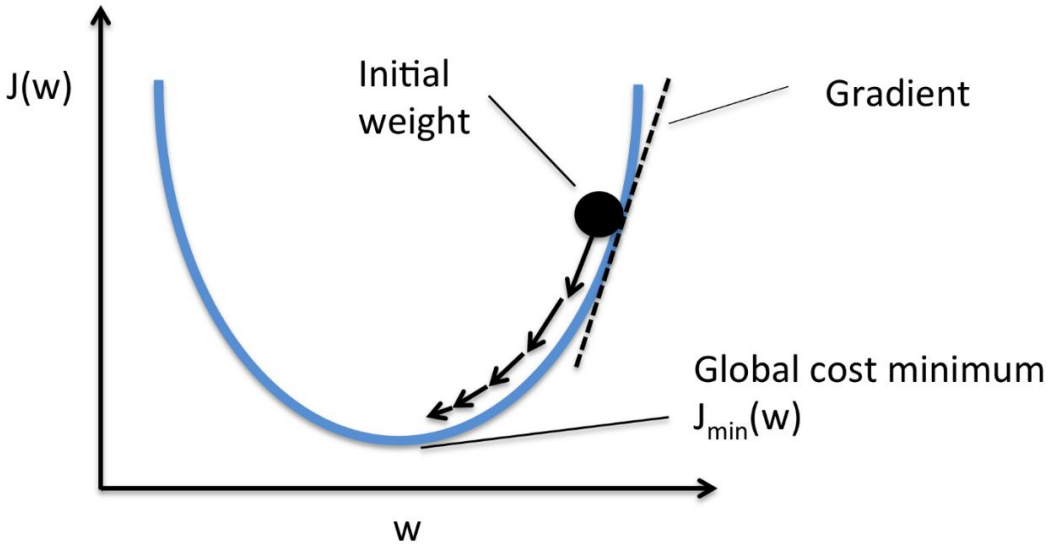
Można wyróżnić kilka grup:

- regulujące proces trenowania
- parametry techniczne
- wskazujące na rodzaj modelu

Spis wszystkich parametrów na stronie dokumentacji:
lightgbm.readthedocs.io/en/latest/Parameters.html

Boosting

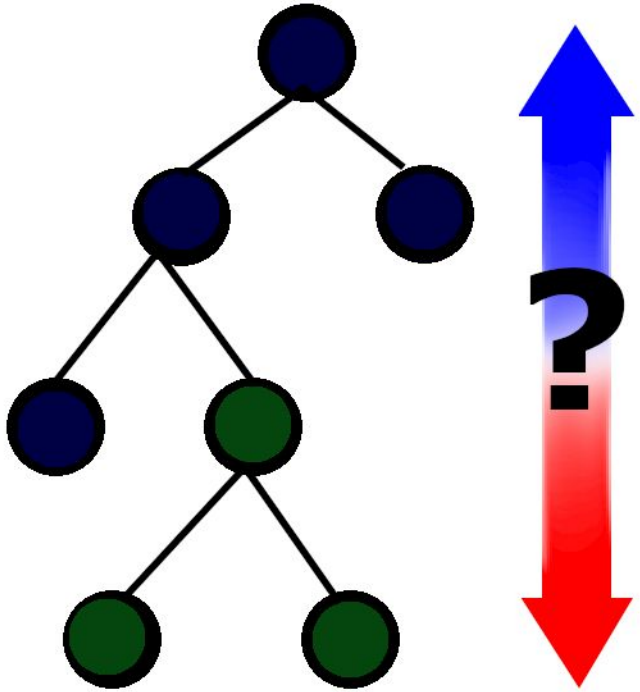
- gbdt
- dart
- goss
- rf



towardsdatascience.com/what-makes-lightgbm-lightning-fast-a27cf0d9785e

Parametry regulujące proces uczenia

- num_leaves
- max_depth
- min_data_in_leaf
- bagging_fraction
- bagging_freq
- feature_fraction
- max_bin
- lambda_l1 i lambda_l2



Parametry techniczne

- num_iterations
- learning_rate
- min_gain_to_split
- early_stopping_round
- categorical_features
- is_unbalance
- scale_pos_weight
- feval

Ind	Feature1	Feature2	Feature3	Target
4563	123.844	3	D	0
4564	156.342	2	D	0
4565	136.938	13	E	1
4566	182.928	5	C	0
4567	142.292	5	B	0
4568	173.291	3	D	0
4569	163.823	7	C	0
4570	182.829	10	C	1
4571	121.203	6	A	0
4572	173.929	5	B	0
4573	183.291	9	E	1
4574	118.931	2	D	0
4575	173.292	11	C	0
4576	134.831	11	E	1
4577	109.820	5	A	0

Regresja a klasyfikacja

Name of the parameter	Note for Classification	Note for regression
objective	Set it binary or multiclass	Set it regression
metric	Binary_logloss or AUC or etc.	RMSE or mean_absolute_error and or etc.
is_unbalance	True or false	–
scale_pos_weight	used only in binary and multi class applications	–
num_class	used only in multi-class classification application	–
reg_sqrt	–	Used to fit sqrt(label) instead of original values for large range label

neptune.ai/blog/lightgbm-parameters-guide

Pakiety w R

W celu budowania tego modelu musimy używać pakietu lightgbm.

Jest on łatwy w instalacji poprzez archiwum CRAN

Poza pakietem do R, lightGBM ma swoje api także dla Pythona oraz C.

Przykładowe wywołanie w R

```
library(lightgbm)
data(agaricus.train, package='lightgbm')
train <- agaricus.train
dtrain <- lgb.Dataset(train$data,
                      label = train$label)
model <- lgb.cv(
    params = list(
        objective = "regression",
        metric = "l2"
    ), data = dtrain)
```

Formaty danych

LightGBM obłsługuje dane w jednym z formatów:

- matrix
- dgCMatrix
- ścieżka do pliku CSV, TSV lub LibSVM
- ścieżka do specjalnego pliku binarnego lgb.Dataset w formacie tworzonym przez ten pakiet

Zmienne kategoryczne

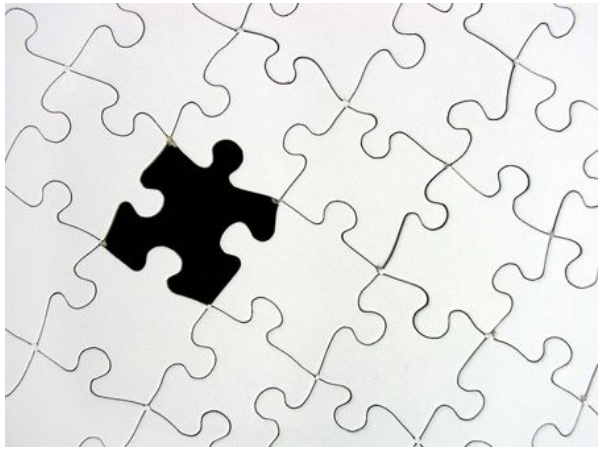
LightGBM obsługuje zmienne kategoryczne

Wedle przeprowadzonego eksperymentu prędkość na zmiennych kategorycznych była 8 razy lepsza, niż na danych zakodowanych za pomocą one-hot encoding

Brakujące dane

LightGBM domyślnie potrafi osłużyć brakujące dane. Można takie zachowanie jednak wyłączyć.

Domyślnie za brak danych uznawana jest wartość NaN, istnieje jednak możliwość za pomocą parametru zmiana, tak aby wartość 0 była uznawana także jako brakująca.



Źródła do nauki

- lightgbm.readthedocs.io/en/latest/ - strona dokumentacji całego projektu
- https://neptune.ai/blog/lightgbm-parameters - blog objaśniajacy najważniejsze parametry
- towardsdatascience.com/how-to-beat-the-heck-out-of-xgboost-with-lightgbm-comprehensive - szybkie objaśnienie oraz wprowadzenie
- https://proceedings.neurips.cc/paper/ - artykuł objaśniacy działanie lightGBM
- https://github.com/microsoft/LightGBM - projekt na GitHubie