Gradient Boosting: modern frameworks and parameter tuning

Emil Kayumov
emil.kayumov@gmail.com
telegram: @emilkayumov

#readml

11.12.2017

Plan

Recap

XGBoost

LightGBM

CatBoost

Parameter tuning

What is it

- An ensemble of weak algorithms (decision trees in most cases)
- Can be used with every differentiable loss function
- Good for heterogeneous data (maybe the best)
- Popular in real tasks (not only competitions), for example a search engine (since Altavista in 2002!)

Recap

«Greedy function approximation: A gradient boosting machine» (Jerome Friedman, 1999)

Weighted sum of basic algorithms:

$$a_N(x) = \sum_{n=0}^{N} \gamma_n b_n(x)$$

Let $L(y, z)$ be differentiable loss function, $b_0(x) = \mathrm{const}$ (0 or some mean value). Iteration N:

$$\sum_{i=1}^{l} L(y_i, a_{N-1}(x_i) + \gamma_N b_N(x_i)) \to \min_{b_N, \gamma_N}$$

Recap

«Greedy function approximation: A gradient boosting machine»
(Jerome Friedman, 1999)

Iteration N:

1. $s_i = -\left.\frac{\partial L}{\partial z}\right|_{z=a_{N-1}(x_i)}$
2. $b_N = \arg\min\limits_{b} \sum_{i=1}^{l} (b(x_i) - s_i)^2$
3. $\gamma_N = \arg\min\limits_{\gamma} \sum_{i=1}^{l} L(y_i, a_{N-1}(x_i) + \gamma b_N(x_i))$

Any problem?

Recap

«Greedy function approximation: A gradient boosting machine»
(Jerome Friedman, 1999)

Iteration N:

1. $s_i = -\left.\frac{\partial L}{\partial z}\right|_{z=a_{N-1}(x_i)}$
2. $b_N = \arg\min\limits_{b} \sum_{i=1}^{l} (b(x_i) - s_i)^2$
3. $\gamma_N = \arg\min\limits_{\gamma} \sum_{i=1}^{l} L(y_i, a_{N-1}(x_i) + \gamma b_N(x_i))$

Regularization: $a_N(x) = a_{N-1}(x) + \eta \gamma_N b_N(x), 0 < \eta < 1$

For trees: $\gamma_N b_N(x) = r_j [x \in R_j]$ (R_j is leaf)

Parameters

- loss
- learning rate (η)
- iteration count
- tree parameters
  - depth
  - subsample
  - max features
  - min samples leaf
  - min split gain
  - ...

Vanilla gradient boosting is implemented in scikit-learn (without any parallelization)

Feature importances

Trees let us calculate importances:

- number of split by features — bad idea
- gain by features — better
- by values in leaves

- No guarantee about low importance features (may be useful)
- Helps to find leaks (overfitted features)

XGBoost

Popular in competitions since 2015 (paper in 2016)

$$\sum_{i=1}^{l} L(y_i, a_{N-1}(x_i) + b_N(x_i)) \approx$$
$$\approx \sum_{i=1}^{l} L(y_i, a_{N-1}(x_i)) + g_i b_N(x_i) + \frac{1}{2} h_i b_N(x_i)^2 + \Omega(b_N)$$

Regularization: $\Omega(b_N) = \gamma T + \lambda \|w\|^2$, T – leaf count, w – leaf values.

Basic rule $\sum_{i=1}^{l}(b(x_i) - s_i)^2 \to \min_b$ like XGBoost without regularization and with $h_i = 1$

XGBoost

- Easy calculating optimal w for every tree structure
- Loss with optimal w can be used as tree criterion
- Builds tree with own criterion (already with regularization)
- Learns best direction for missing values

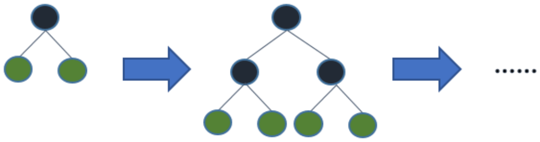
XGBoost

- Also linear model (simplier and less overfit)
- Regression, classification, ranking + custom loss
- Gredy and approximate finding splits (now also by histogram after LightGBM)
- Effective with sparse data
- Now also leaf-wise (best-first) tree growth (after LightGBM)

LightGBM

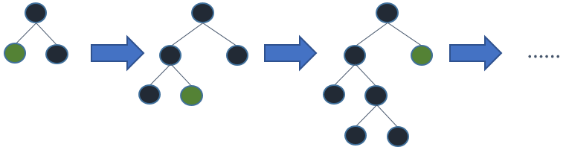
Published in 2016 by Microsoft (paper in 2016 too)

Before:



Level-wise tree growth

LightGBM:



Leaf-wise tree growth

LightGBM

Published in 2016 by Microsoft (paper in 2016 too)

- Finds splits by histogram
- Leaf-wise (best-first) tree growth by default
- Effective with sparse data (exclusive feature bundling)
- Supports categorical features (one-hot encoding and relevance splitting)
- Faster than xgboost (approx 3x)
- Gradient-based one-side sampling – faster mode for big dataset (not default)
- Random forest mode
- A lot of loss functions + custom loss

Catboost

Published in 2017 by Yandex

- Smoothed target encoding for categorical features (also one-hot and feature combinations)

$$\frac{\sum_{j=1}^{p-1}[x_{\sigma_j,k} = x_{\sigma_p,k}]y_{\sigma_j} + a \times prior}{\sum_{j=1}^{p-1}[x_{\sigma_j,k} = x_{\sigma_p,k}] + a}$$

- Oblivious decision trees
- Less overfit in leaf values while calculating gradients (arxiv: «Fighting biases with dynamic boosting»)
- P-value overfit detector (not only by simple decreasing score)
- So slow but will be faster (the fastest GPU mode)
- A lot of loss functions + custom loss

GPU support

- All frameworks support GPU
- Faster than CPU
- Uses for histogram computation

Learning rate vs iterations

$$\textit{learning rate} \times \textit{iterations} \approx \mathrm{const}$$

- You can tune with less number of iterations and increase for final model
- Do not forget that learning rate depends on scale of gradients! For example MAE

Early stopping

- All gradient boosting overfits
- Print loss/metric after every iterations to see overfitting or use early stop
- Should train final model with early stopping too (you can average results for different validation parts) — it is my opinion
- So you can choose learning rate and set big number of iterations (it depends on complexity too)
- Early stop can stop in the beginning because of different mean target in train and test

Parameters

- Complexity: max depth, num leaves, max bins in histogram
- Regularization: L1, L2, min samples leaf, min split gain, ...
- Randomness: feature fraction, object fraction, ...

Tuning

- Fix learning rate, number of iteration, some randomness (or not)
- Deal with underfit vs overfit: tune complexity vs regularization
- Tune randomness
- Decrease learning rate and increase number of iterations for final model

Of course, not only this method

Tuning: easy way

- Grid Search
- HyperOpt
- BayesianOpt
- ...

Hacks and other stuff

- Averaging by seeds: if you have any random simple averaging helps (stabilization)
- Averaging by validation sets if you use early stopping in final model (only data without time order) – my opinion
- xgbfir tool or CatBoost to find interactions and add it manually
- Do not forget about negative sale predictions and other suprise

What to use?

- LightGBM – fast
- CatBoost – needs no tune, automatically works with categorical features
- XGBoost – . . . (maybe for big ensemble)

- Try GPU version too

Links

- Greedy function approximation: A gradient boosting machine
- XGBoost: A Scalable Tree Boosting System
- LightGBM: A Highly Efficient Gradient Boosting Decision Tree
- Fighting biases with dynamic boosting
- А. Дьяконов «Градиентный бустинг»
- Лекция про градиентный бустинг из курса Машинного обучения ФКН ВШЭ
- Алексей Натёкин про градиентный бустинг и фишки (видео)