SCHOOL OF DATA ANALYSIS

Machine Learnig – Part 3

Nikita Kazeev[1,2,3]

[1] National Research University Higher School of Economics (HSE) [2] Yandex [3] Yandex School of Data Analysis

Let’s get acquainted

- Who is (or in near future will) be doing a physics analysis?

Let’s get acquainted

› Who is (or in near future will) be doing a physics analysis?

› Who has fitted a classifier?

Let’s get acquainted

- Who is (or in near future will) be doing a physics analysis?
- Who has fitted a classifier?
- Who has computed cross-validation error of a classifier?

Let’s get acquainted

- Who is (or in near future will) be doing a physics analysis?
- Who has fitted a classifier?
- Who has computed cross-validation error of a classifier?
- Who has trained a deep NN?

Let's get acquainted

- Who is (or in near future will) be doing a physics analysis?
- Who has fitted a classifier?
- Who has computed cross-validation error of a classifier?
- Who has trained a deep NN?
- Who has trained a convolutional NN?

Let’s get acquainted

- Who is (or in near future will) be doing a physics analysis?
- Who has fitted a classifier?
- Who has computed cross-validation error of a classifier?
- Who has trained a deep NN?
- Who has trained a convolutional NN?
- Who has trained a LSTM RNN?

General remarks

- 8 hours is not nearly enough

General remarks

- 8 hours is not nearly enough
- Just enough theory to understand what’s going on

General remarks

- 8 hours is not nearly enough
- Just enough theory to understand what's going on
- Just enough practice to use out-of-shelf state-of-the-art solutions (I really love those of-in-between words)

General remarks

- 8 hours is not nearly enough
- Just enough theory to understand what's going on
- Just enough practice to use out-of-shelf state-of-the-art solutions (I really love those of-in-between words)
- Encourage to attend our summer school

General remarks

- 8 hours is not nearly enough
- Just enough theory to understand what's going on
- Just enough practice to use out-of-shelf state-of-the-art solutions (I really love those of-in-between words)
- Encourage to attend our summer school
- Compete at Kaggle

Environment setup

This lecture plan

- Naive boosting for regression
- Gradient boosting machine
- XGBoost
- Dealing with non-numeric data
- Dealing with overfitting

I'm grateful to Alexei Artemov for his materials.

Naive boosting for regression

Boosting for regression

- Consider a regression problem $\frac{1}{2}\sum_{i=1}^{\ell}(h(x_i) - y_i)^2 \to \min_{h}$

Boosting for regression

- Consider a regression problem $\frac{1}{2}\sum_{i=1}^{\ell}(h(x_i) - y_i)^2 \to \min\limits_{h}$
- Search for solution in the form of weak learner composition $a_N(x) = \sum_{n=1}^{N} h_n(x)$ with weak learners $h_n \in \mathbb{H}$

Boosting for regression

- Consider a regression problem $\frac{1}{2}\sum_{i=1}^{\ell}(h(x_i) - y_i)^2 \to \min_h$
- Search for solution in the form of weak learner composition $a_N(x) = \sum_{n=1}^{N} h_n(x)$ with weak learners $h_n \in \mathbb{H}$
- The **boosting approach:** add weak learners greedily
  1. Start with a "trivial" weak learner $h_0(x) = \frac{1}{\ell}\sum_{i=1}^{\ell} y_i$

Boosting for regression

- Consider a regression problem $\frac{1}{2}\sum_{i=1}^{\ell}(h(x_i) - y_i)^2 \to \min_h$
- Search for solution in the form of weak learner composition $a_N(x) = \sum_{n=1}^{N} h_n(x)$ with weak learners $h_n \in \mathbb{H}$
- The **boosting approach:** add weak learners greedily
  1. Start with a "trivial" weak learner $h_0(x) = \frac{1}{\ell}\sum_{i=1}^{\ell} y_i$
  2. At step N, compute the residuals

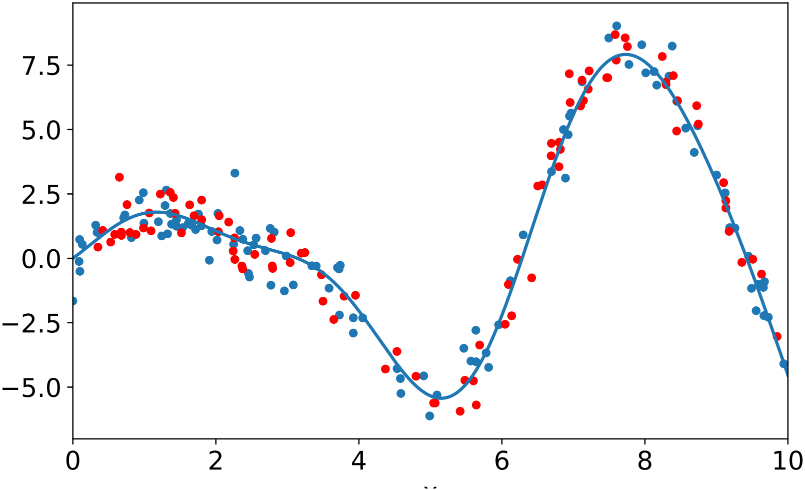
$$s_i^{(N)} = y_i - \sum_{n=1}^{N-1} h_n(x_i) = y_i - a_{N-1}(x_i), \qquad i = 1, \ldots, \ell$$

  3. Learn the next weak algorithm using

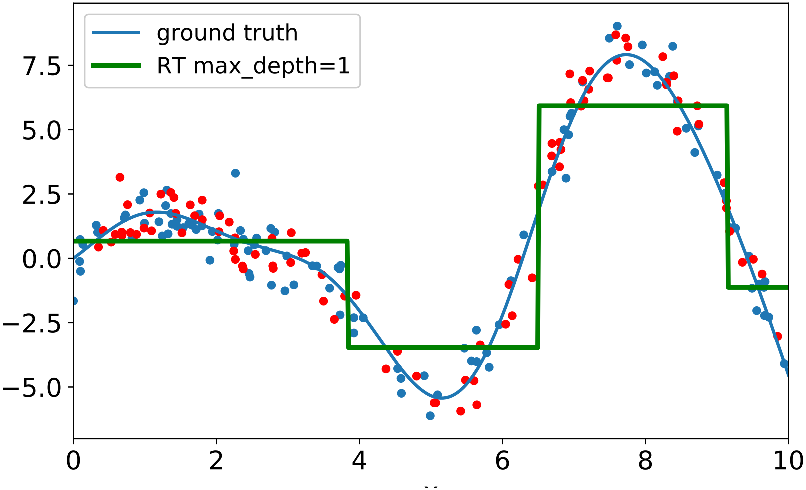
$$a_N(x) := \arg\min_{h \in \mathbb{H}} \frac{1}{2}\sum_{i=1}^{\ell}(h(x_i) - s_i^{(N)})^2$$

(this implementation may be found in, e.g., `scikit-learn`)

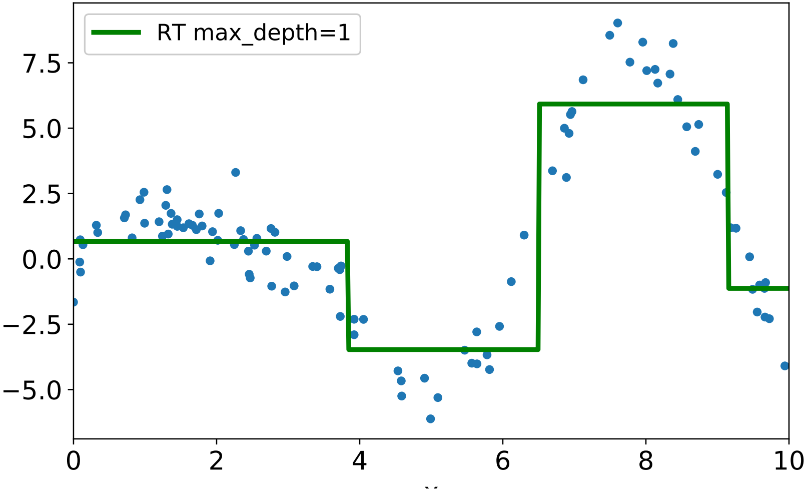
Boosting: an example regression problem



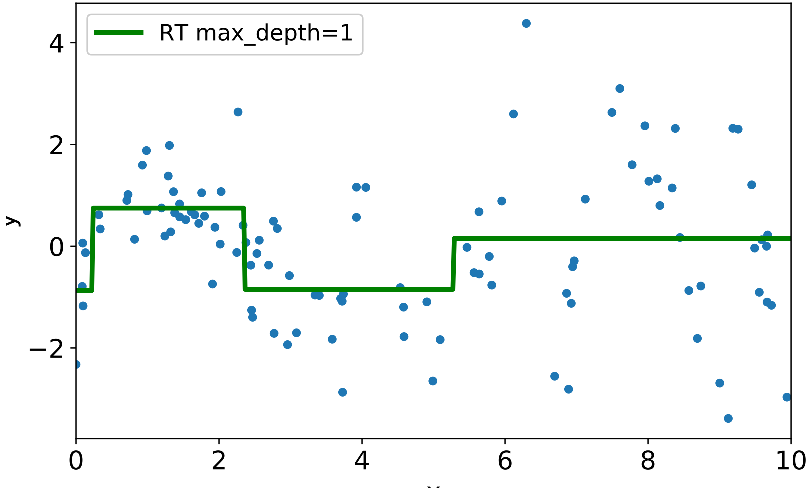
Boosting: an example regression problem



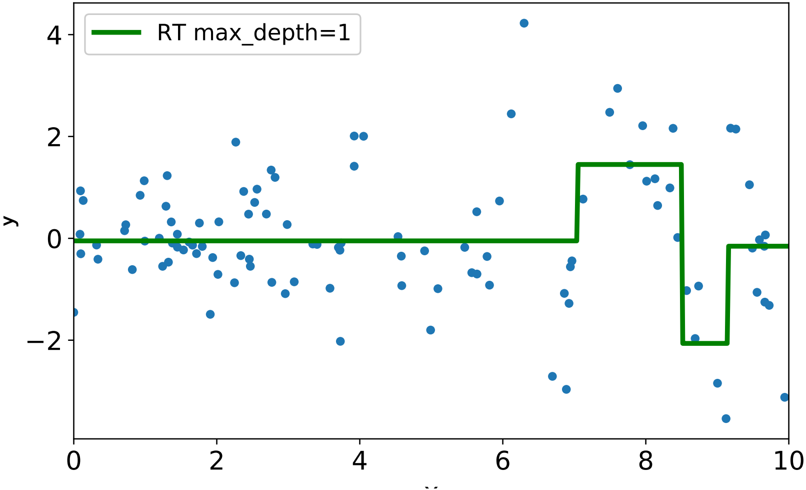
Boosting: an example regression problem



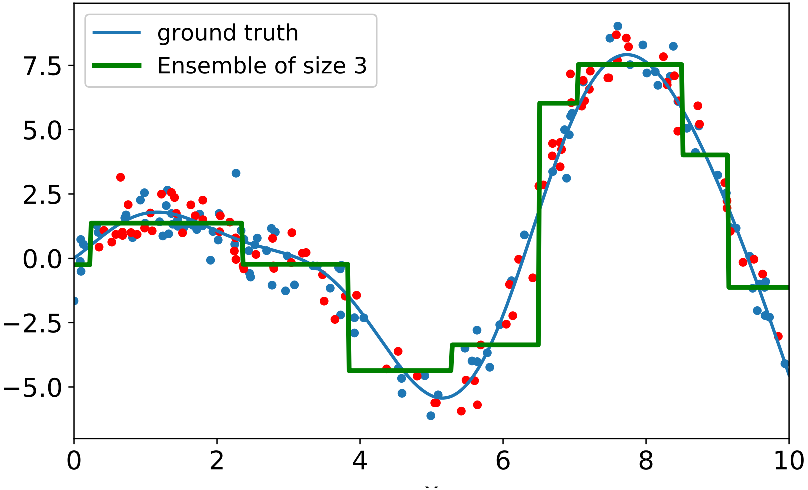
Boosting: an example regression problem



Boosting: an example regression problem



Boosting: an example regression problem



Gradient boosting

Gradient boosting: motivation

- With $a_{N-1}(x)$ already built, how to find the next γ_N and h_N if

$$\sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + \gamma h(x_i)) \to \min_{\gamma, h}$$

Gradient boosting: motivation

- With $a_{N-1}(x)$ already built, how to find the next γ_N and h_N if

$$\sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + \gamma h(x_i)) \to \min_{\gamma, h}$$

- Recall: functions decrease in the direction of negative gradient

Gradient boosting: motivation

- With $a_{N-1}(x)$ already built, how to find the next γ_N and h_N if

$$\sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + \gamma h(x_i)) \to \min_{\gamma, h}$$

- Recall: functions decrease in the direction of negative gradient
- View $L(y, z)$ as a function of z $(= a_N(x_i))$, execute gradient descent on z

Gradient boosting: motivation

- With $a_{N-1}(\mathbf{x})$ already built, how to find the next γ_N and h_N if

$$\sum_{i=1}^{\ell} L(y_i, a_{N-1}(\mathbf{x}_i) + \gamma h(\mathbf{x}_i)) \to \min_{\gamma, h}$$

- Recall: functions decrease in the direction of negative gradient
- View $L(y, z)$ as a function of z $(= a_N(\mathbf{x}_i))$, execute gradient descent on z
- Search for such $s_1, \ldots, s_\ell$ that

$$\sum_{i=1}^{\ell} L(y_i, a_{N-1}(\mathbf{x}_i) + s_i) \to \min_{s_1, \ldots, s_\ell}$$

Gradient boosting: motivation

- With $a_{N-1}(\mathbf{x})$ already built, how to find the next γ_N and h_N if

$$\sum_{i=1}^{\ell} L(y_i, a_{N-1}(\mathbf{x}_i) + \gamma h(\mathbf{x}_i)) \to \min_{\gamma, h}$$

- Recall: functions decrease in the direction of negative gradient
- View $L(y, z)$ as a function of z $(= a_N(\mathbf{x}_i))$, execute gradient descent on z
- Search for such $s_1, \dots, s_\ell$ that

$$\sum_{i=1}^{\ell} L(y_i, a_{N-1}(\mathbf{x}_i) + s_i) \to \min_{s_1, \dots, s_\ell}$$

- Choose $s_i = -\left.\frac{\partial L(y_i, z)}{\partial z}\right|_{z = a_{N-1}(\mathbf{x}_i)}$, approximate s_i's by $h_N(\mathbf{x}_i)$

The Gradient Boosting Machine [Friedman, 2001]

- Input:
  - Training set $X^\ell = \{(\mathbf{x}_i, y_i)\}_{i=1}^{\ell}$
  - Number of boosting iterations N
  - Loss function $Q(y, z)$ with its gradient $\frac{\partial Q}{\partial z}$
  - A family $\mathbb{H} = \{h(\mathbf{x})\}$ of weak learners and their associated learning procedures
  - Additional hyperparameters of weak learners (tree depth, etc.)
- Initialize GBM $h_0(\mathbf{x})$ using some simple rule (zero, most popular class, etc.)
- Execute boosting iterations $t = 1, \ldots, N$ (see next slide)
- Compose the final GBM learner: $a_N(\mathbf{x}) = \sum_{t=0}^{N} \gamma_i h_i(\mathbf{x})$

The Gradient Boosting Machine [Friedman, 2001]

At every iteration:

1. Compute pseudo-residuals: $s_i = -\left.\frac{\partial Q(y_i, z)}{\partial z}\right|_{z=a_{N-1}(x_i)}, i = 1, \ldots, \ell$

The Gradient Boosting Machine [Friedman, 2001]

At every iteration:

1. Compute pseudo-residuals: $s_i = -\left.\frac{\partial Q(y_i, z)}{\partial z}\right|_{z=a_{N-1}(\mathbf{x}_i)}, i = 1, \ldots, \ell$
2. Learn $h_N(\mathbf{x}_i)$ by regressing onto $s_1, \ldots, s_\ell$:

$$h_N(x) = \arg\min_{h \in \mathbb{H}} \sum_{i=1}^{\ell} (h(\mathbf{x}_i) - s_i)^2$$

The Gradient Boosting Machine [Friedman, 2001]

At every iteration:

1. Compute pseudo-residuals: $s_i = -\left.\frac{\partial Q(y_i, z)}{\partial z}\right|_{z=a_{N-1}(\mathbf{x}_i)}, i = 1, \ldots, \ell$
2. Learn $h_N(\mathbf{x}_i)$ by regressing onto $s_1, \ldots, s_\ell$:

$$h_N(x) = \underset{h \in \mathbb{H}}{\arg\min} \sum_{i=1}^{\ell} (h(\mathbf{x}_i) - s_i)^2$$

3. Find the optimal γ_N using plain gradient descent:

$$\gamma_N = \underset{\gamma \in \mathbb{R}}{\arg\min} \sum_{i=1}^{\ell} Q(y_i, a_{N-1}(\mathbf{x}_i) + \gamma h_N(\mathbf{x}_i))$$

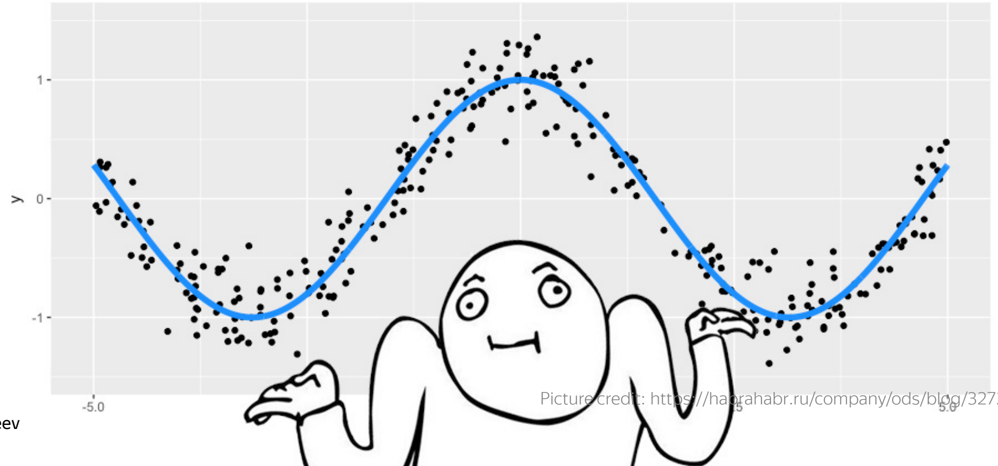
The Gradient Boosting Machine [Friedman, 2001]

At every iteration:

1. Compute pseudo-residuals: $s_i = - \left. \frac{\partial Q(y_i, z)}{\partial z} \right|_{z = a_{N-1}(x_i)}, i = 1, \dots, \ell$
2. Learn $h_N(\mathbf{x}_i)$ by regressing onto $s_1, \dots, s_\ell$:
$$h_N(x) = \arg\min_{h \in \mathbb{H}} \sum_{i=1}^{\ell} (h(\mathbf{x}_i) - s_i)^2$$
3. Find the optimal γ_N using plain gradient descent:
$$\gamma_N = \arg\min_{\gamma \in \mathbb{R}} \sum_{i=1}^{\ell} Q(y_i, a_{N-1}(\mathbf{x}_i) + \gamma h_N(\mathbf{x}_i))$$
4. Update the GBM by $a_N(\mathbf{x}_i) \leftarrow a_{N-1}(\mathbf{x}) + \gamma_N h_N(\mathbf{x})$

GBM: an example regression problem

- Consider a training set for a $X^{300} = \{(x_i, y_i)\}_{i=1}^{300}$ where $x_i \in [-5, 5]$, $y_i = \cos(x_i) + \varepsilon_i$, $\varepsilon_i \sim \mathcal{N}(0, 1/5)$

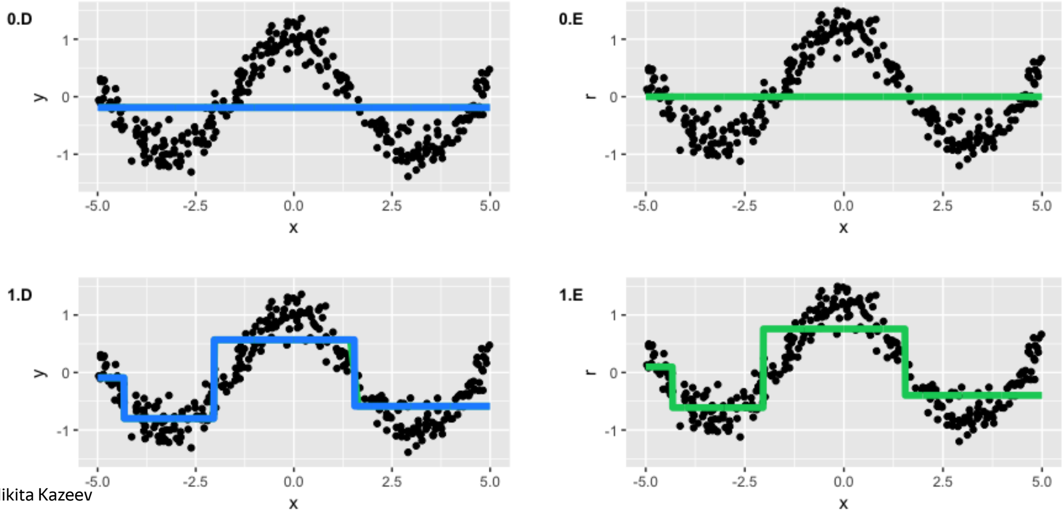


Picture credit: https://habrahabr.ru/company/ods/blog/327250

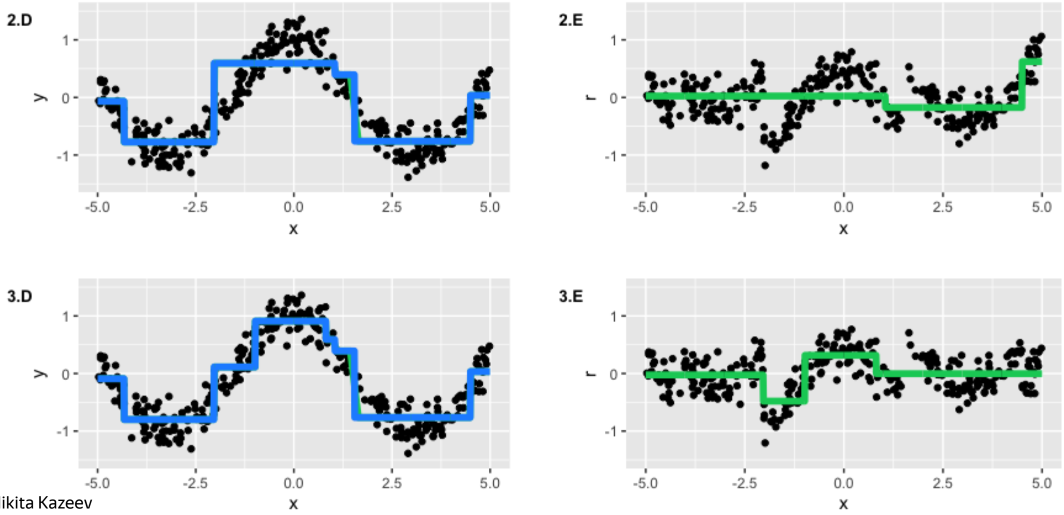
GBM: an example regression problem

- Pick $\mathrm{N} = 3$ boosting iterations
- Quadratic loss $\mathrm{Q}(\mathrm{y}, \mathrm{z}) = (\mathrm{y} - \mathrm{z})^2$
- Gradient of the quadratic loss $\frac{\partial \mathrm{Q}(\mathrm{y_i}, \mathrm{z})}{\partial \mathrm{z}} = (\mathrm{y} - \mathrm{z})$ is just redisuals
- Pick decision trees as weak learners $\mathrm{h_i}(\mathbf{x})$
- Set 2 as the maximum depth for decision trees

GBM: an example regression problem

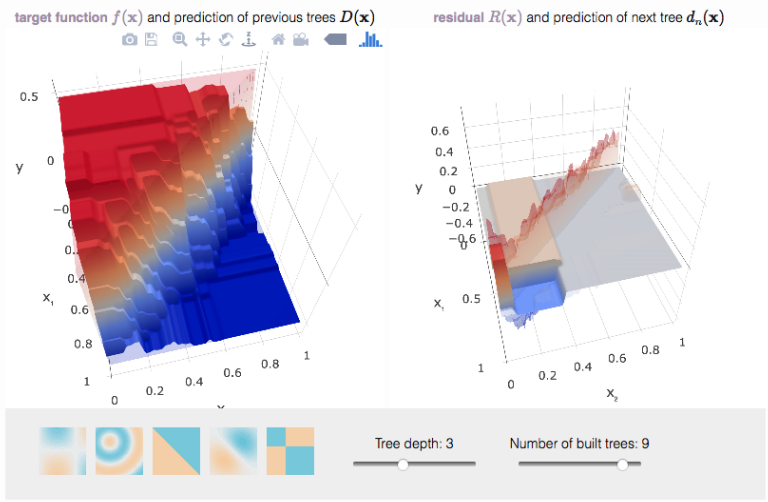


GBM: an example regression problem



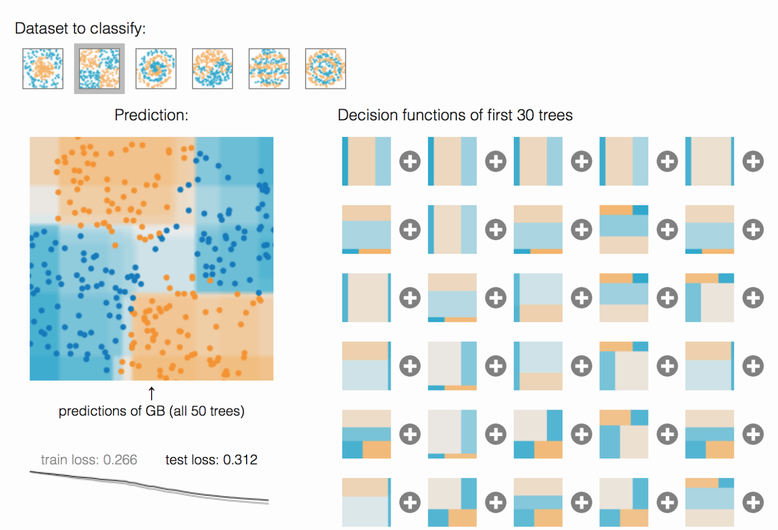
GBM: an interactive demo

http://arogozhnikov.github.io/2016/06/24/gradient_boosting_explained.html



GBM: an interactive demo

http://arogozhnikov.github.io/2016/07/05/gradient_boosting_playground.html



XGBoost

Extreme Gradient Boosting

1. Approximate the descent direction constructed using second order derivatives

$$\sum_{i=1}^{\ell} \left(-s_i h(\mathbf{x}_i) + \frac{1}{2} t_i h^2(\mathbf{x}_i) \right) \to \min_h, \qquad t_i = \frac{\partial^2}{\partial z^2} L(y_i, z) \Big|_{a_{N-1}(\mathbf{x}_i)}$$

Extreme Gradient Boosting

1. Approximate the descent direction constructed using second order derivatives

$$\sum_{i=1}^{\ell}\left(-s_i h(\mathbf{x}_i) + \frac{1}{2}t_i h^2(\mathbf{x}_i)\right) \to \min_h, \qquad t_i = \frac{\partial^2}{\partial z^2} L(y_i, z)\Big|_{a_{N-1}(\mathbf{x}_i)}$$

2. Penalize large leaf counts J and large leaf coefficient norm $\|b\|_2^2 = \sum_{j=1}^{J} b_j^2$

$$\sum_{i=1}^{\ell}\left(-s_i h(x_i) + \frac{1}{2}t_i h^2(x_i)\right) + \gamma J + \frac{\lambda}{2}\sum_{j=1}^{J} b_j^2 \to \min_h$$

where $b(\mathbf{x}) = \sum_{j=1}^{J} b_j[\mathbf{x} \in R_j]$

Extreme Gradient Boosting

3. Choose split $[\mathbf{x}_j < t]$ at node R to maximize

$$Q = H(R) - H(R_\ell) - H(R_r) \to \max,$$

where the impurity criterion

$$H(R) = -\frac{1}{2}\left(\sum_{(t_i,s_i)\in R} s_j\right)^2 \Bigg/ \left(\sum_{(t_i,s_i)\in R} t_j + \lambda\right) + \gamma$$

Extreme Gradient Boosting

3. Choose split $[\mathbf{x}_j < t]$ at node R to maximize

$$Q = H(R) - H(R_\ell) - H(R_r) \to \max,$$

where the impurity criterion

$$H(R) = -\frac{1}{2}\left(\sum_{(t_i,s_i)\in R} s_j\right)^2 \Bigg/ \left(\sum_{(t_i,s_i)\in R} t_j + \lambda\right) + \gamma$$

4. The stopping rule: declare the node a leaf if even the best split gives negative Q

Categorical features

Categorical features

Image: www.petsworld.in

One-hot encoding

$$[\text{proton, pion, kaon}] \to [[1,0,0],[0,1,0],[0,0,1]]$$

One-hot encoding

$$[\text{proton, pion, kaon}] \rightarrow [[1, 0, 0], [0, 1, 0], [0, 0, 1]]$$

- Doesn't scale well with the number of categories

CTR (aka click-through ratio)

For each pair (target_class, categorical_feature_value):

$$\text{ctr}_\text{i} = \frac{\text{countInClass} + \text{prior}}{\text{totalCount} + 1}$$

- countInClass – number of objects in the i-th class with the current categorical feature value
- prior – algorithm parameter
- totalCount – total number of objects with the current categorical feature value

CTR example

fruit	target	ctr
apple	0	0.625
orange	0	0.25
apple	1	0.625
apple	1	0.625

prior $= 0.5$

Classes counter

For each pair (target_class, categorical_feature_value):

$$\text{count}_\text{i} = \frac{\text{curCount} + \text{prior}}{\text{totalCount} + 1}$$

- curCount – number of objects with the current categorical feature value
- prior – algorithm parameter
- totalCount – total number of objects

Counters example

fruit	target	ctr	counter
apple	0	0.625	0.7
orange	0	0.25	0.3
apple	1	0.625	0.7
apple	1	0.625	0.7

prior $= 0.5$

Meet CatBoost

- Gradient boosting on decision trees
- Categorical features handling (even more advanced than discussed!)
- A novel dynamic boosting scheme (submitted to NIPS) [I'm a coauthor]
- Released into open source by Yandex
- Used in the LHCb PID

Overfitting

GBM: regulization via shrinkage

- For too simple weak learners, the negative gradient is approximated badly $\implies$ random walk in space of samples
- For too complex weak learners, a few boosting steps may be enough for overfitting
- **Shrinkage:** make shorter steps using a learning rate $\eta \in (0, 1]$

$$a_N(x_i) \leftarrow a_{N-1}(x) + \eta \gamma_N h_N(x)$$

(effectively distrust gradient direction estimated via weak learners)

GBM: shrinkage

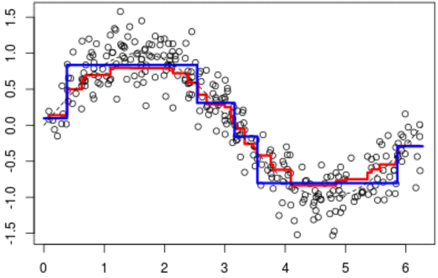


Figure: High shrinkage

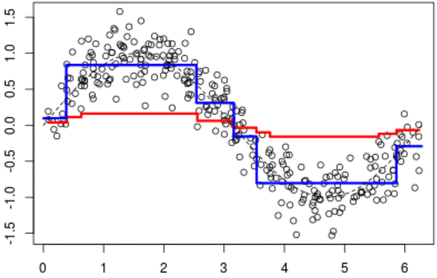
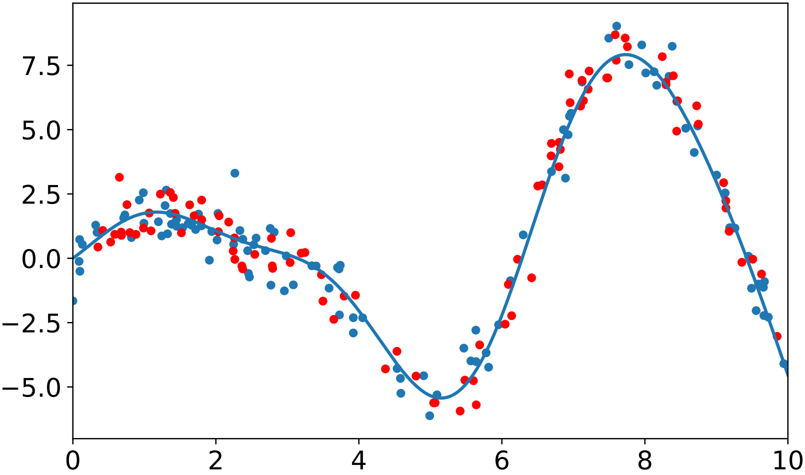
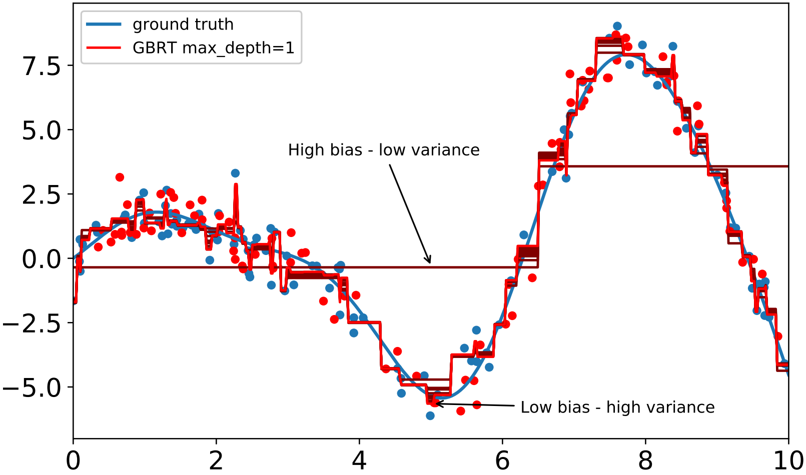


Figure: Low shrinkage

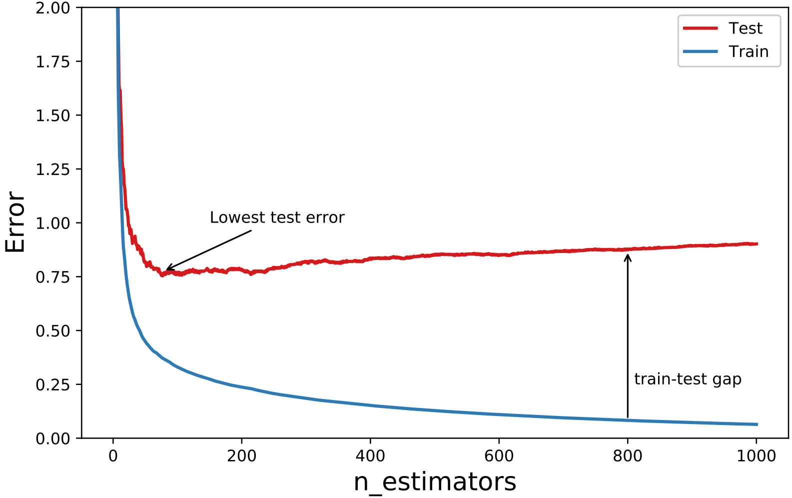
GBM: regularization approaches



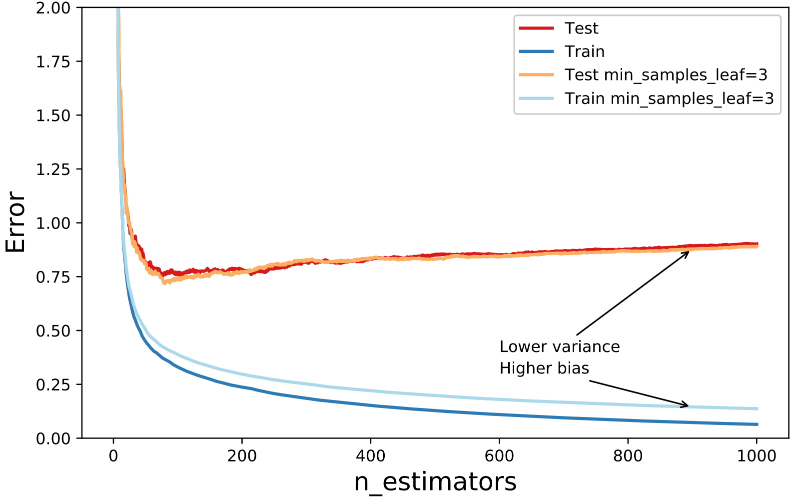
GBM: regularization approaches



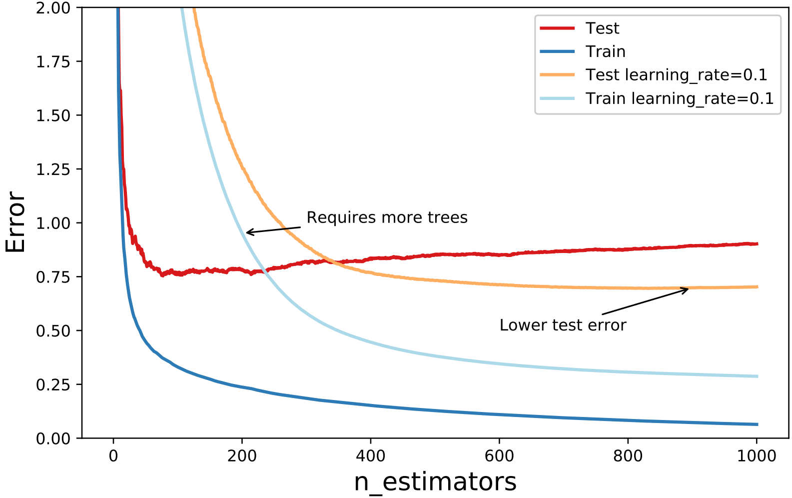
GBM: regularization approaches



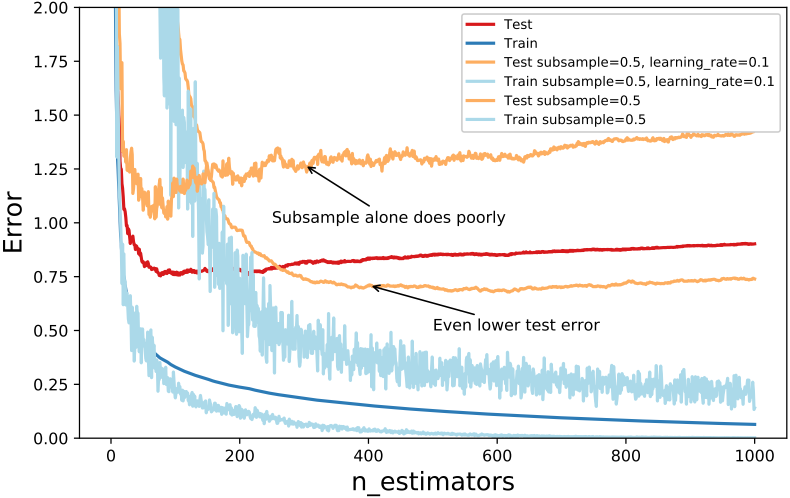
GBM: regularization approaches



GBM: regularization approaches



GBM: regularization approaches



Gradients bias in gradient boosting

- Each subsequent tree is fit to the gradient between the current predictions on train and the true labels

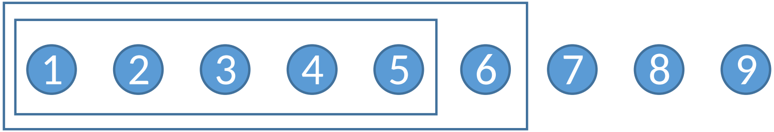
Gradients bias in gradient boosting

- Each subsequent tree is fit to the gradient between the current predictions on train and the true labels
- The gradient is estimated using the model fitted on the very dataset used for training

Gradients bias in gradient boosting

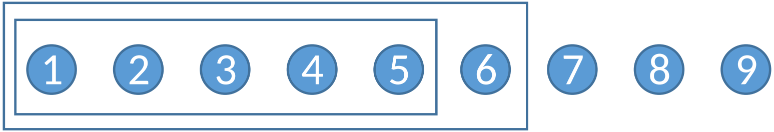
- Each subsequent tree is fit to the gradient between the current predictions on train and the true labels
- The gradient is estimated using the model fitted on the very dataset used for training
- The gradients are likely to be overfitted

Dynamic boosting



› Order data randomly

Dynamic boosting



- Order data randomly
- For each element maintain prediction based on the previous model elements

Summary [theory]

- Boosting: a general meta-algorithm aimed at composing a strong hypothesis from multiple weak hypotheses
- Boosting can be applied for arbitrary losses, arbitrary problems (regression, classification) and over arbitrary weak learners
- The Gradient Boosting Machine: a general approach to boosting adding weak learners that approximate gradient of the loss function
- AdaBoost: gradient boosting with an exponential loss function resulting in reweighting training instances when adding weak learners
- XGBoost: gradient boosting with second order optimization, penalized loss and particular choice of impurity criterion

Summary [practice]

[Disclaimer] Objectively comparing algorithms is hard, but judging from competitions & industry cases…

Summary [practice]

[Disclaimer] Objectively comparing algorithms is hard, but judging from competitions & industry cases…

- As of 2017 Gradient Boosting and Deep Learning rule
    - If you're using something else, think

Summary [practice]

[Disclaimer] Objectively comparing algorithms is hard, but judging from competitions & industry cases…

- As of 2017 Gradient Boosting and Deep Learning rule
  - If you're using something else, think
- There are more-or-less equal implementations in H2O, LightGBM, XGBoost
- You're also invited to try the new catboost [the recommendation is biased, gradients – not so much…]

Contacts

Nikita Kazeev
Researcher

kazeevn@yandex-team.ru

nikita.kazeev.9