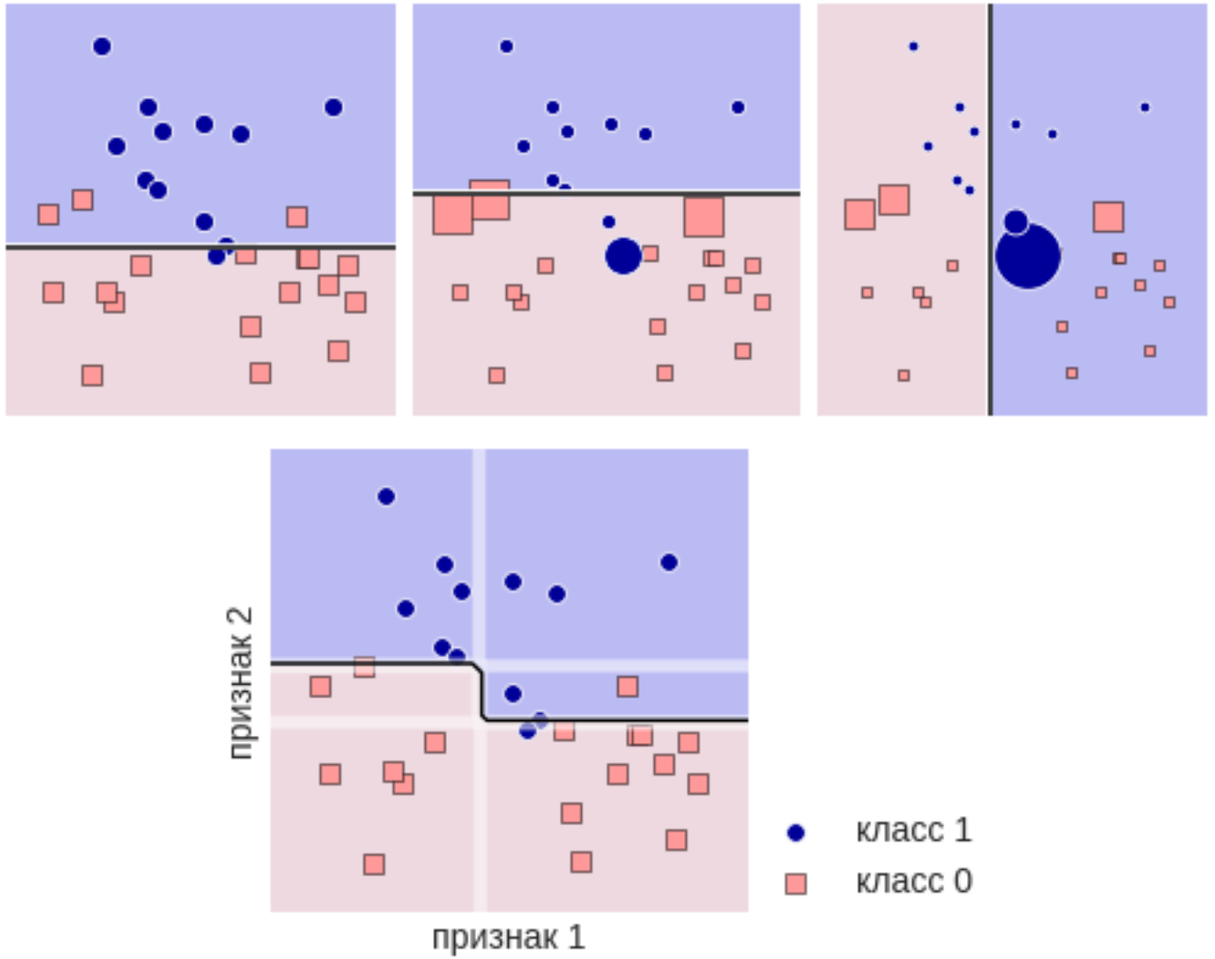
Градиентный бустинг

Александр Дьяконов

14 декабря 2020 года

Градиентный бустинг над деревьями

Вспоминаем идею...



Идея градиентного бустинга

FSAM + минимизация в случае дифференцируемой ф-ии ошибки

Задача регрессии $(x_i, y_i)_{i=1}^m$

дифференцируемая функция ошибки $L(y,a)$

уже есть алгоритм $a(x)$ **строим** $b(x)$**:**

$$a(x_i) + b(x_i) = y_i, i \in \{1,2,\ldots,m\}.$$

т.е. «в некотором смысле» настраиваемся на невязку

$$b(x_i) \approx y_i - a(x_i)$$

формально надо:

$$\sum_{i=1}^m L(y_i, a(x_i) + b(x_i)) \to \min$$

а не

$$\sum_{i=1}^m L(y_i - a(x_i), b(x_i)) \to \min$$

хотя часто они эквивалентны

Проблема

Задача

$$\sum_{i=1}^{m} L(y_i, a(x_i) + b(x_i)) \to \min$$

может не решаться аналитически

$$F = \sum_{i=1}^{m} L(y_i, a(x_i) + b_i) \to \min_{(b_1,\ldots,b_m)}$$

Функция $F(b_1,\ldots,b_m)$ **убывает в направлении антиградиента,**

поэтому выгодно считать

$$b_i = -L'(y_i, a(x_i)), \;\; i \in \{1,2,\ldots,m\},$$

новая задача для настройки второго алгоритма:

$$(x_i, -L'(y_i, a(x_i)))_{i=1}^{m}.$$

Алгоритм градиентного бустинга (примитивный вариант)

- **Строим алгоритм в виде**

$$a_n(x) = \sum_{t=1}^{n} b_t(x),$$

для удобства можно даже считать, что $a_0(x) \equiv 0$**.**

- **Пусть построен** $a_t(x)$**, тогда обучаем алгоритм** $b_{t+1}(x)$ **на выборке**

$$(x_i, -L'(y_i, a_t(x_i)))_{i=1}^{m}$$

- $a_{t+1}(x) = a_t(x) + b_{t+1}(x)$**.**

Итерационно обучаем сумму алгоритмов…

Вот почему называется градиентный бустинг

Частный случай: регрессия с СКО

$$L(y,a) = \frac{1}{2}(y-a)^2$$

$$L'(y,a) = -(y-a)$$

Задача для настройки следующего алгоритма

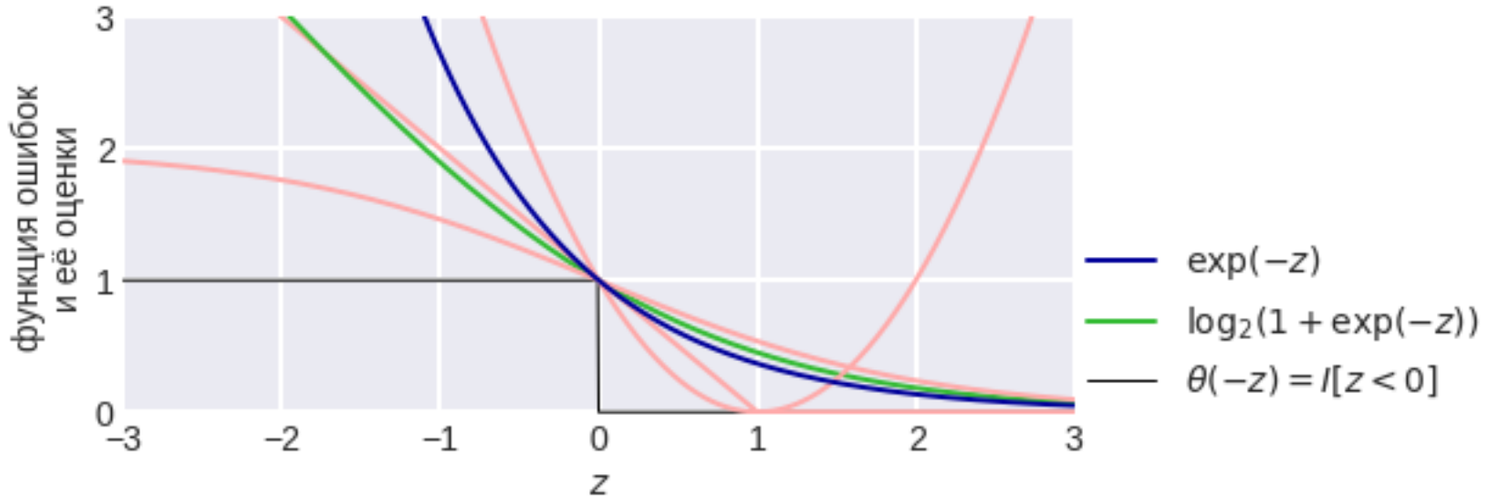
$$(x_i,\ y_i - a_t(x_i))_{i=1}^m$$

т.е. очень логично: настраиваемся на невязку!

Частный случай: классификация на два класса

надо найти дифференцируемую функцию ошибки…

- **предполагаем, что алгоритм выдаёт вещественные значения**
- **делаем функцию похожей на «совпадение»**



Частный случай: классификация на два класса

BinomialBoost – логистическая функция ошибки:

$$L(y,a) = \log(1+e^{-y\cdot a}),\ a \in (-\infty,+\infty),\ y \in \{-1,+1\},$$

$$L'(y,a) = -\frac{y}{1+e^{-y\cdot a}}.$$

Функция ошибки типа Adaboost:

$$L(y,a) = e^{-y\cdot a},\ a \in (-\infty,+\infty),\ y \in \{-1,+1\},$$

$$L(y,a) = -ye^{-y\cdot a}.$$

здесь что-то выводится явно…

Итерация градиентного бустинга

Как решать задачу

$$(x_i, -L'(y_i, a_t(x_i)))_{i=1}^m$$**?**

Любым простым методом!

Мы уже настраиваемся на нужную функцию ошибки.

Проблема

Шаг в сторону антиградиента

- **не приводит в локальный минимум (сразу) $\Rightarrow$ итерации**
- **мы всё равно не можем сделать такой шаг, а лишь шаг по ответам какого-то алгоритма модели $\Rightarrow$ не нужно стремиться шагать именно туда**

Дальше решение проблем…

Наискорейший спуск

$$\sum_{i=1}^{m} L(y_i, a_t(x_i) + \eta \cdot b_t(x_i)) \to \min_{\eta},$$

$$a_{t+1}(x) = a_t(x) + \eta_t \cdot b_t(x) = \eta_1 \cdot b_1(x) + \ldots + \eta_t \cdot b_t(x)$$

Эвристика сокращения – Shrinkage

$$a_{t+1}(x) = a_t(x) + \eta \cdot b_t(x),$$

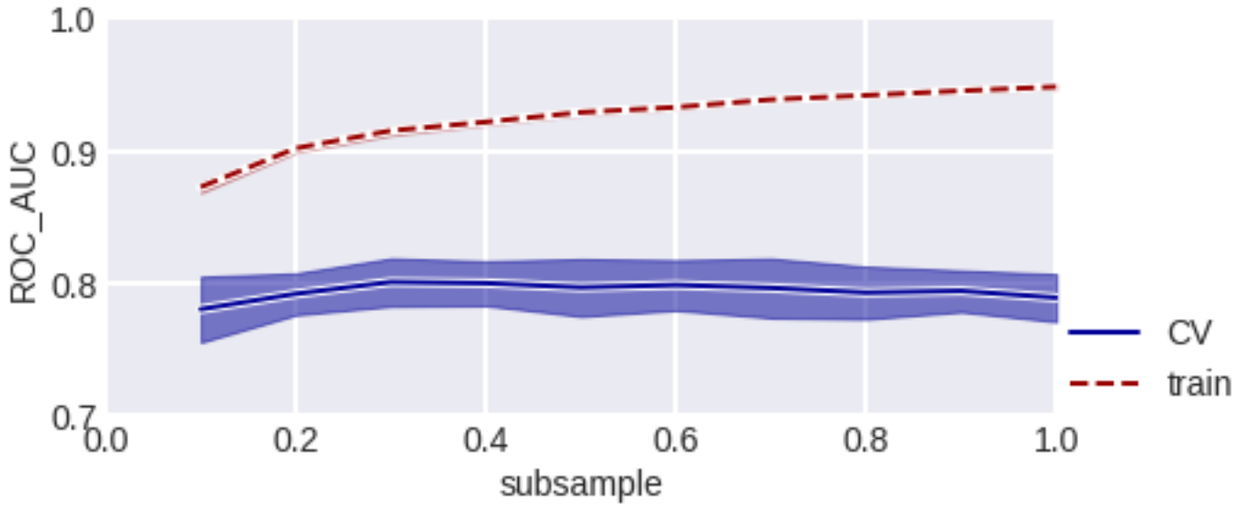
$\eta \in (0, 1]$ – **скорость (темп) обучения (learning rate)**

Видно, что число слагаемых (базовых алгоритмов) – шагов бустинга – надо контролировать (при увеличении можем переобучиться)

Чем меньше скорость, тем больше итераций надо

Стохастический градиентный бустинг (Stochastic gradient boosting)

Идея бэгинга Бреймана: bag fraction ~ берём часть всей выборки



- м.б. лучше качество («регуляризация»)
- быстрее
- аналог обучения по минибатчам
- можно вычислить OOB-ошибки

J. Friedman «Stochastic Gradient Boost» // 1999 http://statweb.stanford.edu/~jhf/ftp/stobst.pdf

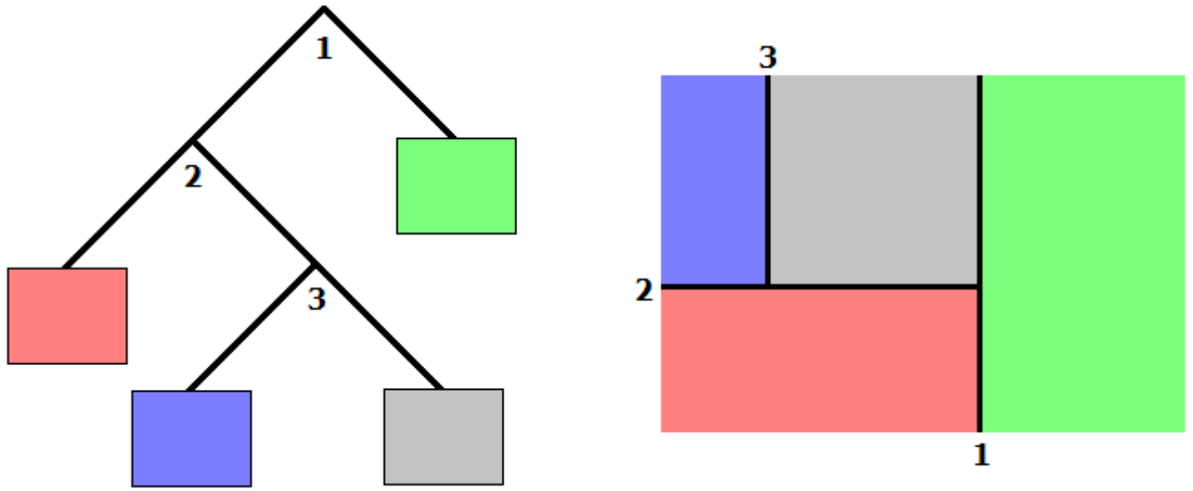
Column / Feature Subsampling for Regularization

аналогичная идея с признаками

TreeBoost – градиентный бустинг над деревьями

Решающее дерево:

$$b(x) = \sum_j \beta_j I[x \in X_j]$$



TreeBoost – градиентный бустинг над деревьями

Наша основная задача

$$\sum_{i=1}^{m} L(y_i, a(x_i) + \sum_j \beta_j I[x \in X_j]) \to \min$$

Разбиваем по областям:

$$\sum_{x_i \in X_j} L(y_i, a(x_i) + \beta_j) \to \min_{\beta_j}.$$

если разбиение выбрано и зафиксировано,

то в каждой области осталось выбрать оптимальную константу

Продвинутые методы оптимизации

Наша основная задача

$$F = \sum_{i=1}^{m} L(y_i, a(x_i) + b_i) \to \min_{(b_1,\ldots,b_m)},$$

заметим, что

$$F = \sum_{i=1}^{m} L(y_i, a(x_i) + b_i) \approx$$

$$\sum_{i=1}^{m} \left[L(y_i, a(x_i)) + L'(y_i, a(x_i)) \cdot b_i + \frac{1}{2} L''(y_i, a(x_i)) \cdot b_i^2 \right]$$

(частные производные по второму аргументу функции ошибки)

Продвинутые методы оптимизации

$$\sum_{i=1}^{m}\left[g_i b_i + \frac{1}{2} h_i b_i^2 \right] \to \min,$$

$$g_i = L'(y_i, a(x_i)),$$

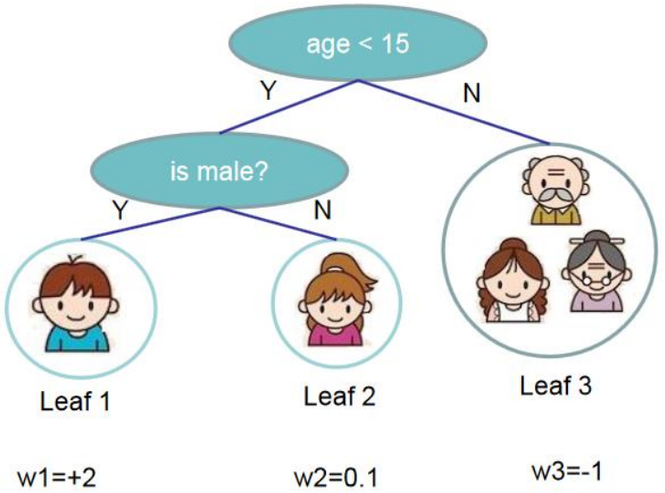
$$h_i = L''(y_i, a(x_i)).$$

Сделаем оптимизацию с регуляризацией.

Продвинутые методы оптимизации

Пусть дерево $b(x)$ делит пространство объектов на T областей $X_1,\ldots,X_T$, в каждой области X_j принимает значение β_j.

$$\Phi = \sum_{i=1}^{m}\left[g_i b_i + \frac{1}{2} h_i b_i^2 \right] + \gamma T + \lambda \frac{1}{2}\sum_{j=1}^{T} \beta_j^2 \to \min$$



$$\Omega = \gamma 3 + \tfrac{1}{2}\lambda(4 + 0.01 + 1)$$

Продвинутые методы оптимизации

$$\Phi = \sum_{j=1}^{T}\left[\sum_{x_i \in X_j}\left[g_i\beta_j + \frac{1}{2}h_i\beta_j^2\right] + \lambda\frac{1}{2}\beta_j^2\right] + \gamma T =$$

$$= \sum_{j=1}^{T}\left[\beta_j\sum_{x_i \in X_j} g_i + \frac{1}{2}\beta_j^2\left(\sum_{x_i \in X_j} h_i + \lambda\right)\right] + \gamma T$$

Приравнивая производную к нулю:

$$\beta_j = -\frac{\sum\limits_{x_i \in X_j} g_i}{\sum\limits_{x_i \in X_j} h_i + \lambda}.$$

Продвинутые методы оптимизации

Минимальное значение (при фиксированной структуре дерева)

$$\Phi_{\min} = -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum\limits_{x_i \in X_j} g_i\right)^2}{\sum\limits_{x_i \in X_j} h_i + \lambda} + \gamma T .$$

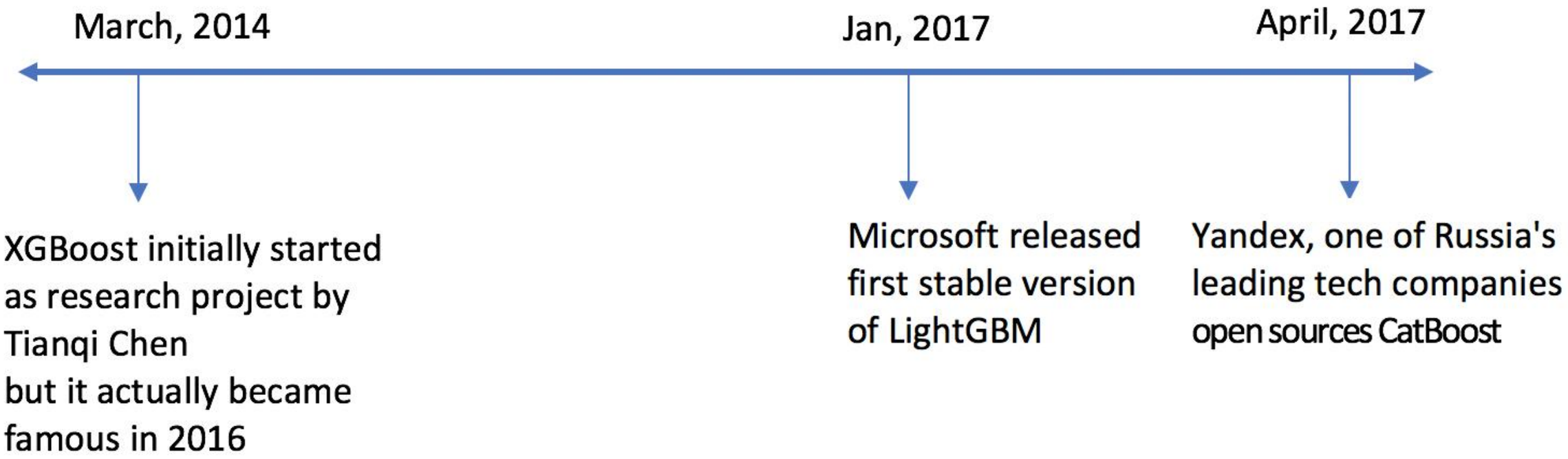
Можно использовать при построении дерева для его оценки:

$$\text{Gain} = \frac{1}{2}\left(\frac{\left(\sum\limits_{x_i \in X_{\text{left}}} g_i\right)^2}{\sum\limits_{x_i \in X_{\text{left}}} h_i + \lambda} + \frac{\left(\sum\limits_{x_i \in X_{\text{right}}} g_i\right)^2}{\sum\limits_{x_i \in X_{\text{right}}} h_i + \lambda} - \frac{\left(\sum\limits_{x_i \in X_{\text{left}}} g_i + \sum\limits_{x_i \in X_{\text{right}}} g_i\right)^2}{\sum\limits_{x_i \in X_{\text{left}}} h_i + \sum\limits_{x_i \in X_{\text{right}}} h_i + \lambda}\right) - \gamma$$

Продвинутые методы оптимизации

Не используем какой-то традиционный критерий расщепления.
Исходим из функции ошибки!

История продвинутых методов



https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db

Особенности

	XGBoost	LightGBM	CatBoost
Построение деревьев	По уровням потом добавили по листьям, но для гистограмм	По листьям Leaf-wise (best-first)	По уровням однородно oblivious trees
Поиск расщеплений	Exact greedy algorithm (полный перебор) + добавили потом гистограммный подход tree_method='hist'	Гистограммный подход (использование бинов) +	
Нули обрабатываются как NaN	+	По умолчанию use_missing=True	

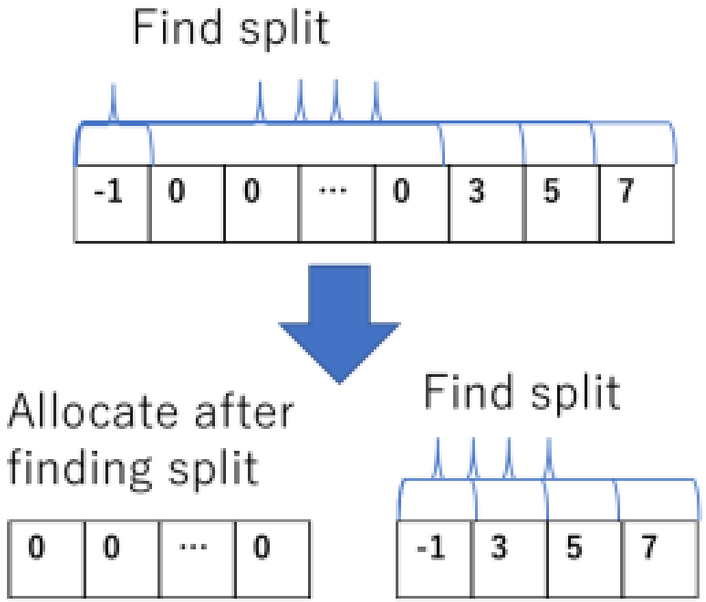
Особенности

	XGBoost	LightGBM	CatBoost
Gradient-based One-Side Sampling (GOSS)	– pre-sorted algorithm & Histogram-based	Среди малых градиентов сэмплируем, но с большим весом	– Но есть динамический бустинг!
Exclusive Feature Bundling		Связываем разреженные признаки, которые одновременно не нули	
Дополнительные фишки		Random forest mode	Динамический бустинг

Особенности

	XGBoost	LightGBM	CatBoost
Подвыборка	subsample	–	feature_fraction
Категориальные признаки	–	+	+ Smoothed target encoding Можно OHE для всех с числом категорий < one_hot_max_size Жадные комбинации категориальных признаков

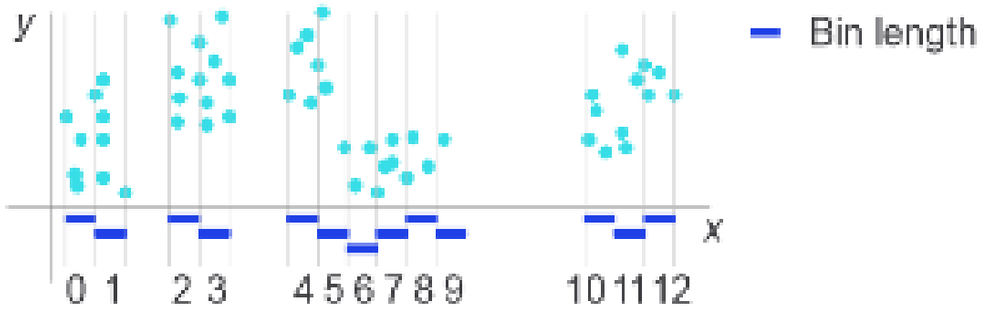
Игнорирование нулей / NaN



https://mlexplained.com/2018/01/05/lightgbm-and-xgboost-explained/

Гистограммный подход (Histogram based algorithm)

каждый вещественный признак дискретизуется – разбивается на бины



теперь число порогов, которые надо посмотреть ~ число бинов

CatBoost = Category + Boosting: проблема смещения

Smoothed target encoding

для категориальных признаков

При построении дерева

значение в листе = сумма антиградиентов

получается утечка

мы оцениваем значение в точке, зная метку на ней!

Динамический бустинг

- случайная перестановка обучения
- оценивания значения, используя информацию до рассматриваемой точки в таблице

oblivious trees

один предикат на каждом уровне

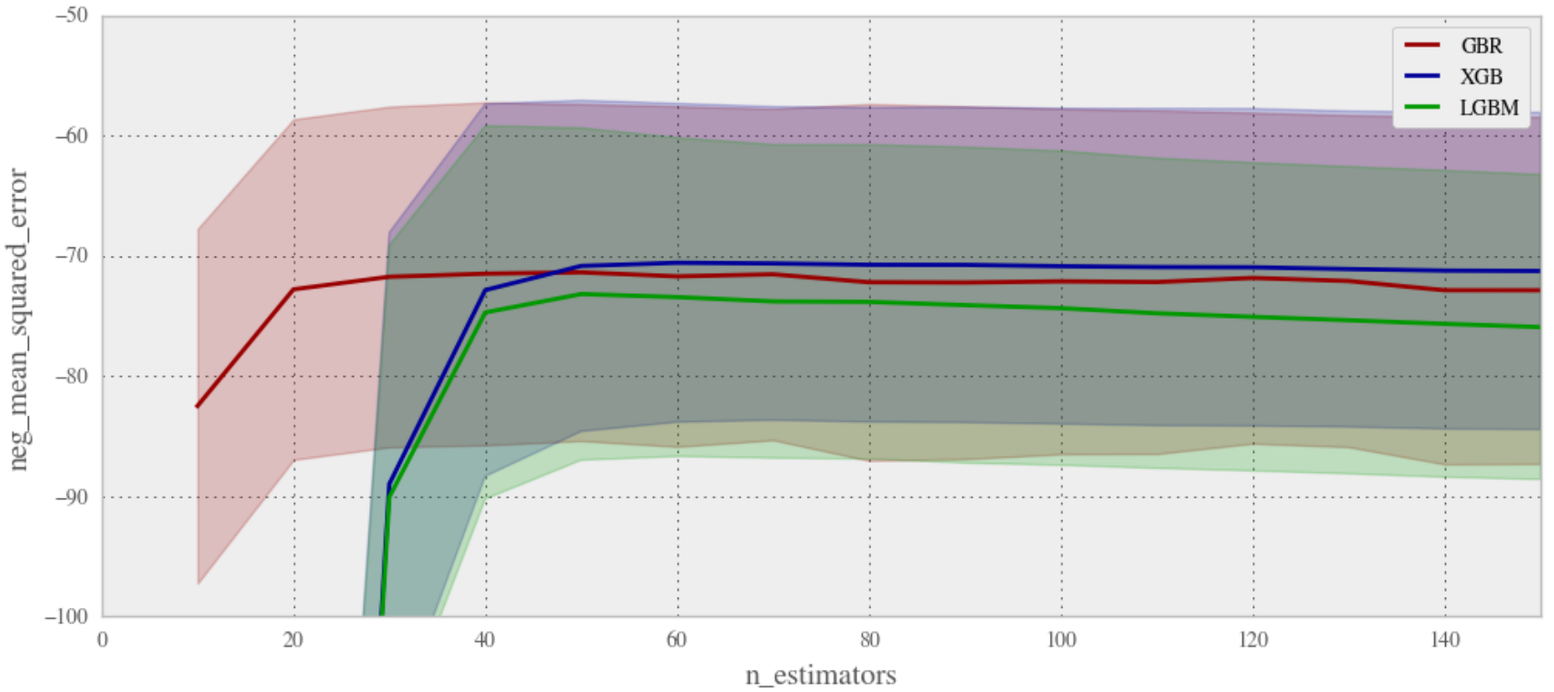
CatBoost = Category + Boosting

работает «из коробки»

с параметрами по умолчанию

Современные реализации градиентного бустинга

sklearn.ensemble.	GradientBoostingRegressor GradientBoostingClassifier
XGBoost (eXtreme Gradient Boosting)	https://github.com/dmlc/xgboost
LightGBM, Light Gradient Boosting Machine	https://github.com/Microsoft/LightGBM
CatBoost	https://github.com/catboost/catboost



Параметры градиентного бустинга

Параметры, определяющие тип бустинга

objective // loss_function	– какая задача решается, какая целевая функция и в каком формате будет ответ
booster boosting_type	– какой бустинг проводить: над решающими деревьями, линейный или dart – lgb: gbdt / dart (Dropouts meet Multiple Additive Regression Trees) / goss / rf
tree_method	– как строить деревья (grow_policy – порядок построения дерева: на следующем шаге расщеплять вершину, ближайшую к корню, или на которой ошибка максимальна)
base_score	– начальный ответ на всех объектах (bias)
eval_metric	– значения какой функции ошибки смотреть на контроле (как правило, задание этого параметра не означает, что эту функцию будем минимизировать при настройке бустинга)

DART: Dropouts meet Multiple Additive Regression Trees

Вместо

$$a_n(x) = \sum_{t=1}^{n} b_t(x)$$

берём подмножество построенных деревьев $Q = \text{randsubset}(\{1, 2, \ldots, n\})$

пытаемся дополнить их

$$b_{t+1} = \arg\min_b \sum_{i=1}^{m} L(y_i, \sum_{t \in Q} b_t(x_i) + b(x_i))$$

потом нужна поправка, что смещали ответ не всего ансамбля (подробно об это не будем)

$$a_n(x) = \sum_{t=1}^{n} b_t(x) + \eta b_{t+1}(x)$$

http://proceedings.mlr.press/v38/korlakaivinayak15.pdf

Параметры градиентного бустинга

Основные параметры

`learning_rate (eta)`	– темп (скорость) обучения
`n_estimators (num_iterations) // iterations`	– число итераций бустинга (базовых алгоритмов)

Параметры градиентного бустинга

Параметры ограничивающие сложность дерева

max_depth	– максимальная глубина
gamma	– порог на уменьшение функции ошибки при расщеплении в дереве
min_child_weight	– минимальная сумма весов объектов в потомках
max_delta_step	– порог на изменение весов
max_leaves num_leaves	– максимальное число вершин в дереве
/ min_split_gain (min_gain_to_split)	– порог на изменение loss-функции
/ min_child_samples (min_data_in_leaf)	– минимальное число объектов в листьях
/ min_sum_hessian_in_leaf	– минимальная сумма весов объектов в листе, минимальное число объектов, при котором делается расщепление

/ subsample_freq (int, optional (default=0)) – частота взятия подвыборок
// sampling_frequency

Что такое веса

min_child_weight – «minimum sum of instance weight (hessian) needed in a child»
max_delta_step – «Maximum delta step we allow each tree's weight estimation to be»

Пример:

$$L_i = \frac{1}{2}(y_i - a_i)^2$$

$$h_i = L_i'' = 1$$

– это как бы вес одного объекта

Параметры градиентного бустинга

Параметры формирования подвыборок

subsample / bagging_fraction	– какую часть объектов обучения использовать для построения одного дерева
colsample_bytree/ feature_fraction	– какую часть признаков использовать для построения одного дерева
colsample_bylevel	– какую часть признаков использовать для построения расщепления в уровне
colsample_bynode	– какую часть признаков использовать для построения расщепления в вершине
scale_pos_weight	– для сбалансирования позитивных и негативных весов
/ class_weight	– веса классов
// bootstrap_type	– тип бутстрепа (для Bayesian bootstrap есть bagging_temperature)

Параметры регуляризации

reg_alpha	– коэффициент L1-регуляризации
reg_lambda // l2_leaf_reg	– коэффициент L2-регуляризации

Параметры градиентного бустинга

Остальные

`Verbosity / silent`	– вывод информации при обучении
`n_jobs`	– число используемых потоков
`random_state`	– инициализация генератора псевдослучайных чисел
`missing`	– что обозначает пропуски
`importance_type`	– как вычислять важность (“gain”, “weight”, “cover”, “total_gain” or “total_cover”)
`num_parallel_tree`	– для режима RF
`/ subsample_for_bin`	– число объектов для бинов

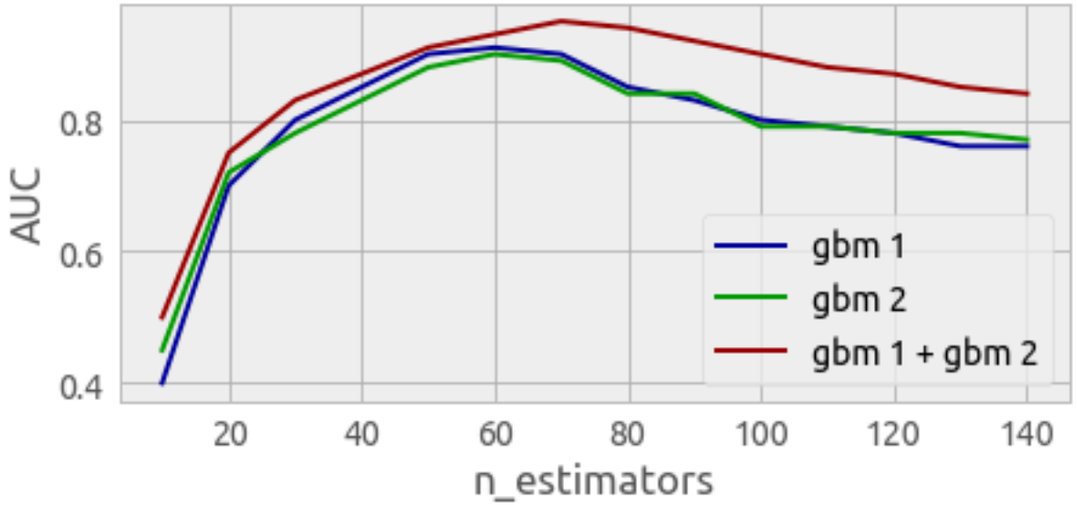
- CPU / GPU
- хранить модель в ОЗУ

Параметры градиентного бустинга

fit

`early_stopping_round (fit)`	– если на отложенном контроле заданная функция ошибки не уменьшается такое число итераций, обучение останавливается
`sample_weight`	– веса объектов
`eval_metric`	– метрика качества
`callbacks`	– какие функции вызывать после каждой итерации

Сумма бустингов



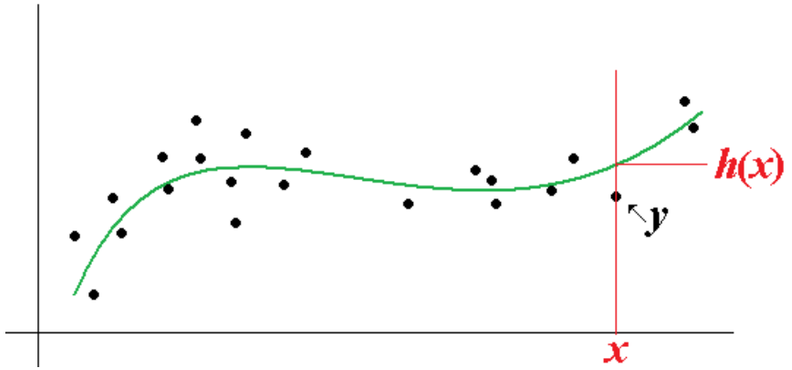
Качество может улучшиться,
но оптимальные параметры меняются!

GBM: Концепция чёрного ящика

```
model <- gbm(is_client_cancel~. , # название целевой переменной
T, # as.data.frame
distribution="gaussian", # распределение... лучше всего gaussian
n.trees=ntrees, # число деревьев (лучше больше, а потом выбрать)
shrinkage=0.07, # скорость сходимости
verbose=TRUE, # вывод сообщений
interaction.depth=12 # сложность модели
)
```

```
class sklearn.ensemble.GradientBoostingClassifier
(loss='deviance', # в классификации - логистическая регрессия или AdaBoost
learning_rate=0.1, , # скорость сходимости
n_estimators=100, # число деревьев
subsample=1.0,
min_samples_split=2,
min_samples_leaf=1,
min_weight_fraction_leaf=0.0,
max_depth=3, # глубина
max_features=None) # сколько признаков смотреть для расщепления
```

Что означает «распределение»



пусть ошибки распределены по нормальному закону

$$p(y \mid x) = \text{const} \cdot e^{-\frac{(y-h(x))^2}{2\sigma^2}}$$

метод максимального правдоподобия

$$\prod_i p(y_i \mid x_i) \sim \prod_i e^{-\frac{(y_i-h(x_i))^2}{2\sigma^2}} \to \max$$

$$\text{const} \cdot \sum_i (y_i - h(x_i))^2 \to \min$$

Что означает «распределение»

пусть ошибки ~ распределение Лапласа

$$p(y \mid x) = \text{const} \cdot e^{-\alpha|y-h(x)|}$$

метод максимального правдоподобия

$$\prod_i p(y_i \mid x_i) \sim \prod_i e^{-\alpha|y_i-h(x_i)|} \to \max$$

это эквивалентно минимизации такой ошибки

$$\text{const} \cdot \sum_i | y_i - h(x_i) | \to \min$$

Встроенные способы контроля

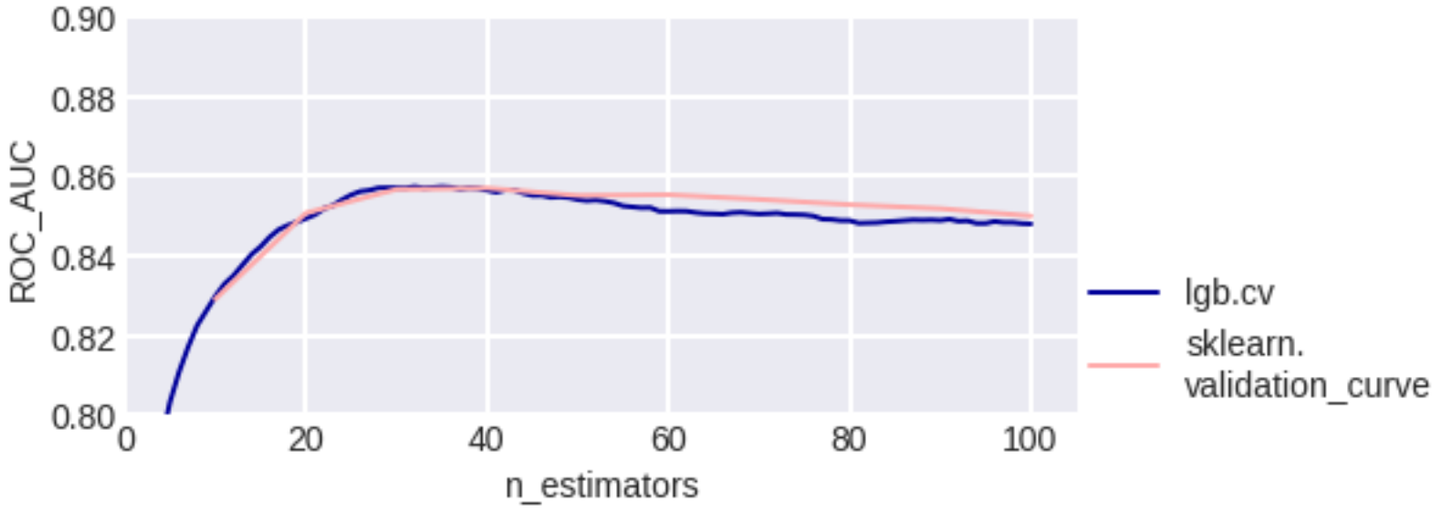
встроенный

```
params = {'objective': 'binary', 'learning_rate': 0.9,
          'num_leaves': 4, 'n_estimators': 100}
tmp = lgb.cv(params,
             train_set=lgb.Dataset(data, y_data),
             metrics=['auc', 'binary_logloss', 'rmse'])
```

универсальный

```
cv = KFold(n_splits=n_splits, shuffle=shuffle, random_state=random_state)
train_errors, test_errors = validation_curve(estimator,
                                             data, y_data,
                                             param_name=param_name,
                                             param_range=pars,
                                             cv=cv,
                                             scoring=scoring,
                                             n_jobs=-1)
```

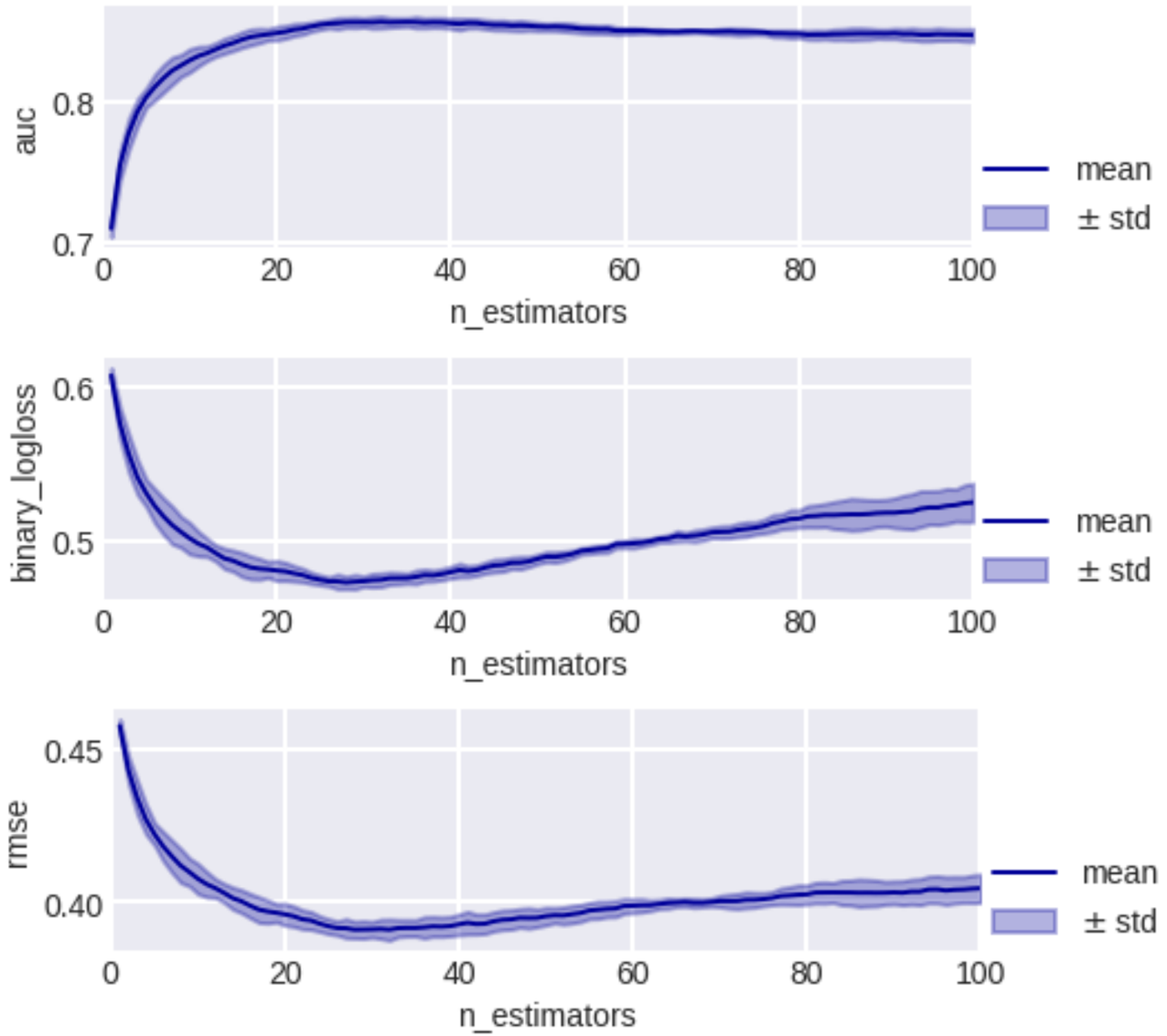
Встроенные способы контроля



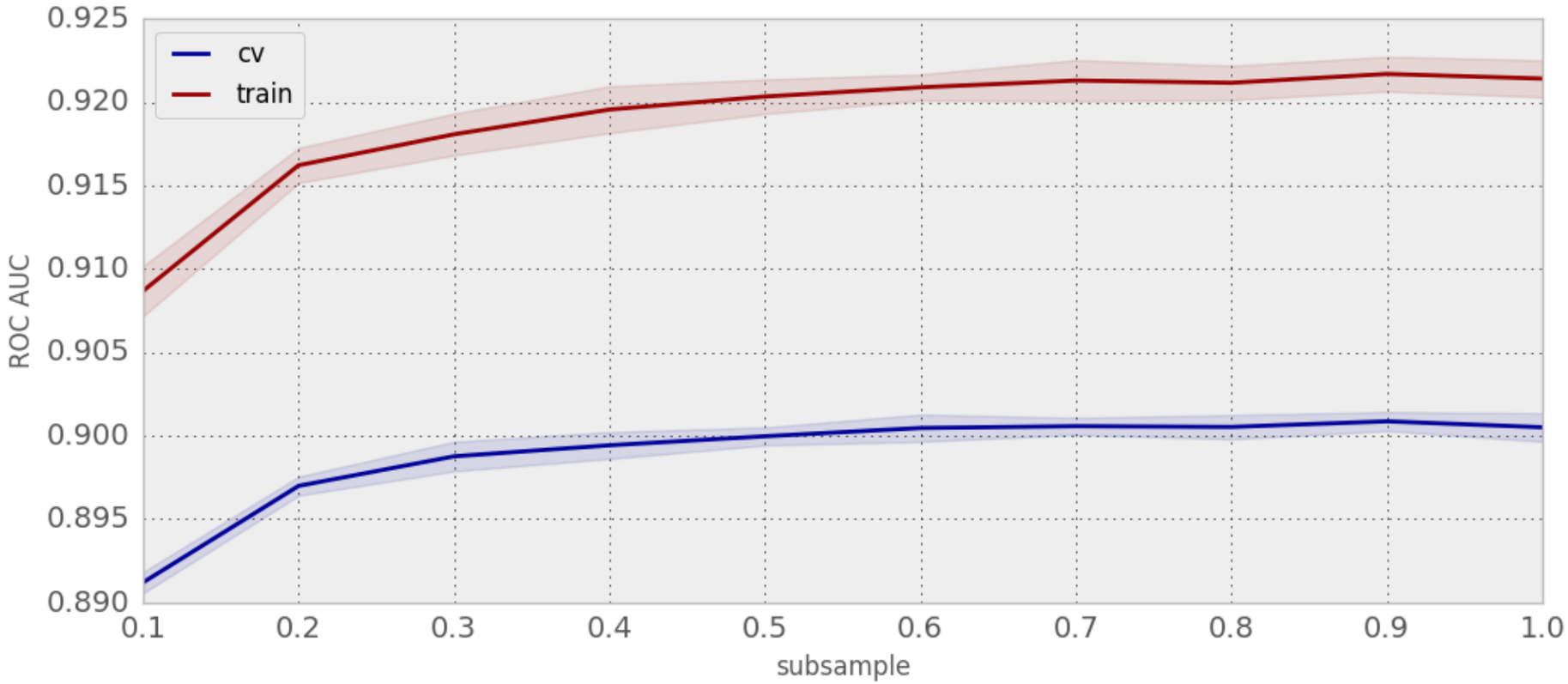
существенно быстрее 2 мин – 2 сек

- **не перестраивают ансамбль с начала**
- **нечестный контроль – упрощённый способ выбора порогов (хитрость!)**
- **сразу получаем с шагом 1**
- **результаты похожи**

Разные метрики качества



Объём выборки `subsample` (ed Бозон)

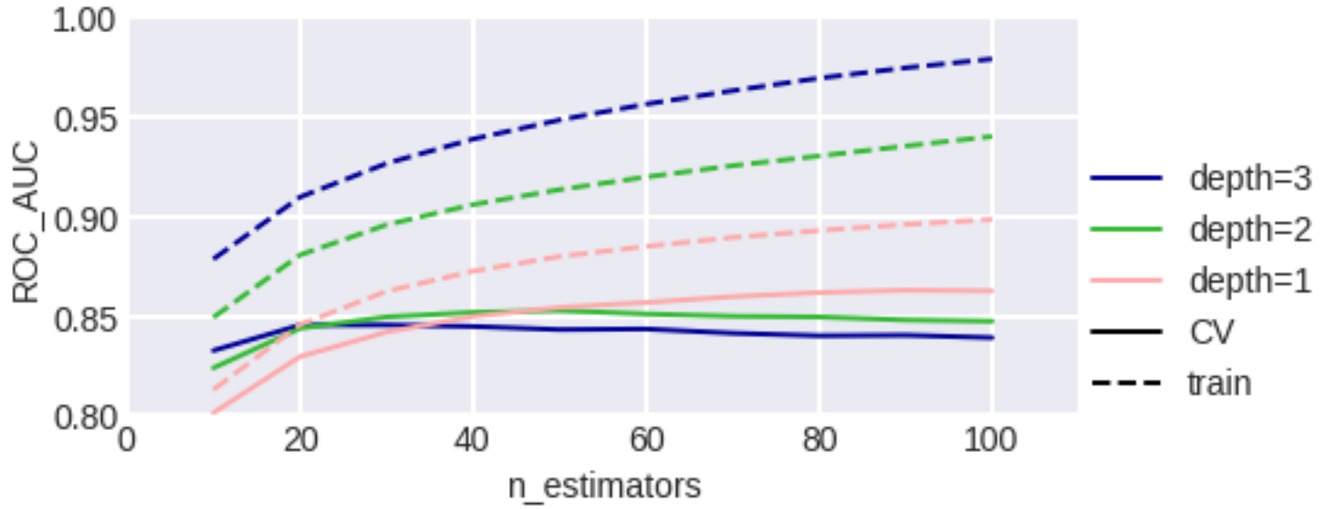


Опять, больше – лучше

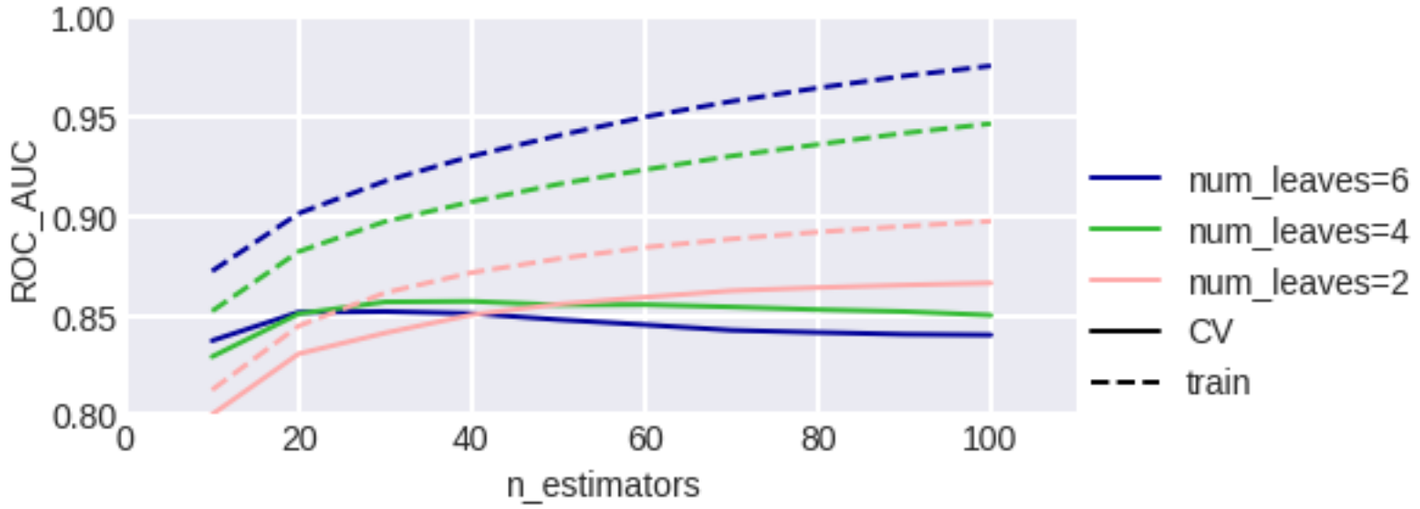
(в XGBoost это не всегда так)

Число деревьев: n_estimators

scikit-learn



lightgbm

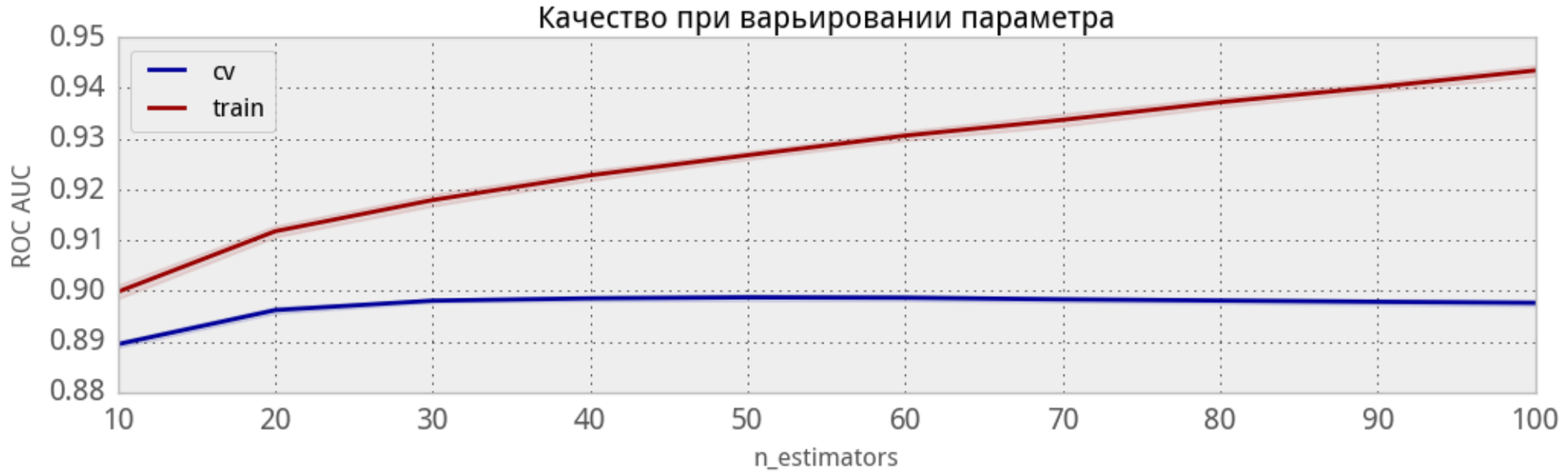


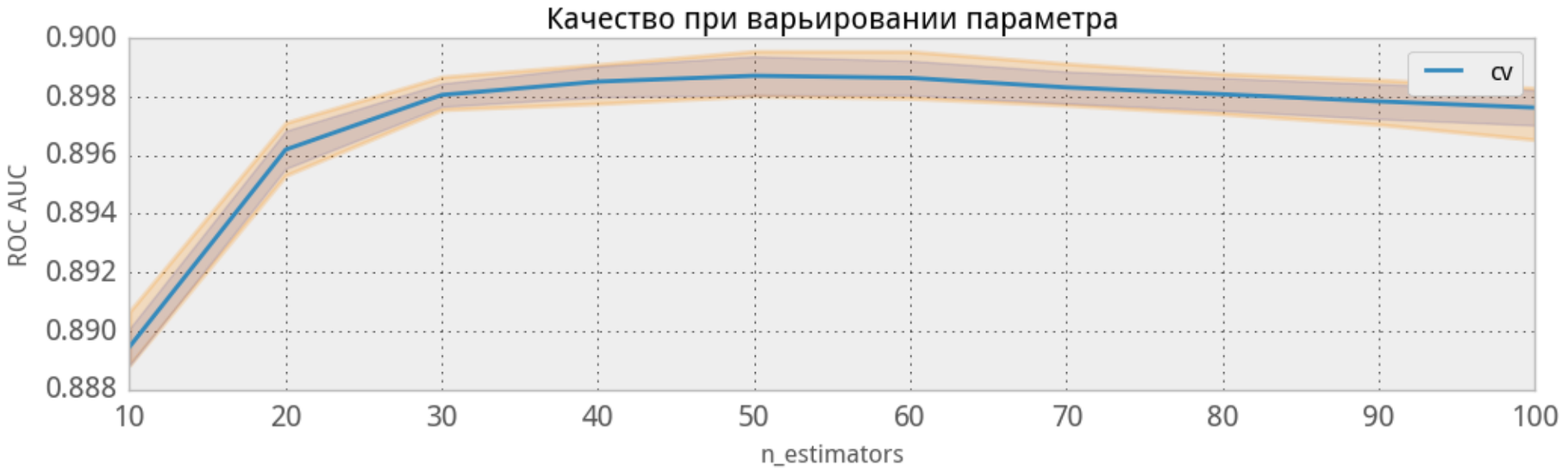
Число деревьев: `n_estimators`

Здесь уже нет логики «чем больше, тем лучше»

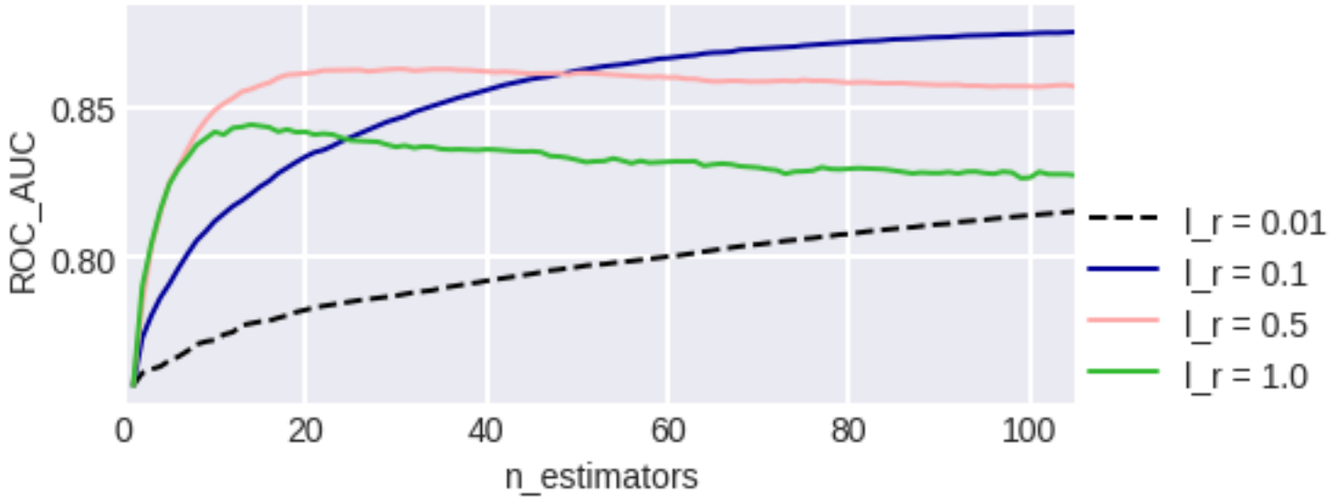
Для разной глубины – разное оптимальное число деревьев

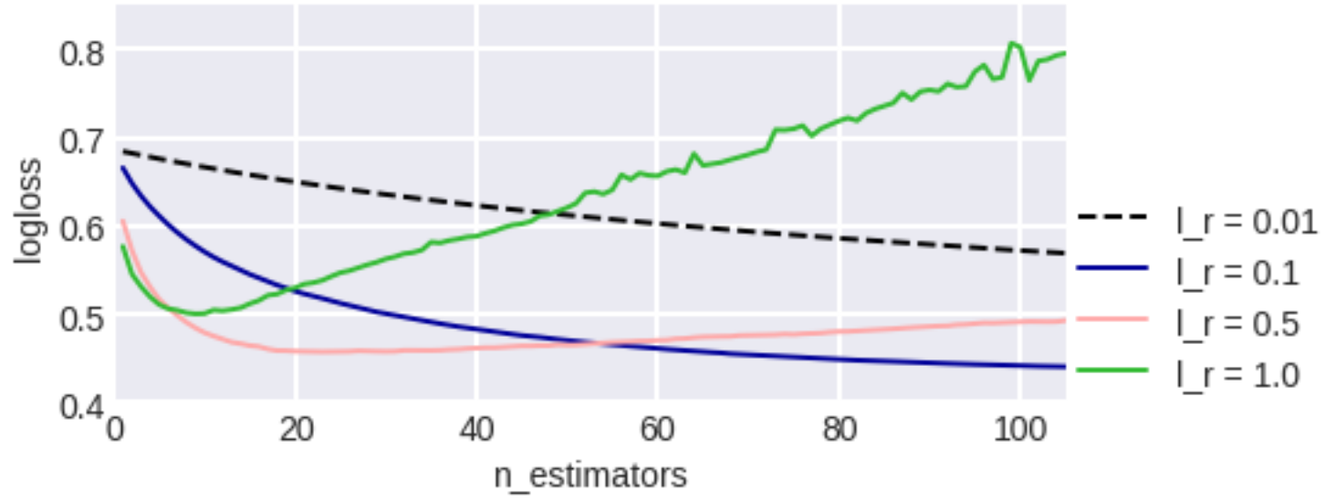
Число деревьев: n_estimators (ed Бозон)





Темп обучения `learning_rate`

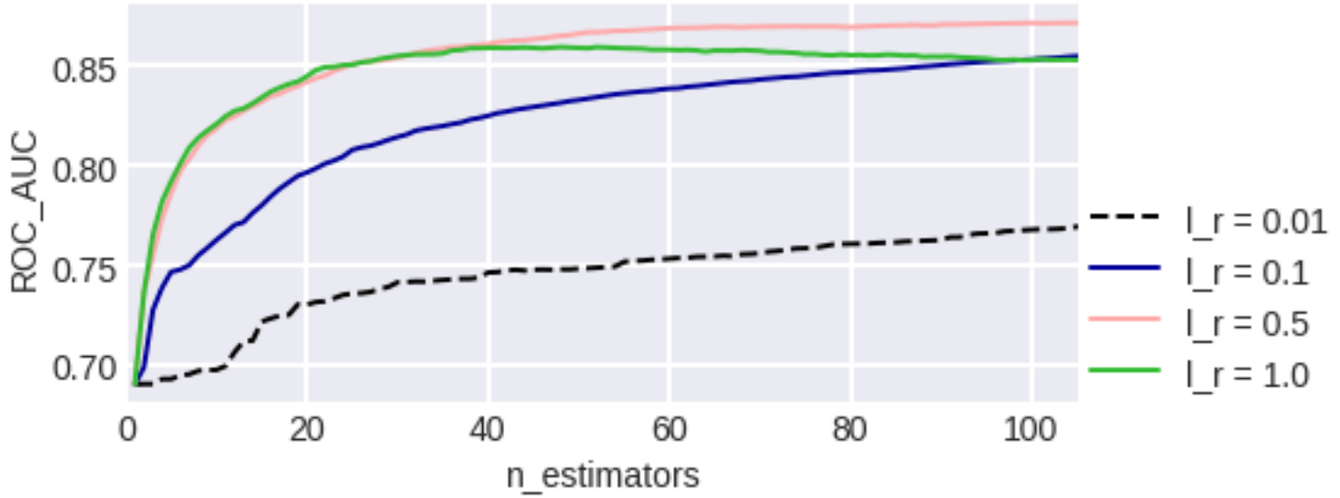




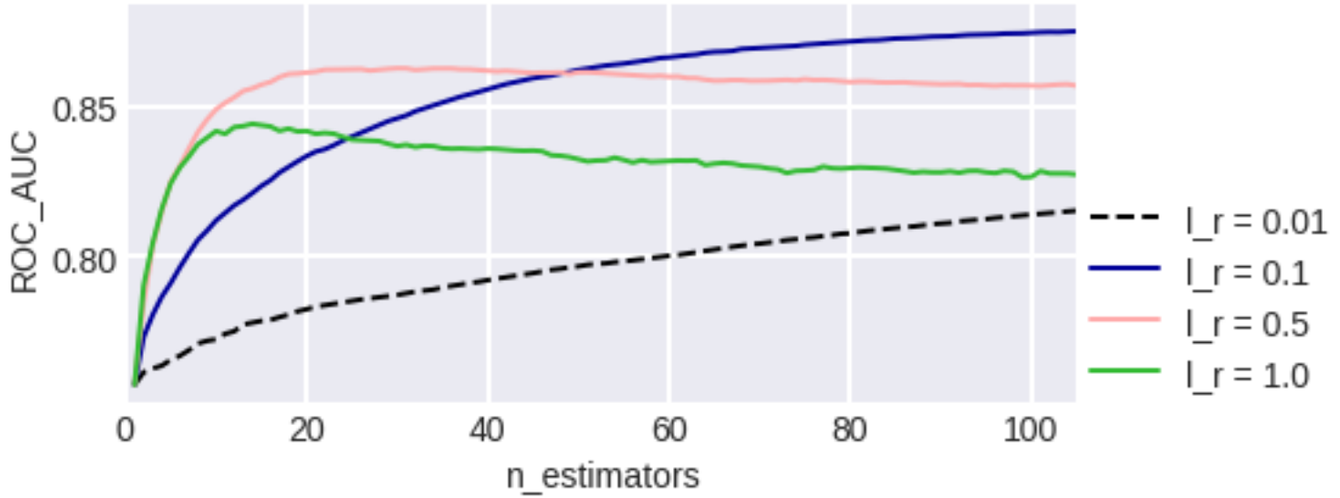
Пример малого, нормального и большого темпов

Темп обучения `learning_rate`

num_leaves = 3

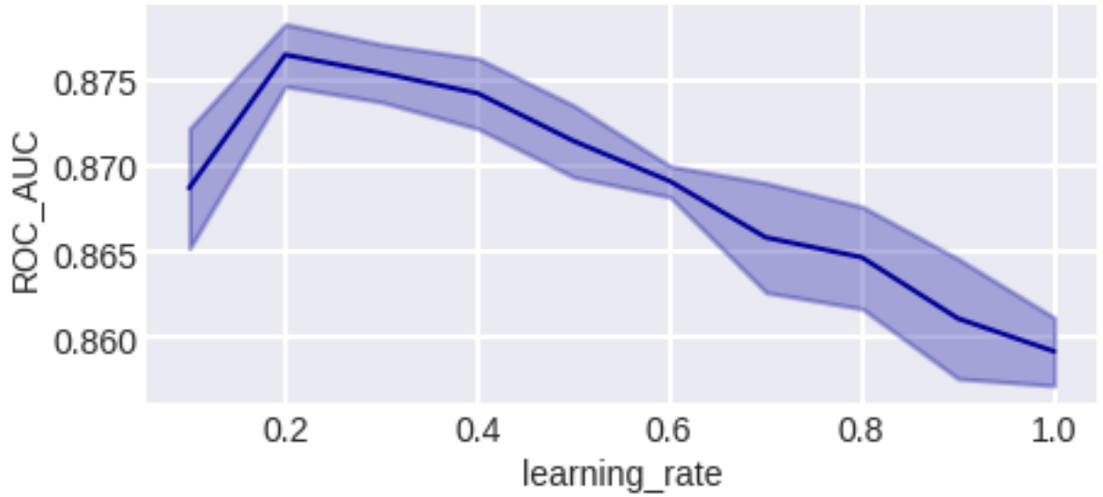


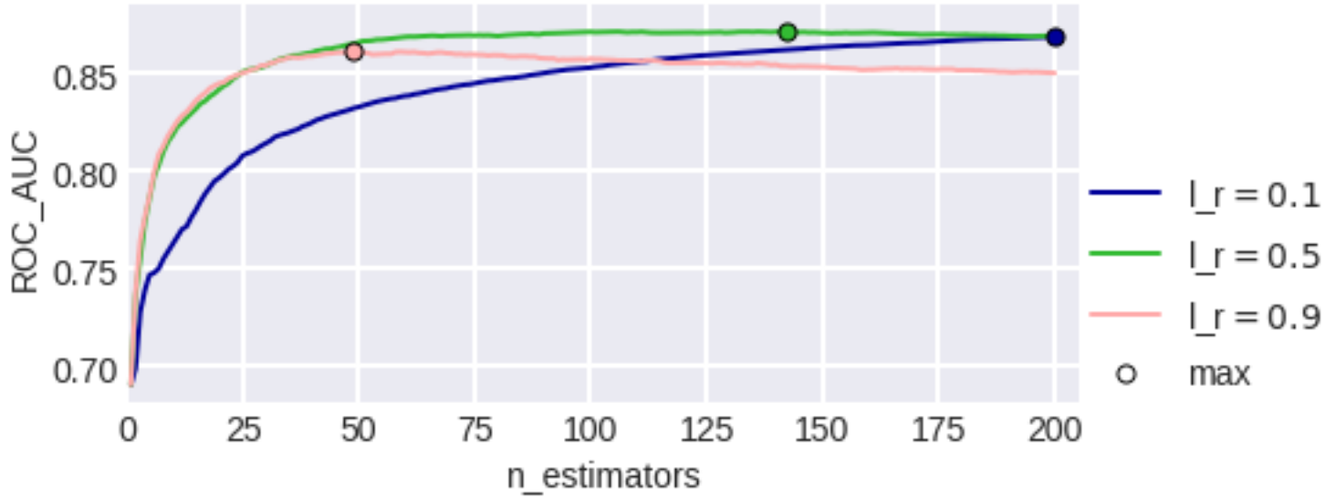
num_leaves = 10



верно ли, что для малых деревьев нет большого темпа?

Темп обучения `learning_rate`





Темп обучения `learning_rate`

Нет логики «уменьшили темп в 2 раза – число деревьев надо увеличить в 2 раза»!

Есть стратегия – сделать очень маленький темп и очень много деревьев
(но для настройки других параметров не годится)

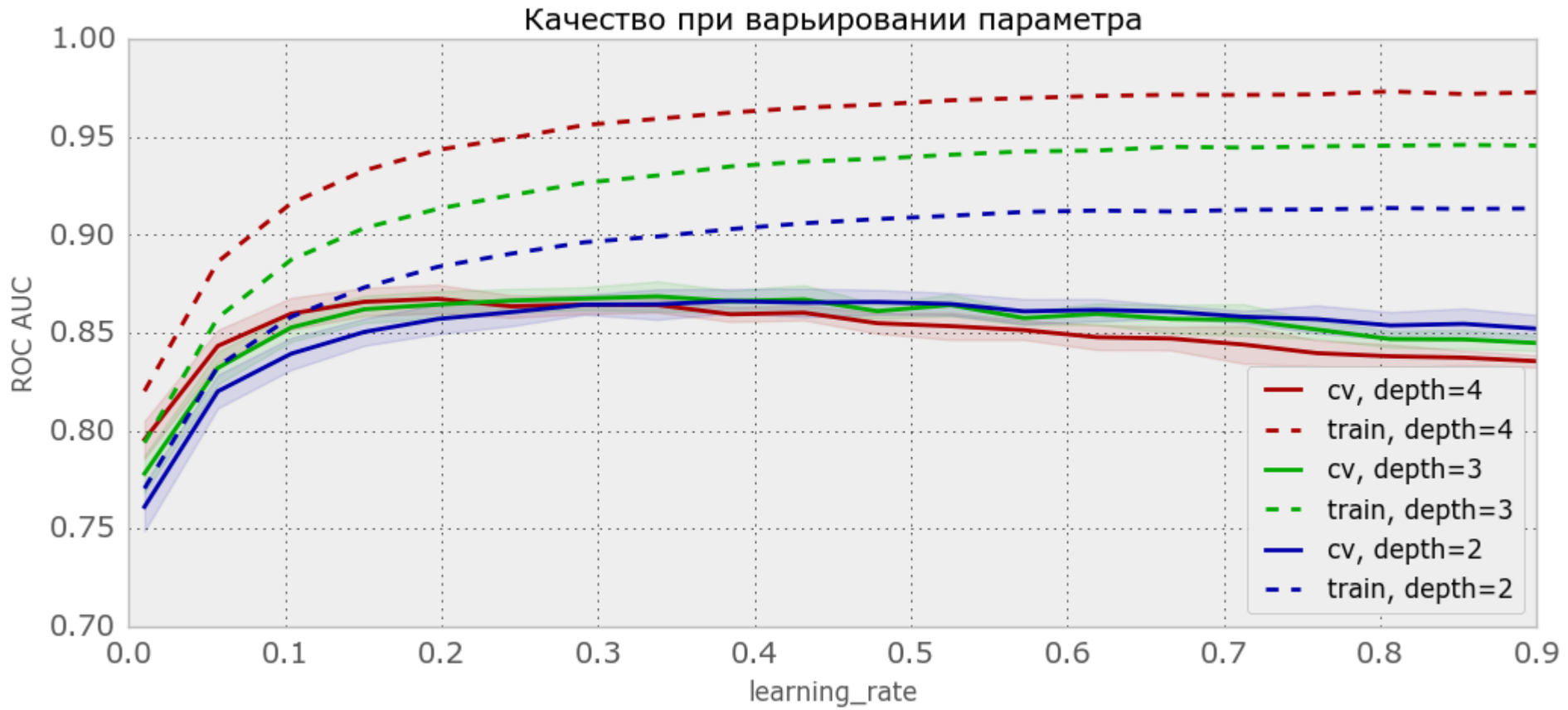
Совет:

- **зафиксируйте достаточно большое число деревьев, которое ещё можно быстро построить**

- **настройте `learning_rate`**

- **настраивайте другие параметры (первым делом – глубину), но помните, что оптимальный темп может поменяться!**

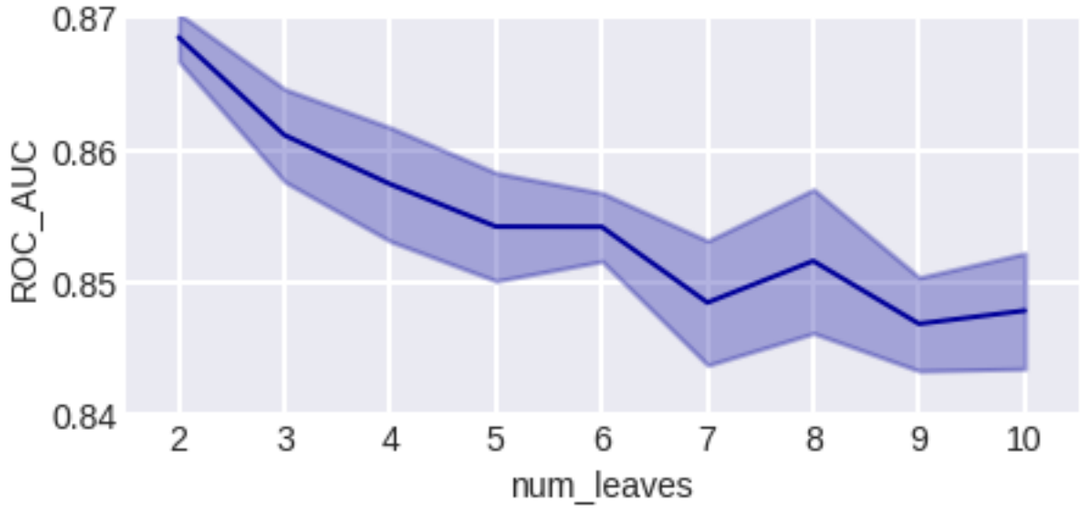
Темп обучения `learning_rate`



Сложность деревьев: число листьев

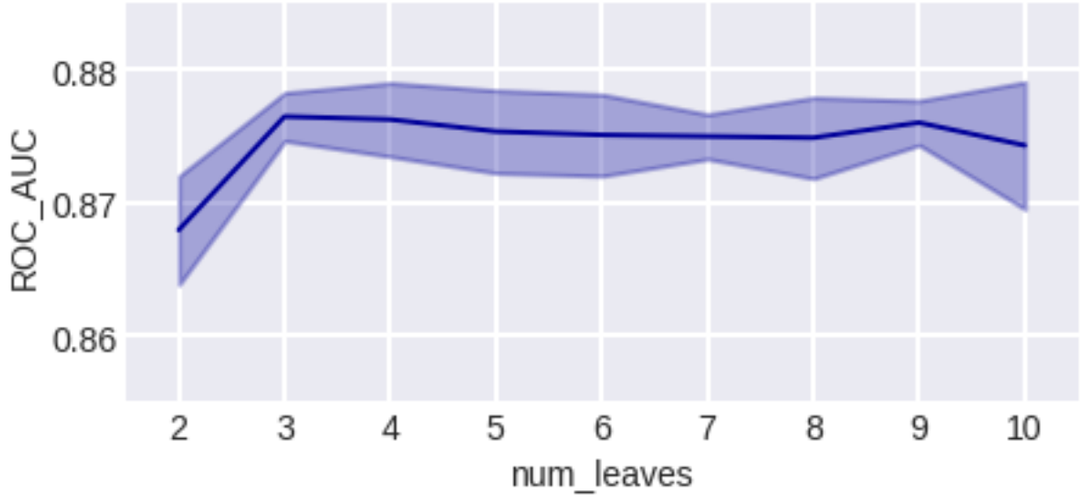
learning_rate = 0.9

деревьев построено достаточно



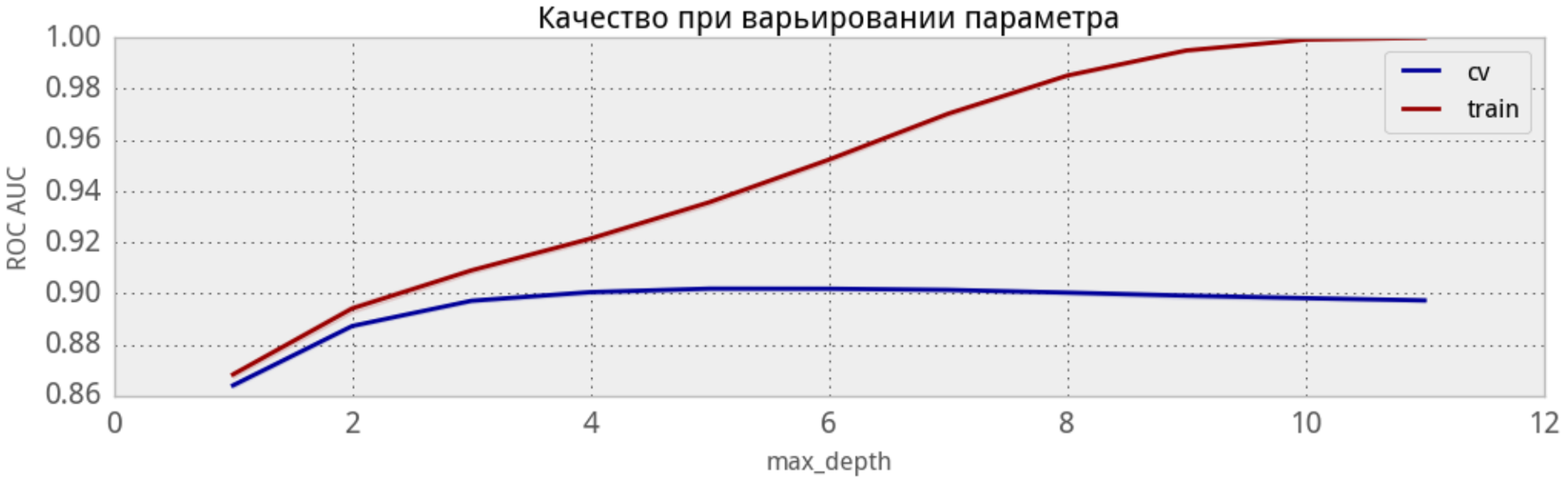
learning_rate = 0.2

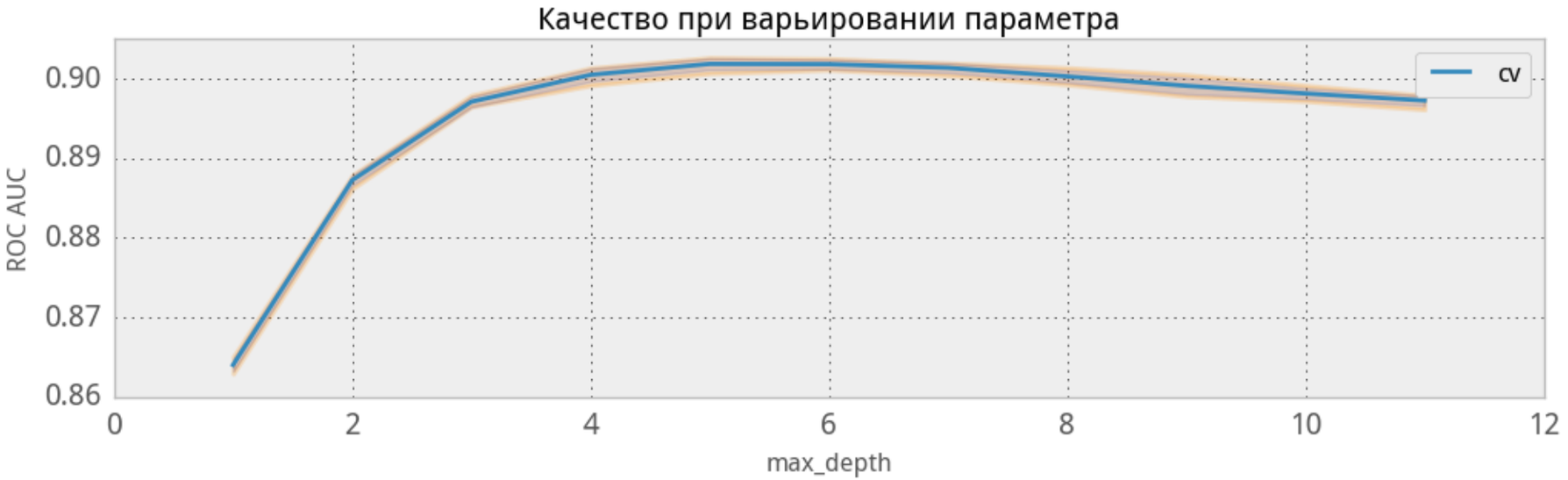
та же задача!



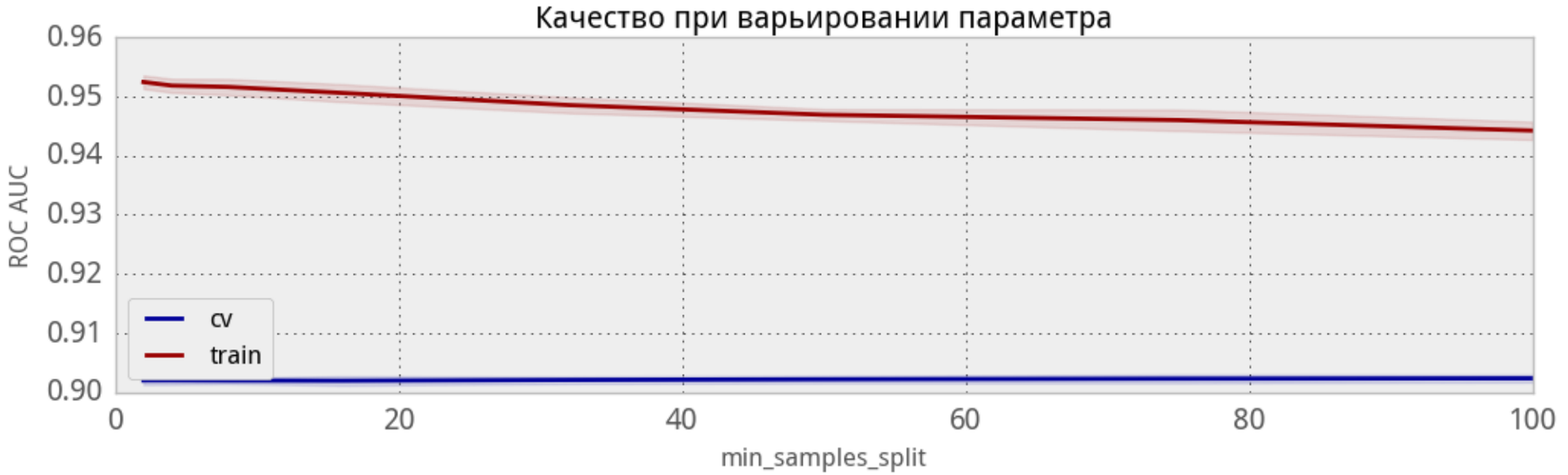
есть тонкость: для каждой сложности своё оптимальное число деревьев

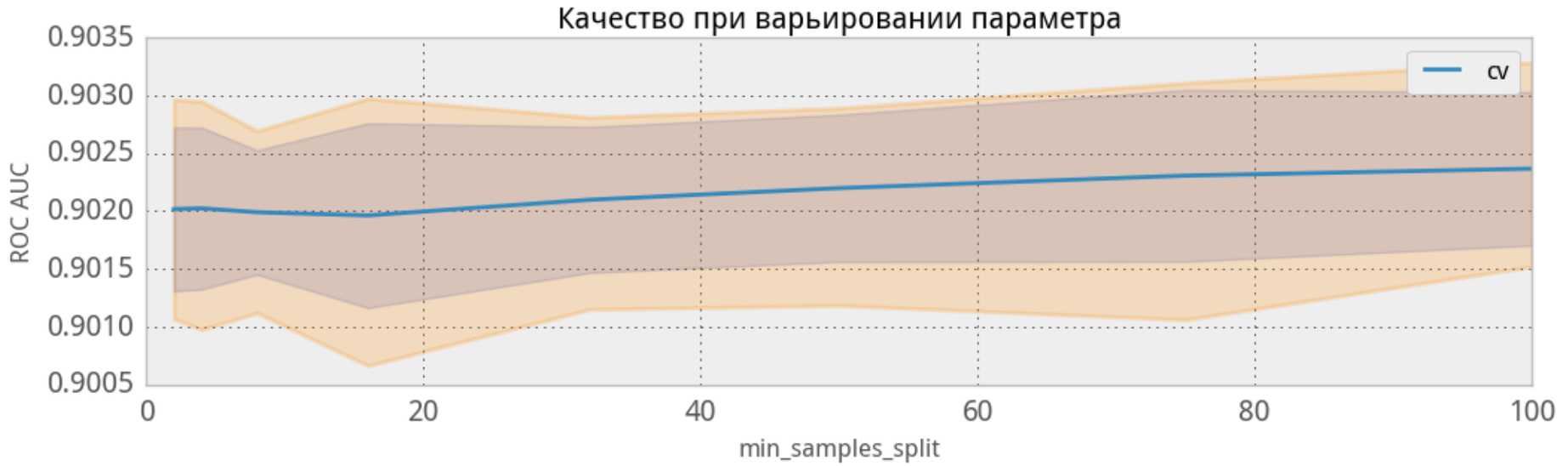
Сложность деревьев: глубина





Ограничение на расщепления / листья





Ограничение на расщепления / листья

Здесь могут быть большие оптимальные значения (10 – 50 – 1000),
но параметры менее значимые, чем другие…

Деревья для GBM

Малой глубины (3 – 6).
– смещённые (high bias), разброс ниже, чем в глубоких

- **смещение как раз устраняется бустингом**
- **модель не должна переобучаться $\Rightarrow$ простая**
- **быстрее строить**

Здесь есть понятие оптимальной глубины!

Тонкости

Бутстрепа обычно нет…

но есть случайные подвыборки и подмножества признаков

Совет

- выбрать критерий расщепления (вид бустинга) из логики

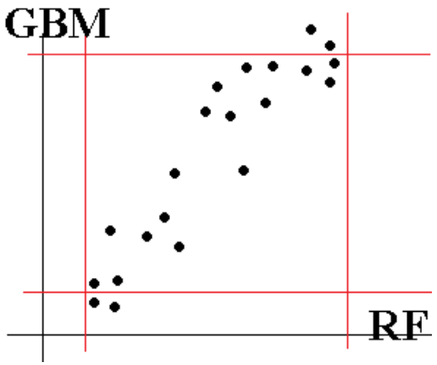
- выбрать число деревьев и темп обучения
(это согласованные параметры)
для настройки можно немного деревьев

- настроить сложность деревьев (варьируя их число) – как и в RF самый важный параметр

- увеличить число деревьев, взять маленький темп обучения

- использовать сумму нескольких gbm

Важно



Значения gbm могут выходить за пределы отрезка!

Вообще говоря, не важно,
как их вернуть обратно…

Приложение GBM: задача скоринга (TKS)

tcs_customer_id	bureau_cd	bki_request_date	inf_confirm_date	type	status	open_date	final_pmt_date	fact_close_date	credit_limit	currency	outstanding	next_pmt	curr_balar
1	2	12Aug2011	20Jul2011	99	00	13May2011	11May2012		28967	RUB	24606,00000	2743,00000	
1	1	12Aug2011	18Feb2009	99	13	27Feb2008	26Feb2009	26Feb2009	30000	RUB	0,00000		
1	1	12Aug2011	21Apr2009	99	13	28Jun2007	30Jun2008	20Apr2009	19421	RUB	0,00000		
1	1	12Aug2011	18Aug2009	9	13	15Jul2008	17Aug2009	17Aug2009	11858	RUB	0,00000		
1	1	12Aug2011	06Sep2010	99	13	09May2009	10May2010	08Sep2010	19691	RUB	0,00000		
1	1	12Aug2011	28Jul2011	7	52	07Sep2010	07Sep2040		10000	RUB			
1	1	12Aug2011	01Aug2011	9	00	31Aug2010	31Aug2015		169000	RUB			
1	1	12Aug2011	03Aug2011	9	00	04Mar2009	03Mar2014		300000	RUB			
1	3	12Aug2011	09Jul2008	9	00	28Jun2007	30Jun2008		19421	RUB	1761,00000		198
1	3	12Aug2011	19Sep2008	9	00	27Feb2008	26Feb2009		30000	RUB	15517,00000		163
1	3	12Aug2011	14Sep2010	9	13	09May2009	10May2010	06Sep2010	19691	RUB	0,00000		
1	3	12Aug2011	11Jul2011	9	00	31Aug2010	31Aug2015		169000	RUB		0,00000	433

Решение = GBM + RF + Линейная регрессия

Приложение GBM: задача скоринга (TKS)

Name	Description	Type
TCS_CUSTOMER_ID	Идентификатор клиента	ID
BUREAU_CD	Код бюро, из которого получен счет	numeric
BKI_REQUEST_DATE	Дата, в которую был сделан запрос в бюро	date
CURRENCY	Валюта договора (ISO буквенный код валюты)	string
RELATIONSHIP	Тип отношения к договору	string
	1 - Физическое лицо	
	2 - Дополнительная карта/Авторизованный пользователь	
	4 - Совместный	
	5 - Поручитель	
	9 - Юридическое лицо	
OPEN_DATE	Дата открытия договора	date
FINAL_PMT_DATE	Дата финального платежа (плановая)	date
TYPE	Код типа договора	string
	1 – Кредит на автомобиль	
	4 – Лизинг. Срочные платежи за наем/пользование транспортным средством, предприятием или оборудованием и т.п.	
	6 – Ипотека – ссудные счета, имеющие отношение к домам, квартирам и прочей недвижимости. Ссуда выплачивается циклично согласно договоренности до тех пор, пока она не будет полностью выплачена или возобновлена.	
	7 – Кредитная карта	
	9 – Потребительский кредит	
	10 – Кредит на развитие бизнеса	
	11 – Кредит на пополнение оборотных средств	
	12 – Кредит на покупку оборудования	
	13 – Кредит на строительство недвижимости	
	14 – Кредит на покупку акций (например, маржинальное кредитование)	
	99 – Другой	
PMT_STRING_84M	Дисциплина (своевременность) платежей. Строка составляется из кодов состояний счета на моменты передачи банком данных по счету в бюро, первый символ - состояние на дату PMT_STRING_START, далее последовательно в порядке убывания дат.	string
	0 – Новый, оценка невозможна	
	X – Нет информации	
	1 – Оплата без просрочек	
	A – Просрочка от 1 до 29 дней	

Приложение GBM: предсказание правильности ответов студентов на вопросы тестов

Разработать алгоритм,
который предсказывает правильность ответа
на вопросы теста.

Зачем?

для рекомендательной системы (алгоритм решает за студента тест и сообщает ему «потенциально неприятные для него» вопросы).

GMAT, SAT, ACT

Победитель – LibFM

#	Team Name	$5,000 • 241 teams Score	Entries
1	Steffen Rendle *	0.24598	16
2	Alexander D'yakonov *	0.24729	38
3	ekla *	0.24745	87
4	PlanetThanet & Birutas	0.24772	51

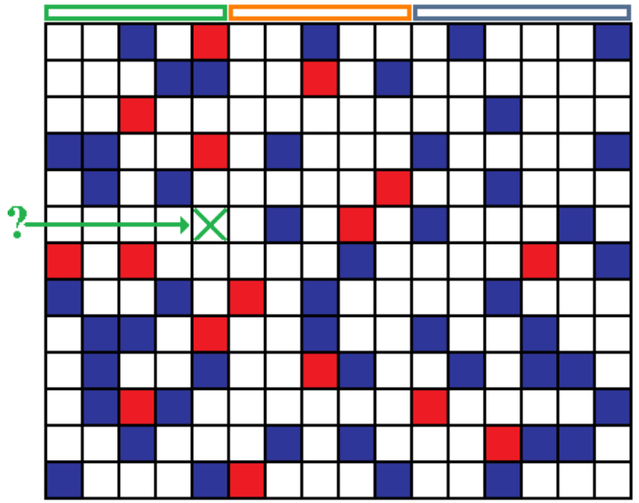
Обычно:

- **контекстная рекомендация**
- **коллаборативная фильтрация**

Наша идея:

свести задачу о рекомендациях

к задаче классификации (регрессии)



пара «студент–вопрос» ~ признаковое описание

генерация кучи признаков

Примеры признаков

Пусть ответы студента:

correct, incorrect, correct, correct, correct, incorrect

$$\text{IQ} \sim \frac{+1-1+1+1+1-1}{6}$$

$$\text{weighted IQ} \sim \frac{+1w_1 - 1w_2 + 1w_3 + 1w_4 + 1w_5 - 1w_6}{w_1 + w_2 + w_3 + w_4 + w_5 + w_6}$$

веса измеряют «похожесть вопросов»

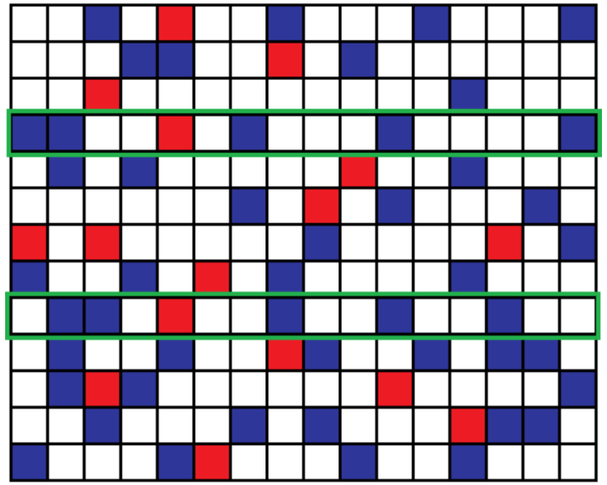
$$w_j = \frac{2}{1+|t-t_j|^{0.3}} - 1 \quad \text{или} \quad w_j = 1 - \sqrt{|t-t_j|}$$

t_j **– время ответа на j-й вопрос,**

t **– время ответа на этот вопрос.**

Ещё веса:

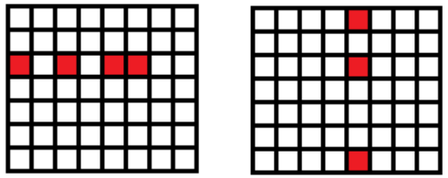
Корреляция столбцов матрицы «студент–ответ»



Аналогично:

Признак «сложность вопроса»

(здесь усредняются ответы на данный вопрос)



Простые признаки:

- время ответа
- 1/(число ответов всего)

SVD-признаки

Восстановление матрицы с помощью SVD-преобразования
(даже по подматрице)

Решение

gbm + glm + NN (CLOP)

Опять: хорошо «смешиваются» разные алгоритмы…
Как настраивать – чуть позже…

Литература

A. Natekin, A. Knoll Gradient boosting machines, a tutorial // Front Neurorobot. 2013; 7: 21.
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3885826/

все статьи по XGBoost, LightGBM, CatBoost