Системы искусственного интеллекта

Лекция 9

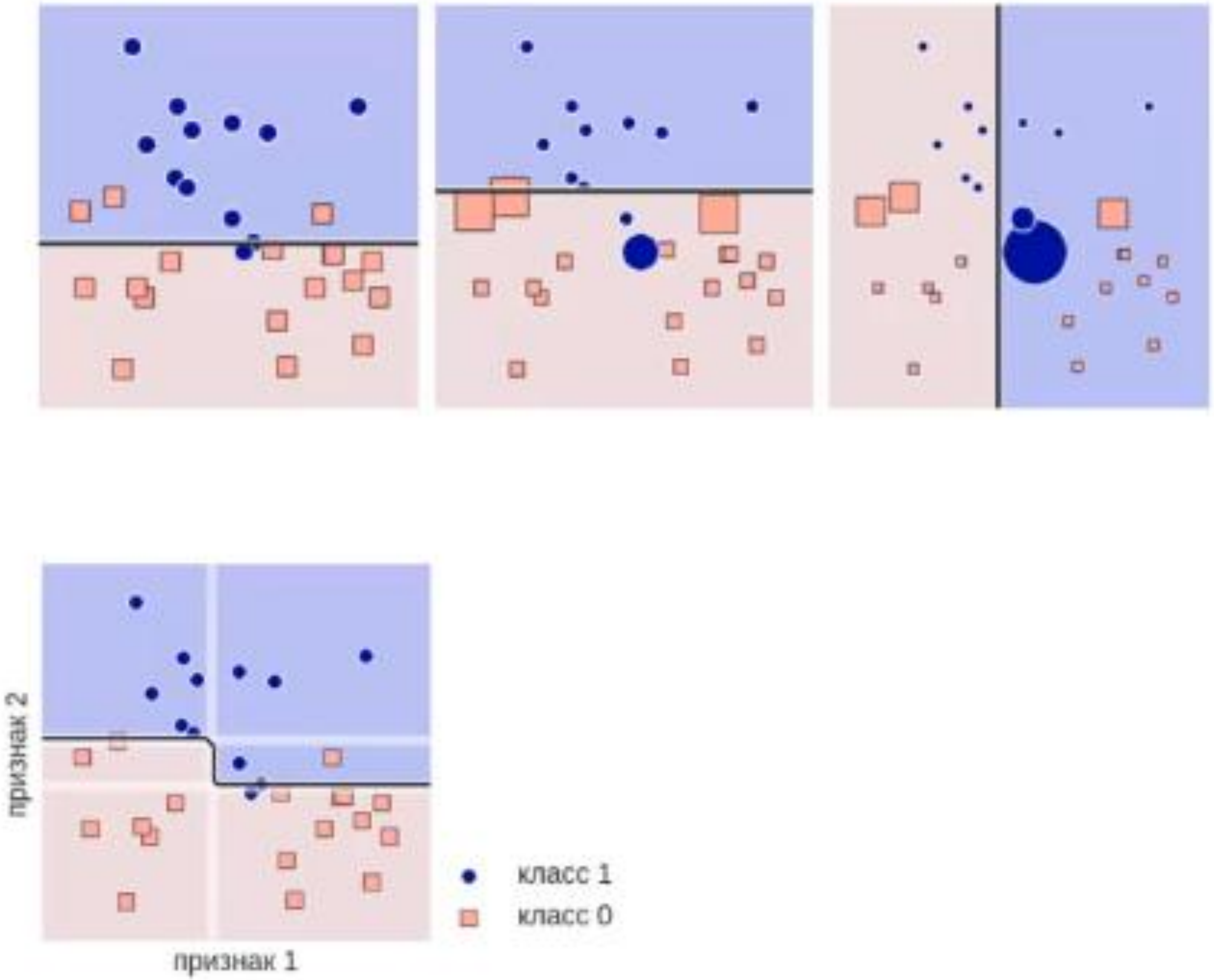
Градиентный бустинг над деревьями

Запорожцев Иван Федорович
zaporozhtsev.if.work@gmail.com

Бустинг

Главная идея

Базовые алгоритмы строятся не независимо, каждый следующий мы строим так, чтобы он исправлял ошибки предыдущих и повышал качество всего ансамбля



Основной принцип реализации бустинга

Forward stagewise additive modeling (FSAM)

Задача регрессии – $(x_i, y_i)_{i=1}^m$

Функция ошибки – $L(y,a)$

Уже есть алгоритм $a(x)$, строим $b(x)$:

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

т.е. в идеале

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

Идея градиентного бустинга

FSAM + минимизация в случае дифференцируемой функции ошибки

Задача регрессии с выборкой $(x_i, y_i)_{i=1}^m$

Дифференцируемая функция ошибки $L(y, a)$

Уже есть алгоритм $a(x)$, строим $b(x)$:

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

Грубо говоря «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

Проблемы

$$w^{(t+1)} = w^{(t)} - \eta \nabla L(w^{(t)})$$

$\eta > 0$ – **шаг / темп обучения** (step size / learning rate)

Хотим $\lim\limits_{t \to \infty} w^{(t)} = \arg\min\limits_{w} L(w)$

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

может не решаться аналитически

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

Функция $F(b_1, \dots, b_m)$ убывает в направлении антиградиента, поэтому выгодно считать

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

→ 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 получаем сумму алгоритмов...

Вот почему называется градиентный бустинг

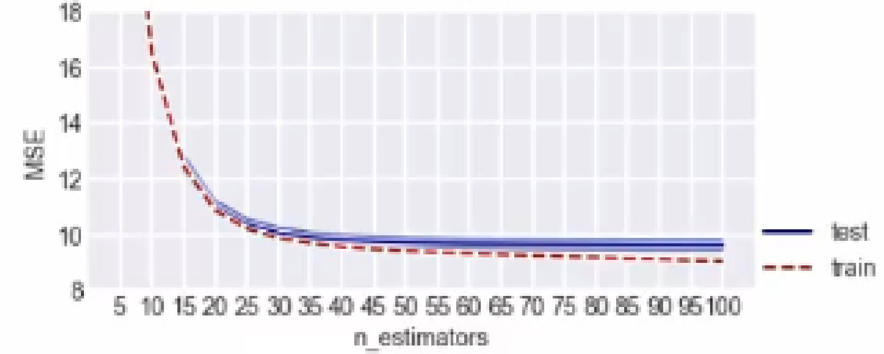
Частный случай: регрессия с СКО

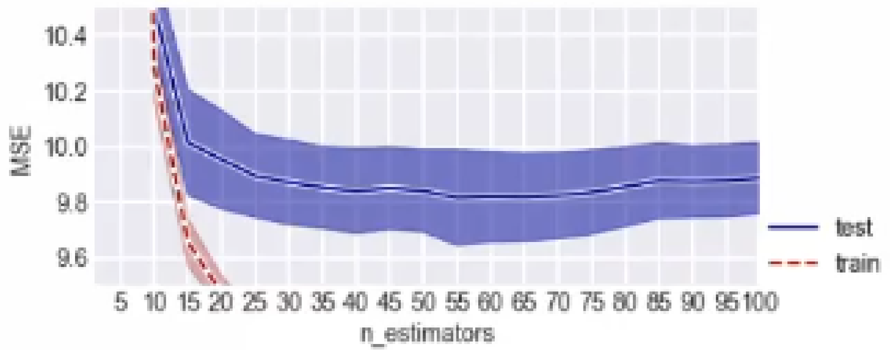
$$L(y,a) = \frac{1}{2}(y-a)^2, \quad L'(y,a) = -(y-a)$$

Задача для настройки следующего алгоритма

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

Здесь совсем логично:
настраиваемся как раз на невязку!





Частный случай: классификация на два класса

BinomialBoost –
логистическая функция ошибки:

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

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

Функция ошибки типа Adaboost:

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

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

Итерация градиентного бустинга

Как решать задачу с выборкой

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

Любым простым методом!
Мы уже настраиваемся
на нужную функцию ошибки.

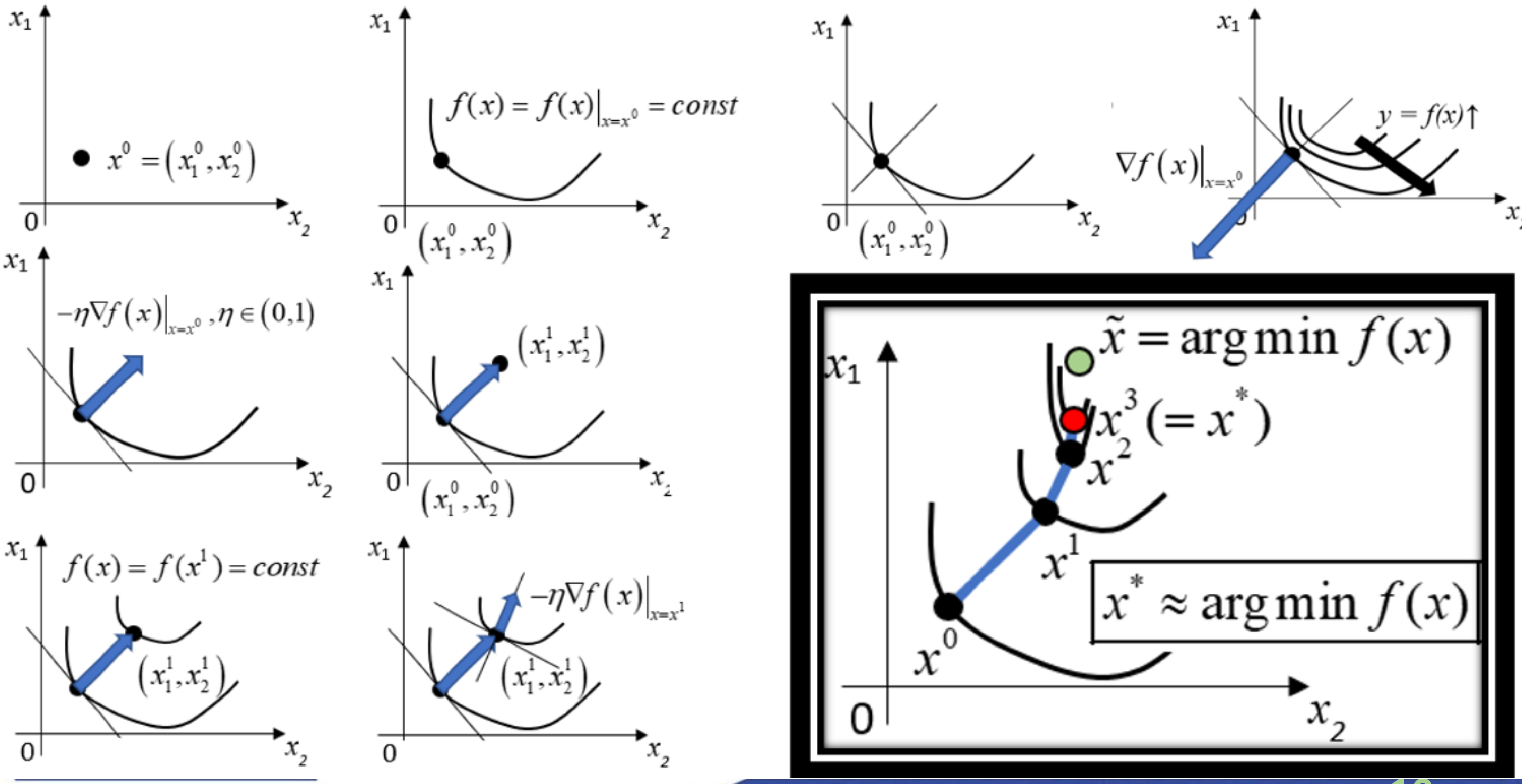
Проблемы:

Шаг в сторону антиградиента

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

Дальше решение проблем

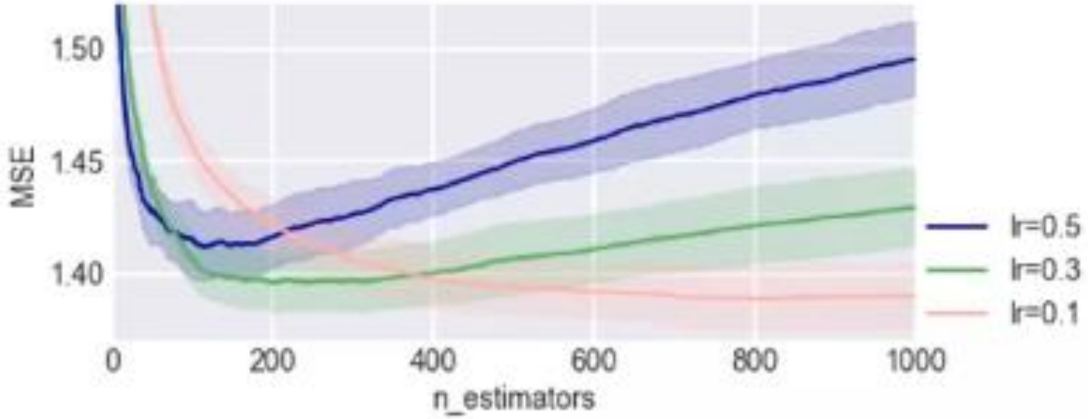
Диаграмма последовательных приближений и линий уровня



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)

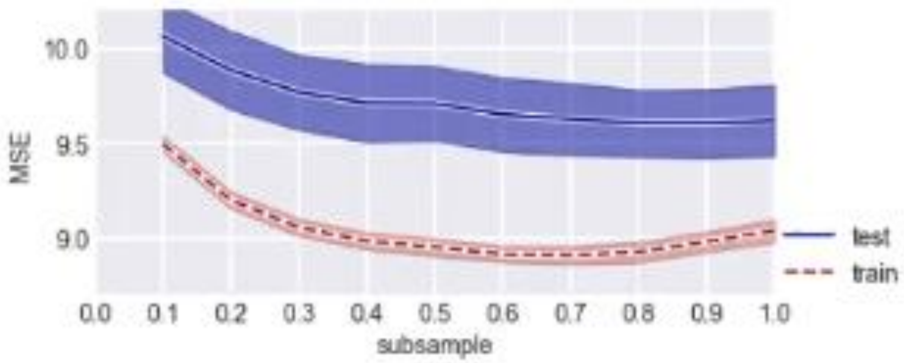
> **Чем меньше скорость, тем больше итераций надо**

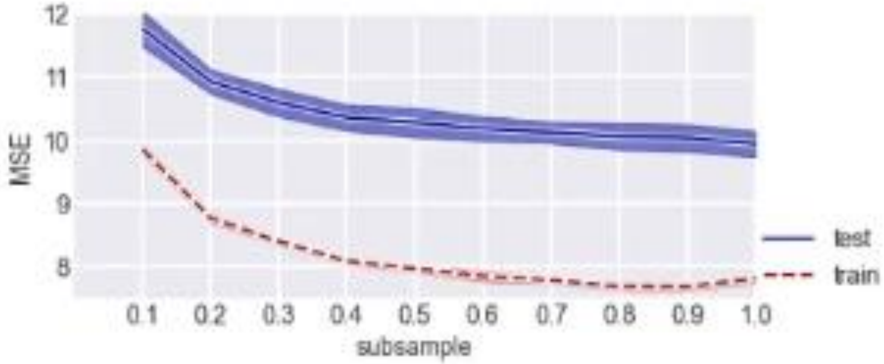
Стохастический градиентный бустинг

Stochastic gradient boosting

Идея бэгинга Breймана:

Bag fraction ~ берём часть всей выборки ⟶ бутстрепа обычно нет...



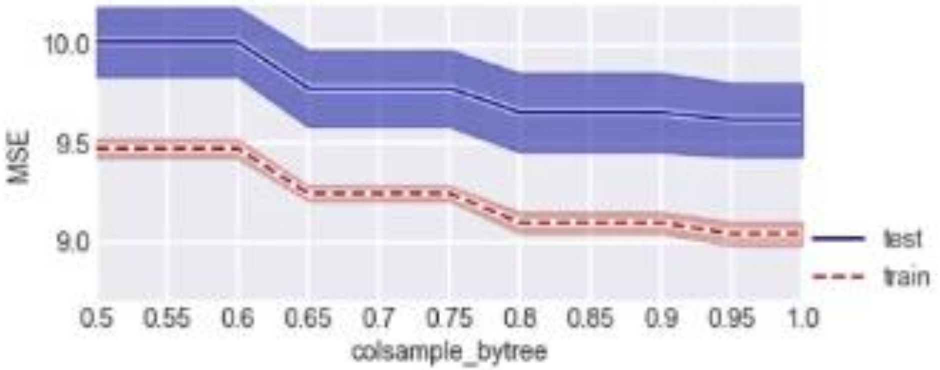


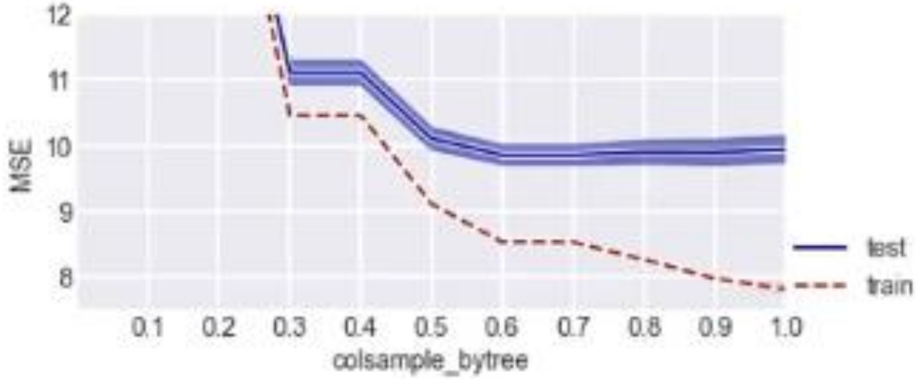
- М.б. лучше качество («регуляризация»)
- Быстрее

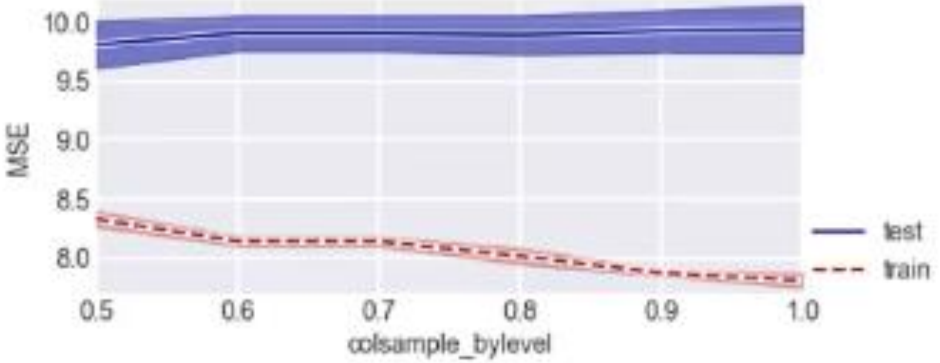
- Аналог обучения по минибатчам
- Можно вычислить ООВ-ошибки (а можно ли?)

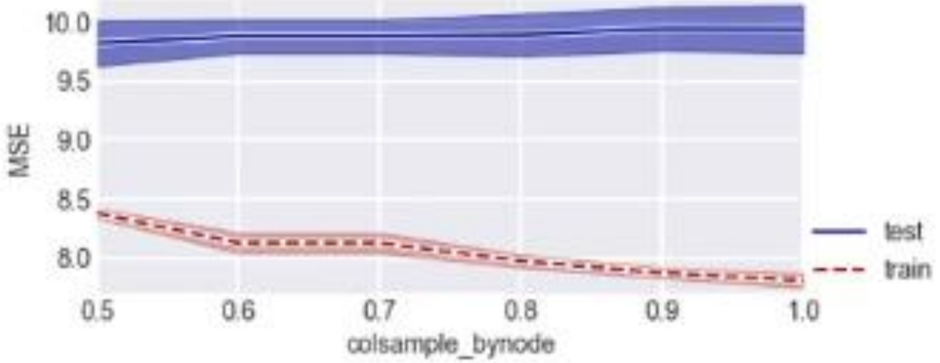
Column / Feature Subsampling for Regularization

Аналогичная идея с признаками





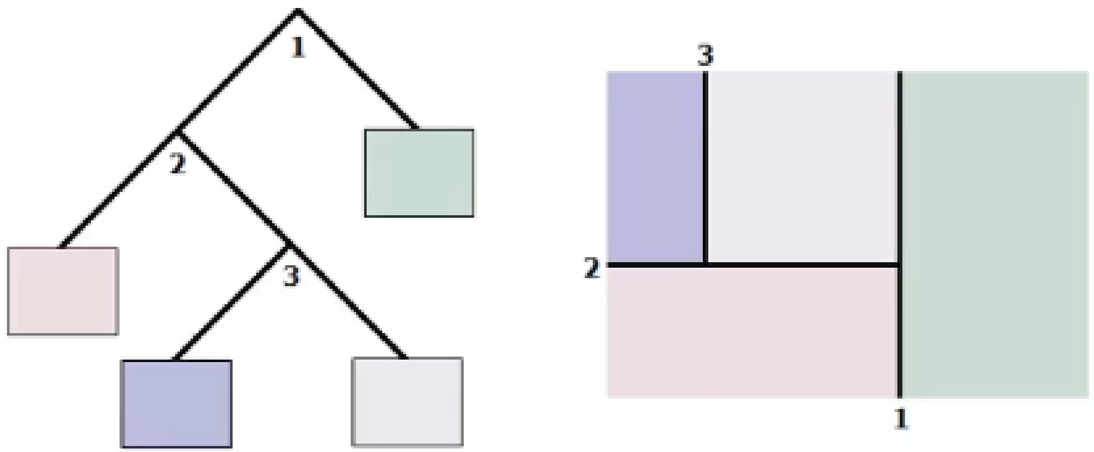




TreeBoost – градиентный бустинг над деревьями

Решающее дерево:

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



TreeBoost – градиентный бустинг над деревьями

Наша основная задача

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

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

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

Заметим, что

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

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

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

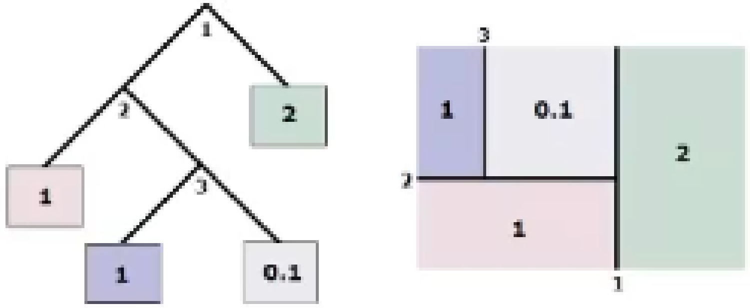
Решающее дерево:

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

Строим алгоритм в виде

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

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



$$\Phi = \ldots + \gamma 4 + \lambda \frac{1}{2}(1 + 1 + 0.01 + 4)$$

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:

$$\beta_j = -\frac{\sum_{x_i \in X_j} g_i}{\sum_{x_i \in X_j} h_i + \lambda}$$

Продвинутые методы оптимизации

Энтропийный → $H(R) = -\sum_j p_j \log_2 p_j$

Джини → $H(R) = \sum_j p_j(1-p_j) = 1 - \sum_j p_j^2$

Минимальное значение (при фиксированной структуре дерева)

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

Можно использовать при построении дерева для его оценки:

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

История продвинутых методов

Современные реализации

01 **March, 2014**
XGBoost был исследовательским проектом Tianqi Chen, использовался на соревновании Higgs Boson MLC

02 **Jan, 2017**
LightGBM представлена компанией Microsoft

03 **April, 2017**
CatBoost представлена компанией Yandex

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

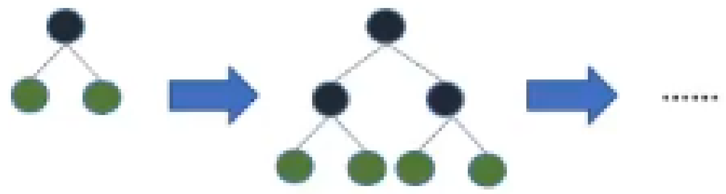
Особенности реализаций продвинутых методов

	XGBoost	LightGBM	CatBoost
Построение деревьев	По уровням (Level-wise) 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) +	Предварительный биннинг

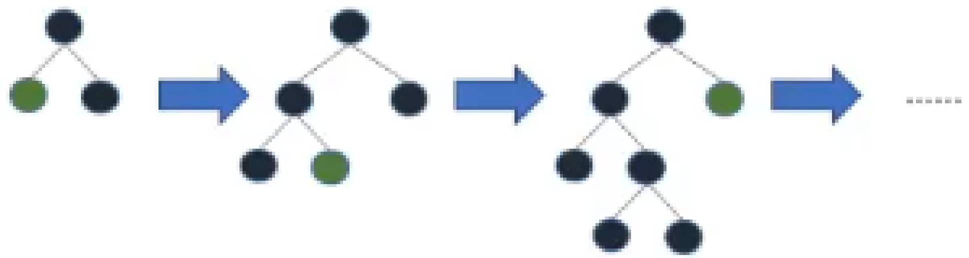
Фишки		Exclusive Feature Bundling
		Связываем разреженные признаки, которые одновременно не нули
		
		Random forest mode
			Динамический бустинг
			
			Overfitting Detector
			Ранний останов
			`od_type='Iter'`
			`use_best_model=True`
			`eval_metric=...`

Построение деревьев

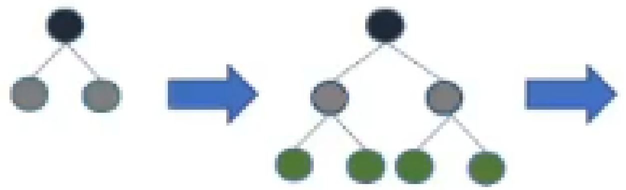
По уровням
(Level-wise)



По листьям
(Leaf-wise)



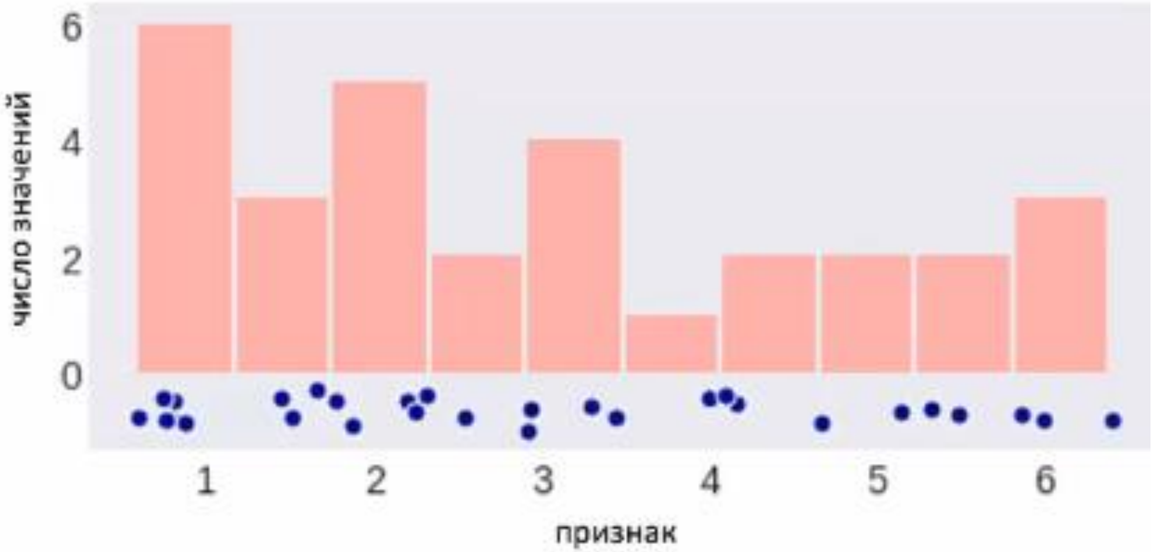
По уровням однородно
(oblivious trees)



Гистограммный подход

Histogram based algorithm

Каждый вещественный признак дискретизуется – разбивается на бины

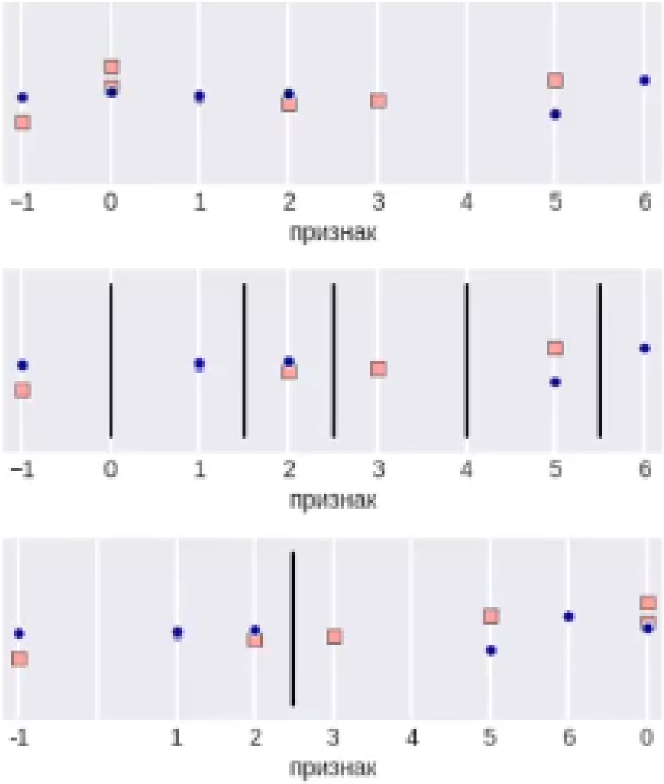


Теперь число порогов, которые надо посмотреть ~ число бинов

Особенности реализаций продвинутых методов

	XGBoost	LightGBM	CatBoost
Сэмплирование / градиенты / сплиты	– Сплиты медленнее, чем у конкурентов pre-sorted algorithm & Histogram-based	Gradient-based One-Side Sampling (GOSS) Среди малых градиентов сэмплируем, но с большим весом не выбран ли по умолчанию	Бернулли или байесовская подвыборка
Важности признаков	Gain / Frequency или Weight / Coverage	Gain / split	Prediction Values Change / Loss Function Change / Internal Feature Importance / SHAP
Нули обрабатываются как NaN	+ см. ниже Sparsity-aware Split Finding	+ По умолчанию `use_missing=True`	
Неизвестные значения	На оптимальную сторону сплита	На оптимальную сторону сплита	Min / Max

Игнорирование нулей / NaN



- Убираем нули
- Выбираем сплит
- Нули добавляем в «выгодное поддерево»

Итог

Выбрать вид бустинга / критерий расщепления / функцию ошибки «по задаче»

Три самых важных параметра: сложность, темп, число деревьев

При разных сложностях (глубина / число листьев)

Настроить два остальных связных параметра

Для настройки можно немного деревьев

В продакшене: увеличить число деревьев, взять маленький темп обучения

Использовать сумму нескольких gbm

- Проверить, помогает ли это
- Проверить, как нужно менять параметры для суммы

Спасибо за внимание!

Запорожцев Иван Федорович
zaporozhtsev.if.work@gmail.com