# Занятие 10. Ансамбли моделей

Гирдюк Дмитрий Викторович

25 ноября 2023

СПбГУ, ПМ–ПУ, ДФС

# Задача обучения с учителем

- Постановка задачи обучения с учителем (supervised learning): необходимо предсказать значение целевой переменной $y \in Y$ объекта по набору его признаков $\boldsymbol{x} \in X$.
- Среда описывается совместным распределением $f_{X,Y}(\boldsymbol{x}, y)$, а выборкой из нее является набор пар $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$.
- Результирующая модель возвращает значение $y$ по признакам $\boldsymbol{x}$: $y = h(\boldsymbol{x}; \boldsymbol{\theta})$.
- Классификация:
  - $Y = \{0, 1\}$ – бинарная (binary)
  - $Y = \{1, 2, \ldots, K\}$ – многоклассовая (multiclass)
  - $Y = \{0, 1\}^K$ – многозначная (multi-label)
- Регрессия:
  - $Y = R$ – одномерная (ordinal)
  - $Y = R^K$ – многомерная (multiple)

# Введение

- На предыдущей лекции мы познакомились с решающими деревьями. Отлично интерпретируемой моделью для решения задач классификации и регрессии.
- Тем не менее, как было показано, решающие деревья весьма нестабильны в своем обучении. Исключив/добавив новые наблюдения, можем получить на выходе существенно различающиеся модели.
- Т.е. дисперсия модели достаточно высока.
- А если другая ситуация? Обучили, например, Лассо регрессию. Дисперсия в пределах разумного, а вот смещение не дотягивает.
- Вопрос: что делать, если модель обучена, параметры подобраны, а ее качество не устраивает?

# Задача

- Допустим, что перед вами стоит следующая задача.
- Вам предлагают выбрать из двух альтернатив:
  1. бросить в кольцо лишь раз и победить при успешном попадании,
  2. бросить трижды, но победа лишь в случае 2 заброшенных мячей.
- Считаем вероятность попадания постоянной и равной $p$.
- Что предпочтете?

# Ансамблирование моделей

- Ансамбль моделей – общий и весьма эффективный подход построения комбинации (базовых) моделей с целью понизить смещение и/или дисперсию относительно одиночных моделей.
- Подходов к ансамблированию хватает. В простейшем случае для задачи регрессии предсказание множества моделей усередняется, а в классификации выбирается наиболее частое. Есть и более продвинутые техники, разберем самые популярные из них:
  - Bootstrap aggregation или Bagging
  - Blending и Stacked generalization или Stacking
  - Boosting
- Методы ансамблирования моделей принято называть метаалгоритмами или метамоделями.

# Bagging i

- Начнем с бэггинга.
- Используя выборку $\mathcal{D}$, обучаем некоторую модель $h(\boldsymbol{x}; \boldsymbol{\theta})$.
- На основе $\mathcal{D}$ сформируем новую выборку размера $\widetilde{N}$ путем извлечения объектов из исходной с повторением. Формирование множества таких новых выборок $\{\mathcal{D}\}_{k=1}^{K}$ называется бутстрепом – это общий метод исследования распределений в статистике.
- На каждой из таких выборок $\mathcal{D}^k$ обучается та же самая модель $h(\boldsymbol{x}; \boldsymbol{\theta}^k)$.

# Bagging ii

- Тогда ансамбль моделей $a(\boldsymbol{x})$ в случае регрессии представляет собой среднее предсказание всех моделей

$$a(\boldsymbol{x}) = \frac{1}{K}\sum_{k=1}^{K} h(\boldsymbol{x}; \boldsymbol{\theta}^k)$$

  а в случае классификации – класс, за который "проголосует"большее число моделей.
- Если $\widetilde{N} = N$ вероятность объекту попасть в новую выборку равна 63.2%. Т.е. каждая модель обучается на бутстрапированной выборке из 63.2% исходных объектов. В общем случае чем меньше данных, тем хуже качество. Но есть и плюсы: оставшиеся данные (out-of-bag, OOB) могут быть использованы для валидации модели.

# Эффект бэггинга i

- Посчитаем смещение ансамбля

$$bias_{\mathcal{D}}\left[a(\boldsymbol{x};\mathcal{D})\right] = E_{\mathcal{D}}\left[f(x) - a(\boldsymbol{x};\mathcal{D})\right] =$$
$$= f(x) - \frac{1}{k}\sum_{k=1}^{K} E_{\mathcal{D}}\left[h(\boldsymbol{x};\mathcal{D}^k)\right] =$$
$$= f(x) - E_{\mathcal{D}}\left[h(\boldsymbol{x};\mathcal{D})\right] = bias_{\mathcal{D}}\left[h(\boldsymbol{x};\mathcal{D})\right]$$

- Смещение ансамбля равно смещению отдельной модели в нем.

# Эффект бэггинга ii

$$Var_{\mathcal{D}}\left[a(\boldsymbol{x};\mathcal{D})\right] = E_{\mathcal{D}}\left[a(\boldsymbol{x};\mathcal{D}) - E_{\mathcal{D}}\left[a(\boldsymbol{x};\mathcal{D})\right]\right]^2 =$$

$$= E_{\mathcal{D}}\left[\frac{1}{k}\sum_{k=1}^{K} h(\boldsymbol{x};\mathcal{D}^k) - E_{\mathcal{D}}\left[\frac{1}{k}\sum_{k=1}^{K} h(\boldsymbol{x};\mathcal{D}^k)\right]\right]^2 =$$

$$= \frac{1}{k^2} E_{\mathcal{D}}\left[\sum_{k=1}^{K}\left(h(\boldsymbol{x};\mathcal{D}^k) - E_{\mathcal{D}}\left[h(\boldsymbol{x};\mathcal{D}^k)\right]\right)\right]^2 =$$

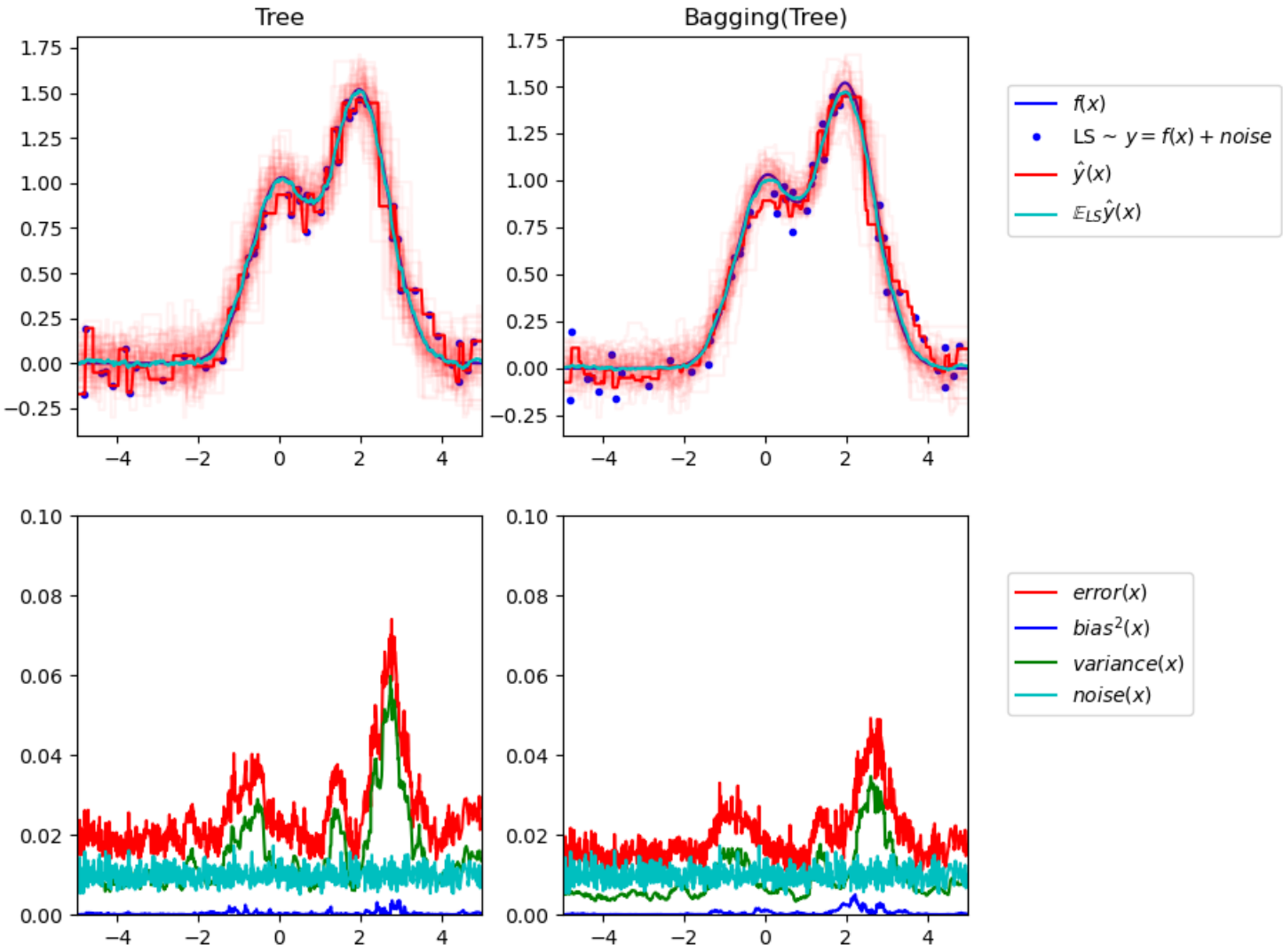
$$= \frac{1}{k^2}\sum_{k=1}^{K} E_{\mathcal{D}}\left[h(\boldsymbol{x};\mathcal{D}^k) - E_{\mathcal{D}}\left[h(\boldsymbol{x};\mathcal{D}^k)\right]\right]^2 +$$

$$+ \frac{1}{k^2}\sum_{k_1\neq k_2} E_{\mathcal{D}}\left[\left(h(\boldsymbol{x};\mathcal{D}_1^k) - E_{\mathcal{D}}h(\boldsymbol{x};\mathcal{D}_1^k)\right)\left(h(\boldsymbol{x};\mathcal{D}_2^k) - E_{\mathcal{D}}h(\boldsymbol{x};\mathcal{D}_2^k)\right)\right] =$$

$$= \frac{1}{k^2}\sum_{k=1}^{K} Var_{\mathcal{D}}\left[h(\boldsymbol{x};\mathcal{D}^k)\right] + \underbrace{\frac{1}{k^2}\sum_{k_1\neq k_2} cov\left[h(\boldsymbol{x};\mathcal{D}_1^k), h(\boldsymbol{x};\mathcal{D}_2^k)\right]}_{=0,\ \text{если некоррелированы}} =$$

$$= \frac{1}{k} Var_{\mathcal{D}}\left[h(\boldsymbol{x};\mathcal{D})\right] \implies \text{дисперсия ансамбля снижается в } k \text{ раз}$$

# Пример снижения дисперсии [1]



# Бэггинг: обсуждение

- Проблема состоит в том, что предсказания базовых моделей не независимы: базовые модели зачастую есть модели из одного класса (например, деревья), а обучение происходит под ту же самую задачу.
- Случайные подвыборки действительно доставляют "разнообразие". Но есть и другие способы: например, обучаться по подмножеству признаков, добавлять рандомизацию при обучении (деревья со случайным выбором точки сплита), или вовсе аугментировать данные.
- Реализован в scikit-learn.

# Random forest

- Случайный лес (Random forest) – продвинутый метод бэггинга над решающими деревьями, который пытается в еще большую "декорреляцию" базовых моделей.
- При обучении решающих деревьев в лесу (ну и фраза), на каждом этапе разбиения поиск оптимиального разделяющего признака производится по подмножеству исходных признаков.
- Доля признаков в подмножестве выступает одним из многочисленных гиперпараметров. Распространенные варианты: $n//3, \sqrt{n}, \log_2 n$.
- Итоговое предсказание получается аналогичным бэггингу образом.

# Сравнение подходов к ансамблированию [2]

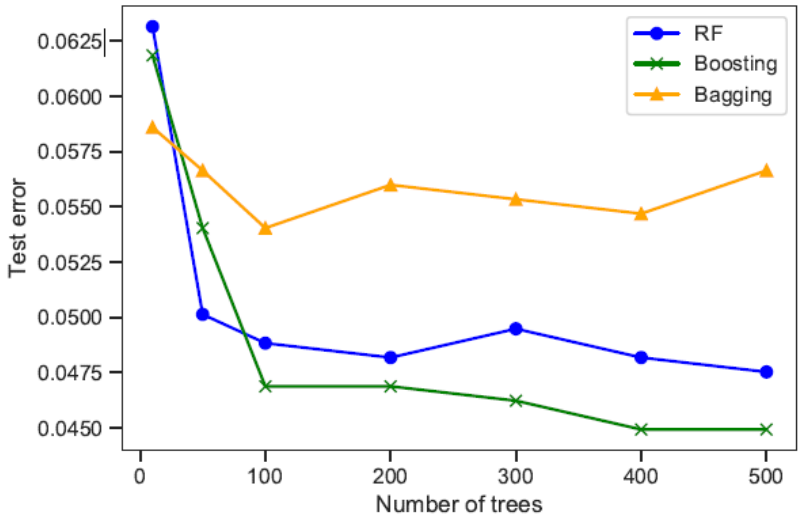



*Figure 18.5: Preditive accuracy vs size of tree ensemble for bagging, random forests and gradient boosting with log loss. Adapted from Figure 15.1 of [HTF09]. Generated by spam_tree_ensemble_compare.ipynb.*

# Случайный лес: обсуждение i

- Кажется, не так уж и сложно.
- Учитывая закладываемую в алгоритм идею, следует строить деревья максимальной глубины – мы ведь исходим из того, что хотим существенно уменьшить дисперсию. Кроме разве что экстремальных случаев: например, очень много данных, много выбросов.
- Случайный лес – один из немногих примеров универсальных алгоритмов в машинном обучении с точки зрения области применения. И весьма эффективных!
- Судите сами, подходит как для классификации и регрессии, так и для отбора признаков, поиска аномалий. Обучение деревьев спокойно параллелится. OOB-объекты за счет сэмплирования выборки, на которых можно получать несмещенную оценку качества и вычислять важность признаков.

# Случайный лес: обсуждение ii

- Как вычислять важность признаков? В случайном лесе делают следующее: замеряют качество дерева в ансамбле на OOB примерах, перемешивают значения признака, замеряют снова. Разница в значениях метрики усредняется по всем деревьям.
- Замечание про переобучение. Конечно, построить миллион деревьев не выйдет, ведь размеры выборки ограничены в том смысле, что бесконечное сэмплирование, потому необходимо отслеживать размер ансамбля и отрезать лишнее, когда метрика качества перестает значительно улучшаться.
- Без недостатков однобоко выглядит. Например, если доля информативных признаков невелика, сэмплирование признаков при разбиении может существенно отразиться на качестве работы алгоритма. Подбирайте и констурируйте признаки с умом!
- Реализован в scikit-learn.

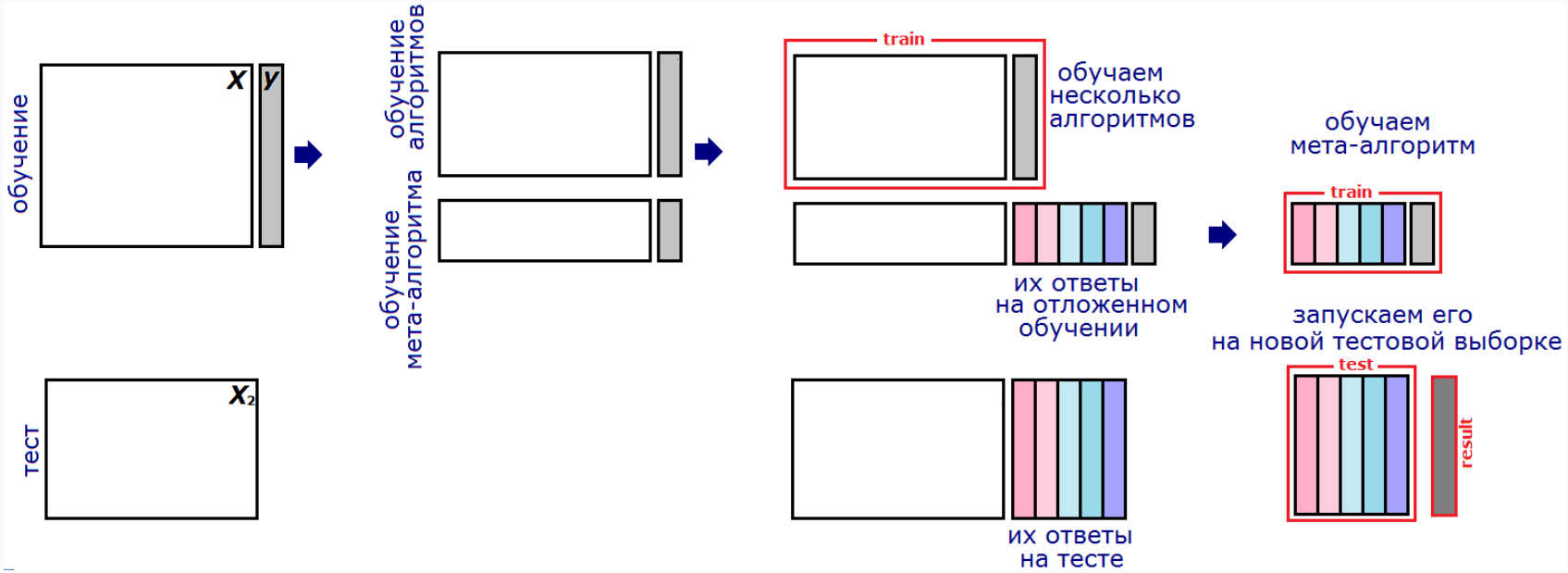
# Blending и Stacking

- Почему бы не пойти дальше? Не просто усреднять, а строить новую модель поверх предсказаний базовых моделей?
- Способов того, как это сделать, тоже хватает. Разберем 2 основных: блэндинг и стэкинг.

# Blending

- Делим обучающую (!) выборку на 2 части.
- На первой части обучаем базовые модели. На отложенной части делаем предсказания обученными моделями и получаем набор из так называемых метапризнаков.
- Используем известные таргеты для обучения метамодели на метапризнаках.

# Схема блендинга [3]



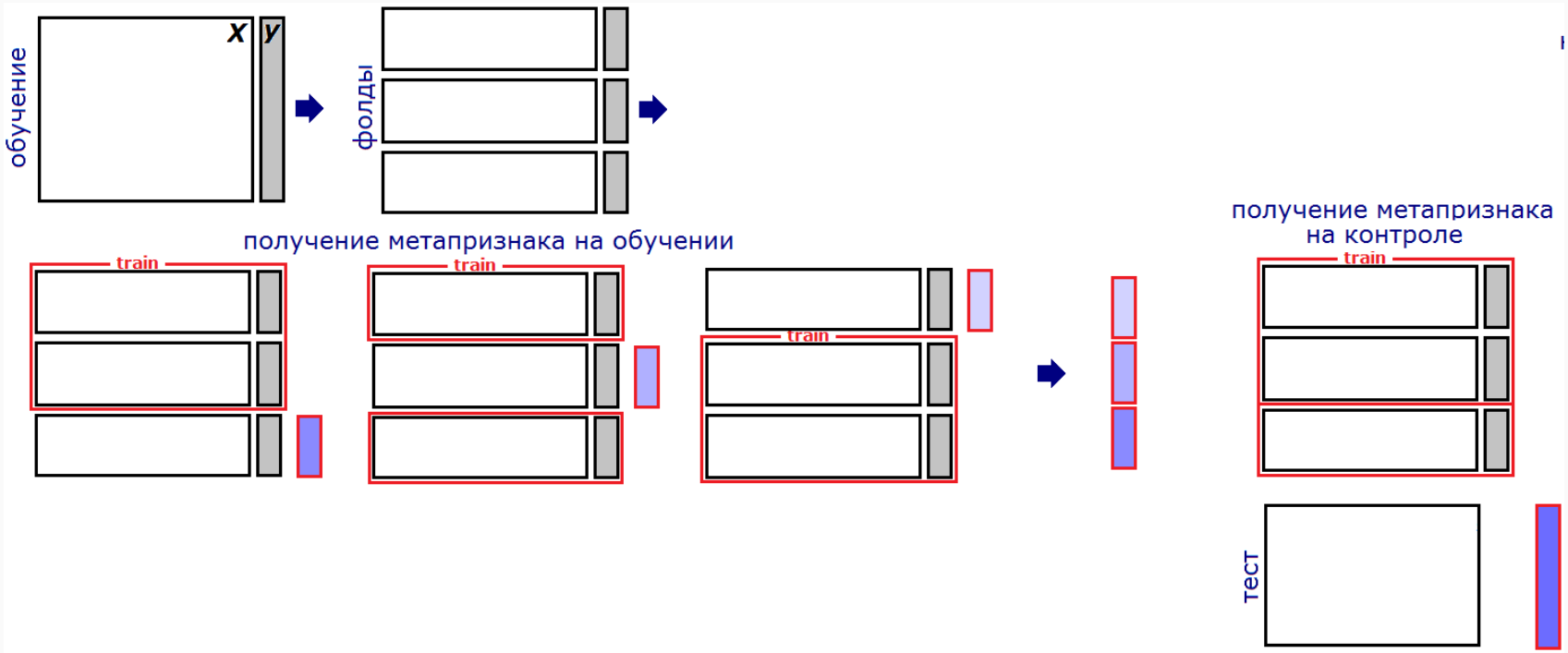
# Stacking i

- Проблема блендинга очевидна: приличная часть данных уходит на обучение метамодели. Частично побороть это можно путем проведения обучения блендинга несколько раз с разными разбиениями данных.
- Но есть и другая идея (скорее даже набор идей), называемая стэкингом.

# Stacking ii

- Как в кросс-валидации делим обучающую выборку на $K$ частей/фолдов. Последовательно обучаем все базовые модели на подмножествах из $K-1$ фолдов, предсказываем на оставшемся.
- Из предсказаний на них собираем для каждой базовой модели свой метапризнак (можно также несколько раз разбить на фолды и усреднить предсказания на объектах). На полученных метапризнаках обучаем метамодель.
- После чего все базовые алгоритмы обучаются на всей тренировочной выборке. Ими производится предсказание на тестовой выборке для получения метапризанков, которые подаются в обученную метамодель.

# Схема стекинга [3]



# Блендинг и стекинг: обсуждение i

- В случае с блендингом и стекингом можно (и нужно) использовать разные типы моделей. Почему бы не объеденить kNN, логистические регрессии с различными параметрами регуляризации и случайный лес, а в качестве метаалгоритма еще раз использовать случайный лес? Правда сильно нафантазировать тут может не выйти ввиду природы данных и различных типов препроцессинга для различных моделей.
- Строгой теории, связывающей эти подходы с дилеммой смещения-дисперсии нет, блендинг и стекинг зачастую просто немного улучшают качество наилучшей модели в ансамбле. Все сильно зависит от задачи, от лосс-функции.

# Блендинг и стекинг: обсуждение ii

- Махинация в стекинге с переобучением базовых моделей на всем тренировочном датасете очевидна. Признаки базовых моделей на обучении и тесте разные. Есть разные способы побороться с ней, например, регуляризацией метамодели или зашумление метапризнаков.
- Стекинг и блендинг можно делать многоуровневым! Муторно, в реальных проектах почти не встречается, но на соревнованиях таким не брезгуют заниматься, если время и вычислительные ресурсы позволяют.
- Добавлять исходные признаки к метапризнакам можно, но очень осторожно.
- В scikit-learn есть реализация стэкинга.

# Boosting

- Бустинг – альтернативный подход к ансамблированию, основная задача которого состоит в уменьшении смещения (хотя дисперсия обычно также снижается).
- Достигается это за счет построения ансамбля из достаточно "слабых" моделей, причем не параллельно, как в бэггинге, а последовательно, где каждая новая базовая модель обучается таким образом, чтобу уменьшить суммарную ошибку текущего ансамбля.
- Бустинг – общая идея, на основе которой в свое время было построено достаточно много разнообразных алгоритмов.
- Наиболее общая версия бустинга, обобщающая известные ранее, "градиентный бустинг", была разработана в конце прошлого века. На текущий момент все основные библиотеки, реализующие бустинг (xgboost [4], catboost [5], lightgbm [6]), реализуют именно ее.

# Последовательное конструирование ансамбля i

- Если вспомнить общую идею того, как мы оптимизируем модели на основе выборочных данных и учтем описание бустинга с предыдущего слайда, то приходим к следующему набору оптимизационных задач

$$\arg\min_{\lambda,\boldsymbol{\theta}} \sum_{i=1}^{N} L\left(y^{(i)}, a_{t-1}(\boldsymbol{x}^{(i)}) + \lambda h(\boldsymbol{x}^{(i)};\boldsymbol{\theta})\right), t = 1, \ldots, T$$

- Решения которых последовательно собирают ансамбль

$$a_t(\boldsymbol{x}) = a_{t-1}(\boldsymbol{x}) + \lambda_t h(\boldsymbol{x};\boldsymbol{\theta}_t)$$

- При решении новой задачи параметры предыдущих моеделей в ансамбле не обновляются, также как не обновляются и их весовые коэффициенты (хотя чаще всего весовые коэффициенты имеют одно фиксированное значение).

# Последовательное конструирование ансамбля ii

- Очевидно, что решение оптимизационной задачи зависит от конкретной лосс-функции и конкретного типа модели.
- Например, для квадратичной функции потерь в задаче регрессии

$$L\left(y^{(i)}, a_{t-1}(\boldsymbol{x}^{(i)}) + \lambda h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})\right) =$$
$$= \frac{1}{2}\left(y^{(i)} - a_{t-1}(\boldsymbol{x}^{(i)}) - \lambda h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})\right)^2 = \left(r_t^{(i)} - \lambda h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})\right)^2$$

- Если положить $\lambda = 1$, то мы по сути подгоняем модель $h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$ предсказывать невязку между таргетом и предсказанием текущего ансамбля. А что такое невязка в данном случае?

$$y^{(i)} - a(\boldsymbol{x}^{(i)}) = -\frac{\partial L(y^{(i)}, a(\boldsymbol{x}))}{\partial a(\boldsymbol{x})}\Big|_{a(\boldsymbol{x})=a_{t-1}(\boldsymbol{x}^{(i)})}$$

# Градиентный бустинг i

- Хорошо, с квадратичной функцией ошибки все понятно, а что с другими лосс-функциями? Да все то же самое! Рассмотрим разложение лосс-функции в ряд Тейлора до первого члена

$$L\left(y^{(i)}, a_{t-1}(\boldsymbol{x}^{(i)}) + \lambda h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})\right) \approx$$

$$\approx L\left(y^{(i)}, a_{t-1}(\boldsymbol{x}^{(i)})\right) + \lambda h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \frac{\partial L(y^{(i)}, a(\boldsymbol{x}))}{\partial a(\boldsymbol{x})}\Big|_{a(\boldsymbol{x})=a_{t-1}(\boldsymbol{x}^{(i)})}$$

- Первое слагаемое постоянное, потому при оптимизации параметров базовой модели итоговая лосс-функция может быть записана следующим образом

$$-\arg\min_{\lambda, \boldsymbol{\theta}} \sum_{i=1}^{N} \lambda h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) g_{t-1}^{(i)},\ t = 1, \ldots, T$$

# Градиентный бустинг ii

- По сути имеем скалярное произведение, и чтобы его минимизировать, можно обучать базовую модель $h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$ предсказывать антиградиент $g_{t-1}^{(i)}$ (и для задачи регрессии, и для классификации!)

$$\boldsymbol{\theta}_t = \arg\min \sum_{i=1}^{N} \left( h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) - g_{t-1}^{(i)} \right)^2$$

- $\lambda$ же можно находить бинарным/линейным поиском для исходной оптимизационной задачи уже после того, как модель $h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$ будет обучена. По крайней мере так было в исходной статье по градиентному бустингу.

# Градиентный бустинг: алгоритм

1. Инициализируем первую модель в ансамбле константой:

$$a_0(\boldsymbol{x}) = \arg\min_{\gamma} \sum_{i=1}^{N} L(y^{(i)}, \gamma)$$

2. Для $t = 1, \ldots, T$ повторяем шаги 3-6.
3. Вычисляем псевдоостатки/антиградиенты

$$g_{t-1}^{(i)} = -\frac{\partial L(y^{(i)}, a(\boldsymbol{x}))}{\partial a(\boldsymbol{x})}\Big|_{a(\boldsymbol{x})=a_{t-1}(\boldsymbol{x}^{(i)})}, \; i = 1, \ldots, N$$

4. Подгоняем по ним базовую модель $h(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$ на выборке $\{(\boldsymbol{x}^{(i)}, g_{t-1}^{(i)})\}_{i=1}^{N}$.
5. Линейным поиском находим оптимальное значение $\lambda_t$.
6. Обновляем ансамбль $a_t(\boldsymbol{x}) = a_{t-1}(\boldsymbol{x}) + \lambda_t h(\boldsymbol{x}; \boldsymbol{\theta}_t)$

# Пример бустинга [2]

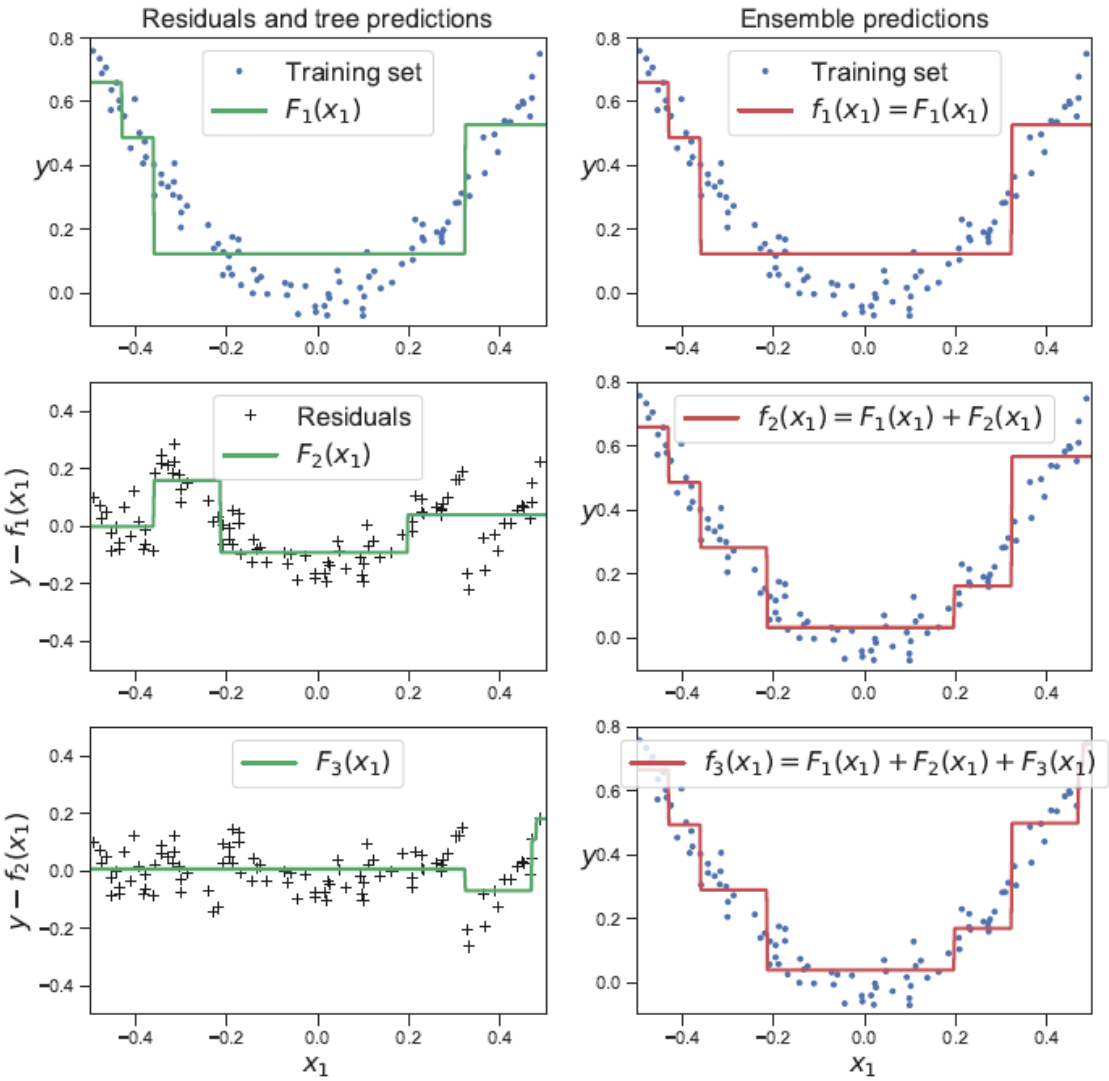



*Figure 18.6: Illustration of boosting using a regression tree of depth 2 applied to a 1d dataset. Adapted from Figure 7.9 of [Gér19]. Generated by boosted_regr_trees.ipynb.*

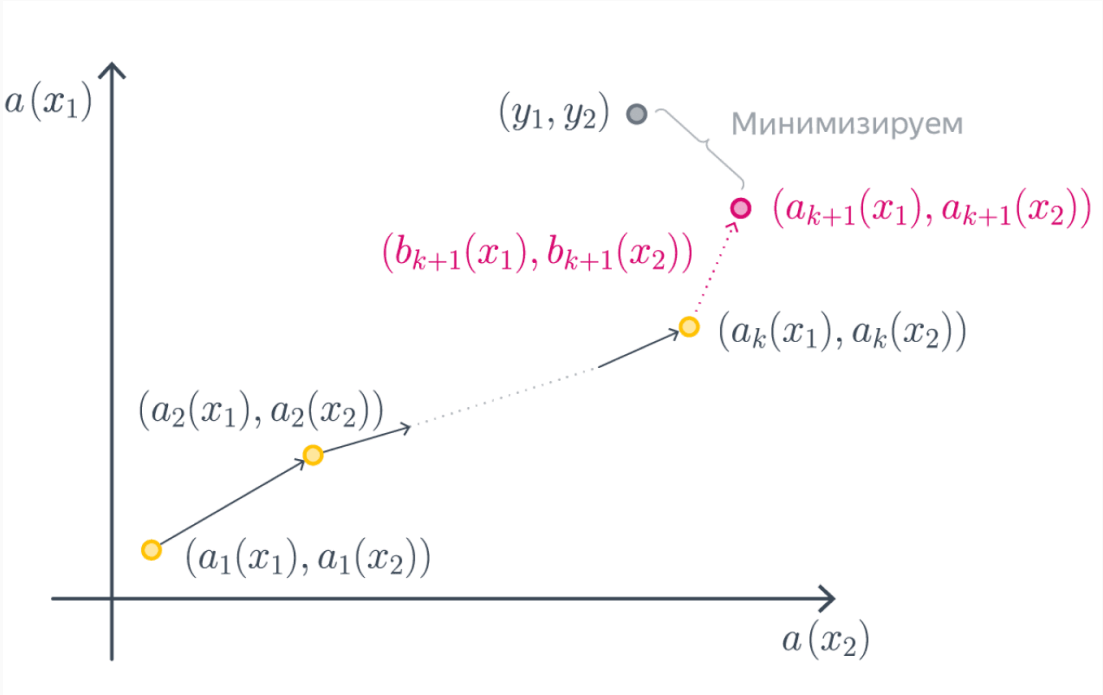
# Градиентный бустинг: лосс-функции

| Name | Loss | $-\partial \ell(y_i, f(\boldsymbol{x}_i))/\partial f(\boldsymbol{x}_i)$ |
|---|---|---|
| Squared error | $\frac{1}{2}(y_i - f(\boldsymbol{x}_i))^2$ | $y_i - f(\boldsymbol{x}_i)$ |
| Absolute error | $\lvert y_i - f(\boldsymbol{x}_i)\rvert$ | $\text{sgn}(y_i - f(\boldsymbol{x}_i))$ |
| Exponential loss | $\exp(-\tilde{y}_i f(\boldsymbol{x}_i))$ | $-\tilde{y}_i \exp(-\tilde{y}_i f(\boldsymbol{x}_i))$ |
| Binary Logloss | $\log(1 + e^{-\tilde{y}_i f_i})$ | $y_i - \pi_i$ |
| Multiclass logloss | $-\sum_c y_{ic} \log \pi_{ic}$ | $y_{ic} - \pi_{ic}$ |

*Table 18.1: Some commonly used loss functions and their gradients. For binary classification problems, we assume $\tilde{y}_i \in \{-1, +1\}$, and $\pi_i = \sigma(2f(\boldsymbol{x}_i))$. For regression problems, we assume $y_i \in \mathbb{R}$. Adapted from [HTF09, p360] and [BH07, p483].*

# Еще одна интерпретация градиентного бустинга [7]

- Помните из курса оптимизации, что такое метод градиентного спуска? Так вот, градиентный бустинг есть метод градиентного спуска в функциональном пространстве. Только функции у нас заданы таблично на тренировочных объектах. И вместо градиентного шага, мы сдвигаемся в функциональном пространстве на значения новой базовой модели, которая обучалась предсказывать антиградиент.



# Градиентный бустинг: обсуждение i

- Итого, градиентный бустинг – общая методология ансамблирования, нацеленная на уменьшение смещения. Хотя дисперсия обычно также снижается.
- Достаточно быстро поняли, что $\lambda_t$ можно заменить константой и выделить как отдельный гиперпараметр. Этот параметр по сути является скоростью обучения (learning rate), и позволяет контролировать вклад каждой базовой модели в итоговый ансамбль.
- На практике чаще всего градиентный бустинг строят над решающими деревьями. Причем важно использовать достаточно неглубокие деревья (вплоть до глубины 1, известные также как рещающие пни). Иначе для глубоких деревьев очень быстро базовые модели начнут предсказывать шум, что приведет к переобучению.

# Градиентный бустинг: обсуждение ii

- Контроль глубины деревьев в ансамбле наряду со скоростью обучения – основные способы борьбы с переобучением градиентного бустинга.
- Градиентный бустинг над решающими деревьями зачастую имеет способы оценки важности признаков. Например, можно замерять суммарное изменение информативности в каждой нелистовой вершине дерева со сплитом по конкретному признаку. А затем усреднять эти значения по всему ансамблю. И выводить нормализованные относительные значения (признак с наибольшим суммарным вкладом ставится в соответствии 100%).
- Несмотря на эффективность случайного леса, на практике бустинг зачастую показывает себя эффективнее с точки зрения итоговой точности.

# Градиентный бустинг: обсуждение iii

- По большому счету, градиентный бустинг на табличных данных является де-факто алгоритмом номер 1. И используется повсеместно.
- Как было ранее отмечено, уже продолжительное время существует 3 приличные библиотеки для обучения градиентного бустинга (xgboost [4], catboost [5], lightgbm [6]). Своих тонкостей у каждой очень много. Впрочем, основные идеи уже так или иначе позаимствованы друг у друга. Например, изначально поддержка категориальных переменных была лишь у catboost. Но с некоторых пор остальные две также умеют работать с ними.
- В scikit-learn также присутствует 2 реализации градиентного бустинга. Но их результаты зачастую хуже, чем у 3 указанных библиотек.

# Использованные источники i


1. **Bagging variance.** URL: https://scikit-learn.org/stable/auto_examples/ensemble/plot_bias_variance.html#sphx-glr-auto-examples-ensemble-plot-bias-variance-py.
2. *Murphy K. P.* **Probabilistic Machine Learning: An introduction.** MIT Press, 2022. URL: probml.ai.
3. *Дьяконов А.* **Стекинг (Stacking) и блендинг (Blending).** URL: https://alexanderdyakonov.wordpress.com/2017/03/10/c%D1%82%D0%B5%D0%BA%D0%B8%D0%BD%D0%B3-stacking-%D0%B8-%D0%B1%D0%BB%D0%B5%D0%BD%D0%B4%D0%B8%D0%BD%D0%B3-blending/.
4. **XGBoost.** URL: https://xgboost.readthedocs.io/en/stable/.
5. **CatBoost.** URL: https://xgboost.readthedocs.io/en/stable/.
6. **LightGBM.** URL: https://lightgbm.readthedocs.io/en/stable/.

# Использованные источники ii

7. **Глава по градиентному бустингу из учебника по машинному обучению школы анализа данных yandex.** URL: https://education.yandex.ru/handbook/ml/article/gradientnyj-busting.