



# Машинное обучение: КОМПОЗИЦИИ МОДЕЛЕЙ

Эмили Драль

MADE academy, Москва 2020

# Алгоритмы машинного обучения

1. Обучение с учителем: линейные модели
2. **Обучение с учителем: ансамбли моделей**
3. Обучение с учителем: нейросетевые модели
4. Обучение без учителя: обзор методов
5. (optional) Рекомендательные системы
6. (optional) Обучение с подкреплением

# План занятия

1. Идея композиции моделей
2. Оценка модели: noise, bias & variance
3. Bagging
4. Boosting
5. Обзор композиций

Идея композиции моделей

# Идея КОМПОЗИЦИИ моделей

## Композиция моделей

$X^l = (x_i, y_i)_{i=1}^l \subset X \times Y$  – обучающая выборка

$a(x) = C(b(x))$  – алгоритм, где

$b: X \rightarrow R$  – базовый алгоритм

$C: R \rightarrow Y$  – решающее правило

$R$  – пространство оценок

Композиция базовых алгоритмов  $b_1, \dots, b_N$ :

$$a(x) = C \left( F(b_1(x), \dots, b_N(x)) \right),$$

где  $F: R^N \rightarrow R$  – корректирующая операция.

Решающее правило вводится для удобства работы с задачами классификации, где часто для получения метки класса требуется преобразование ответа модели.

# Идея композиции моделей

## Композиция моделей

Для чего строить композицию?

Идея: за счет использования комбинации из нескольких моделей вместо одной снизить ошибку прогноза

# Идея КОМПОЗИЦИИ моделей

## Композиция моделей

Для чего строить композицию?

Идея: за счет использования комбинации из нескольких моделей вместо одной снизить ошибку прогноза

Проблема: если просто взять выборку  $X$  и построить  $N$  моделей, они все будут одинаковыми

# Идея КОМПОЗИЦИИ моделей

## Bootstrap

Проблема: если просто взять выборку  $X$  и построить  $N$  моделей, они все будут одинаковыми

Идея: с помощью технологии bootstrap сгенерируем подвыборки из исходной выборки с возвращением

Тогда:

$X_1, \dots, X_n$  - подвыборки, полученные методом bootstrap

$b_1, \dots, b_n$  - базовые модели

$y(x)$  - верные ответы



# Идея КОМПОЗИЦИИ моделей

## Bootstrap

$X_1, \dots, X_n$  - подвыборки, полученные методом bootstrap

$b_1, \dots, b_n$  - базовые модели

$y(x)$  - верные ответы

Определим  $E_x(b_j(x) - y(x))^2$  - матожидание ошибки базового алгоритма по всем  $x$ . Обозначим ошибку  $\varepsilon_j(x)$

Предположим, что:

$E_x \varepsilon_j(x) = 0$  – ошибки несмещенные

$E_x \varepsilon_j(x) \varepsilon_k(x) = 0, j \neq k$  – ошибки независимы (не коррелированы)

# Идея композиции моделей

## Bootstrap

Определим  $E_x(b_j(x) - y(x))^2$  - матожидание ошибки базового алгоритма по всем  $x$ . Обозначим ошибку  $\varepsilon_j(x)$

Предположим, что:

$$E_x \varepsilon_j(x) = 0$$

$$E_x \varepsilon_j(x) \varepsilon_k(x) = 0, \quad j \neq k$$

Рассмотрим  $a(x) = \frac{1}{N} \sum_{j=1}^N b_j(x)$  - среднее  $N$  базовых алгоритмов

Давайте оценим ошибку композиции и сравним с ошибкой базового алгоритма

# Идея КОМПОЗИЦИИ МОДЕЛЕЙ

## Bootstrap

$$a(x) = \frac{1}{N} \sum_{j=1}^N b_j(x)$$

Оценим ошибку композиции

$$E_x \left( \frac{1}{N} \sum_{j=1}^N b_j(x) - y(x) \right)^2$$

внесем  $y(x)$  внутрь суммы

$$E_x \left( \frac{1}{N} \sum_{j=1}^N (b_j(x) - y(x)) \right)^2 = E_x \left( \frac{1}{N} \sum_{j=1}^N \varepsilon_j(x) \right)^2$$

раскроем квадрат суммы

$$\frac{1}{N^2} (E_x \sum_{j=1}^N \varepsilon_j^2(x) + \sum_{j \neq k} E_x \varepsilon_j(x) \varepsilon_k(x))$$

второе слагаемое равно 0

$$\frac{1}{N^2} E_x \sum_{j=1}^N \varepsilon_j^2(x)$$

# Идея композиции моделей

## Bootstrap

$$a(x) = \frac{1}{N} \sum_{j=1}^N b_j(x)$$

Пусть ошибки одинаково распределены

$\frac{1}{N^2} E_x \sum_{j=1}^N \varepsilon_j^2(x) = \frac{1}{N} E_x \varepsilon_j^2$  - таким образом ошибка усредненной модели в  $n$  раз меньше

Значит, построение композиции для снижения ошибки имеет смысл

# Оценка модели

## Разложение ошибки

Ошибка любого алгоритма может быть представлена в виде суммы трёх составляющих:

- **Сложность задачи:** насколько задача поддается решению?
- **Смещение:** насколько выбранная модель подходит для решения задачи?
- **Разброс:** насколько разнообразно выбранное семейство моделей (дисперсия ответов модели)

## Оценка модели

# Разложение ошибки

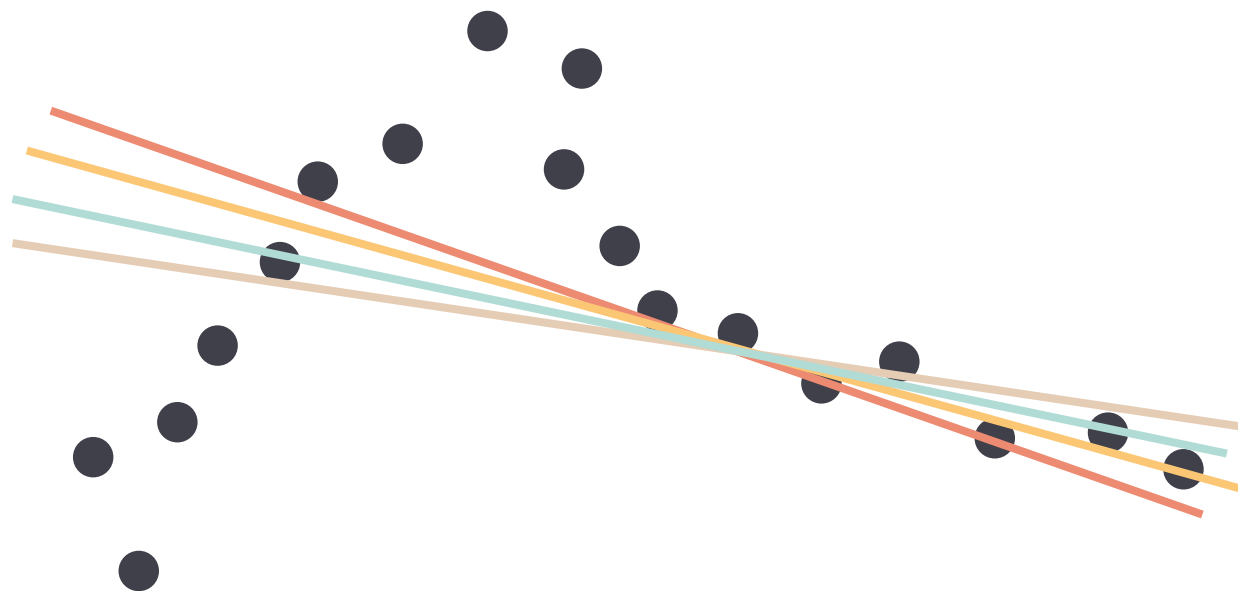


Какова сложность задачи?

- задача умеренно сложная
- для нас это не принципиально, так как не влияет на модель

Оценка  
модели

# Разложение ошибки

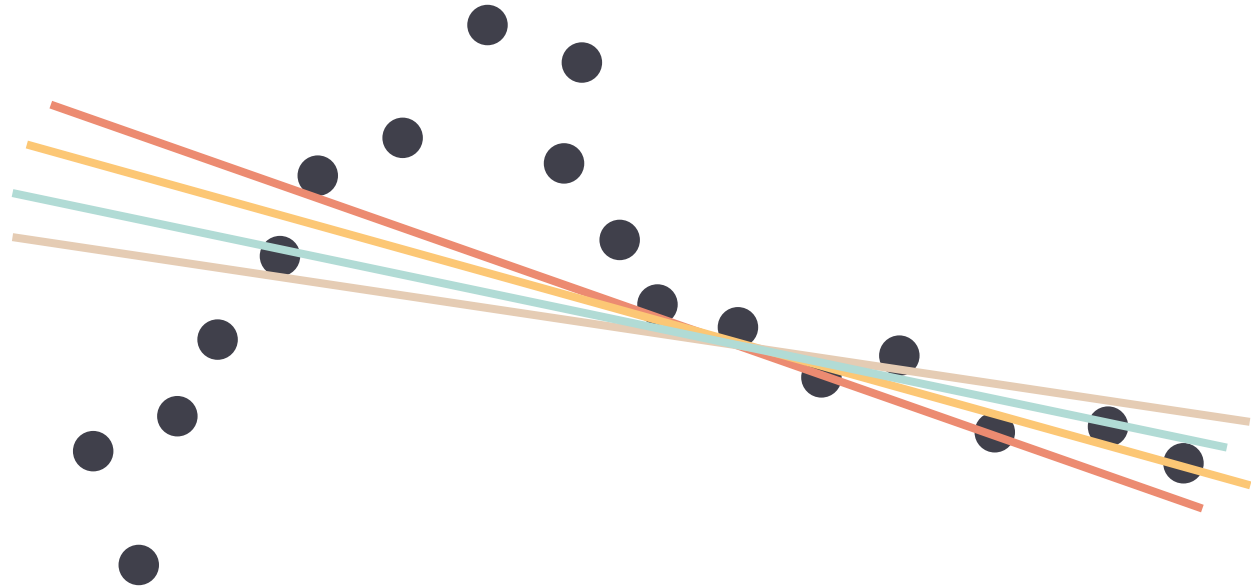


Каковы смещение и разброс?



## Оценка модели

# Разложение ошибки

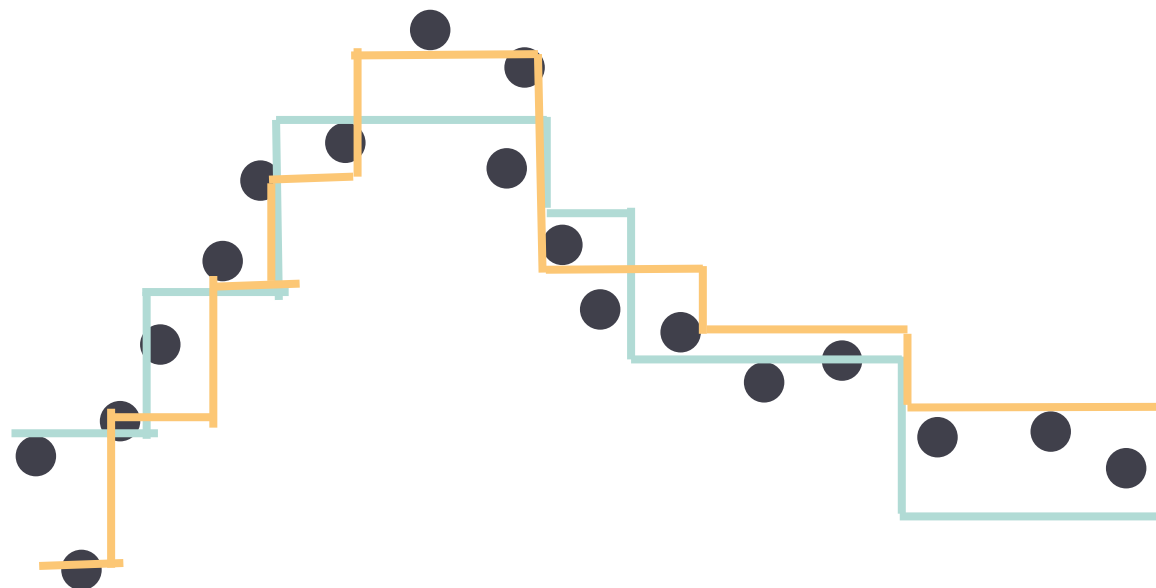


Каковы смещение и разброс?

- высокое смещение
- низкий разброс

## Оценка модели

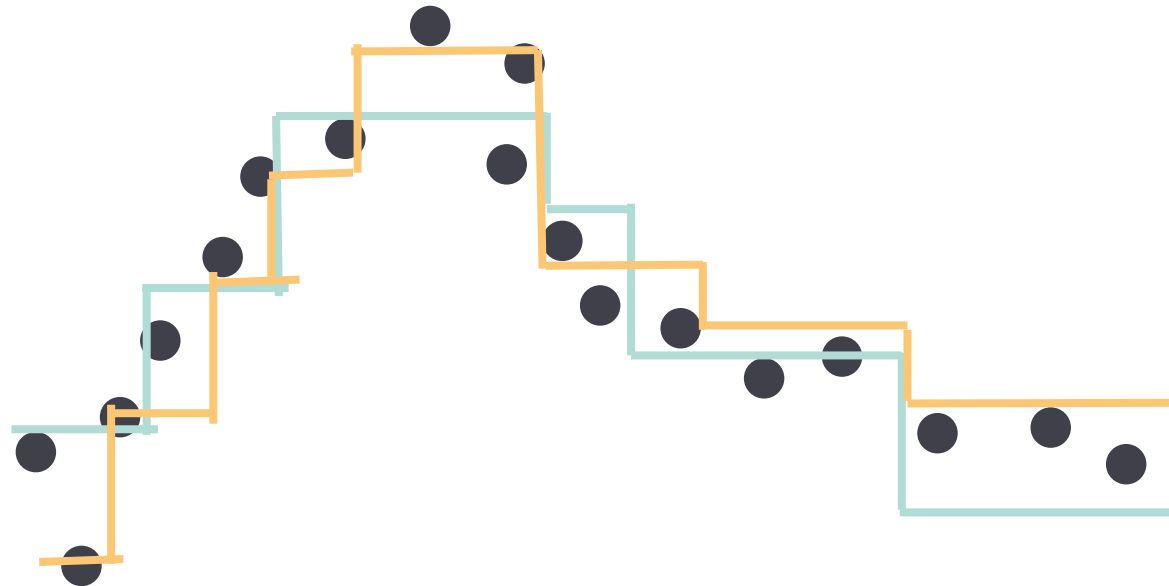
# Разложение ошибки



Каковы смещение и разброс?

## Оценка модели

# Разложение ошибки

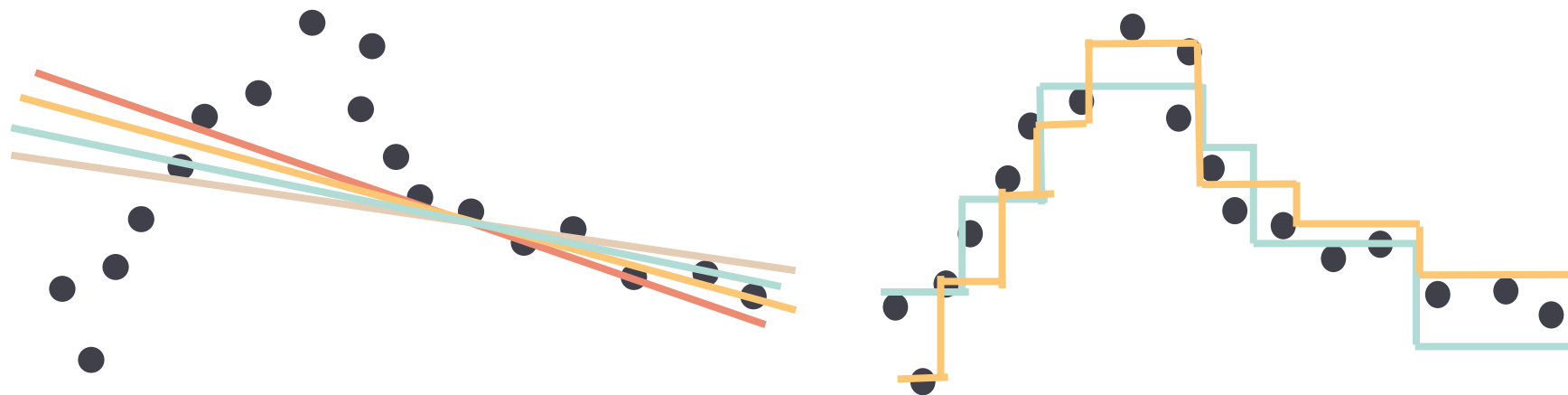


Каковы смещение и разброс?

- низкое смещение
- высокий разброс

## Оценка модели

# Разложение ошибки



Компромисс между смещением и разбросом называют bias-variance tradeoff

Идея: с помощью построения композиции снизить разброс при сохранении смещения

# Bagging

# Bagging

## Bootstrap

Bagging = Bootstrap aggregation

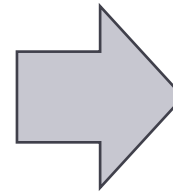
№	значение
1	
2	
3	
N	

# Bootstrap

Bagging = Bootstrap aggregation

Bagging

№	значение
1	
2	
3	
N	



№	значение
1	
25	
1	

№	значение
67	
24	
13	

№	значение
9	
9	
9	

...

# Bagging

## Bagging

Bagging = Bootstrap aggregation

$\mu: (X \times Y)^l \rightarrow A$  - метод обучения

$\tilde{X}$  - подвыборка, полученная с помощью bootstrap

$b_1(\tilde{X}), \dots, b_N(\tilde{X})$  – базовые модели (одинаковые), обученные на разных подвыборках

$a(x) = \frac{1}{N} \sum_{i=1}^N b_i(x)$  - композиция моделей



# Bagging

## Bagging

Random Forest = Bagging over decision trees

Для достижения эффекта снижения разброса деревья должны быть разными (некоррелированными)

\*помните, мы требовали, чтобы  $E_x \varepsilon_j(x) \varepsilon_k(x) = 0, j \neq k$

# Bagging

## Bagging

Random Forest = Bagging over decision trees

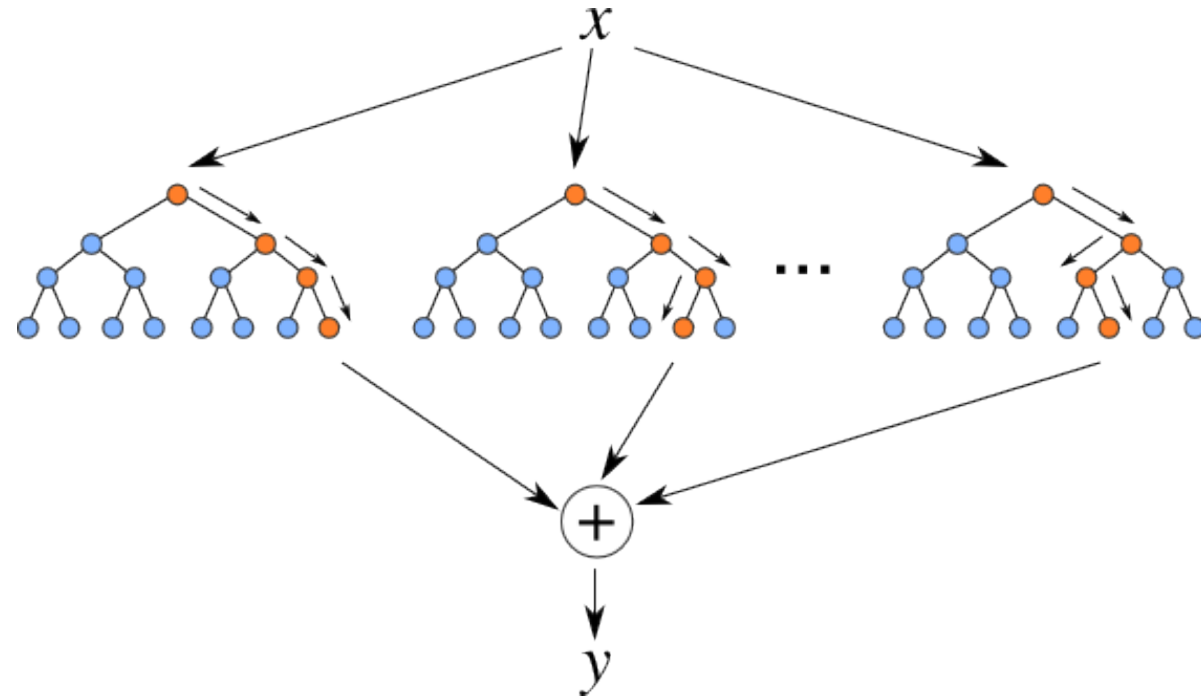
Для достижения эффекта снижения разброса деревья должны быть разными (некоррелированными):

- каждое дерево строится на своей подвыборке
- разбиение в каждом узле производится по признаку из ограниченного подмножества признаков
- используются деревья большой глубины (всего несколько объектов в листе)

## Bagging

# Random forest: построение

1. Генерируем  $M$  выборок на основе имеющейся
2. Строим на них деревья с рандомизированными разбиениями в узлах: выбираем  $k$  случайных признаков и ищем наиболее информативное разбиение по ним
3. При прогнозировании усредняем ответ всех деревьев

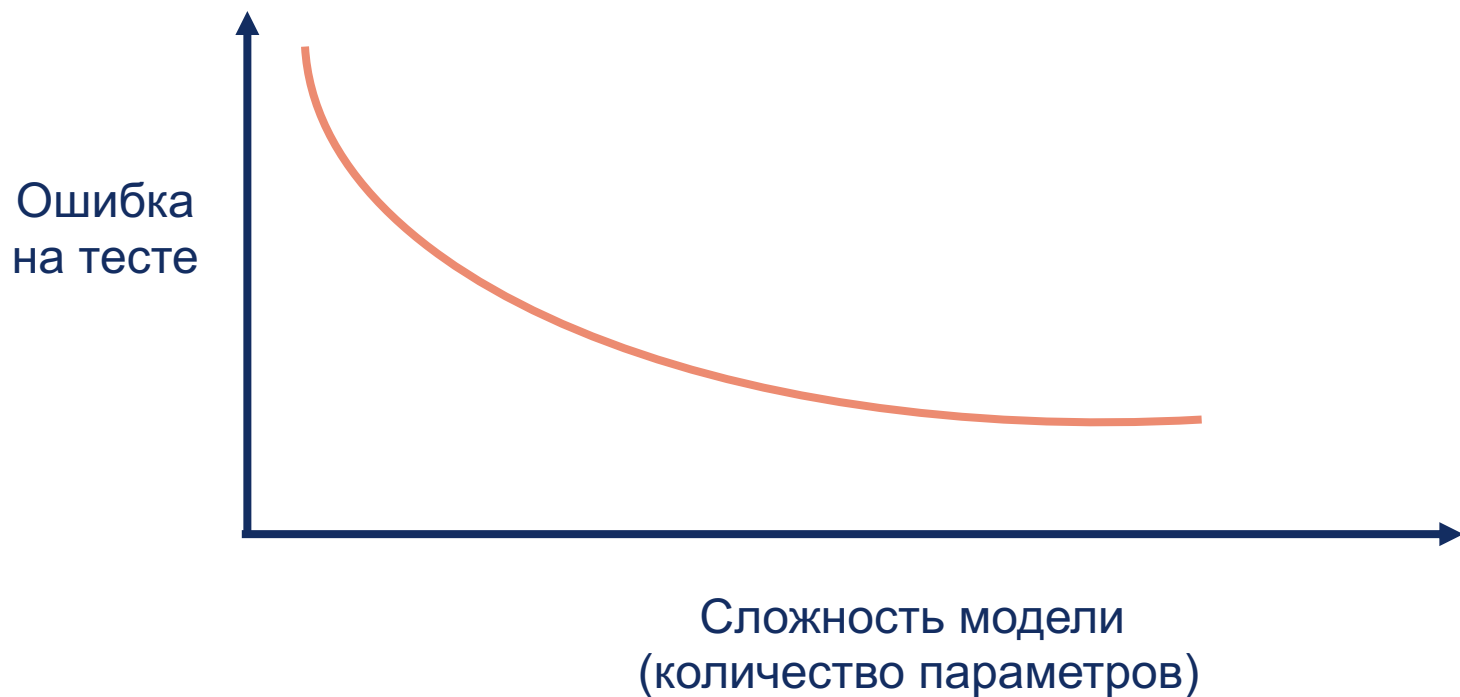


## Bagging

# Random Forest: переобучение

Random forest:

- Не требует тонкой настройки
- Устойчив к переобучению



## Bagging

# Out of bag estimation

Random forest позволяет делать оценку обобщающей способности модели без использования тестовой выборки

Идея: давайте оценим качество для каждого объекта, только по тем деревьям, в построении которых он не участвовал (а такие есть и их много, так как используется bagging)

## Bagging

# Out of bag estimation

Random forest позволяет делать оценку обобщающей способности модели без использования тестовой выборки

$$OOBE = \sum_{i=1}^l L(y_i, \frac{1}{\sum_{n=1}^N [x_i \notin \tilde{X}_n]} \sum_{n=1}^N [x_i \notin \tilde{X}_n] b_n(x_i))$$

для каждого объекта посчитали ошибку среднего прогноза по деревьям, в обучении которых он не участвовал

## Bagging

# Out of bag estimation

Random forest позволяет делать оценку обобщающей способности модели без использования тестовой выборки

$$OOBE = \sum_{i=1}^l L(y_i, \frac{1}{\sum_{n=1}^N [x_i \notin \tilde{X}_n]} \sum_{n=1}^N [x_i \notin \tilde{X}_n] b_n(x_i))$$

для каждого объекта посчитали ошибку среднего прогноза по деревьям, в обучении которых он не участвовал

## Bagging

# Random forest

$T_n(x)$  – номер листа, в который попал объект  $x$  в дереве  $b_n$

Тогда можно записать прогноз базового алгоритма:

$$b_n(x) = \sum_{i=1}^l w_n(x, x_i) y_i, \quad w_n(x, x_i) = \frac{[T_n(x)=T_n(x_i)]}{\sum_{j=1}^l [T_n(x)=T_n(x_j)]}$$

фактически здесь записано среднее по объектам обучающей выборке в листе, в который попал объект  $x$



## Bagging

# Random forest

$T_n(x)$  – номер листа, в который попал объект  $x$  в дереве  $b_n$

Тогда можно записать прогноз базового алгоритма:

$$b_n(x) = \sum_{i=1}^l w_n(x, x_i) y_i, \quad w_n(x, x_i) = \frac{[T_n(x)=T_n(x_i)]}{\sum_{j=1}^l [T_n(x)=T_n(x_j)]}$$

Выразим через эту формулу прогноз всей композиции:

$$a(x) = \frac{1}{N} \sum_{n=1}^N \sum_{i=1}^l w_n(x, x_i) y_i = \sum_{i=1}^l \left( \frac{1}{N} \sum_{n=1}^N w_n(x, x_i) \right) y_i$$

# Random forest

Прогноз всей композиции:

$$a(x) = \frac{1}{N} \sum_{n=1}^N \sum_{i=1}^l w_n(x, x_i) y_i = \sum_{i=1}^l \left( \frac{1}{N} \sum_{n=1}^N w_n(x, x_i) \right) y_i$$

## Bagging

$\sum_{n=1}^N w_n(x, x_i)$  - количество деревьев, в которых  $x$  и  $x_i$  попали в один лист. Или сходство  $x$  и  $x_i$  в терминах деревьев

Можно понимать RF как способ построения метрики близости на объектах на основе их попадания в одни и те же листья.

## Bagging

# Random Forest

- Bagging над деревьями решений
- Не требует тонкой настройки
- Устойчив к переобучению
- Умеренно интерпретируем
- Позволяет обучать модель распределено

# Boosting

## Boosting

# Gradient Boosted Decision Trees

Bagging: строим базовые алгоритмы независимо друг от друга

Идея: давайте строить базовые алгоритмы последовательно. Причем, строить будем таким образом, чтобы каждый следующий алгоритм корректировал прогнозы всей предыдущей композиции

## Boosting

# Gradient Boosted Decision Trees

Идея: давайте строить базовые алгоритмы последовательно таким образом, чтобы каждый следующий алгоритм корректировал прогнозы всей предыдущей композиции

Рассмотрим простой случай:

$$a(x) = \sum_{n=1}^N b_n(x)$$

$$\sum_{i=1}^l (a(x_i) - y_i)^2 \rightarrow \min_a$$

## Boosting

# Gradient Boosted Decision Trees

Попробуем построить  $b_i$  последовательно так, чтобы каждый следующий алгоритм корректировал ошибки предыдущих.

Шаг 1:

$$b_1(x) = \arg \min_b \sum_{i=1}^l (b(x_i) - y_i)^2$$

Проанализируем прогнозы этой модели:

$b_1(x_1)$	$y_1$
$b_1(x_2)$	$y_2$
...	...
$b_1(x_n)$	$y_n$

Что должен прогнозировать следующий алгоритм, чтобы скорректировать уже построенный?

## Boosting

# Gradient Boosted Decision Trees

Попробуем построить  $b_i$  последовательно так, чтобы каждый следующий алгоритм корректировал ошибки предыдущих.

Шаг 1:

$$b_1(x) = \arg \min_b \sum_{i=1}^l (b(x_i) - y_i)^2$$

Шаг 2: 
$$b_2(x) = \arg \min_b \sum_{i=1}^l (b(x_i) - (y_i - b_1(x_i)))^2$$

$b_1(x_1)$	$y_1$	$b_2(x_1) = y_1 - b_1(x_1)$
$b_1(x_2)$	$y_2$	$b_2(x_2) = y_2 - b_1(x_2)$
...	...	...
$b_1(x_n)$	$y_n$	$b_2(x_n) = y_n - b_1(x_n)$



## Boosting

# Gradient Boosted Decision Trees

Попробуем построить  $b_i$  последовательно так, чтобы каждый следующий алгоритм корректировал ошибки предыдущих.

Шаг 1:

$$b_1(x) = \arg \min_b \sum_{i=1}^l (b(x_i) - y_i)^2$$

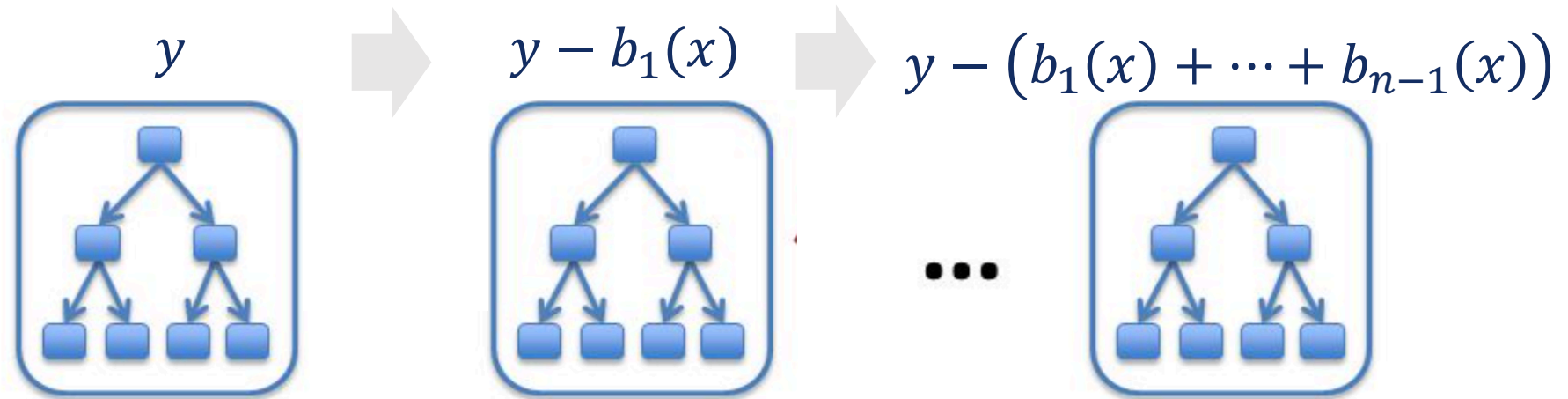
Шаг 2: 
$$b_2(x) = \arg \min_b \sum_{i=1}^l (b(x_i) - (y_i - b_1(x_i)))^2$$

таким образом, если предыдущие модели работают корректно, мы учим следующий приближать 0; в противном случае – приближать разницу

# Gradient Boosted Decision Trees

$$a(x) = b_1(x) + \cdots + b_n(x)$$

Boosting



## Boosting

# Gradient Boosted Decision Trees

Обобщим подход для произвольной дифференцируемой функции потерь  $L$ :

$$a(x) = \sum_{n=1}^N b_n(x)$$

Обозначим  $s_i = y_i - b_k(x_i)$  ошибка на объекте  $i$

Допустим композиция  $a_{n-1}(x)$  уже построена

Нужно построить  $b_N(x)$  так, чтобы:

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

## Boosting

# Gradient Boosted Decision Trees

Прежде чем искать модель  $b_N(x)$  давайте оценим, как стоило бы сделать правку на ответы композиции  $a_{N-1}$ , чтобы уменьшить ошибку

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

Какие  $s_i$  точно уменьшат ошибку?

## Boosting

# Gradient Boosted Decision Trees

Прежде чем искать модель  $b_N(x)$  давайте оценим, как стоило бы сделать правку на ответы композиции  $a_{N-1}$ , чтобы уменьшить ошибку

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

Какие  $s_i$  точно уменьшат ошибку?

Вариант 1:  $s_i = y_i - a_{N-1}(x_i)$  – можно, но такой вариант не учитывает особенности выбранной  $L$ .

Что если, например, перепрогнозирование и недопрогнозирование стоят неодинаково?

## Boosting

# Gradient Boosted Decision Trees

Прежде чем искать модель  $b_N(x)$  давайте оценим, как стоило бы сделать правку на ответы композиции  $a_{N-1}$ , чтобы уменьшить ошибку

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

Какие  $s_i$  точно уменьшат ошибку?

Вариант 1:  $s_i = y_i - a_{N-1}(x_i)$  – нет учета специфики  $L$

Вариант 2:  $s_i = -\frac{\partial L}{\partial z} \big|_{z=a_{N-1}(x_i)}$  - сдвигаем ответ на объекте  $x_i$  в сторону наискорейшего убывания ошибки

## Boosting

# Gradient Boosted Decision Trees

$$s_i = -\frac{\partial L}{\partial z} \Big|_{z=a_{N-1}(x_i)}$$

Вектор частных производных – антиградиент суммарной ошибки

$$\left( -\frac{\partial L}{\partial z} \Big|_{z=a_{N-1}(x_i)} \right) = -\nabla_z \sum_{i=1}^l L(y_i, z_i) \Big|_{z_i=a_{N-1}(x_i)}$$

т.е. взяли  $L$  на всей обучающей выборке и посчитали градиент по всем прогнозам

## Boosting

# Gradient Boosted Decision Trees

$$s_i = -\frac{\partial L}{\partial z} \Big|_{z=a_{N-1}(x_i)}$$

$$\left(-\frac{\partial L}{\partial z} \Big|_{z=a_{N-1}(x_i)}\right) = -\nabla_z \sum_{i=1}^l L(y_i, z_i) \Big|_{z_i=a_{N-1}(x_i)}$$

Мы договорились искать модель  $b_N$  так, чтобы она приближала  $s_i$ . Можем сделать это таким образом:

$$b_N(x) = \arg \min_b \sum_{i=1}^l (b(x_i) - s_i)^2$$

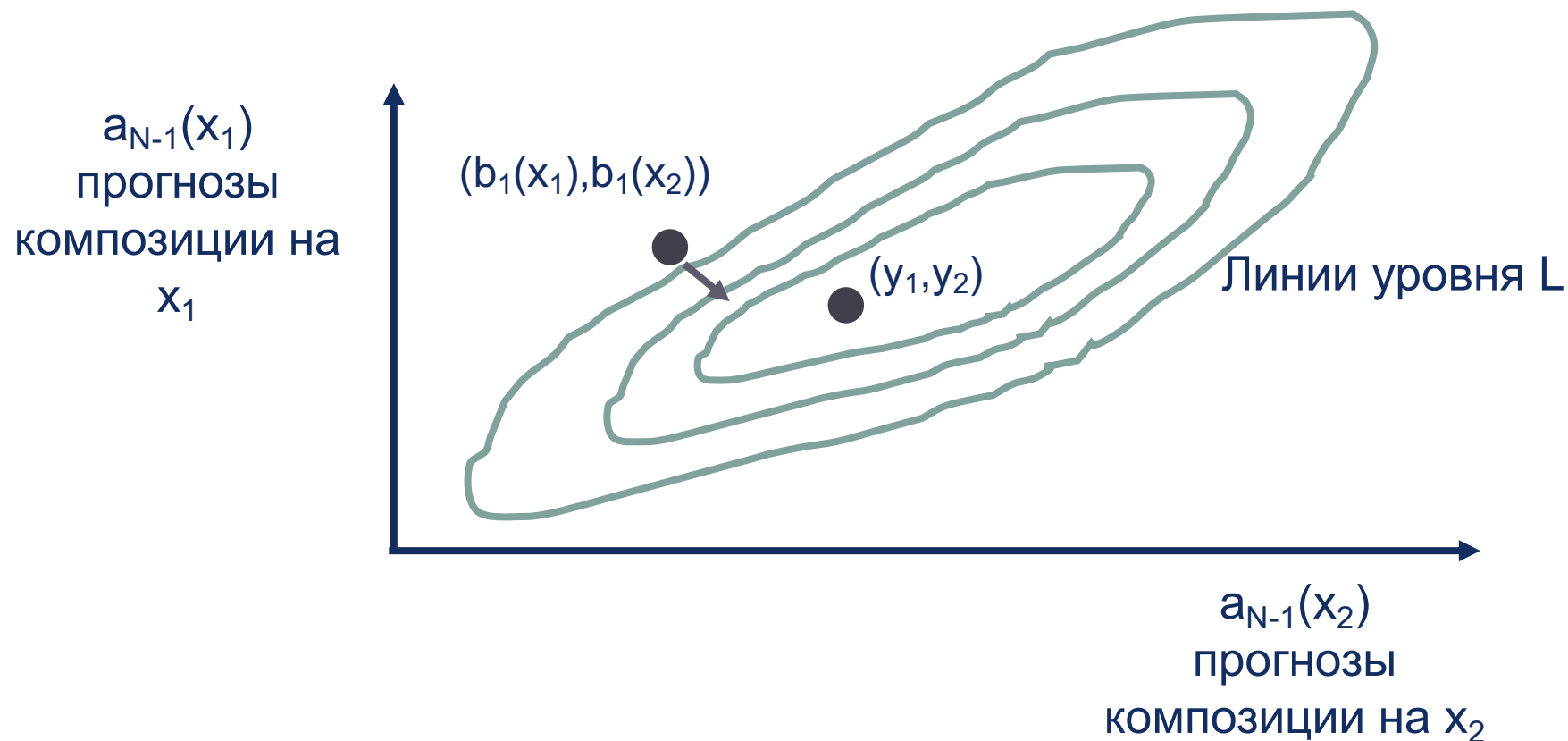
Интересно, что в данном случае нас устраивает квадратичная функция потерь, так как особенности  $L$  уже учтены в  $s$



# Gradient Boosted Decision Trees

Boosting

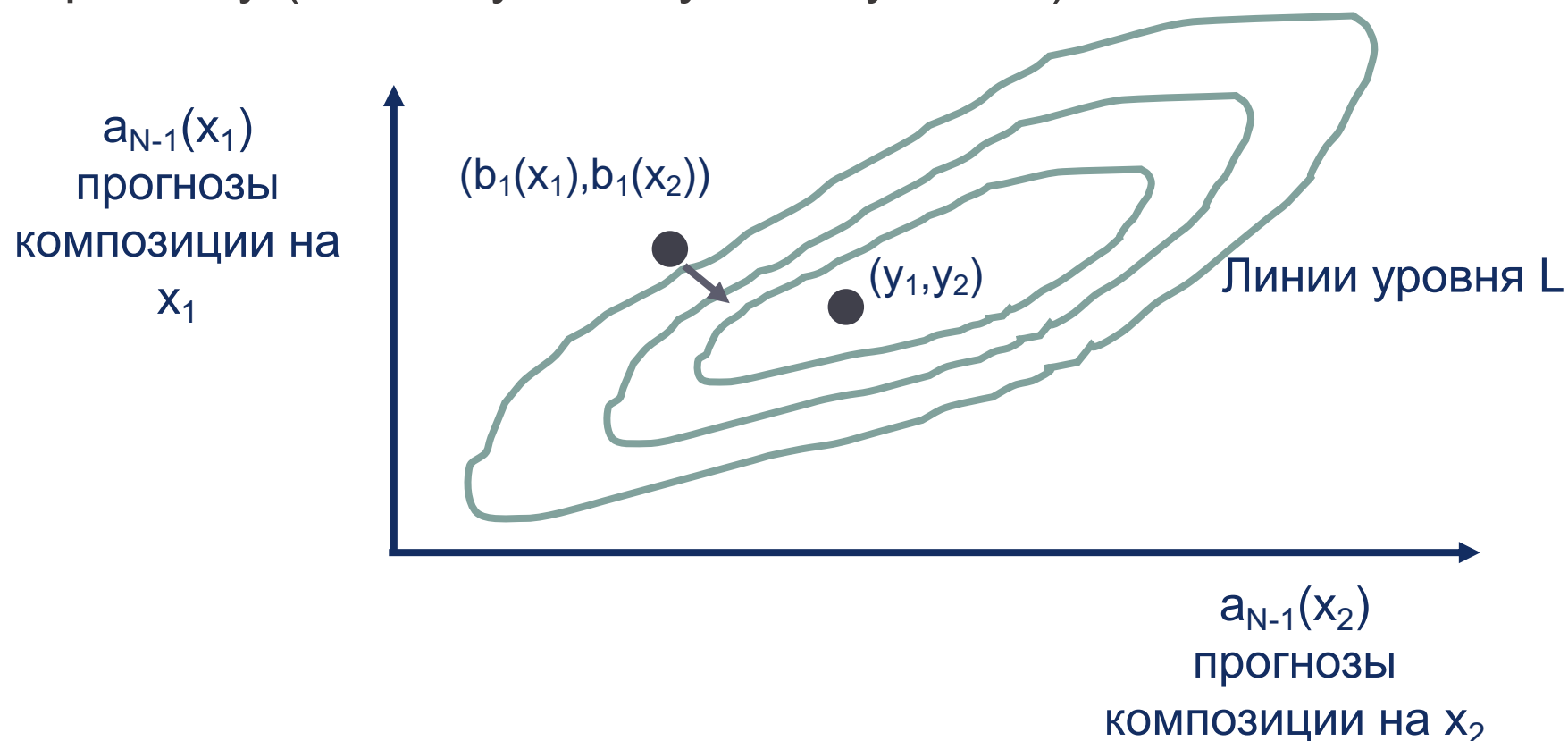
$$\left(-\frac{\partial L}{\partial z}\bigg|_{z=a_{N-1}(x_i)}\right) = -\nabla_z \sum_{i=1}^l L(y_i, z_i)\bigg|_{z_i=a_{N-1}(x_i)}$$



## Boosting

# Gradient Boosted Decision Trees

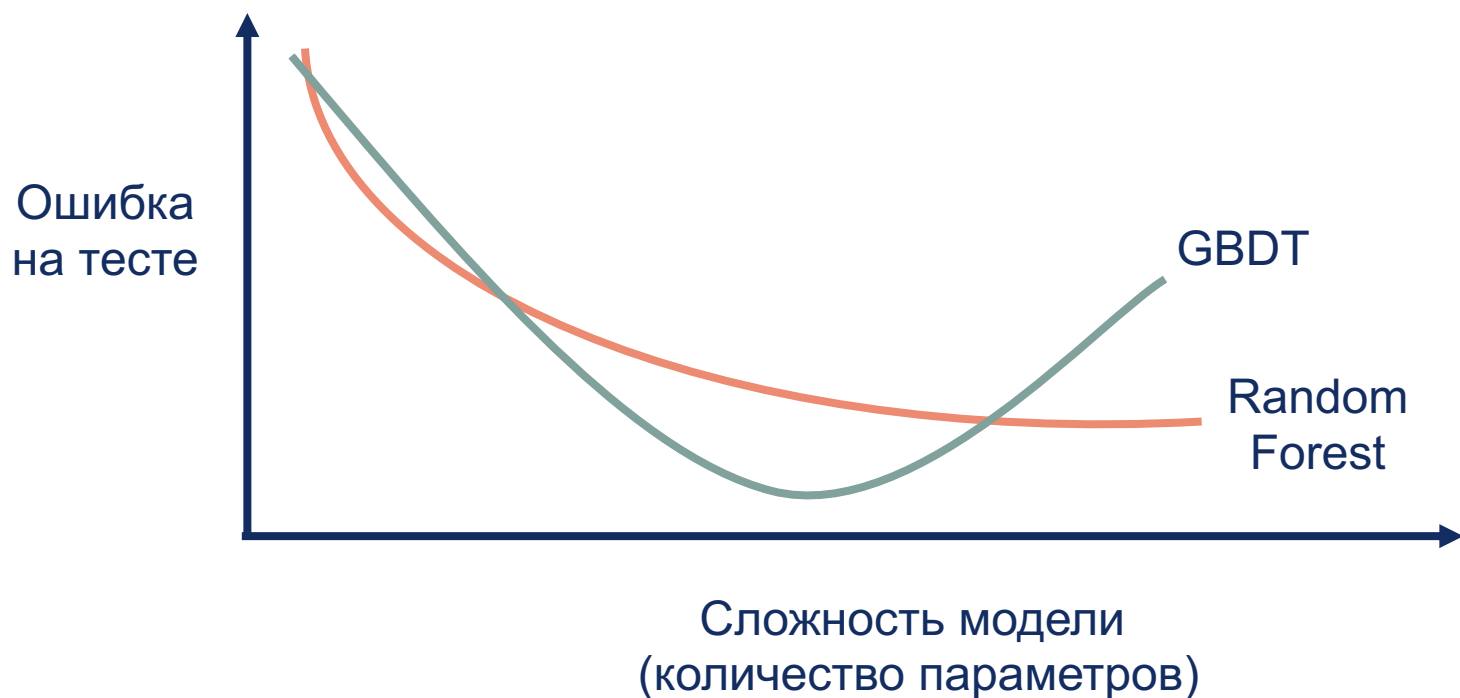
Получается, что мы двигаемся в пространстве прогнозов композиции алгоритмов, где каждый шаг – новый алгоритм, так, чтобы сойтись к оптимальному прогнозу (точному ответу на обучении)



# Boosting

## GBDT: регуляризация

В отличие от random forest склонен к переобучению, однако позволяет получить меньшее смещение (ошибку)



# Boosting

## GBDT: регуляризация

Движение в сторону антиградиента суммарной ошибки:

- С помощью простых моделей
- С помощью сложных моделей

# Boosting

## GBDT: регуляризация

Движение в сторону антиградиента суммарной ошибки:

- С помощью простых моделей: движение в нужную сторону может быть осложнено. Можем идти очень медленно, или вовсе не сможем двигаться по сложной траектории
- С помощью сложных моделей

# Boosting

## GBDT: регуляризация

Движение в сторону антиградиента суммарной ошибки:

- С помощью простых моделей: движение в нужную сторону может быть осложнено. Можем идти очень медленно, или вовсе не сможем двигаться по сложной траектории
- С помощью сложных моделей: первая модель может переобучиться, далее идея корректировки будет неприменима

# Boosting

## GBDT: регуляризация

Движение в сторону антиградиента суммарной ошибки:

- С помощью простых моделей: движение в нужную сторону может быть осложнено: можем идти очень медленно, или вовсе не сможем двигаться по сложной траектории
- С помощью сложных моделей: первая модель может переобучиться, далее идея корректировки будет неприменима

Идея: ограничим шаг!

# Boosting

## GBDT: скорость обучения

Ограничим шаг:

$$a_N(x) = a_{N-1}(x) + \eta b_n(x)$$

$\eta \in (0; 1]$ - длина шага или скорость обучения

Скорость обучения связана с количеством итераций:

- чем меньше скорость обучения, тем больше итераций нужно
- можно получить лучший минимум, хотя и потребуются больше итераций



# Boosting

## GBDT: регуляризация

Движение в сторону антиградиента суммарной ошибки:

- С помощью простых моделей: движение в нужную сторону может быть осложнено. Можем идти очень медленно, или вовсе не сможем двигаться по сложной траектории
- С помощью сложных моделей: первая модель может переобучиться, далее идея корректировки будет неприменима

Идея 2: будем обучать  $b_N$  на подвыборке объектов. Это стохастический градиентный бустинг.

## Boosting

# GBDT: оптимизация ответов в ЛИСТЯХ

Аналитическая запись дерева решений:

$$b_n(x) = \sum_{j=1}^{J_N} b_{Nj} [x \in R_{Nj}]$$

$J_N$  – регионы, на которые дерево разбивает объекты

Запишем GB, подставив в качестве  $b_N$  дерево решений

$$a_N(x) = a_{N-1} + \sum_{j=1}^{J_N} b_{nj} [x \in R_{Nj}]$$

## Boosting

# GBDT: оптимизация ответов в ЛИСТЬЯХ

Запишем GB, подставив в качестве  $b_N$  дерево решений

$$a_N(x) = a_{N-1} + \sum_{j=1}^{J_N} b_{nj} [x \in R_{Nj}]$$

Тогда потери примут вид:

$$\sum_{i=1}^l L(y_i, a_{N-1}(x_i) + \sum_{j=1}^{J_N} b_{Nj} [x_i \in R_{Nj}]) \rightarrow \min_{b_{Nj} | j=1, J_N}$$

Идея: не меняя структуру дерева, скорректировать ответ в листе так, чтобы он был оптимален с точки зрения  $L$

Это возможно, так как мы строим деревья оптимизируя квадрат отклонения от смещения, а не саму  $L$

## Boosting

# GBDT: оптимизация ответов в листьях

Не меняя структуру дерева, скорректировать ответ в листе так, чтобы он был оптимален с точки зрения  $L$

$$\sum_{i=1}^l L(y_i, a_{N-1}(x_i) + \sum_{j=1}^{J_N} b_{Nj} [x_i \in R_{Nj}]) \rightarrow \min_{b_{Nj} | j=1, J_N}$$

Для того, чтобы удобнее сделать правки в листьях сведем многомерную задачу к  $J_N$  одномерных, оптимизируя по каждому листу ( $x_i \in R_{Nj}$ ) отдельно

$$\sum_{x_i \in R_{Nj}} L(y_i, a_{N-1}(x_i) + b_{Nj}) \rightarrow \min_{b_{Nj}}$$

Эти задачи решать легко, так как  $b_{Nj}$  — число (ответ в листе) и каждая задача одномерная

## Boosting

# Gradient Boosted Decision Trees

- Очень универсальный
- Позволяет точно приблизить восстанавливаемую функцию или разделяющую поверхность классов
- Умеренно интерпретируем
- Композиции могут содержать десятки тысяч базовых моделей и долго обучаться
- Переобучение на выбросах при избыточном количестве классификаторов

# Обзор композиций

# Ансамбли моделей

## Способы комбинирования моделей

- Bagging
- Boosting
- Stacking
- Blending

# Ансамбли моделей

## Bagging

Bagging = Bootstrap aggregation

№	значение
1	
2	
3	
N	

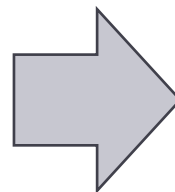


# Ансамбли моделей

## Bagging

Bagging = Bootstrap aggregation

№	значение
1	
2	
3	
N	



№	значение
1	
25	
1	

№	значение
67	
24	
13	

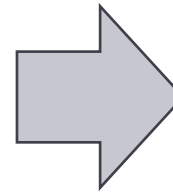
№	значение
9	
9	
9	

# Ансамбли моделей

## Bagging

Bagging = Bootstrap aggregation

№	значение
1	
2	
3	
N	



№	значение
1	
25	
1	


№	значение
67	
24	
13	


№	значение
9	
9	
9	

По схеме выбора с возвращением, генерируем  $M$  обучающих выборок такого же размера, обучаем на них модели и усредняем результат

# Boosting

Бустинг – жадное построение взвешенной суммы базовых алгоритмов  $b_k(x)$


$$a(x) = \eta_1 b_1(x)$$


$$a(x) = \eta_1 b_1(x) + \eta_2 b_2(x)$$




...


# Ансамбли моделей

## Boosting

Бустинг – жадное построение взвешенной суммы базовых алгоритмов  $b_k(x)$

$$a(x) = \sum_{t=1}^T \eta_t b_t(x)$$


$$a(x) = \eta_1 b_1(x)$$


$$a(x) = \eta_1 b_1(x) + \eta_2 b_2(x)$$



...


# Ансамбли моделей


## Boosting

Бустинг – жадное построение взвешенной суммы базовых алгоритмов  $b_k(x)$

$$a(x) = \sum_{t=1}^T \eta_t b_t(x)$$

$b_k(x)$  – как правило, решающие деревья небольшой глубины или линейные модели


$$a(x) = \eta_1 b_1(x)$$


$$a(x) = \eta_1 b_1(x) + \eta_2 b_2(x)$$



...

# Ансамбли моделей

# Stacking

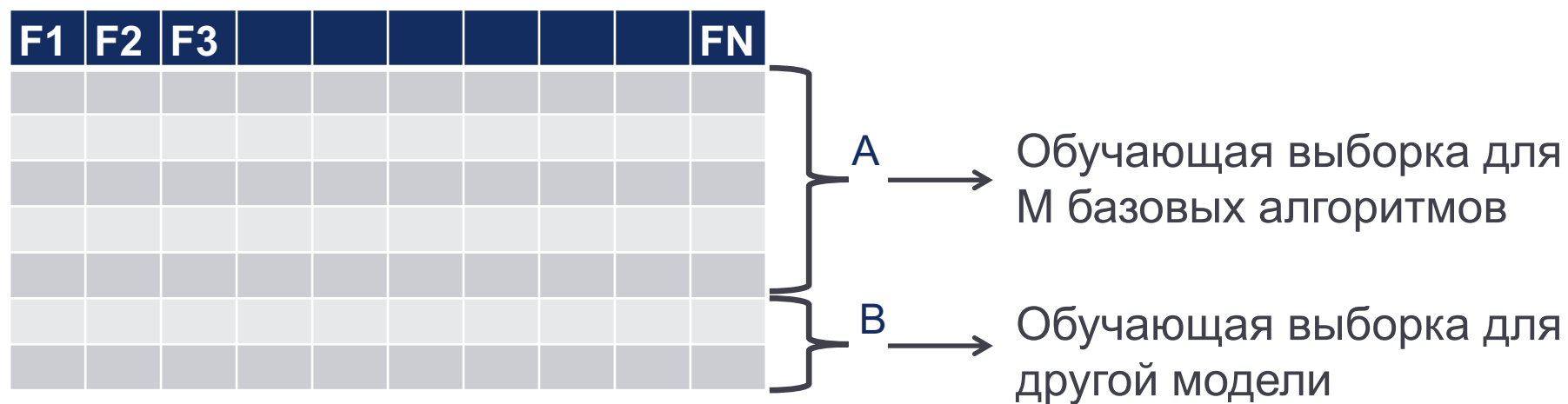
## Обучающая выборка:

[illegible]

# Ансамбли моделей

## Stacking

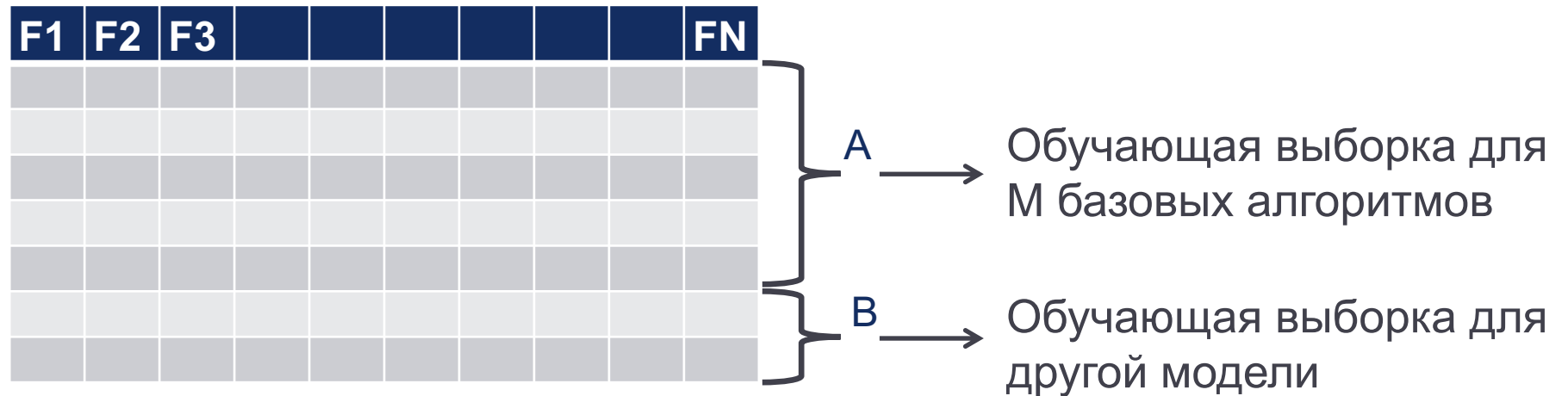
Обучающая выборка:



# Ансамбли моделей

## Stacking

Обучающая выборка:



Обучаем M базовых  
алгоритмов на  
выборке A



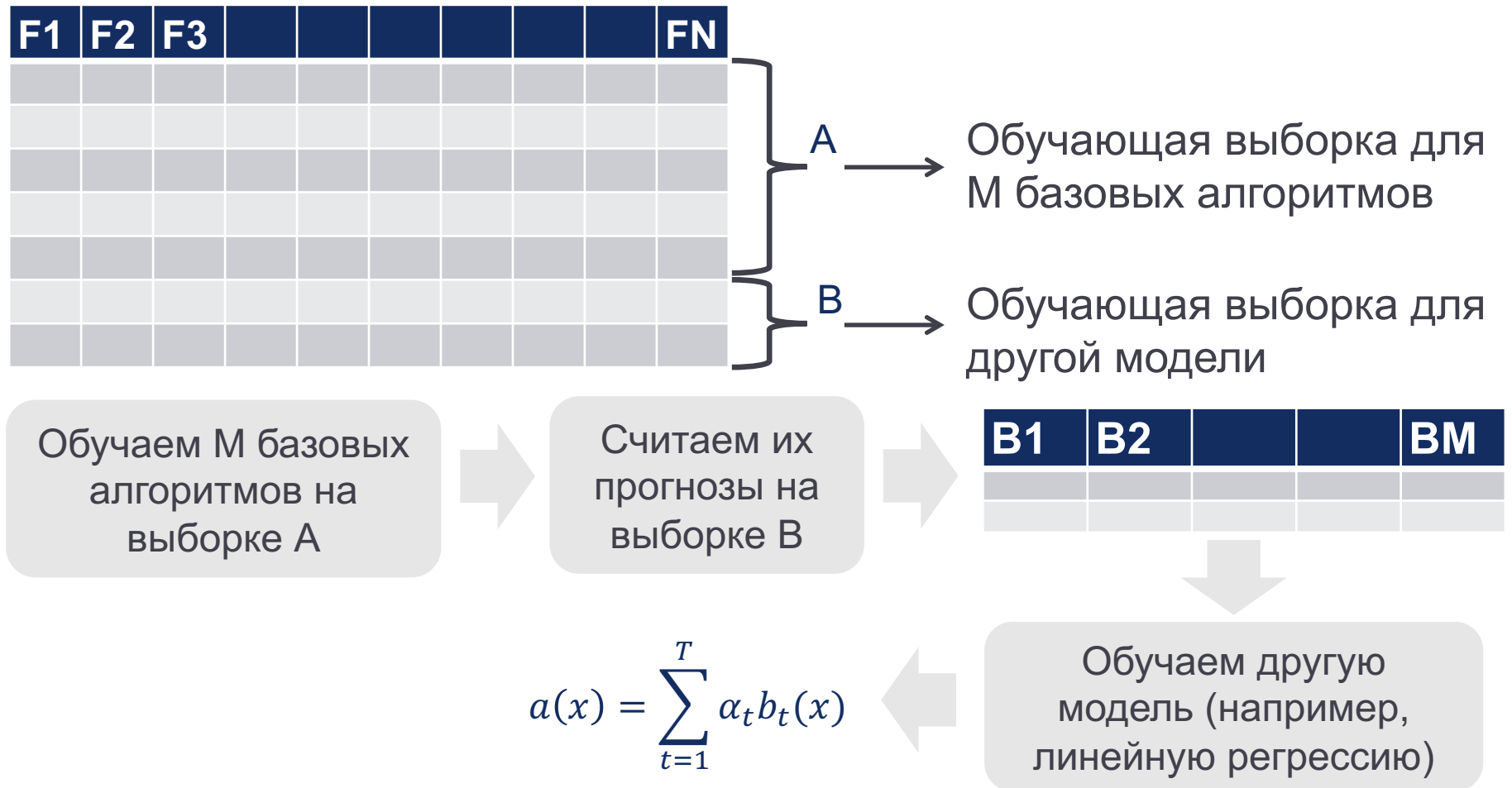
Считаем их  
прогнозы на  
выборке B



# Ансамбли моделей

## Stacking

Обучающая выборка:



# Ансамбли моделей

## Blending

Смесь нескольких сильных классификаторов:

$$a(x) = \sum_{t=1}^T \alpha_t b_t(x)$$

+ веса неотрицательны и дают в сумме единицу

## Ансамбли моделей

# Blending

Смесь нескольких сильных классификаторов:

$$a(x) = \sum_{t=1}^T \alpha_t b_t(x)$$

+ веса неотрицательны и дают в сумме единицу

### Преимущества и недостатки:

- Очень прост идейно, хорошо работает
- Иногда надо перебирать веса или использовать дискретную оптимизацию
- Не всегда композиция в виде взвешенной суммы — то, что надо. Иногда нужна более сложная композиция

# Машинное обучение: КОМПОЗИЦИИ МОДЕЛЕЙ

Спасибо!  
Эмели Драль