Ансамбли

Ансамбли

- Деревья решений могут восстанавливать очень сложные закономерности, но при этом неустойчивы к малейшим изменениям в данных
- Поэтому деревья решений самостоятельно используются редко, но в составе *ансамблей* (*композиций*, *комитетов*) показывают хорошие результаты

Ансамбли

Основные подходы:

- Усреднение – несколько моделей строятся независимо, а их ответы усредняются
  - Бэггинг (bagging), случайные леса (random forests)
- Бустинг – модели строятся последовательно, и каждая следующая модель исправляет ошибки предыдущей
  - AdaBoost, градиентный бустинг
- Стекинг (stacking) – использование в качестве признаков предсказаний базовых классификаторов
- Голосование (voting)

Bagging

- Бэггинг (bagging = bootstrap aggregation) – независимо строит несколько моделей и усредняет их ответы
  - Breiman Leo. Bagging Predictors // Technical Report No. 421. 1994
- Модели строятся на данных, полученных при помощи бутстрэпа
- Бутстрэп (bootstrap) – способ многократной генерации выборок на основе имеющейся выборки методом Монте-Карло:
  - Пусть дана выборка $X = (x_i, y_i)$
  - Равномерно возьмем из выборки ℓ объектов с возвращением
  - Из-за возвращения среди них окажутся повторы
  - Обозначим новую подвыборку через X_1
  - Повторив процедуру N раз, сгенерируем N подвыборок $X_1, \ldots, X_N$

Bagging

- Пусть имеется некоторый метод обучения $\mu(X)$
- Построим на его основе метод $\tilde{\mu}(X)$, который генерирует случайную подвыборку $\tilde{X}$ с помощью бутстрэпа и подает ее на вход метода μ:

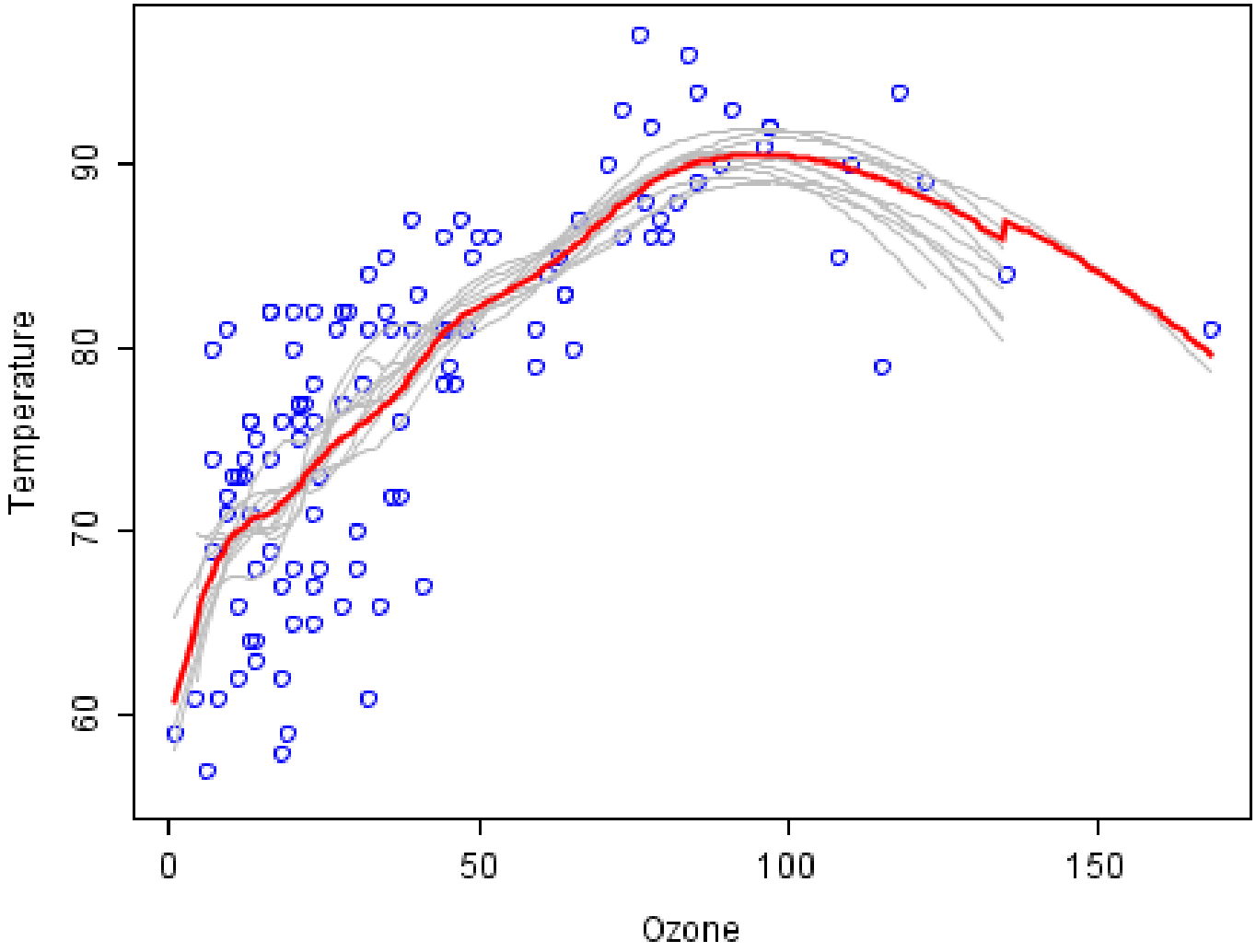
$$\tilde{\mu}(X) = \mu(\tilde{X})$$

- В бэггинге обучается некоторое число базовых алгоритмов b_n с помощью метода $\tilde{\mu}$, и строится итоговая композиция как среднее базовых алгоритмов:

$$a_N(x) = \frac{1}{N}\sum_{n=1}^{N} b_n(x) = \frac{1}{N}\sum_{n=1}^{N} \tilde{\mu}_n(X)(x)$$

Bagging

Позволяет снизить дисперсию
и борется с переобучением



Bagging – scikit-learn

```
class sklearn.ensemble.BaggingClassifier(
      estimator=None,       # DecisionTreeClassifier
      n_estimators=10,
      *,
      max_samples=1.0,
      max_features=1.0,
      bootstrap=True,       # Whether samples are drawn with replacement
      bootstrap_features=False,
      oob_score=False,      # use out-of-bag samples
      warm_start=False,
      n_jobs=None,
      random_state=None,
      verbose=0)
```

Bagging – scikit-learn

```
class sklearn.ensemble.BaggingRegressor (
      estimator=None,       # DecisionTreeRegressor
      n_estimators=10,
      *,
      max_samples=1.0,
      max_features=1.0,
      bootstrap=True,       # Whether samples are drawn with replacement
      bootstrap_features=False,
      oob_score=False,      # use out-of-bag samples
      warm_start=False,
      n_jobs=None,
      random_state=None,
      verbose=0)
```

Random forests

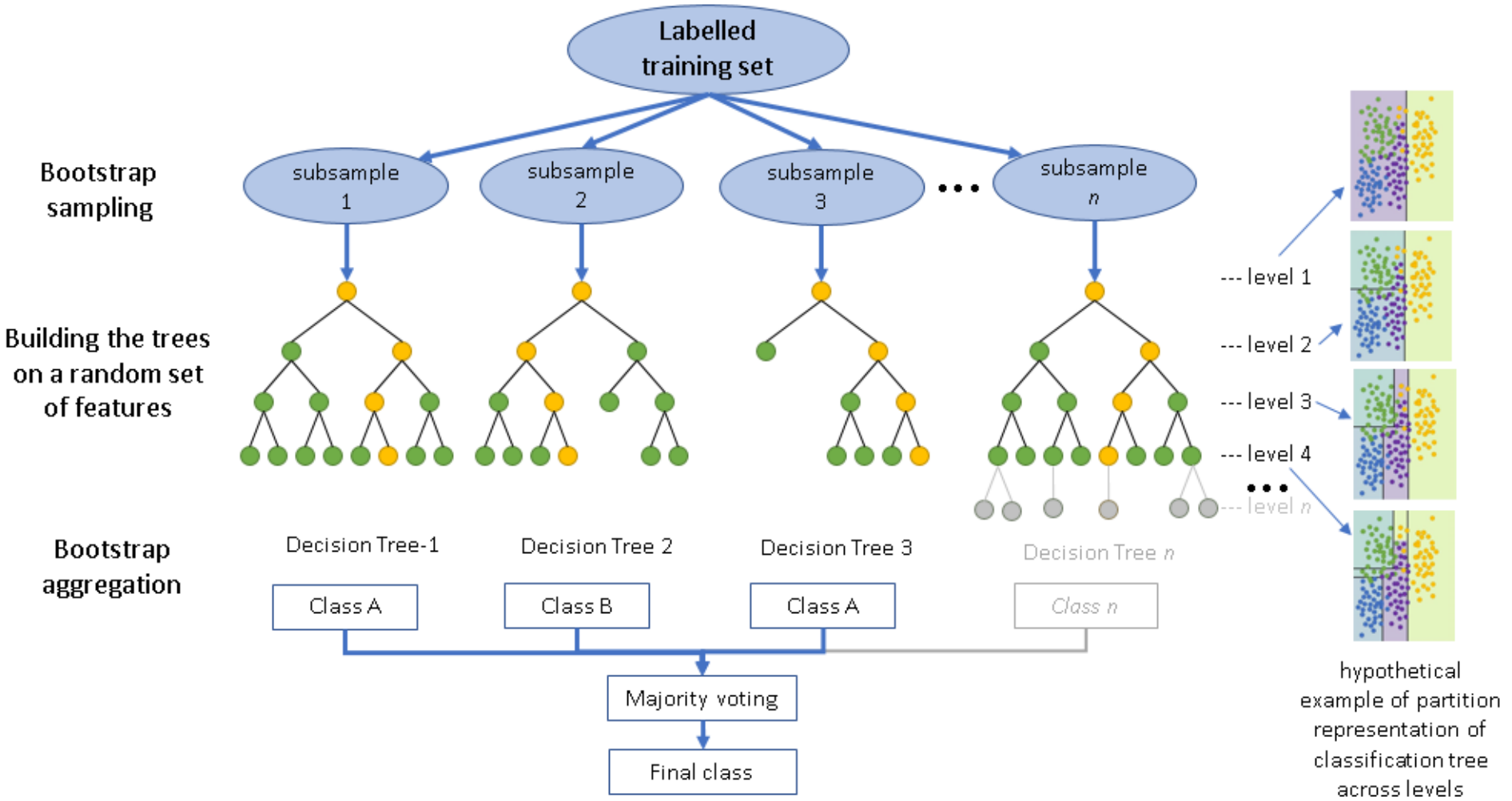
- Метод случайных лесов (random forests) основан на бэггинге над решающими деревьями и методе случайных подпространств (random subspace)
  - Breiman Leo. Random Forests // Machine Learning. 2001
  - Ho Tin Kam. Random Decision Forests // 3rd International Conference on Document Analysis and Recognition, 1995
    - Random subspace method
- Вводятся два источника случайности:
  - случайная выборка обучающих объектов (бутстрэп)
  - случайное подмножество признаков при выборе оптимального разбиения в процессе построения дерева решений

Random forests

- Для $n = 1, \dots, N$:
  - Сгенерировать выборку $\tilde{X}_n$ на основе бутстрэпа
  - Построить дерево решений $b_n(x)$ по выборке $\tilde{X}_n$:
    - Дерево строится, пока в каждом листе не окажется n_{min} или меньше объектов
    - При каждом разбиении сначала выбирается m случайных признаков и оптимальное разбиение ищется только среди них
- Вернуть ансамбль:

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

Random forests



Random forests

Рекомендации (d – число признаков):

- в задачах классификации: $m = \lfloor \sqrt{d} \rfloor$ (в scikit-learn именно так)
- в задачах регрессии: $m = \lfloor d/3 \rfloor$ (в scikit-learn $m = d$)

Отличия от Bagging:

- базовая модель – всегда дерево
- случайные подмножества признаков на каждом разбиении

Random forests – scikit-learn

```
class sklearn.ensemble.RandomForestClassifier(
      n_estimators=100,     # The number of trees
      *,
      criterion='gini',     # 'gini', 'entropy', 'log_loss'
      max_depth=None,       # None -> pure leaves or min_samples_split
      min_samples_split=2,  # n_min – minimum number of samples to split
      min_samples_leaf=1,
      max_features='sqrt',  # max_features=sqrt(n_features)
      bootstrap=True,       # Whether samples are drawn with replacement
      oob_score=False,      # use out-of-bag samples
      n_jobs=None,
      random_state=None,
      verbose=0,
      warm_start=False,
      class_weight=None,
      max_samples=None)     # X.shape[0]
```

Random forests – scikit-learn

```
class sklearn.ensemble.RandomForestRegressor(
      n_estimators=100,     # The number of trees
      *,
      criterion='squared_error',
      max_depth=None,       # None -> pure leaves or min_samples_split
      min_samples_split=2,  # n_min – minimum number of samples to split
      min_samples_leaf=1,
      max_features=1.0,
      bootstrap=True,       # Whether samples are drawn with replacement
      oob_score=False,      # use out-of-bag samples
      n_jobs=None,
      random_state=None,
      verbose=0,
      warm_start=False,
      max_samples=None)     # X.shape[0]
```

AdaBoost

- AdaBoost (Adaptive Boosting) – последовательное построение моделей, каждая из которых исправляет ошибки предыдущей
  - Freund Yoav, Schapire Robert. A decision-theoretic generalization of on-line learning and an application to boosting // European Conference on Computational Learning Theory. 1995
- В качестве базовых моделей используются *слабые классификаторы* (weak learner) – распознают объекты немного лучше, чем случайное угадывание
  - Как правило, используется дерево решений с высотой 1 (decision stump – «решающий пень»)

AdaBoost

- Входные данные: $(x_1, y_1), \ldots, (x_m, y_m)$, $x_i \in X$, $y_i \in Y = \{-1, +1\}$
- Инициализация весов: $D_1(i) = 1/m$, $i = 1, \ldots, m$
- Для $t = 1 \ldots T$:
  1. Обучение слабого классификатора h_t с использованием распределения D_t
  2. Вычисление ошибки для h_t:

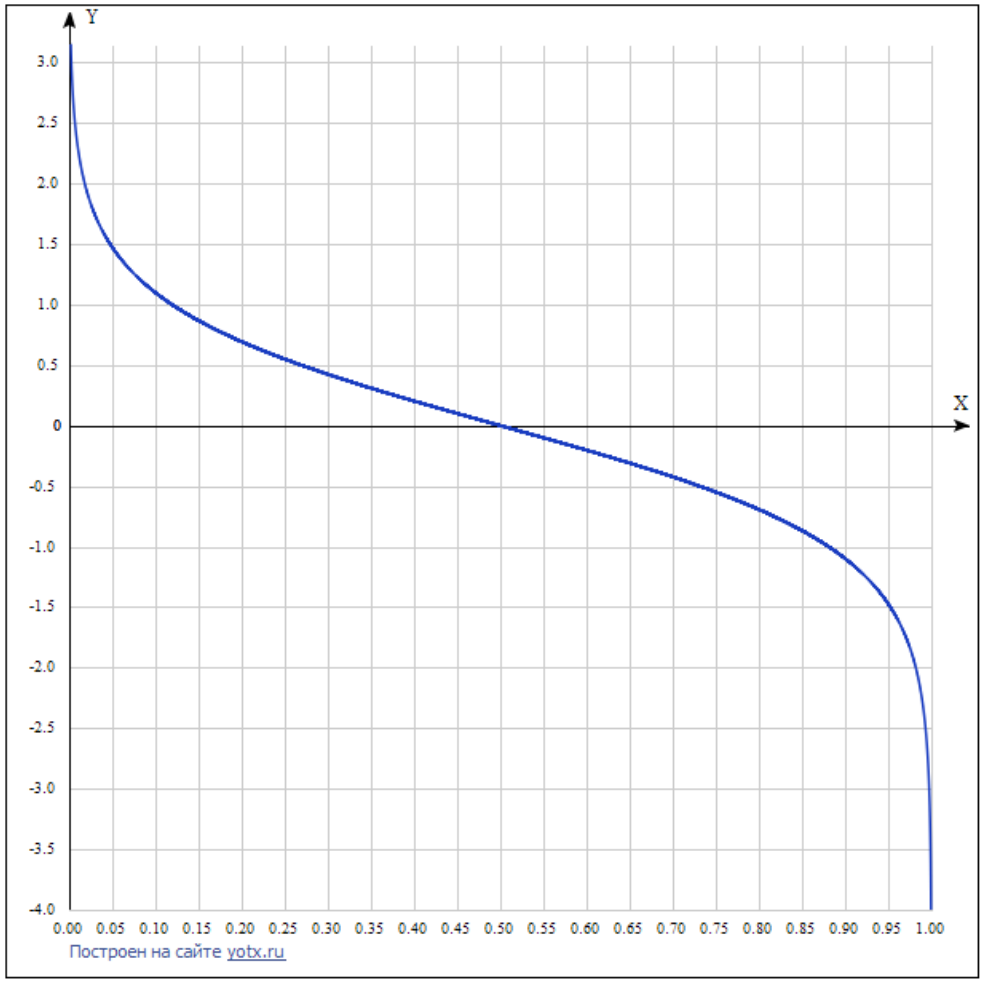
$$\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$$

  3. Если $\epsilon_t \geq 0.5$, то останов (слабый классификатор должен давать не менее 50% правильных ответов)

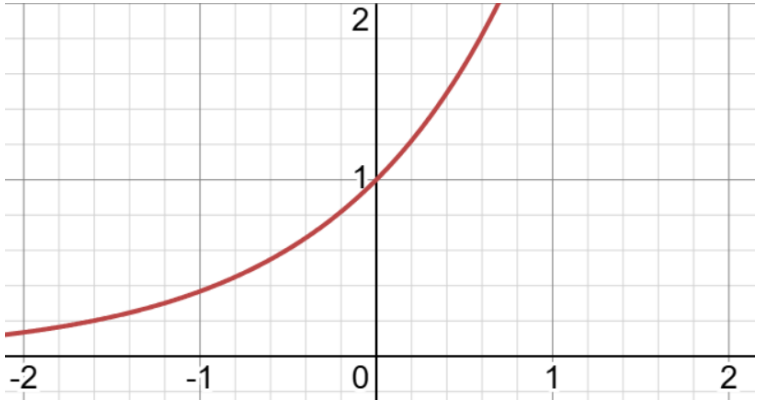
AdaBoost

4. Вычисление уверенности классификатора:

$$\alpha_t = \frac{1}{2} ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$



AdaBoost



5. Обновление весов:

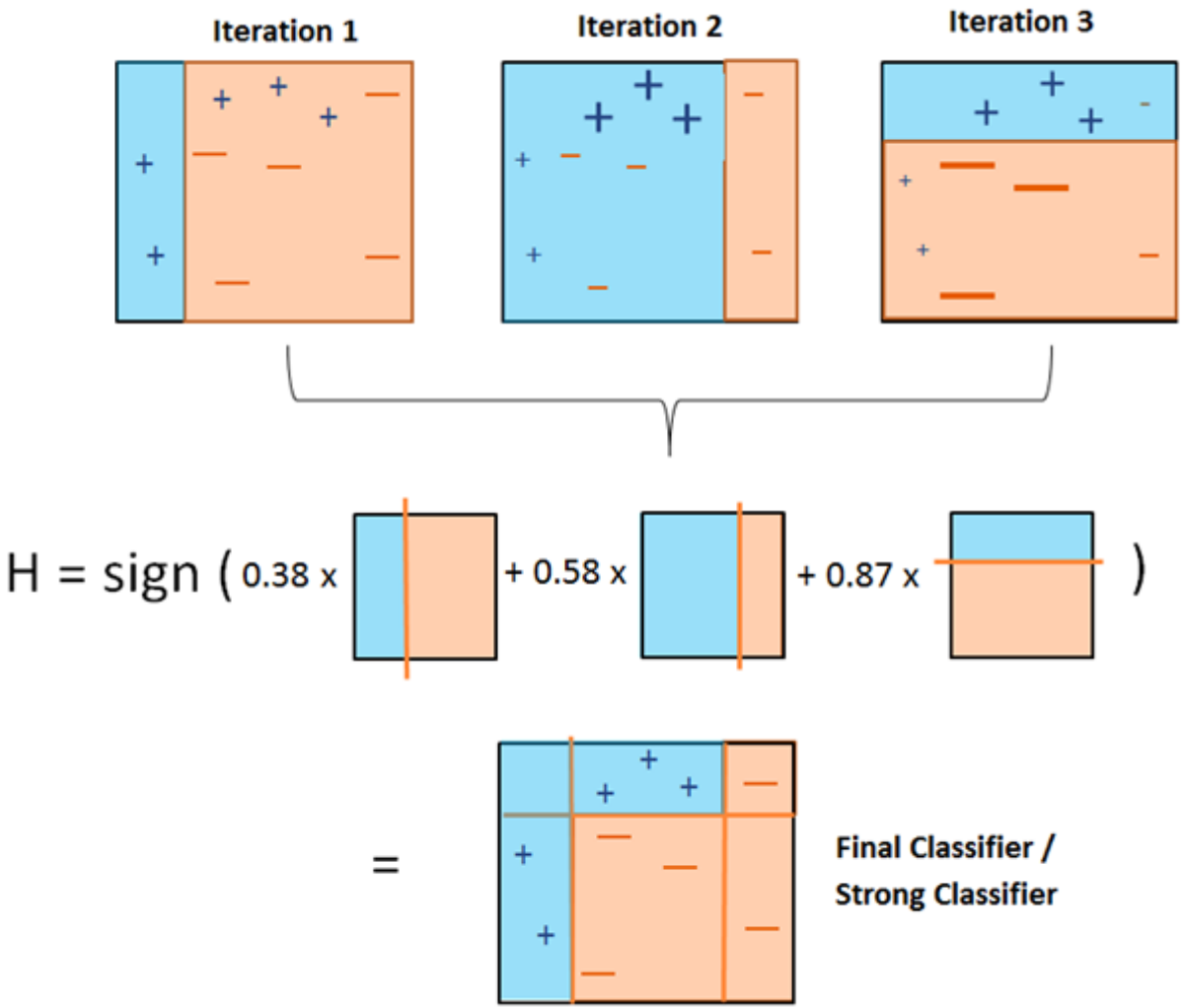
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} \text{ если } h_t(x_i) = y_i \\ e^{\alpha_t} \text{ если } h_t(x_i) \neq y_i \end{cases} = \frac{D_t(i)}{Z_t} e^{-\alpha_t y_i h_t(x_i)}$$

где Z_t – нормализующий параметр, выбранный так, чтобы D_{t+1} являлось распределением вероятностей, т.е. $\sum_{i=1}^{m} D_{t+1}(i) = 1$

6. Результирующий классификатор:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

AdaBoost


Iteration 1
Iteration 2
Iteration 3
H = sign (0.38 x
+ 0.58 x
+ 0.87 x
)
=
Final Classifier /
Strong Classifier

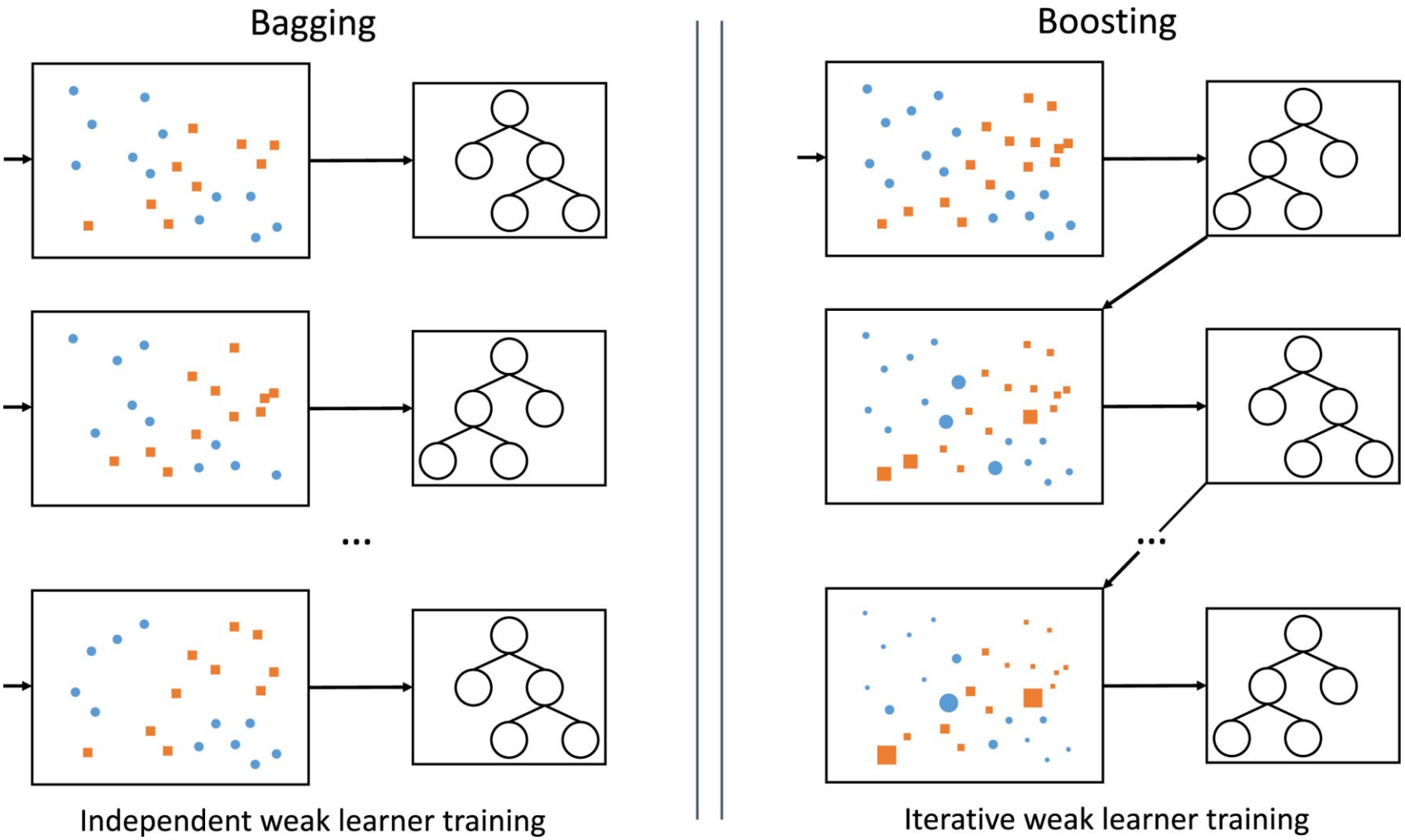
AdaBoost – scikit-learn

```
class sklearn.ensemble.AdaBoostClassifier(
      estimator=None,       # DecisionTreeClassifier(max_depth=1)
      *,
      n_estimators=50,
      learning_rate=1.0,
      algorithm='SAMME.R', # Zhu et al. Multi-class AdaBoost (2009)
      random_state=None
)
```

AdaBoost – scikit-learn

```
class sklearn.ensemble.AdaBoostRegressor(
      estimator=None,        # DecisionTreeRegressor(max_depth=3)
      *,
      n_estimators=50,
      learning_rate=1.0,
      loss='linear',        # 'linear', 'square', 'exponential'
      random_state=None
)
```

Bagging vs. Boosting



Градиентный бустинг

- Градиентный бустинг (Gradient Boosting Machine, GBM) – обобщение бустинга на произвольные (дифференцируемые) функции потерь
- Friedman Jerome. Greedy Function Approximation: A Gradient Boosting Machine // Annals of Statistics, 2001 (draft 1999)
- На практике используется градиентный бустинг над деревьями решений (Gradient Boosting over Decision Trees, GBDT)

https://education.yandex.ru/handbook/ml/article/gradientnyj-busting

Градиентный бустинг

- Пусть дана дифференцируемая функция потерь $L(y, z)$
- Будем строить ансамбль как взвешенную сумму базовых моделей:

$$a_N(x) = \sum_{n=0}^{N} \gamma_n b_n(x)$$

- Начальная модель $b_0(x)$ (лист):
  - $b_0(x) = 0$
  - $b_0(x) = \arg\max_{y \in \mathbb{Y}} \sum_{i=1}^{l} [y_i = y]$ – в задачах классификации
  - $b_0(x) = \frac{1}{l} \sum_{i=1}^{l} y_i$ – в задачах регрессии
  - $\gamma_0 = 1$

Градиентный бустинг

- Пусть построен ансамбль $a_{N-1}(x)$ из $N-1$ модели и нужно выбрать следующую базовую модель $b_N(x)$ так, чтобы минимизировать ошибку:

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

- Задачу выбора функции $\gamma_N b_N(x_i)$ сводим к задаче выбора числовых значений $s_1, \dots, s_l$ (*остатков*), которые должна аппроксимировать эта функция

Градиентный бустинг

- Рассмотрим l+1-мерное пространство, в котором функционал ошибки зависит от предсказаний текущего ансамбля $a_{N-1}(x)$
  - Каждая ось соответствует обучающему примеру
  - Значения на оси соответствуют предсказаниям текущего ансамбля для данного примера
- В градиентном бустинге значения $s_1, \dots, s_l$ выбираются равными антиградиенту функционала ошибок в этом пространстве:

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

- Таким образом, осуществляется один шаг градиентного спуска

Градиентный бустинг

- Требуется построить функцию $b_N(x)$ такую, которая аппроксимирует значения $s_1, \dots, s_l$ – задача обучения с учителем
- Можно использовать в качестве функционала среднеквадратичную ошибку:

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

- После нахождения $b_N(x)$ определяем γ_N:

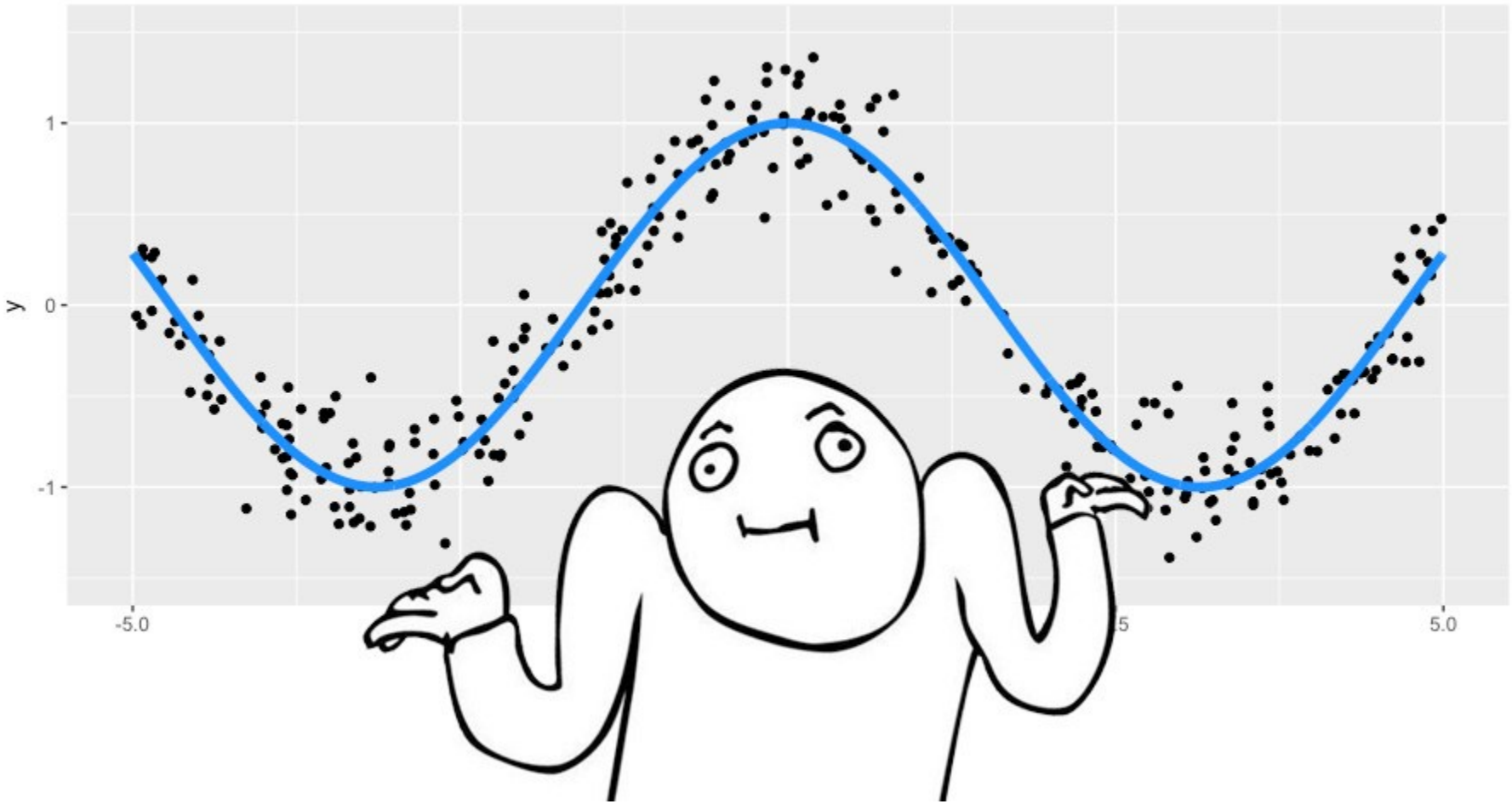
$$\gamma_N = \arg\min_\gamma \sum_{i=1}^{l} L\big(y_i, a_{N-1}(x_i) + \gamma b_N(x_i)\big)$$

Градиентный бустинг: функции потерь

- Регрессия:
  - Среднеквадратическая: $L(y,z) = \frac{1}{2}(y-z)^2$
    - $L'(y,z) = z - y$
  - Модуль отклонения: $L(y,z) = |y-z|$
    - $L'(y,z) = -sign(z-y)$
- Классификация:
  - Логистическая: $L(y,z) = \log(1+e^{-yz})$
    - $L'(y,z) = -\frac{y}{1+e^{-yz}}$
  - Экспоненциальная (AdaBoost): $L(y,z) = e^{-yz}$
    - $L'(y,z) = -ye^{-yz}$

Градиентный бустинг: пример

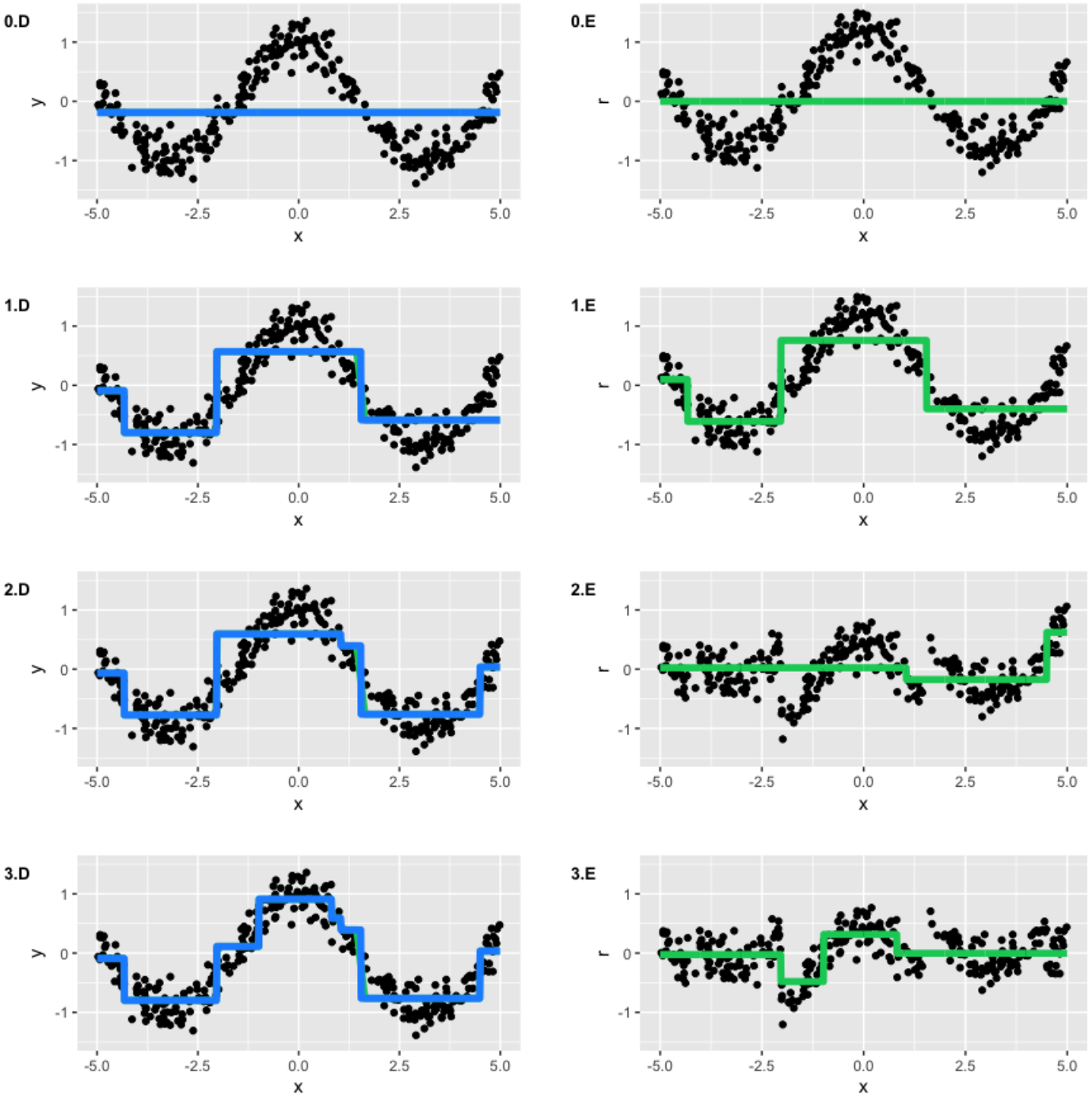
$$y = \cos(x) + \epsilon\,, \epsilon \sim \mathcal{N}\left(0, \frac{1}{5}\right), x \in [-5, 5]$$



https://habr.com/ru/company/ods/blog/327250

Градиентный бустинг: пример

- Данные: $\{(x_i, y_i)\}_{i=1,\ldots,300}$
- Число итераций: $M = 3$
- Среднеквадратическая функция потерь: $L(y, z) = \frac{1}{2}(y - z)^2$
- Градиент функции потерь (остатки): $s = z - y$
- Базовые алгоритмы: деревья решений
- Гиперпараметры: глубина деревьев равна 2


0.D
0.E
1.D
1.E
2.D
2.E
3.D
3.E
y
r
x
1
0
-1
-5.0
-2.5
0.0
2.5
5.0

Градиентный бустинг

Реализации градиентного бустинга:

- LightGBM (Microsoft)
  - https://habr.com/ru/companies/tochka/articles/751012
- Scikit-learn:
  - GradientBoostingClassifier, GradientBoostingRegressor
  - HistGradientBoostingClassifier, HistGradientBoostingRegressor
    - на основе LightGBM
    - быстрее, если в датасете десятки тысяч примеров
    - встроенная обработка пропущенных значений и категориальных признаков
- XGBoost (University of Washington)
- CatBoost (Yandex)
  - https://habr.com/ru/companies/tochka/articles/751012

Голосование (voting)

- VotingClassifier
  - Majority / Hard Voting
  - Weighted Average Probabilities / Soft Voting
- VotingRegressor

Stacking

- StackingClassifier
- StackingRegressor