Линейные модели

Легченко Антон 14.07.25

Антон Легченко

Korona tech, Старший дата-аналитик

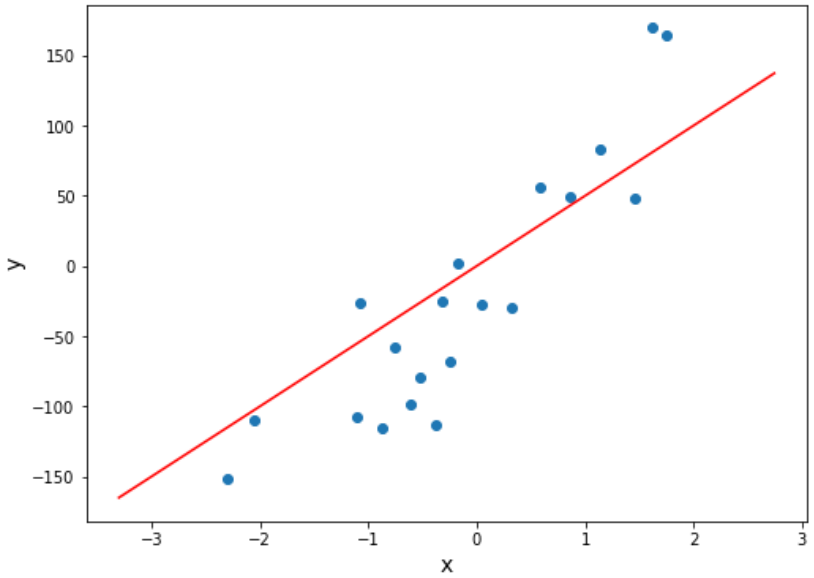
- Работаю в R&D 3 года
- Аспирант ММФ НГУ
- Работаю преймущественно с NLP, распознаванием речи
- Занимал 1ое место на AIJ 24 и Avito ML cup 25

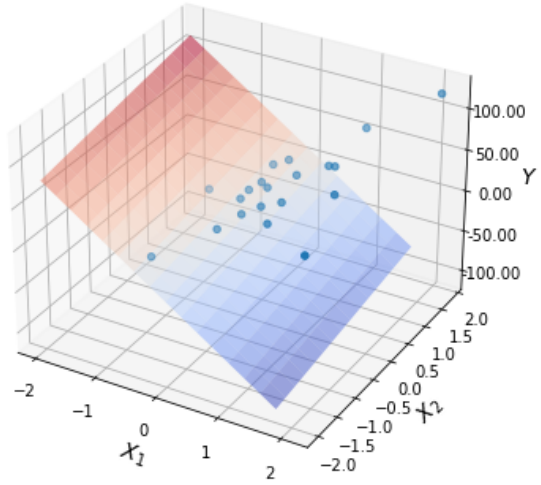
Линейные методы обучения с учителем

- Модель: линейная комбинация признаков

$$y = w_0 + w_1 x_1 + \ldots + w_n x_n$$

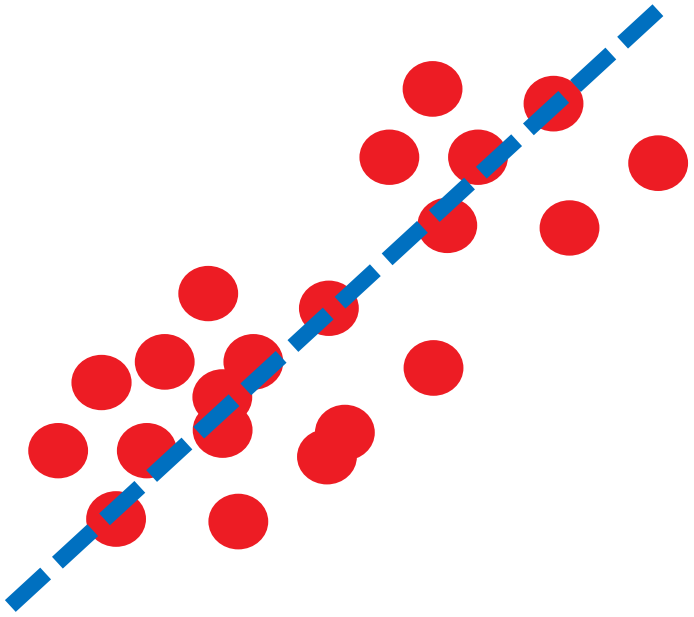
- Примеры:
  - Линейная регрессия – регрессия
  - Логистическая регрессия – классификация



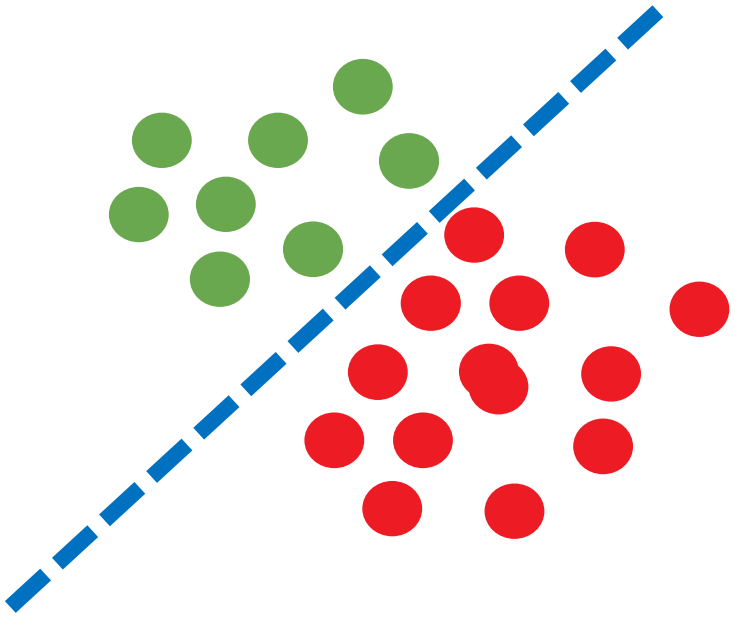


Регрессия / Классификация

Регрессия



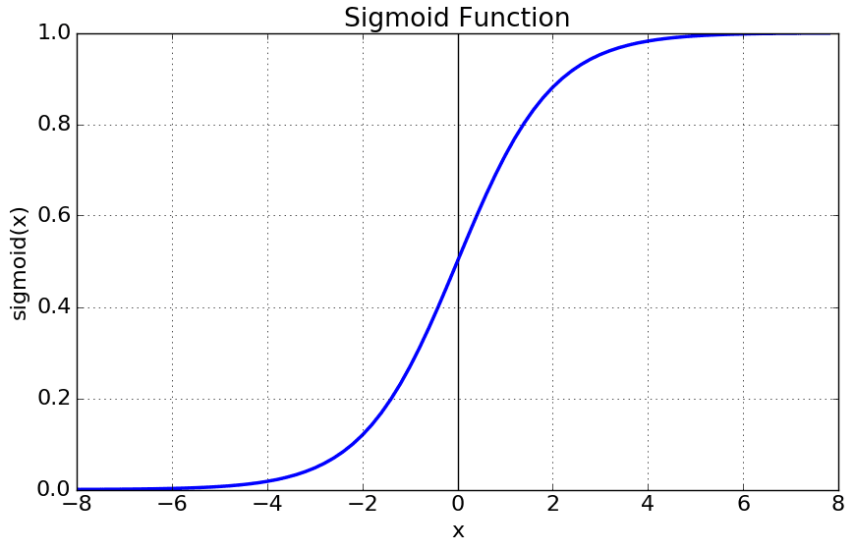
Классификация



Логистическая регрессия

- Модель для решения задачи классификации
- Прогнозирует вероятность P(y=1|x)
- Использует логистическую функцию (сигмоиду)

$$P(y=1|x) = \sigma(w^T x) = \frac{1}{1+e^{-w^T x}}$$



Функция правдоподобия для логистической регрессии

- Вероятностный смысл логистической регрессии: максимизация правдоподобия

$$L(w) = \prod_{i=1}^{N} P(y_i|x_i, w)$$

- Чаще максимизируют логарифм правдоподобия (log-likelihood), чтобы упростить вычисления:

$$\log L(w) = \sum_{i=1}^{N} [y_i \log \sigma(w^T x_i) + (1 - y_i) \log(1 - \sigma(w^T x_i))]$$

Многоклассовая классификация

One-vs-Rest (OvR, «один против всех»)

- Для задачи с K классами обучается K отдельных бинарных классификаторов.
- Каждый классификатор отличает один класс от остальных.
- Для нового объекта выбирается класс с наибольшим значением отклика

Softmax

- Обобщение логистической функции на более чем два класса
- Для объекта *x* вероятность принадлежности к классу *k* вычисляется как:

$$P(y=k|x)=\frac{\exp(\langle w_k,x\rangle+b_k)}{\sum_{j=1}^{K}\exp(\langle w_j,x\rangle+b_j)}$$

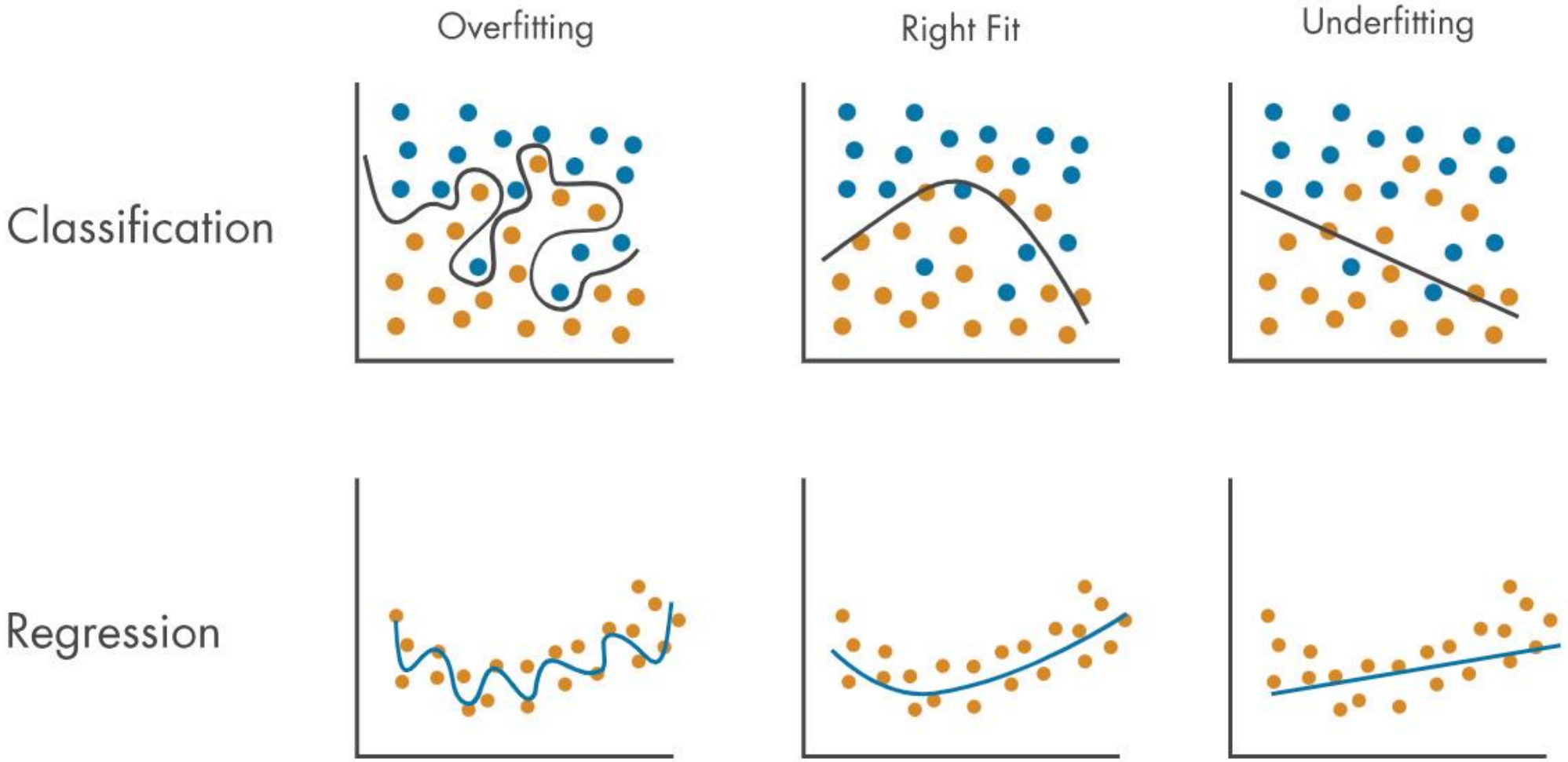
- Мягко распределяет вероятности между всеми классами (сумма равна 1).

$$L=-\sum_{i=1}^{N}\sum_{k=1}^{K}y_{ik}\log p_{ik}$$

Проблема переобучения

- Модель слишком точно описывает обучающую выборку
- Хуже работает на новых данных
- Причины:
  - слишком сложная модель
  - мало данных
  - ошибки в данных

Проблема переобучения



L1-регуляризация

L1-регуляризация — это способ борьбы с переобучением посредством добавления к оптимизируемой функции штрафа за "сложность" модели.

- Добавляет штраф за сумму абсолютных значений весов

$$\log L(w) - \lambda \sum |w_j|$$

- Способствует разреженным решениям (feature selection)

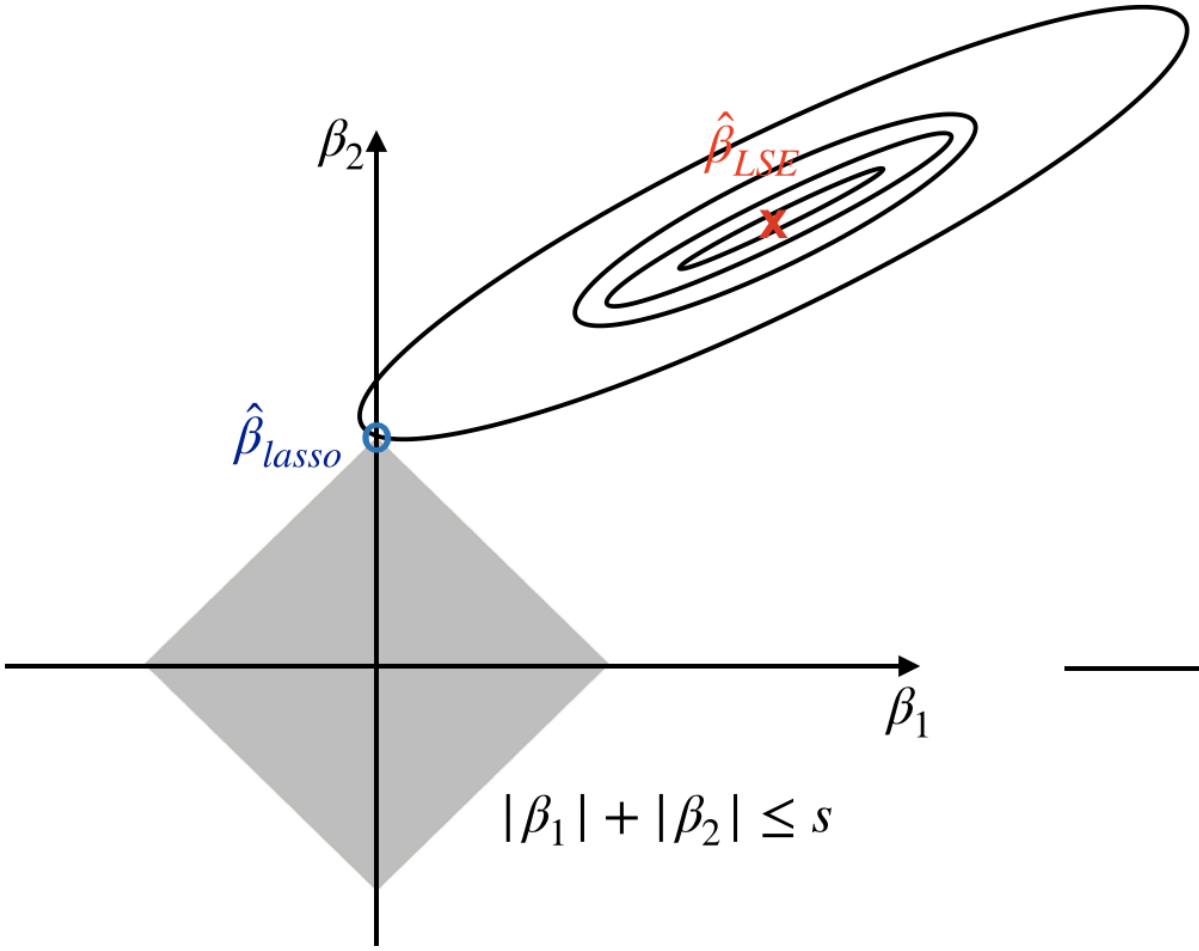
L2-регуляризация

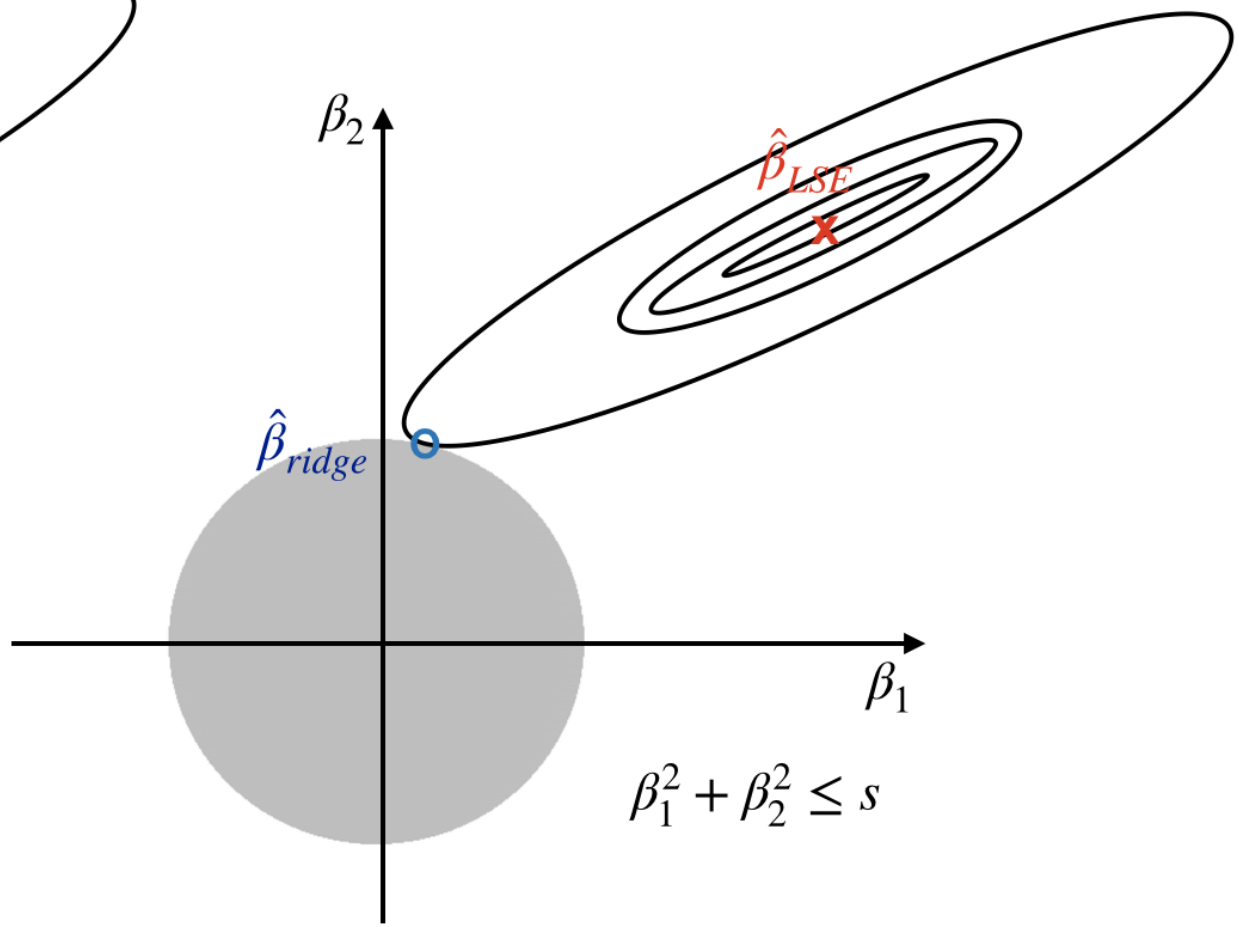
- Вносит штраф за сумму квадратов весов

$$-\log L(w) + \lambda \sum_{j=1}^{d} w_j^2$$

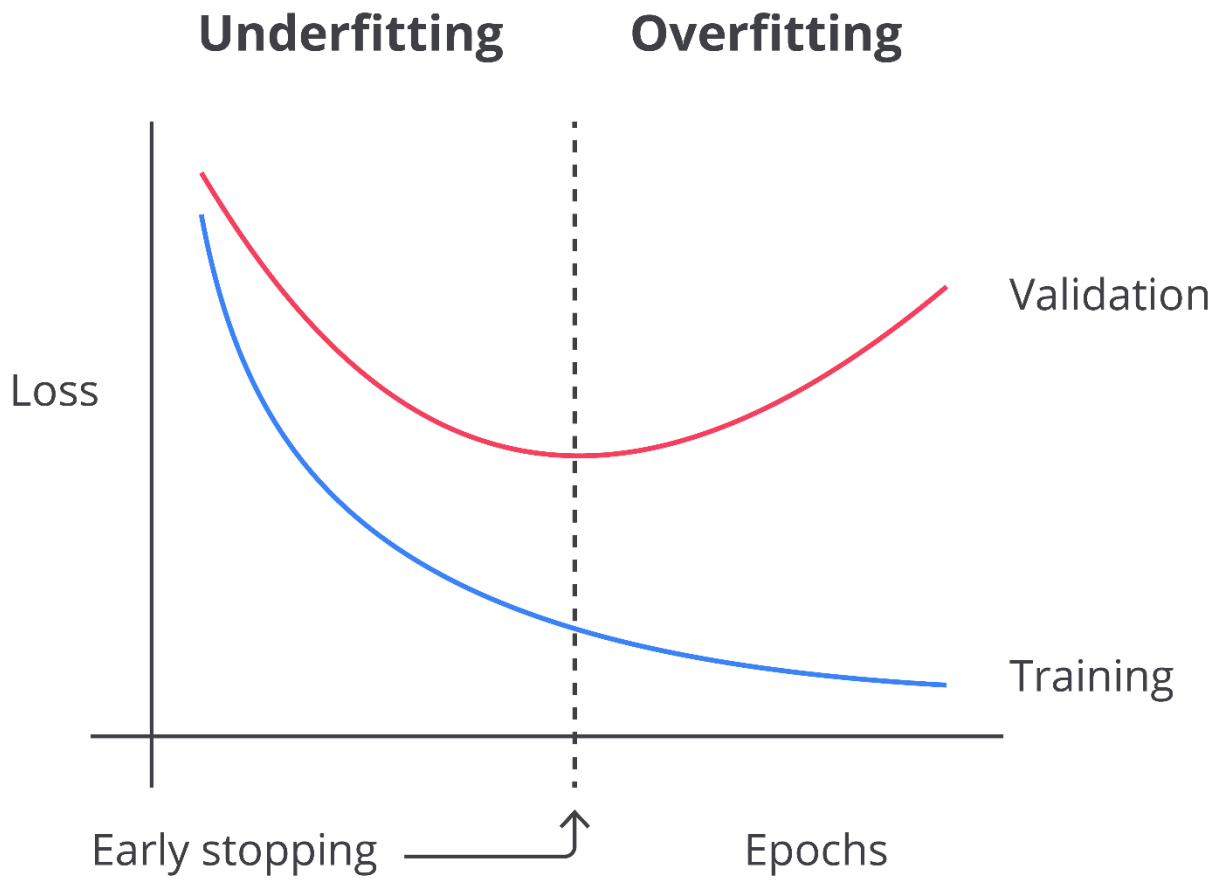
- Модель — менее чувствительна к выбросам и колебаниям обучающих данных

Геометрическая интерпретация



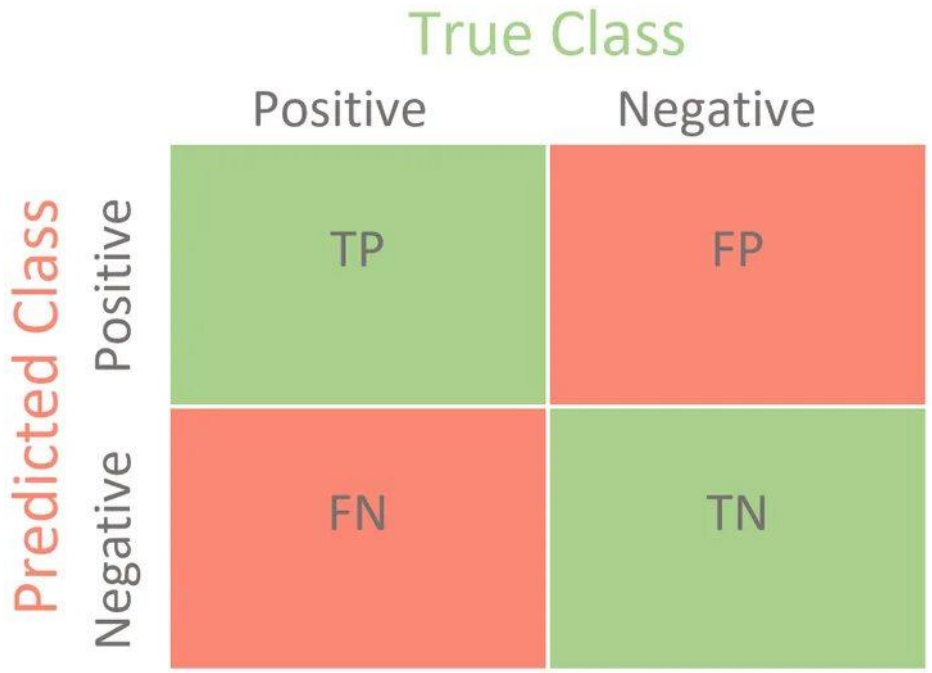


Ранний останов (early stopping)



Оценка качества бинарной классификации

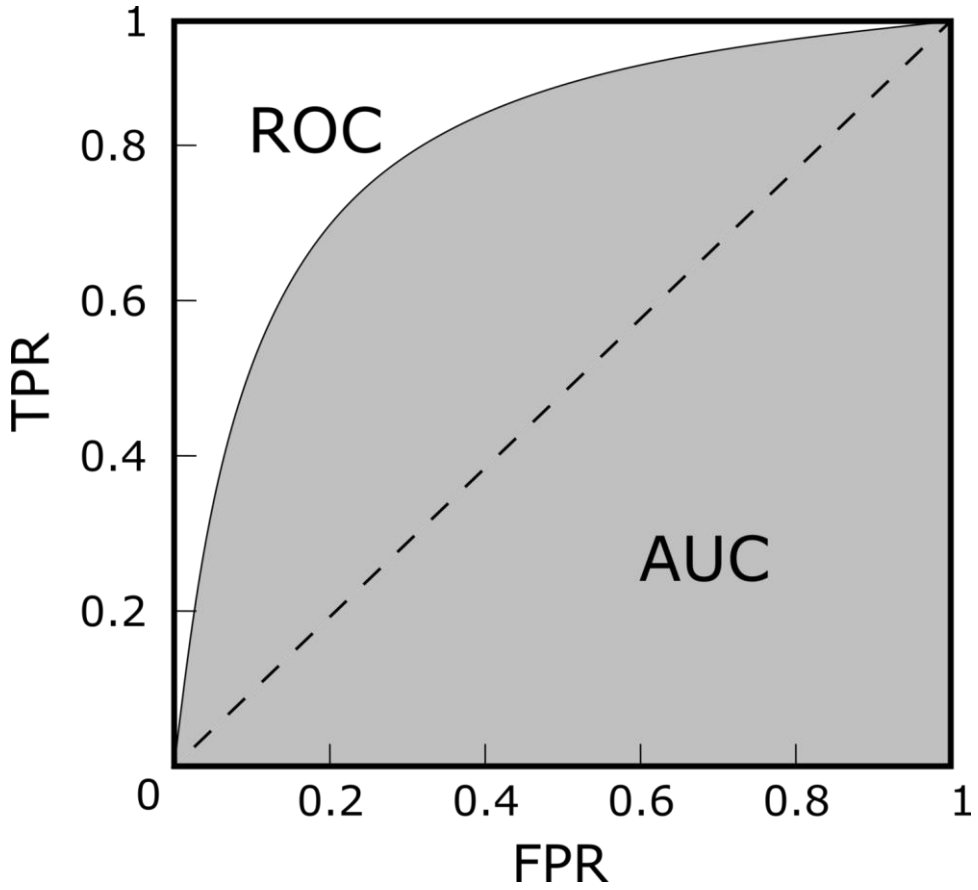
Матрица ошибок



$$Recall = \frac{TP}{TP+FN}$$

$$Specificity = \frac{TN}{TN+FP}$$

ROC-кривая



- ROC — зависимость TPR и FPR при разных порогах
- AUC — интегральная оценка

Обобщение метрик на многоклассовый случай

- *One-vs-Rest (OvR)* — один против всех
  каждый класс считается положительным поочерёдно

- Усреднение метрики для нескольких классов
  - Micro
  - Macro
  - Weighted

Гиперпараметры логистической регрессии

- Коэффициент регуляризации (λ)
- Тип регуляризации: L1/L2
- Критерий и алгоритм оптимизации

Методы оптимизации

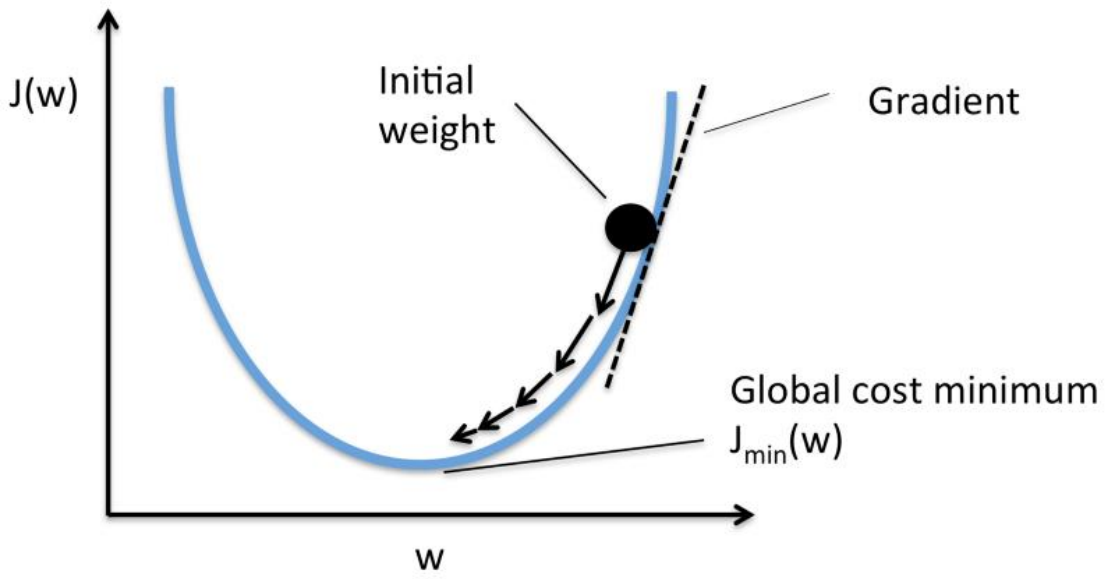
- Метод наименьших квадратов

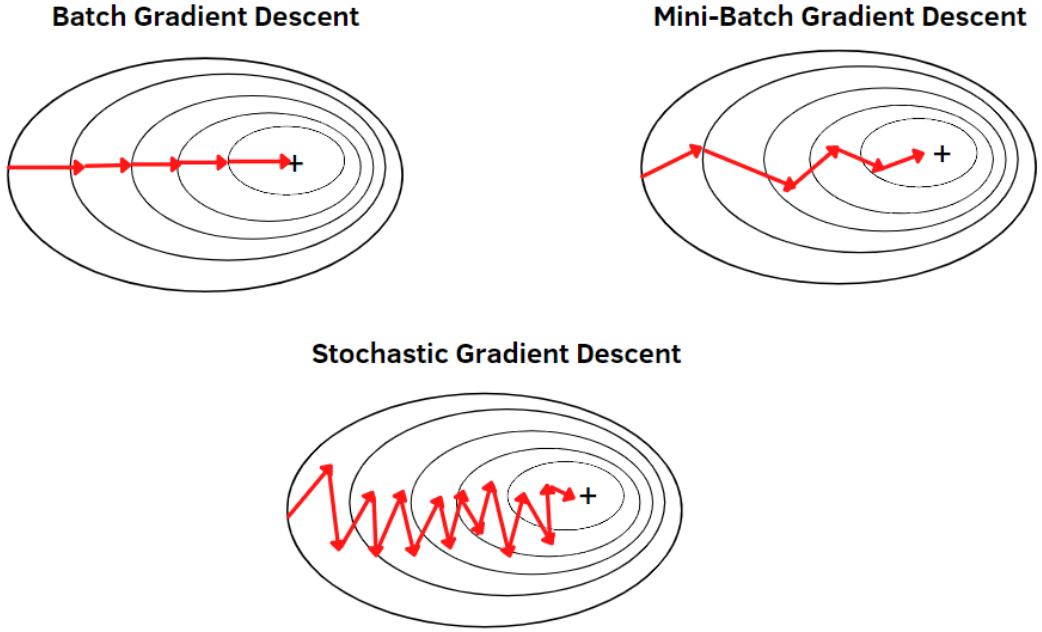
$$\min_w \|y - Xw\|^2$$

$$w^* = (X^T X)^{-1} X^T y$$

Методы оптимизации

- Градиентный спуск (GD / SGD)



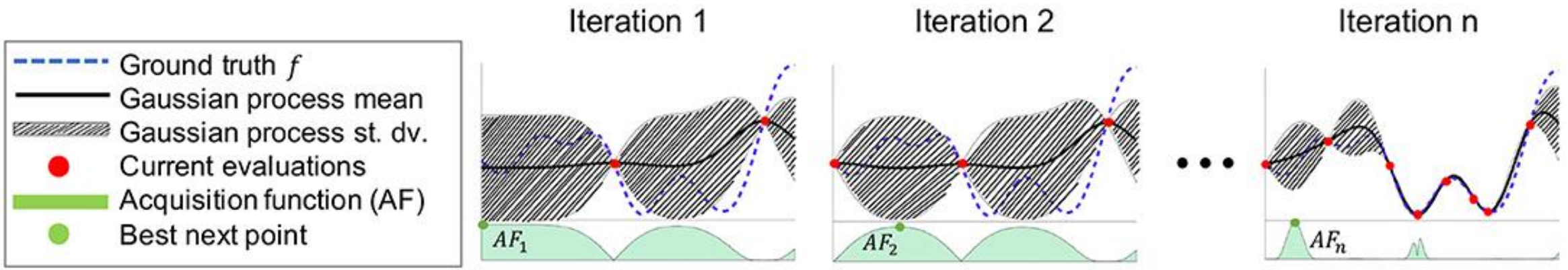


Подбор гиперпараметров

- Выбор способа отбора (перебор, случайный поиск, байесовская оптимизация)
- Определение диапазонов и шагов поиска
- Метрика качества (MAE, MSE, R^2 для регрессии)
- Отложенные данные / кроссвалдиация
- Агрегация результатов

Какие наборы параметров проверяем

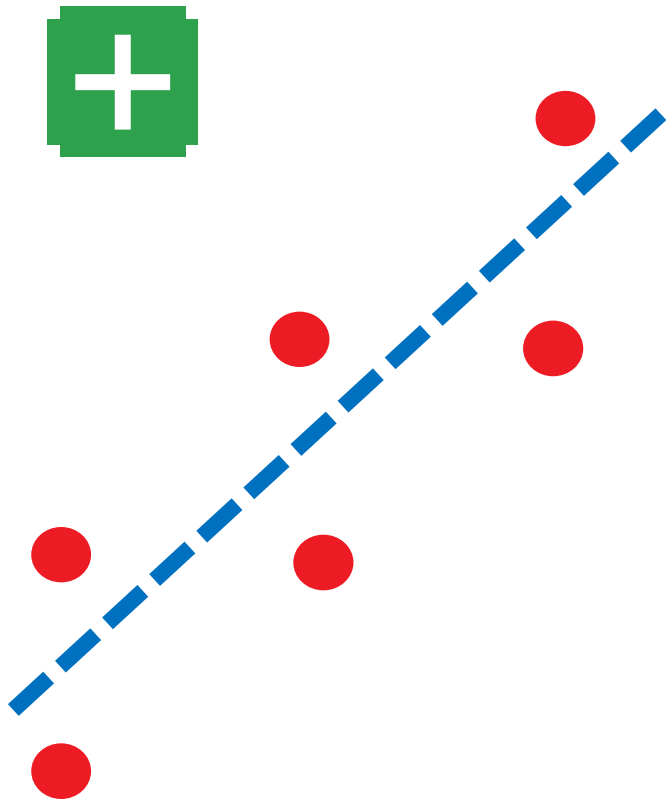
- *Grid Search* — перебор по сетке
- *Randomized Search* — случайный перебор
- *Байесовская оптимизация*
  - Не только случай, но и учёт уже полученных результатов
  - Примеры: hyperopt, optuna, scikit-optimize
  - Особенно актуально, если обучение долгое

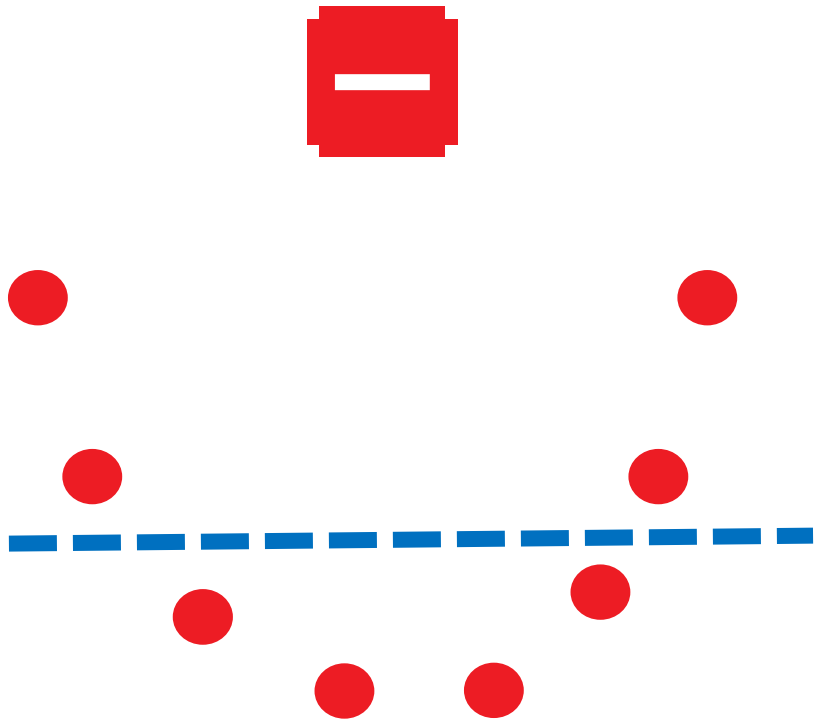


Готовые инструменты

- scikit-learn: GridSearchCV, RandomizedSearchCV
- optuna, hyperopt, scikit-optimize (айтили и облачные решения)
- AutoML-платформы (H2O, TPOT)

Ограничения





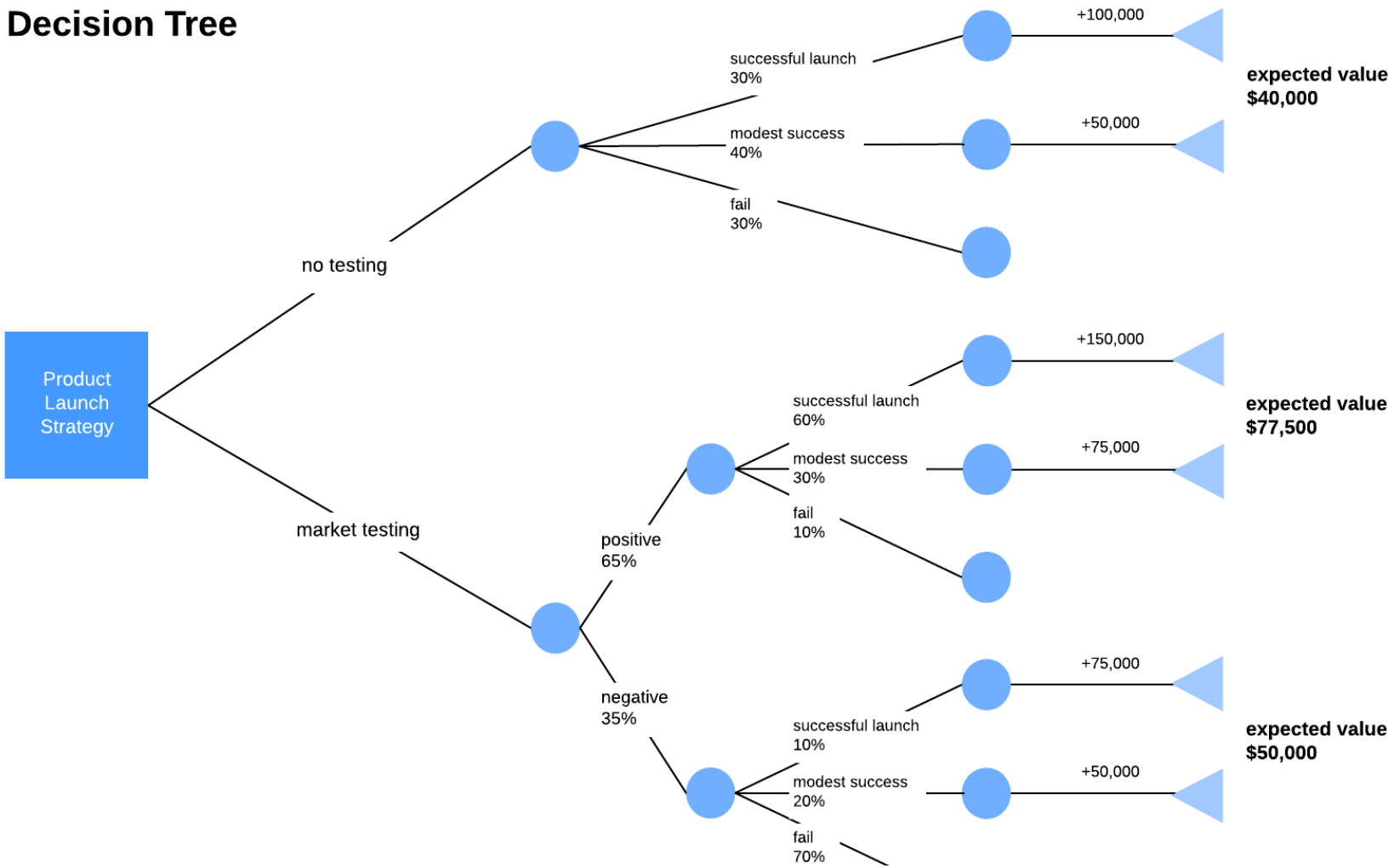
Ограничения

- Не могут аппроксимировать сложные (нелинейные) зависимости
- Склонны к underfitting на сложных задачах
- Требуют ручного конструирования признаков

Деревья решений

- Могут аппроксимировать нелинейные зависимости
- Простая интерпретация
- Идея: последовательное разбиение выборки на подгруппы

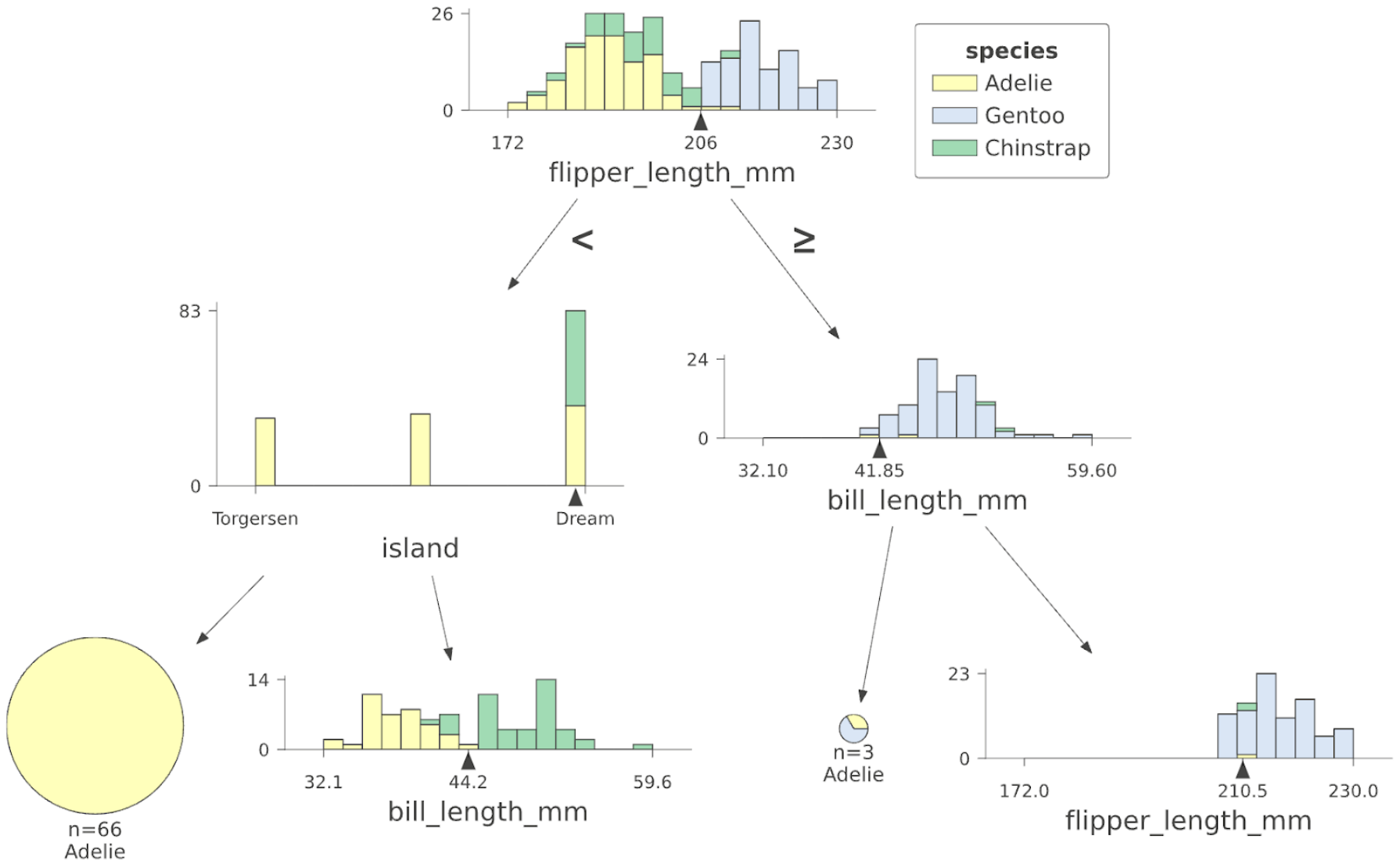
Деревья решений



Как строится дерево решений?

- Каждый узел делит выборку по признаку
- Деление выбирается так, чтобы максимально "очистить" дочерние группы
- Строим дерево до заданного ограничения (глубины, минимального числа элементов и пр.)

Как строится дерево решений?



CART (Classification and Regression Tree)

- Строит строго бинарные деревья решений.
- Для задачи классификации разделяет данные по признаку, чтобы минимизировать показатель "нечистоты" в дочерних узлах.
- Главный критерий разбиения — индекс Джини.

$$Gini(S) = 1 - \sum_{k=1}^{K} p_k^2$$

где p_k — доля объектов класса k в текущей выборке S, K — число классов.

CART

- Признак и порог разбиения выбираются так, чтобы минимизировать средневзвешенный *Gini* в полученных подмножествах

$$Q(S, a, t) = \frac{|S_{left}|}{|S|} Gini(S_{left}) + \frac{|S_{right}|}{|S|} Gini(S_{right})$$

где S_{left}, S_{right} — разделённые части множества S по признаку a и порогу t.

- CART выбирает *a,t,* для которых *Q* — минимум.

C4.5

- Может делать не только бинарные разбиения.
- Для задачи классификации применяет энтропийный критерий (информационный прирост, Information Gain), основываясь на энтропии Шеннона.

- **Энтропия (Shannon Entropy)**

$$H(S) = -\sum_{k=1}^{K} p_k \log_2 p_k$$

где p_k — доля объектов класса k в выборке S.

C4.5

- **Информационный прирост (Information Gain):**

$$IG(S, a) = H(S) - \sum_{v \in values(a)} \frac{|S_v|}{|S|} H(S_v)$$

- a — выбранный признак;
- v — возможные значения признака a;
- S_v — часть выборки с $a = v$.

C4.5

- **Gain Ratio** (с поправкой на количество значений признака)

$$Gain\ Ratio(a) = \frac{IG(S,a)}{IV(a)}$$

- $IV(a)$ — intrinsic value (внутренняя стоимость расщепления):

$$IV(a) = -\sum_{v \in values(a)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$$

- C4.5 выбирает признак и разбиение с максимальным Gain Ratio.

C4.5

- Может делать не только бинарные разбиения.
- Для задачи классификации применяет энтропийный критерий (информационный прирост, Information Gain), основываясь на энтропии Шеннона.

- **Энтропия (Shannon Entropy)**

$$H(S) = -\sum_{k=1}^{K} p_k \log_2 p_k$$

где p_k — доля объектов класса k в выборке S.

Пример

ID	ПРИЗНАК X (ВОЗРАСТ ≤ 30?)	КЛАСС Y
1	Да	0
2	Да	1
3	Нет	0
4	Нет	0
5	Да	1
6	Нет	1

Пример

- *Gini* для исходного множества

p_0 (доля класса 0): 3 из 6 = **0.5**

p_1 (доля класса 1): 3 из 6 = **0.5**

$$Gini_{root} = 1 - (p_0^2 + p_1^2) = 1 - (0.5^2 + 0.5^2) = 1 - (0.25 + 0.25) = 0.5$$

Пример

- Группа 1: X = "Да”

  Y (классы): 0, 1, 1

  $p_0 = 1/3, p_1 = 2/3$

$$Gini_{\text{Да}} = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 = 1 - \frac{1}{9} - \frac{4}{9} = 1 - \frac{5}{9} = \frac{4}{9} \approx 0.444$$

- Группа 2: X = "Нет"

  Y: 0, 0, 1

  $p_0 = 2/3, p_1 = 1/3$

$$Gini_{\text{Нет}} = 1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2 = 1 - \frac{4}{9} - \frac{1}{9} = 1 - \frac{5}{9} = \frac{4}{9} \approx 0.444$$

Пример

- Взвешенный *Gini* после разбиения

$$Gini_{split} = \frac{3}{6} \cdot 0.444 + \frac{3}{6} \cdot 0.444 = 0.222 + 0.222 = 0.444$$

- Прирост "чистоты" после разбиения

$$0.5 - 0.444 = 0.056$$

Пример

- Энтропия (Shannon Entropy) для исходного множества

$$H_{root} = -p_0 \log_2 p_0 - p_1 \log_2 p_1 = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = -0.5 * (-1) - 0.5 * (-1) =$$

$$(\log_2 0.5 = -1)$$

- Группа 1: X = "Да”

$p_0 = 1/3, p_1 = 2/3$

$$H_{\text{Да}} = -\frac{1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3}$$

$$\log_2 \frac{1}{3} \approx -1.585, \ \log_2 \frac{2}{3} \approx -0.585$$

$$H_{\text{Да}} = -\frac{1}{3} * (-1.585) - \frac{2}{3} * (-0.585) = 0.528 + 0.389 = 0.918$$

Пример

- Группа 2: X = "Нет"

$$H_{\text{Нет}} = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.389 + 0.528 = 0.918$$

- Взвешенная энтропия после разбиения:

$$H_{split} = \frac{3}{6} * 0.918 + \frac{3}{6} * 0.918 = 0.459 + 0.459 = 0.918$$

- Information Gain

$$IG = H_{root} - H_{split} = 1.0 - 0.918 = 0.082$$

Дискретизация вещественных признаков в деревьях

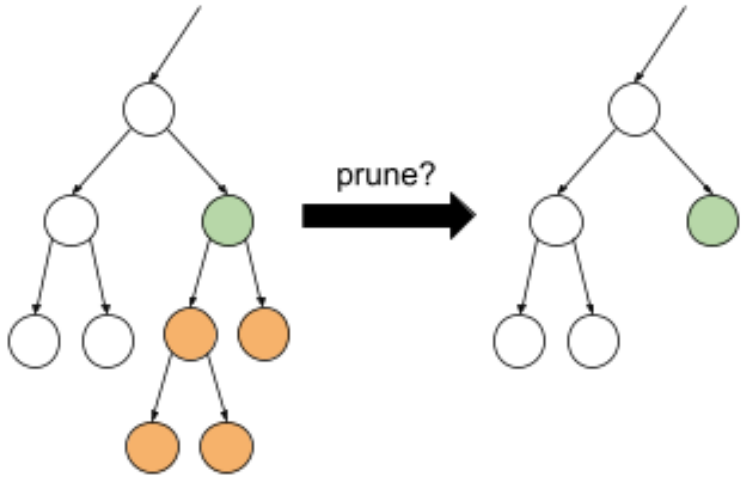
- Для непрерывного (веществennозначного) признака, алгоритм запускает перебор всех возможных порогов
- Учитывается, где меняется класс
- Ищет тот порог, при котором итоговое разделение будет наиболее чистым

Работы с категориальными признаками

- CART: перебор всех "разделений на 2 группы" вариантов.
- C4.5: разбиение по всем уникальным значениям признака.

Pruning

- Обрезка — это способ уменьшить сложность и повысить обобщающую способность модели
- Если строить дерево «до конца», оно будет идеально описывать обучающие данные, в том числе случайный шум.
- Такое дерево получается слишком сложным, плохо работает на новых данных



Pre-pruning

Ограничения накладываются уже в процессе построения дерева, например:

- максимальная глубина дерева (max_depth).
- минимальное число объектов в листе (min_samples_leaf).
- минимальное число объектов для разбиения узла (min_samples_split).
- минимальное улучшение чистоты при разбиении (min_impurity_decrease).

- Преимущество — эффективность, быстрее строится более компактное дерево.
- Недостаток — можно остановиться слишком рано и упустить тонкие закономерности.

Post-pruning

- **Cost complexity pruning**

$$R_\alpha(T) = R(T) + \alpha|T|$$

где

$R(T)$ — ошибка дерева T,

$|T|$ — число листьев в дереве,

α — параметр, регулирующий баланс между ошибкой и размером дерева.

- **Reduced error pruning**
  - Для каждого поддерева временно его убирают (заменяют родительским листом)
  - Если ошибка на валидации не выросла или даже уменьшилась — обрезку оставляют.

Сильные стороны деревьев

- Могут строить сложные нелинейные границы
- Устойчивы к пропускам, не требуют нормализации данных
- Хорошо работают с категориальными признаками
- Интерпретируемы
- Подходят для автоматизации построения экспертных правил.)

Слабые стороны деревьев решений

- Склонны к переобучению
- Могут быть неустойчивы к малым флуктуациям данных
- Не всегда хорошо работают, когда отношения между признаками сложные (например, мультиколлинеарность)

Принцип «Нет бесплатных завтраков»

(No Free Lunch Theorem)

- Для любой задачи нет идеального универсального алгоритма
- Успех метода зависит от задачи, данных и гипотез о структуре данных

Смещение и дисперсия

- Смещение (bias): насколько "далеко" среднее предсказание модели от истинных значений
- Дисперсия (variance): насколько сильно меняется предсказание модели при небольшой смене выборки

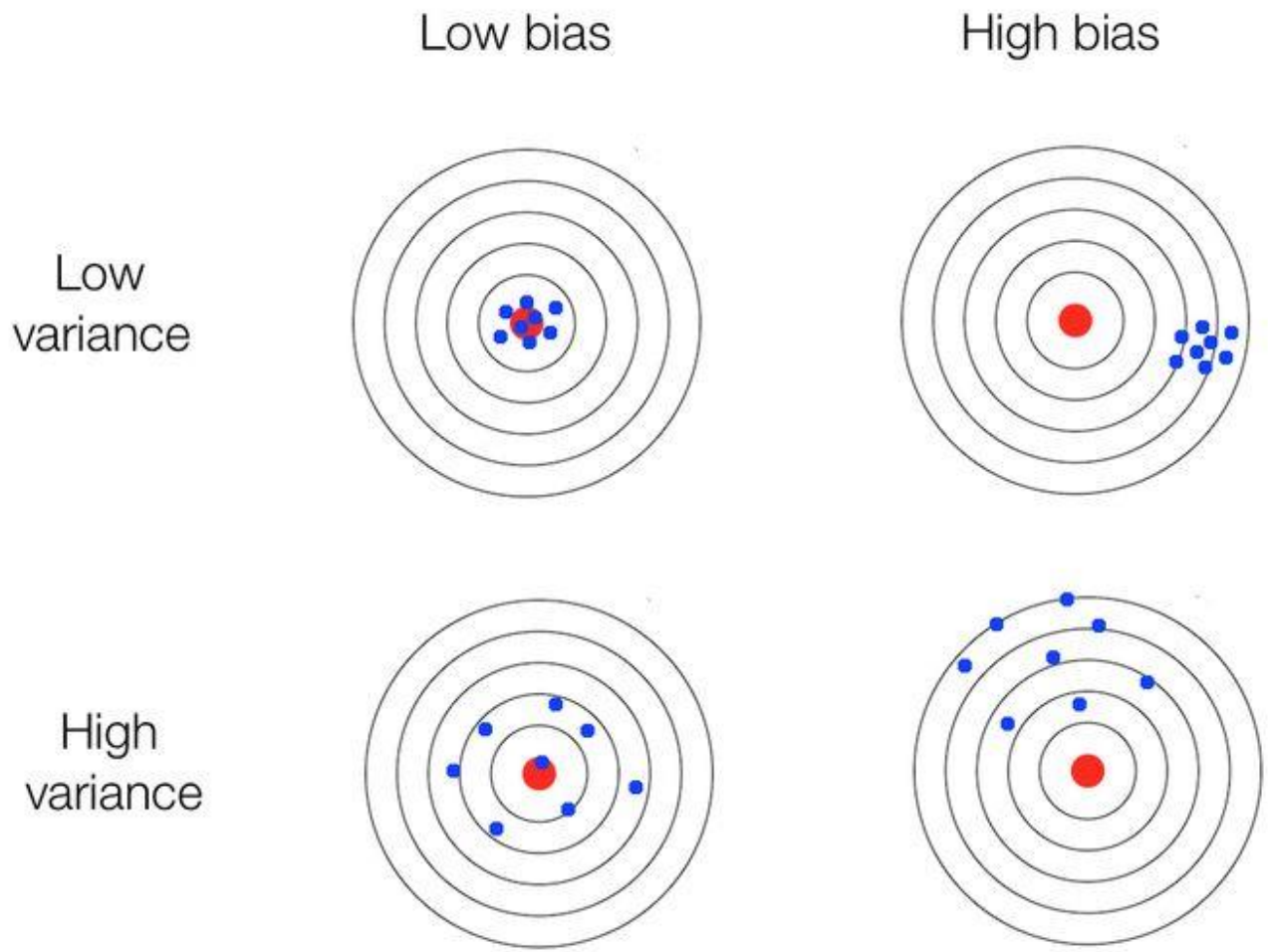
Пусть $\hat{f}(x)$ — предсказание обученной модели, а $y(x)$ — истинное значение.

Тогда для некоторого объекта x:

$$\mathrm{MSE}(x) = \underbrace{\left(\mathbb{E}\hat{f}(x) - y(x)\right)^2}_{\mathrm{Bias}^2} + \underbrace{\mathbb{E}\left[(\hat{f}(x) - \mathbb{E}\hat{f}(x))^2\right]}_{\mathrm{Variance}} + \sigma^2$$

где σ^2 — необъяснимый шум (ирр.шум).

Смещение и дисперсия



Смещение и дисперсия

- **Высокое смещение:**
  модель слишком простая, всегда ошибается примерно в одну и ту же сторону (underfitting)
- **Высокая дисперсия**:
  модель слишком чувствительна к обучающей выборке и переобучается (overfitting)
- Баланс bias-variance важен для высокой точности на новых данных

Идея ансамблирования

- Ансамбль = комбинация нескольких моделей (базовых алгоритмов)
- Итоговое решение основывается на их "коллективном" мнении
  - Голосование
  - Усреднение
  - взвешенная сумма
  - и др.

Варианты ансамблирования

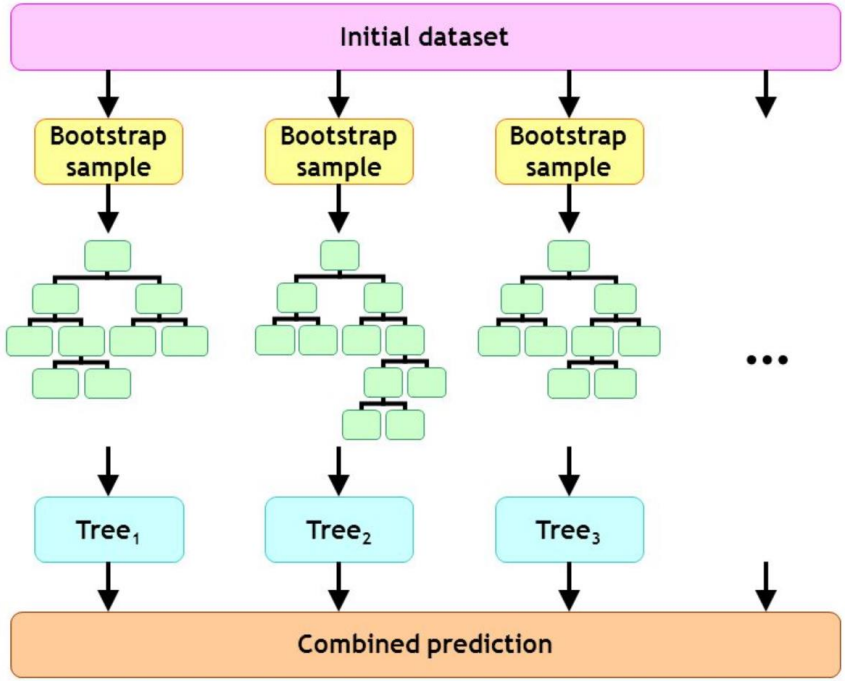
- Бэггинг (bagging, bootstrap aggregation)
- Бустинг (boosting)
- Внешние (stacking, blending)

Бэггинг

- Bagging ("bootstrap aggregation") — создаём множество новых обучающих выборок случайной выборкой с возвращением из исходной
- Обучаем на них много простых независимых моделей
- Добавляет "разнообразие" в данные для каждой базовой модели, делая деревья менее похожими друг на друга и уменьшая итоговую дисперсию ансамбля.

Бэггинг + Деревья = Случайный лес

- Случайный лес = бэггинг над деревьями + случайный поднабор признаков для каждого разбиения
- Итоговое решение - голосование деревьев



Случайный лес

Плюсы

- Снижает дисперсию (variance)
- Не увеличивает смещение (bias) по сравнению с одним деревом
- Обучается параллельно

Бустинг

- Серию слабых моделей обучают последовательно, на исправление ошибок предыдущих
- Итоговое решение — взвешенная комбинация всех моделей

В отличие от беггинга, базовые модели строятся НЕ независимо, а каждая следующая фокусируется на тех объектах, которые плохо предсказывает ансамбль в данный момент.

AdaBoost (Adaptive Boosting)

Пытаемся "переучить" примеры, ошибочно классифицированные предыдущими моделями за счёт изменения весов объектов.

- Инициализация весов

$$w_i^{(1)} = \frac{1}{N} \quad \forall i = 1, 2, \ldots, N$$

- Для каждого шага *m=1,2,…,M*

Обучаем слабый классификатор $h_m(x)$ на обучающей выборке с весами $w_i^{(m)}$

Считаем ошибку:

$$\varepsilon_m = \frac{\sum_{i=1}^{N} w_i^{(m)} \cdot \mathbb{I}(h_m(x_i) \neq y_i)}{\sum_{i=1}^{N} w_i^{(m)}}$$

AdaBoost (Adaptive Boosting)

Вычисляем вес модели:

$$\alpha_m = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_m}{\varepsilon_m} \right)$$

Обновляем веса объектов:

$$w_i^{(m+1)} = w_i^{(m)} \cdot \exp\left(-\alpha_m y_i h_m(x_i)\right)$$

(где $y_i \in \{-1, +1\}$)

Затем нормируем веса, чтобы $\sum_i w_i^{(m+1)} = 1$.

- Финальное решение:

$$H(x) = \text{sign} \left(\sum_{m=1}^{M} \alpha_m h_m(x) \right)$$

Gradient Boosting

В каждом шаге строится слабая модель, которая аппроксимирует отрицательный градиент функции потерь на текущих ошибках ансамбля ("остатках")

Алгоритм (регрессия, квадратичная функция потерь):

- Инициализация весов

$$F_0(x) = \arg\min_c \sum_{i=1}^{N} L(y_i, c)$$

где L — функция потерь (например, $L(y, F) = (y - F)^2$ для MSE).

Gradient Boosting

- Для шагов *m=1…M*

  Считаем "остатки" (градиенты):

$$r_i^{(m)} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x_i)=F_{m-1}(x_i)}$$

  Обучаем слабую модель $h_m(x)$ так, чтобы она аппроксимировала $r_i^{(m)}$.

  Находим оптимальный шаг (или используем фиксированный learning rate ν):

$$F_m(x) = F_{m-1}(x) + \nu \cdot h_m(x)$$

- Финальное решение:

  $\hat{y}(x) = F_M(x)$

Gradient Boosting

Для классификации:

Функция потерь — логистическая (log-loss):

$$L(y, F(x)) = \log(1 + e^{-2yF(x)})$$

Следовательно, градиент по $F(x)$:

$$r_i^{(m)} = \frac{2y_i}{1 + \exp(2y_i F_{m-1}(x_i))}$$

Дальше шаги аналогичны.

Stochastic Gradient Boosting

- Отличие:
  на каждом шаге для обучения дерева берется случайная подвыборка объектов (обычно 50%-80%), а также зачастую случайный поднабор признаков.

Конец