Лекция 3. Регуляризация, композиции алгоритмов

Юрий Яровиков

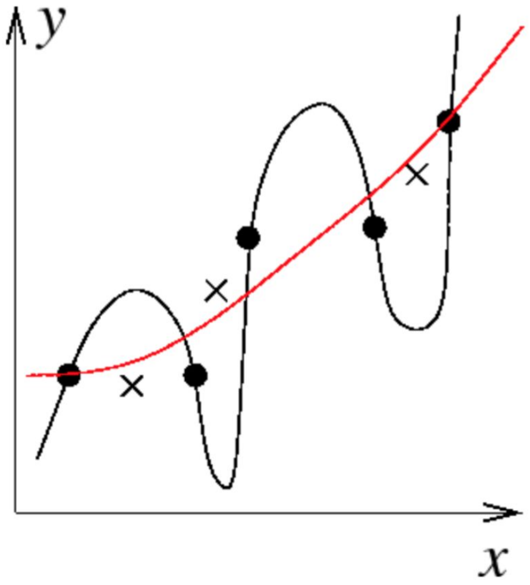
План лекции

- Методы борьбы с переобучением
  - Напоминание: переобучение и методы борьбы с ним
  - Напоминание: линейные алгоритмы
  - Ridge-регрессия и Lasso-регрессия в Python
- Композиции алгоритмов
  - Напоминание: метрические алгоритмы и решающие деревья
  - Композиции алгоритмов: бэггинг, бустинг
  - Градиентный бустинг над решающими деревьями

Методы борьбы с переобучением

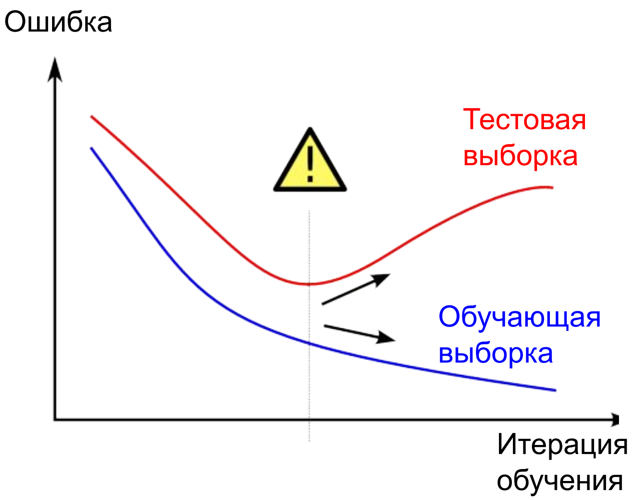
Переобучение

- Из-за чего возникает переобучение?
    - Переобучение есть всегда, когда выбор делается на основе заведомо неполной информации
    - Слишком сложная/гибкая модель может чрезмерно подстроиться под обучающую выборку и потерять способность находить нижележащие закономерности в новых данных



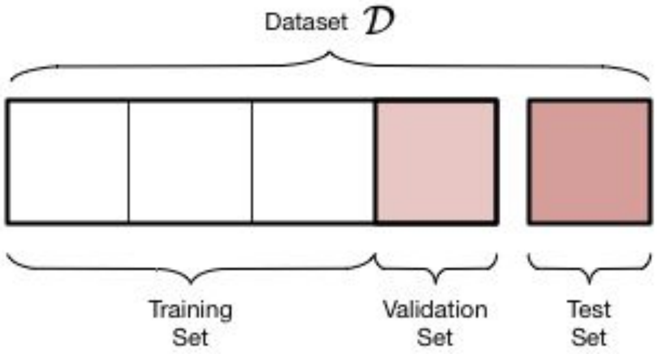
Переобучение

- Как обнаружить?
  - Разделить выборку на обучающую и контрольную
  - Следить за качеством на контрольной выборке
- Минусы?
  - Уменьшение размера обучающей выборки может негативно сказаться на качестве
  - Малый размер тестовой выборки может давать сильное смещение оценки.
  - Можно переобучиться под **тестовую выборку**



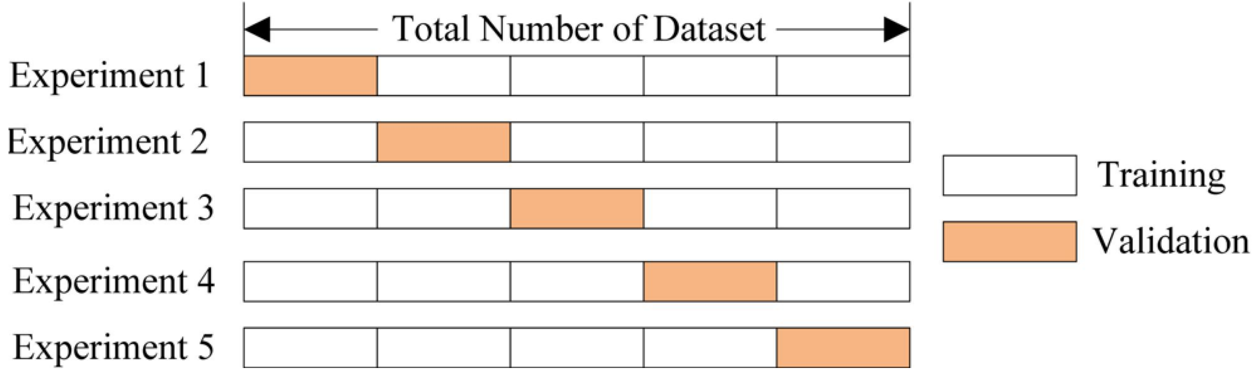
Переобучение под Тестовую Выборку

- Используя Train/Test split подход
  - Малый размер тестовой выборки
  - Перебор гипер-параметров модели
  - Выбор по наилучшему результату
- Train/Validate/Test split
  - Обучаем на Train
  - Корректируем параметры алгоритма по Validate
  - Итоговое качество на Test



Кросс-Валидация

- Кросс-Валидация
  - Разделить выборку на K корзин
  - Запустить K экспериментов, исключая одну корзину из обучения и замеряя на ней качество
  - Итоговое качество получить усреднением

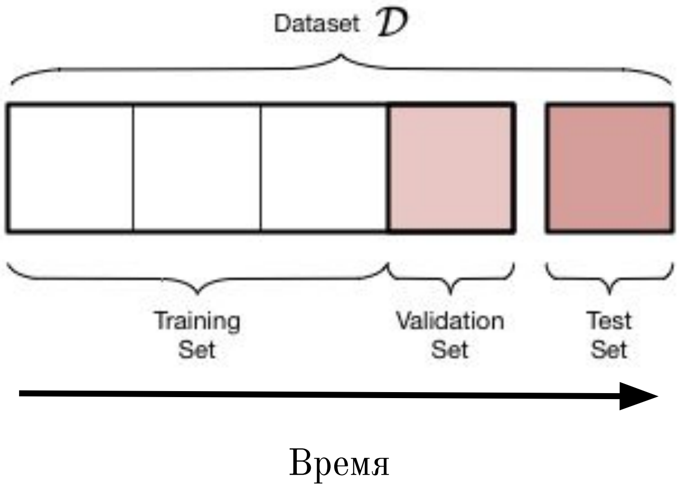


Кросс-Валидация

- Плюсы
  - Качество измеряется на всем наборе данных
  - Качество не зависит от выбора конкретного тестового набора
  - Сложнее переобучиться под test
- Минусы
  - Скорость!
- Что выбрать
  - Мало обучающих данных → Кросс-Валидация
  - Много обучающих данных → Train/Validate split
- Не забыть
  - Отложить Test для замера итогового качества
  - Обучить итоговую модель на всех данных

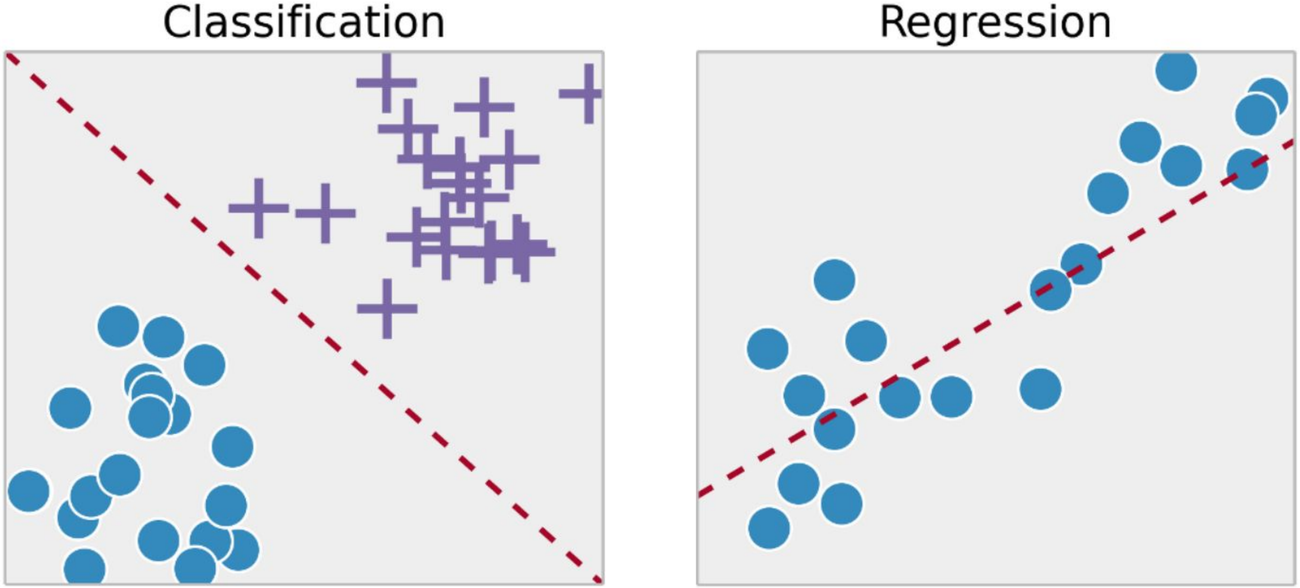
Кросс-Валидация По Времени

- Используется для анализа временных рядов
  - Тестовый набор выбирается из самых свежих данных, обучение на более старых
- Полезно в реальных задачах
  - Если в качестве признаков используется множество сигналов, которые могут меняться от времени
  - Есть возможность определить дату наблюдения



Напоминание: линейные алгоритмы

$$y(x) = \mathrm{sign}(<w, x> + b) \qquad y(x) = <w, x> + b$$



Регуляризация в линейных моделях

Разные линейные модели отличаются разными функциями потерь.

- Линейная регрессия: $\sum_{i=1}^{l}(\langle x_i, w\rangle - y_i)^2 \to \min_w$
- Ridge-регрессия: $\sum_{i=1}^{l}(\langle x_i, w\rangle - y_i)^2 + C||w||^2 \to \min_w$
- Lasso-регрессия: $\sum_{i=1}^{l}(\langle x_i, w\rangle - y_i)^2 + C||w|| \to \min_w$

Покажем отличия алгоритмов на примере датасета boston.

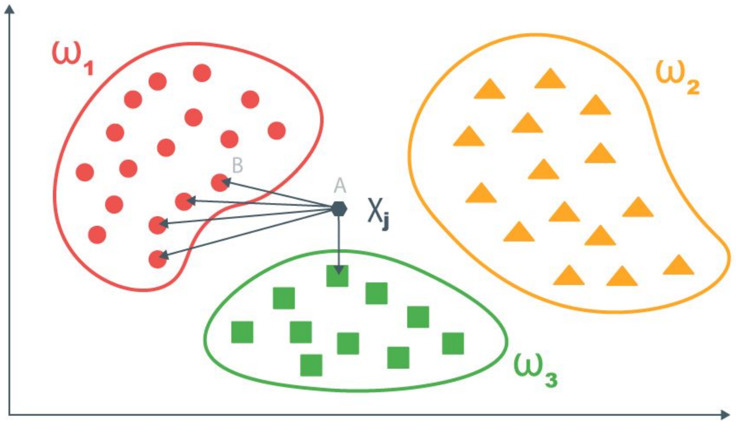
Композиции алгоритмов

Напоминание: метрические алгоритмы

Метрический алгоритм — алгоритм, опирающийся на геометрическую структуру данных в пространстве объектов.

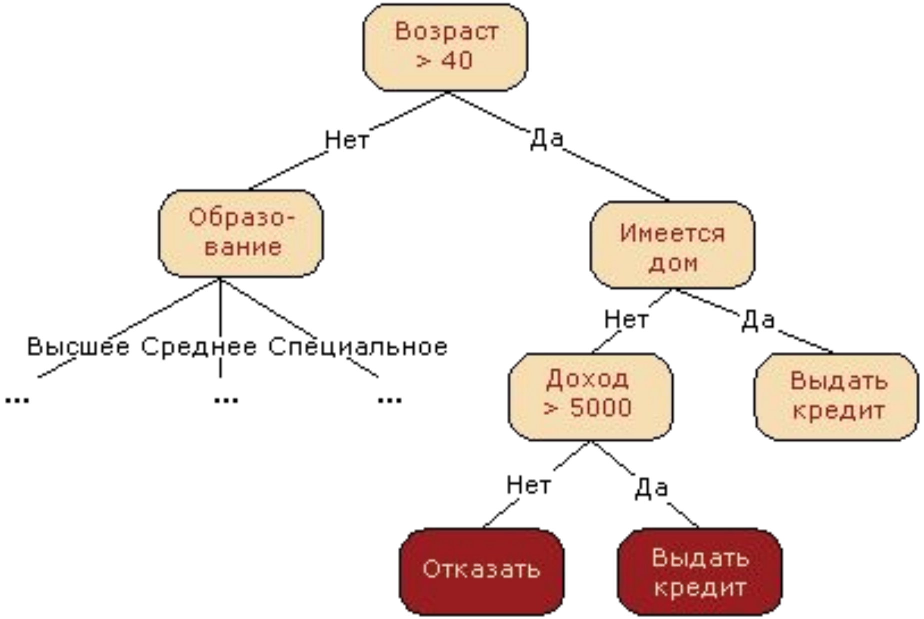
Алгоритм k ближайших соседей:

- Хотим предсказать класс объекта x
- Вычисляем $f_1(x), f_2(x), \ldots, f_n(x)$
- Находим k ближайших объектов из обучающей выборки
- Предсказание = самый популярный класс среди соседей



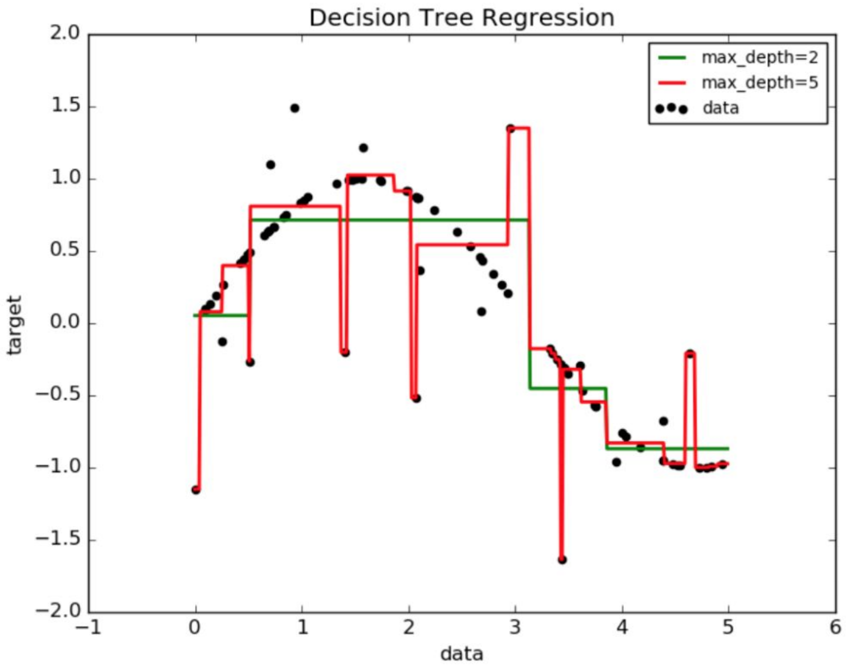
Напоминание: решающие деревья

- В каждой вершине дерева находится вопрос
- В зависимости от ответа на вопрос, алгоритм направляется в нужную ветвь дерева
- Листы дерева соответствуют решению алгоритма



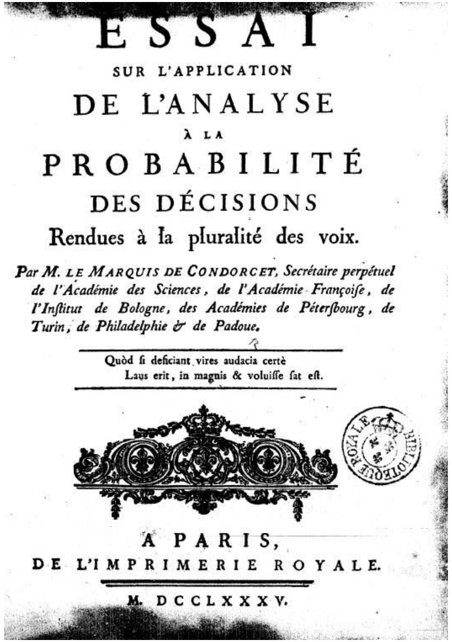
Решающее дерево: проблема переобучения

- Легко переобучиться, так как число листьев растет экспоненциально
- Для набора данных $N = 1000$ хватит дерева глубины 10, чтобы покрыть каждый объект листо



Композиции алгоритмов: простое голосование

- Если каждый член жюри имеет независимое мнение, и если вероятность правильного решения члена жюри больше 0.5, то тогда вероятность правильного решения присяжных в целом возрастает с увеличением количества членов жюри и стремится к единице.
- Если же вероятность быть правым у каждого из членов жюри меньше 0.5, то вероятность принятия правильного решения присяжными в целом монотонно уменьшается и стремится к нулю с увеличением количества присяжных.


ESSAI
SUR L'APPLICATION
DE L'ANALYSE
À LA
PROBABILITÉ
DES DÉCISIONS
Rendues à la pluralité des voix.

Par M. LE MARQUIS DE CONDORCET, Secrétaire perpétuel de l'Académie des Sciences, de l'Académie Françoise, de l'Institut de Bologne, des Académies de Péterſbourg, de Turin, de Philadelphie & de Padoue.

Quòd ſi deficiant vires audacia certè
Laus erit, in magnis & voluiſſe ſat eſt.

À PARIS,
DE L'IMPRIMERIE ROYALE.
M. DCCLXXXV.

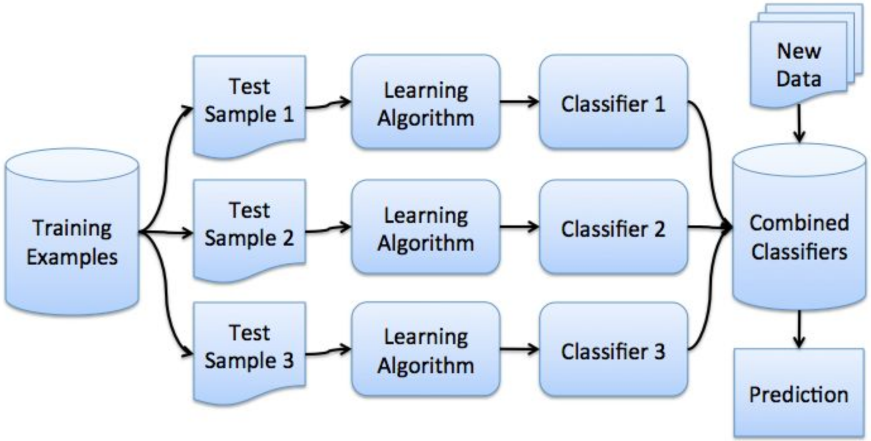
принцип Кондорсе, 1784

Эксперимент Гальтона, 1906

- Собралось около 800 человек, которые попытались угадать вес быка на ярмарке. Бык весил 1198 фунтов.
- Ни один крестьянин не угадал точный вес быка
- Среднее предсказание оказалось равным 1197 фунтов.

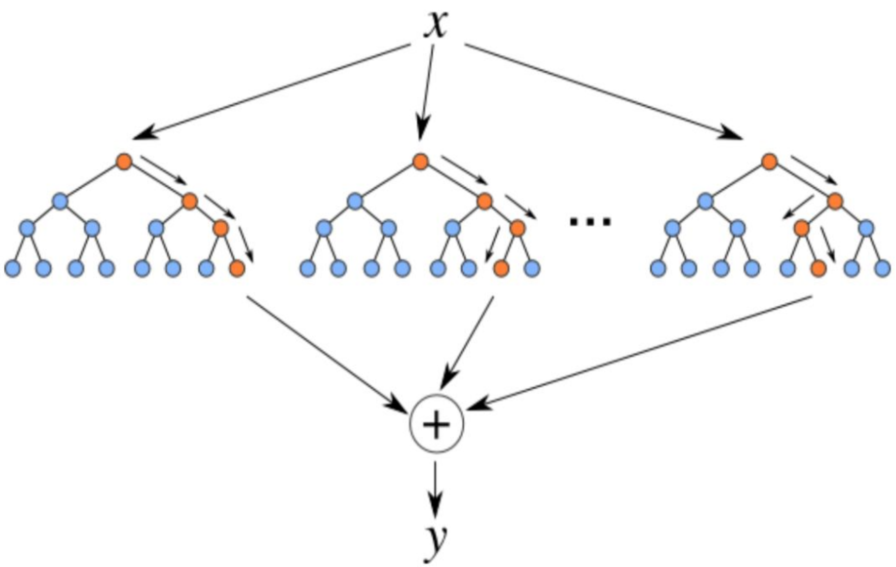
Бэгинг

- С помощью бутстрэпа генерируем M выборок
- На каждой выборке обучим свой классификатор
- Итоговый классификатор будет усреднять ответы всех этих алгоритмов

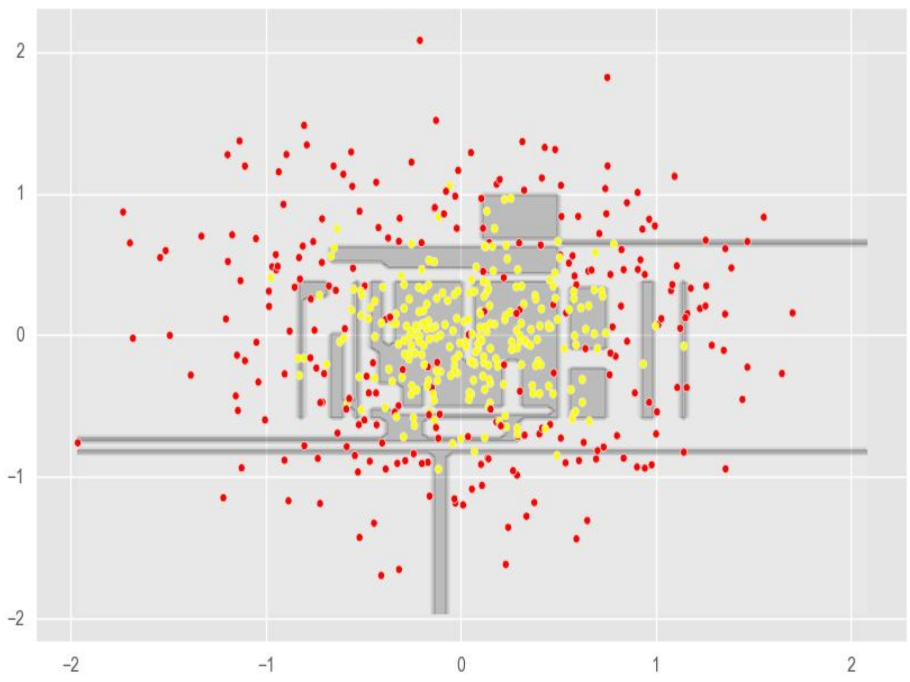


Решающий лес

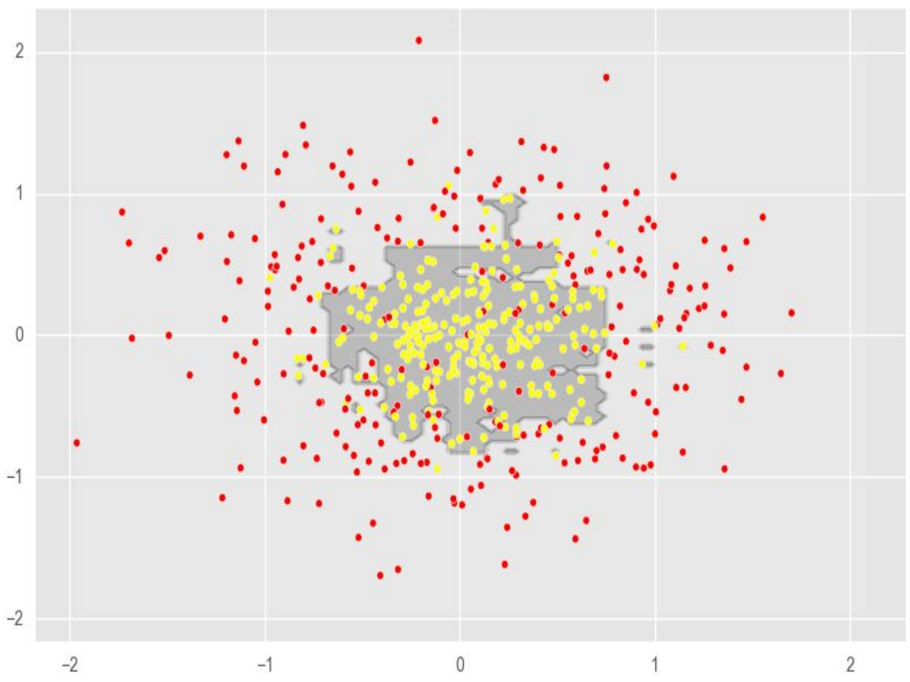
- Построим совокупность решающих деревьев, каждое из которых будем обучать по случайной подвыборке и случайному подмножеству признаков
- Каждое дерево имеет малую глубину
- Итоговый ответ — усреднение ответов по всем деревьям



Решающее дерево



Решающий лес



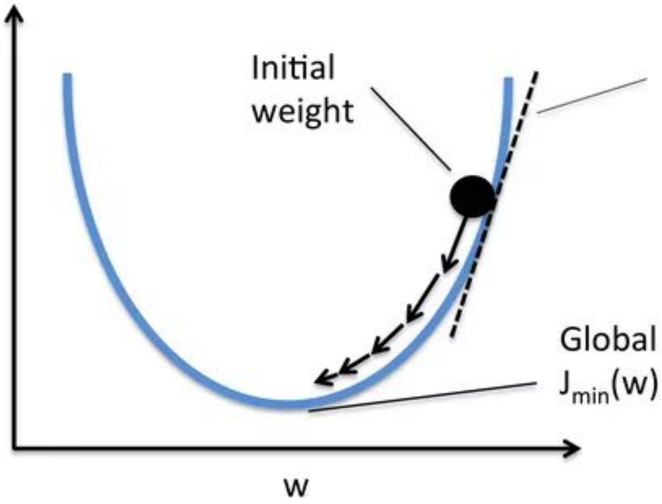
Бустинг

- Строим алгоритмы по очереди
- Каждый новый алгоритм выбирается так, чтобы вся композиция работала наилучшим образом
- Новый алгоритм исправляет ошибки предыдущих алгоритмов
- Итоговое решение принимается взвешенным голосованием

Напоминание: градиентный спуск

- Минимизируем функцию $f(x)$
- Выбираем x_0 — начальное приближение
- Пусть x_n — текущая найденная точка
- Вычисляем градиент (производную) $f'(x_n)$
- Вычисляем следующее приближение:

$$x_{n+1} = x_n - \alpha f'(x_n)$$



Градиентный бустинг

Линейная (выпуклая) комбинация базовых алгоритмов:

$$a(x) = \sum_{t=1}^{T} \alpha_t b_t(x), \quad x \in X, \quad \alpha_t \in \mathbb{R}_+$$

Функционал качества с произвольной функцией потерь $\mathscr{L}(a, y)$:

$$Q(\alpha, b; X^\ell) = \sum_{i=1}^{\ell} \mathscr{L}\Bigg(\underbrace{\underbrace{\sum_{t=1}^{T-1} \alpha_t b_t(x_i)}_{f_{T-1,i}} + \alpha b(x_i)}_{f_{T,i}}, y_i \Bigg) \to \min_{\alpha, b}$$

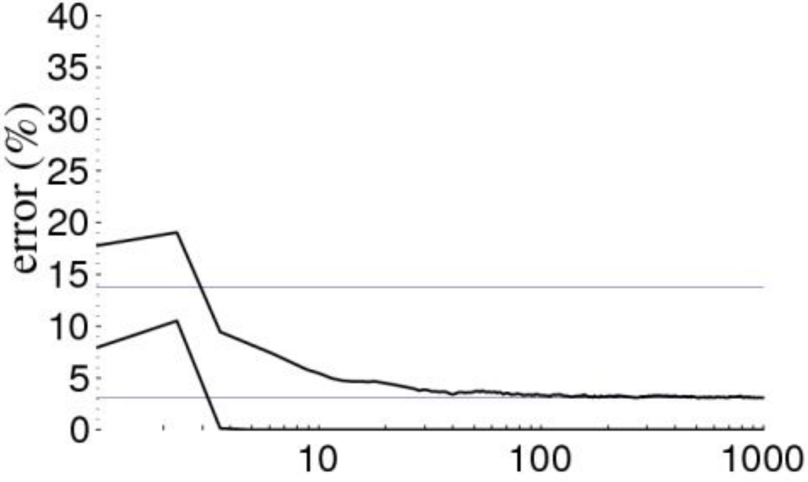
Настраиваем b_T на антиградиент функции потерь $\mathscr{L}(a, y)$ в точке $\sum_{t=1}^{T-1} \alpha_t b_t(x_i)$

Градиентный бустинг

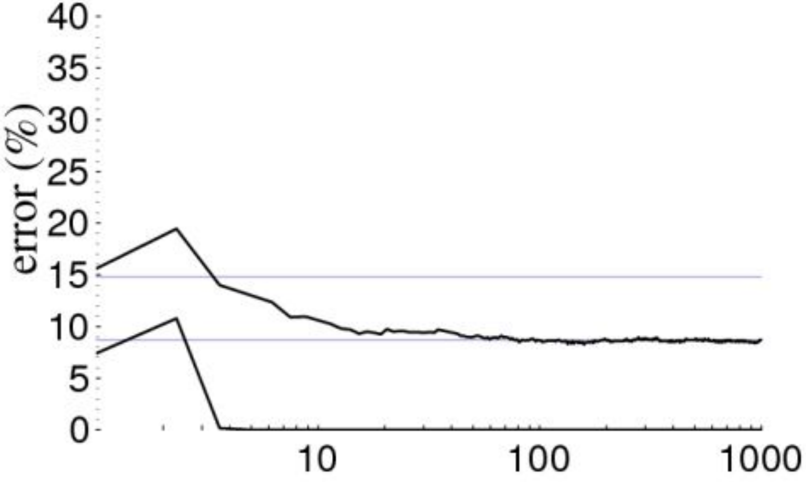
- Базовые алгоритмы b_t должны быть простыми и быстро обучаемыми
- Часто в качестве базовых алгоритмов используются решающие деревья
- Реализации градиентного бустинга над решающими деревьями:

Бустинг: почти полное отсутствие переобучения

Задача UCI:letter



Задача UCI:satimage



Бустинг: преимущества и недостатки

- Преимущества:
  - Позволяет очень точно восстанавливать искомую функцию
  - Почти не переобучается
- Недостатки:
  - Медленный
  - Плохо интерпретируемый
  - Переобучение на выбросах при избыточном количестве алгоритмов
  - Нужна довольно большая обучающая выборка

Стекинг

Стекинг — ещё один способ построить композицию алгоритмов

- Обучим несколько различных алгоритмов на одних и тех же данных
- Будем использовать ответы этих алгоритмов как новые признаки!