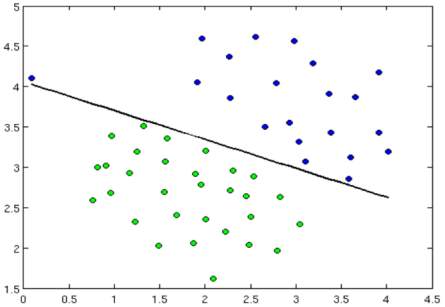
Напоминание: задача обучения с учителем

- Есть множество объектов X и множество ответов Y
- Каждый объект $x \in X$ характеризуется своим признаковым описанием
- Также есть неизвестная функция y, которая каждому элементу X ставит в соответствие какой-то элемент y
- В задаче обучения с учителем имеем *обучающую выборку* $x_1, \ldots, x_\ell \in X$, для каждого объекта которой известен правильный ответ $y_i = y(x_i)$
- Цель: построить на основе обучающей выборки алгоритм $a(x)$, который будет как можно лучше приближать функцию y на всём множестве X

Напоминание: линейные модели



- Каждый объект x – вектор со значениями признаков
- Ответ y определяется по формуле

$$y = f(\langle x, w \rangle + b)$$

- w – это вектор весов, той же длины, что и x, b – коэффициент сдвига
- Если $y \in \{0, 1\}$ (задача бинарной классификации), то $f(z) = \mathrm{sng(z)}$
- Если $y \in \mathbb{R}$ (задача регрессии), то $f(z) = z$

Ссылка на источник картинки

Напоминание: метрические модели

- **Обучение с учителем: knn**
- Имеем объекты X и метрику $\rho(x_1, x_2)$
- Для каждого объекта есть ответ y
- В простейшем случае модель запоминает обучающие объекты
- И когда приходит новый, выставляет ему y в зависимости от того, какие k объектов у нему ближе всего по ρ
- Если y – метка класса, то присваиваем самый частый класс среди соседей (можно взвешенно)
- Если $y \in \mathbb{R}$, то присваиваем среднее значение по соседям (взвешенно)

- **Обучение без учителя: K-means**
- Имеем объекты X и метрику $\rho(x_1, x_2)$
- Для объектов нет ответов
- Хотим распределить все объекты по K плотным группам (кластерам)
- Для этого по-очерёдно повторяем
  1. выбираем центры кластеров и относим все точки к ближайшим кластерам
  2. внутри каждого кластера пересчитываем центр

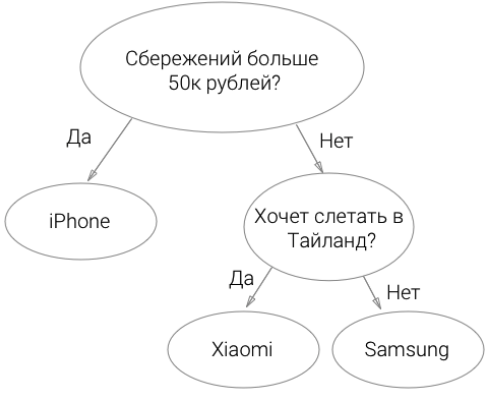
Логические модели

Наиболее известный представитель – *решающее дерево*

Ссылка на источник картинки

Логические модели

- Наиболее известный представитель – *решающее дерево*
- Представляет собой бинарное дерево
- Каждый внутренний узел содержит условие, которое отправляет объект в правое или левое поддерево
- Каждый лист соответствует какому-то классу (или числу, если задача регрессии)



Ссылка на источник картинки

Как строится решающее дерево

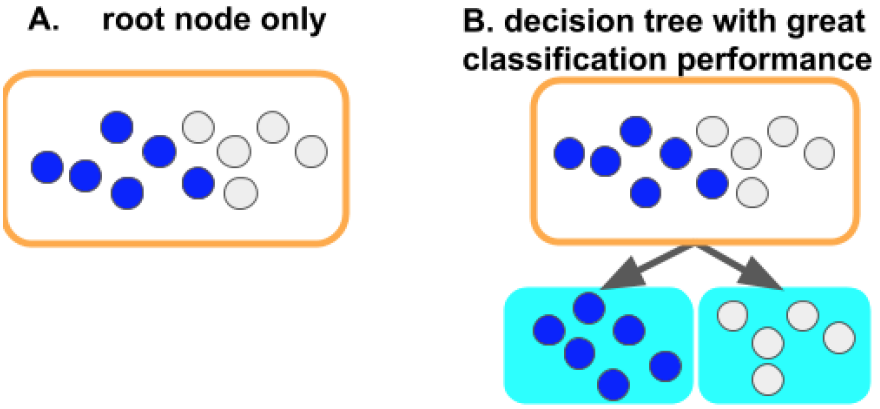
Решающие деревья обычно строят *жадным образом*

- Берём объекты, попавшие в текущую вершину (истории болезней)
- Перебираем признаки (температура, рост, вес)
- Для каждого признака перебираем пороги (35.0°, 35.1°, 35.2°, . . .)
- Смотрим, как объекты будут разбиваться на две части по этому признаку с этим порогом
- Если разбились хорошо, то добавляем эту вершину, и в неё записываем правило:

  «если этот признак больше этого порога – иди в левое поддерево, иначе – в правое»

Ссылка на источник изложения

Построение одного узла



Ссылка на источник картинки

А в чём жадность?

Решающие деревья обычно строят *жадным образом*

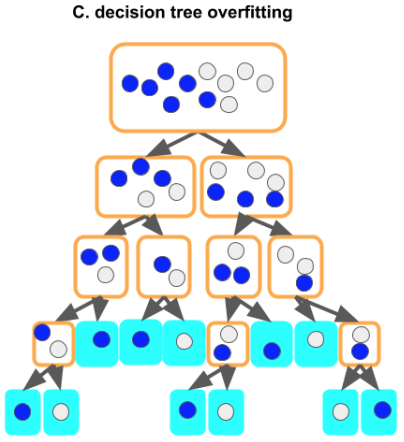
- Алгоритм называется **жадным**, если он на каждом своём шаге выбирает действие, которое оптимально на этом шаге, но может быть глобально неоптимальным
- **Пример:**
  - я хочу поиграть на компьютере, поиграю сегодня
  - на этом шаге (сегодня) я поиграл 2 часа
  - но сегодня из-за этого я не сделал уроки
  - меня наказали и не дали поиграть на выходных 4 часа
  - глобально за неделю я поиграл неоптимальнot время
- В случае обучения дерева жадность в том, что на каждом шаге мы смотрим только на то, как хорошо разбиваются объекты в этом узле
- Так делается потому, что обучении дерева – дискретная задача, её полноценная оптимизации возможна только перебором

Что такое «хорошее разбиение объектов»?

- При добавлении очередного узла мы ищем признак и порог для него, которые разбивают объекты, попавшие в этот узел, наилучшим образом
- Критерии «хорошести» разбиения зависят от задачи и бывают разные
- Для регрессии обычно используются величины, показывающие, насколько большой разброс значений в получающихся подвыборках
- Для классификации есть разные варианты, самый простой – считать, сколько объектов относятся не к самому большому классу
- Этот критерий не очень хорош, поскольку оценивает частоту только самого большого класса в подвыборке
- На практике для классификации обычно используются *критерий Джини* и *энтропийный критерий*

Деревья и переобучение

- Очевидно, что для любой выборки можно построить дерево, которое будет её идеально классифицировать
- Но для предсказания такое дерево бесполезно
- Это переобучение – излишняя подгонка под обучающие данные
- Деревья сильно склонны к переобучению



Ссылка на источник картинки

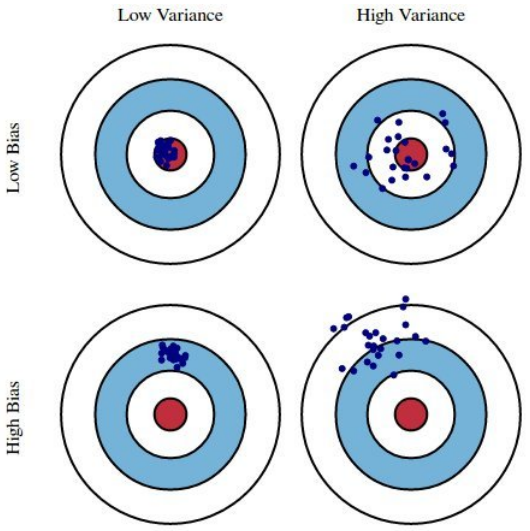
Деревья и композиции

«Если вы будете вместе, то вас никто не сломит, а по отдельности вас также легко победить, как и сломать пару соломинок»

Фрагмент древней притчи

- Деревья сильно склонны к переобучению
- Большое качественное дерево обучать сложно
- Из-за этого сами по себе решающие деревья редко используются на практике
- Зато можно объединять много деревьев в *композиции*
- И вот эти композиции являются одними из самых крутых алгоритмов машинного обучения!

Проблема Bias-Variance



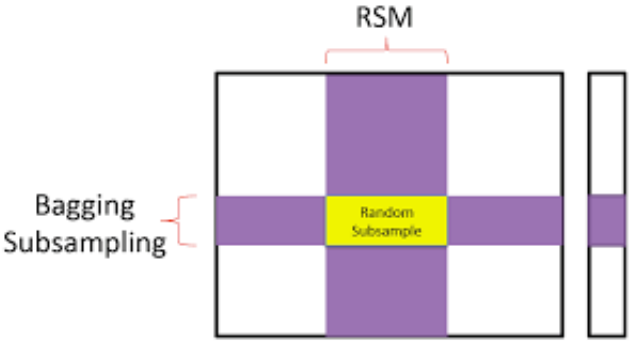
Ссылка на источник картинки

Композиции: бэггинг

- Есть два основных вида композиций: *бэггинг* и *бустинг*
- **Бэггинг:**
  1. Набираем m случайных объектов с повторением
  2. Обучаем на них модель
  3. Повторяем шаги 1 и 2 N раз
  4. Дальше усредняем ответы всех полученных моделей
- Можно показать математически, что если модели достаточно различные, то бэггинг уменьшает variance и не растит bias
- Значит, надо использовать модели, у которых низкий bias, то есть сложные
- Например, можно использовать достаточно глубокие деревья

Random Subspace Method

- Критически важно обучать как можно более разные модели
- В обычном бэггинге мы случайно выбираем объекты для обучения
- Давайте для каждой модели ещё и случайно выбирать признаки!



Ссылка на источник картинки

Ура, мы получили случайный лес!

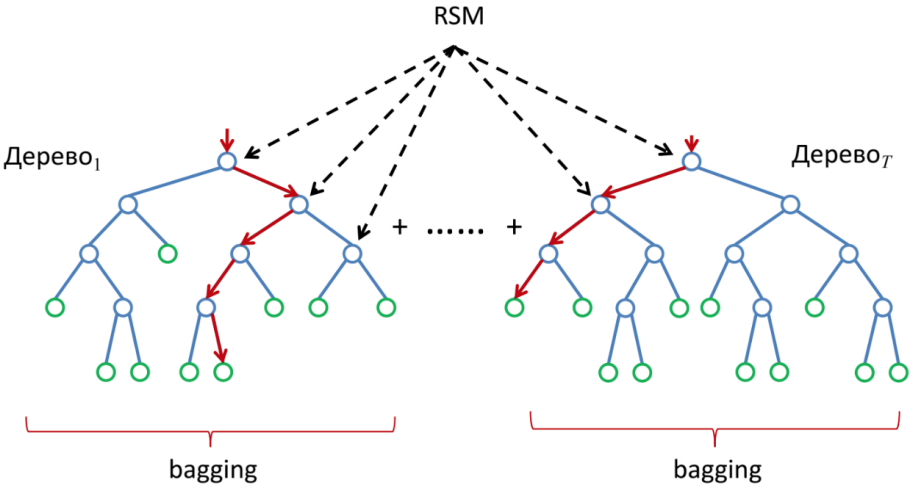
Ссылка на источник картинки

Случайный лес

- Random Forest – один из наиболее известных и популярных алгоритмов машинного обучения, появился в начале 2000-х

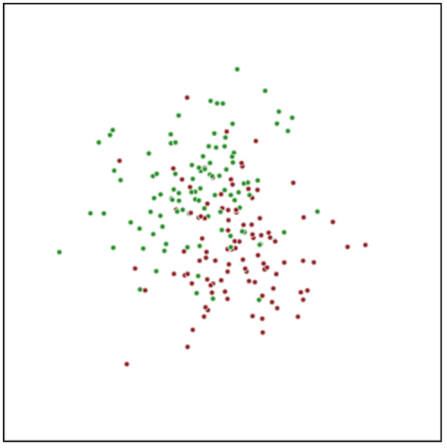
- **Алгоритм обучения:**
  1. для $n = 1, \ldots, N$:
  2. Сгенерировать подвыборку $\tilde{X}_n$ с возвращением
  3. Построить решающее дерево $b_n(x)$ по выборке $\tilde{X}_n$:
     - дерево строится, пока в каждом листе не окажется не более n_{min} объектов
     - в каждом узле сперва выбирается m случайных признаков, и оптимальное разбиение ищется только среди них
  4. Вернуть композицию $a_N(x) = \frac{1}{N} \sum_{i=1}^{N} b_n(x)$

Алгоритм Random Forest



Как работает случайный лес

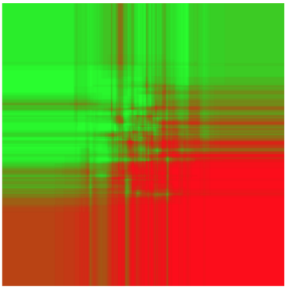
Сгенерируем выборку из двух двумерных нормальных распределений:



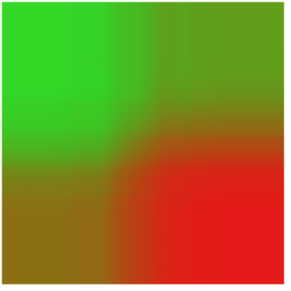
Ссылка на источник картинки

Как работает случайный лес

Реальная оптимальная границы



Результат работы Random Forest (50 деревьев)



Результат работы Random Forest (2000 деревьев)

Ссылка на источник картинки

Резюме по случайным лесам

Преимущества:

- ✓ Простая идея и реализация, простор для эвристических расширений
- ✓ Несложно настраивается при обучении
- ✓ Не переобучается с количеством итераций
- ✓ Работает с признаками разной природы, обрабатывает пропущенные значения
- ✓ Легко параллелится

Недостатки:

- ✗ Плохо обрабатывает линейные закономерности и большое число признаков одной природы
- ✗ Медленно работает на больших данных

Градиентный бустинг

- *Градиентный бустинг* – другой способ построения композиции
- Будем обучать N простых моделей $b_n(x)$, ответ композиции a_N на объекте x будет определяться по формуле

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

- Здесь γ_n – обучаемый вес (вклад) каждой модели
- Будем обучать каждую следующую модель так, чтобы она уменьшала ошибку всех уже построенных
- γ_0 и b_0 выбираются простыми, на их ошибку настраивается первая модель, и так далее
- Градиентный – потому что каждая новая модель приближает антиградиент суммы значений функции потерь текущей композиции во всех объектах
- Бустинг – жадный алгоритм, на каждом шаге мы пытаемся построить самую оптимальную модель для этого шага

Ссылка на источник изложения

Почему градиентный?

- **Напоминание:** в любой задаче обучения с учителем есть функция потерь
- На вход L получает настоящий ответ и предсказание модели, на выходе считает ошибку модели
- Например, в задаче классификации L может возвращать 0 при совпадении настоящего и предсказанного классов, и 1 – в противном случае
- Обучая модель, мы хотим минимизировать ошибку на всех ℓ объектах:

$$\sum_{i=1}^{\ell} L(y_i, a(x_i)) \to \min_{a_N}$$

- В случае градиентного бустинга $a = a_N = \sum_{n=0}^{N} \gamma_n b_n$
- Допустим, мы обучили $N-1$ моделей и хотим обучить N-ю
- Иными словами, надо, чтобы новая модель давала такие на объектах x_i такие значения s_i, чтобы

$$\sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + s_i) \to \min_{s_1,\ldots,s_\ell}$$

Почему градиентный?

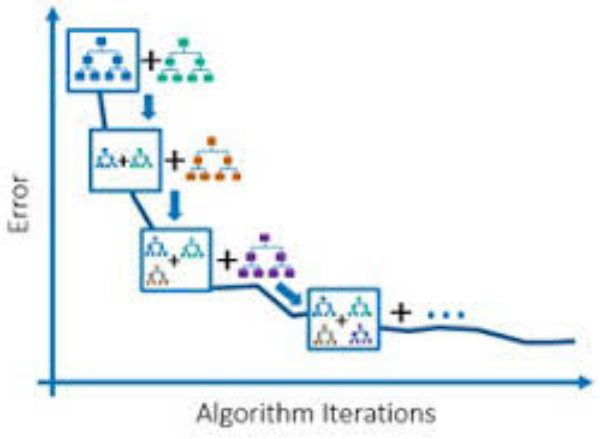
- Надо, чтобы новая модель давала такие на объектах x_i такие значения s_i, чтобы

$$\sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + s_i) \to \min_{s_1,\dots,s_\ell}$$

- Как выбрать значения s_i, которые должна выдавать новая модель?
- **Напоминание:** производная функции в точке показывает направление и скорость её изменения
- Мы хотим выбрать значения s_i, чтобы они уменьшали функцию суммарную ошибку композиции – давайте посчитаем производную L в точке $a_{N-1}(x_i)$ и присвоим s_i значение, противоположное производной
- Векторы $[s_1, \dots, s_\ell]$ будет вектором антиградиента функции $\sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i))$ – отсюда название алгоритма
- Дальше как обычно обучаем N-ю модель предсказывать значения s_i

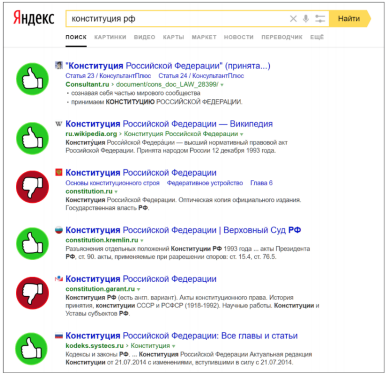
Градиентный бустинг над решающими деревьями

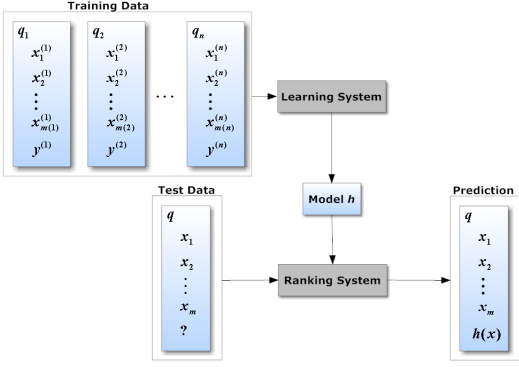
- Бустинг позволяет уменьшить bias моделей
- Variance либо сохраняется, либо растёт
- Базовые модели должны быть простыми
- Хорошо подойдут неглубокие деревья



Ссылка на источник картинки

Задача ранжирования поисковой выдачи





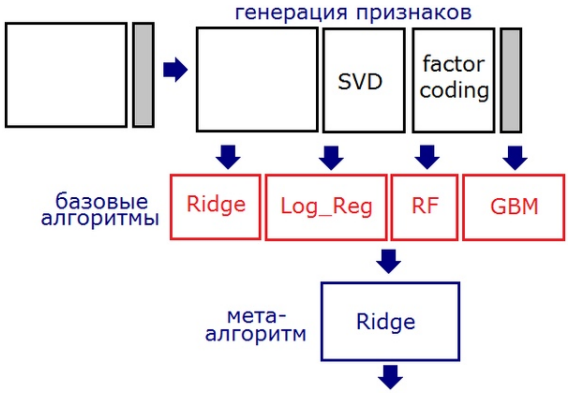
Ссылка на источник картинки

MatrixNet и CatBoost

- MatrixNet – библиотека Яндекса, в которой эффективно реализовано обучение и применений модели градиентного бустинга над решающими деревьями
- MatrixNet из коробки умеет решать задачу обучения ранжированию
- На MatrixNet основаны очень многие ML-системы Яндекса, в частности ранжирование в Поиске и в Дзене
- CatBoost – новая open-source библиотека для ГБ, работает как MatrixNet или лучше
- Главная особенность – умеет из коробки работать с категориальными признаками (пример: цвет, марка автомобиля)
- ГБ активно используется многими компаниями и институтами, например Яндексом, Yahoo, ЦЕРНом

Оффтоп: стекинг

- Бэггинг и бустинг – не единственные варианты композиции моделей
- Ещё одна популярная техника – *стекинг*
- Идея проста: давайте сначала обучим базовые модели, а потом на их выходах обучим ещё одну модель
- Ответ этой модели и будет ответом композиции



Ссылка на источник картинки

Итоги занятия

- Композиции моделей машинного обучения часто работают лучше, чем отдельные модели
- Случайный лес – популярная модель на конкурсах по анализу данных
- Градиентный бустинг над решающими деревьями – одна из самых мощных моделей машинного обучения, есть много эффективных реализаций
- На ГБ построены алгоритмы ранжирования Яндекса
- Стекинг – общая методика построения композиций, которая используется везде

Успехов!:)