Решающие деревья и ансамбли

Шевкунов Кирилл

ФИВТ МФТИ

Москва, 2018

План

- **Решающие деревья**
  - Терминология
  - Дерево как классификатор
  - Дерево как регрессор
  - Построение деревьев
- **Случайный лес**
  - Идея ансамблирования
  - Алгоритм
- **Градиентный бустинг**
  - Идея
  - Градиентный спуск
  - Алгоритм

Терминология

Определение

Граф - множество вершин и рёбер между ними

Терминология

Определение

Граф - множество вершин и рёбер между ними

Определение

Дерево - связный граф без циклов

Терминология

Определение

Граф - множество вершин и рёбер между ними

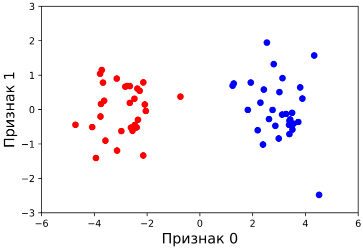
Определение

Дерево - связный граф без циклов

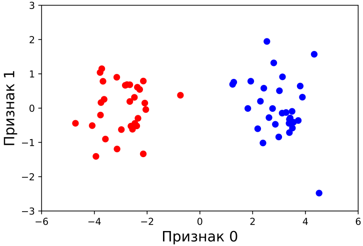
Определение

Лес - граф состоящий из несвязанных деревьев (любой граф без циклов)

Пример с разделимыми выборками

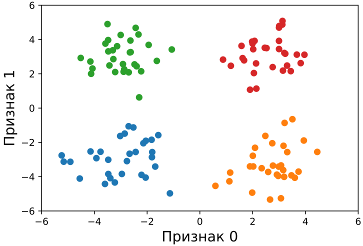


Пример с разделимыми выборками

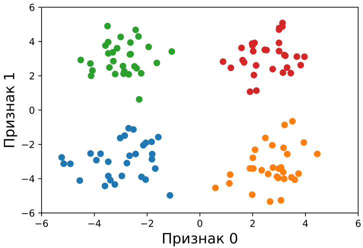



```python
def classify(X):
    if X[0] < 0:
        return "red"
    else:
        return "blue"
```

Пример с разделимыми выборками

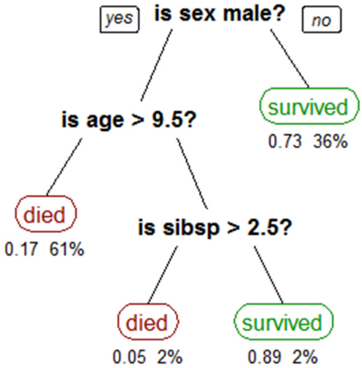


Пример с разделимыми выборками

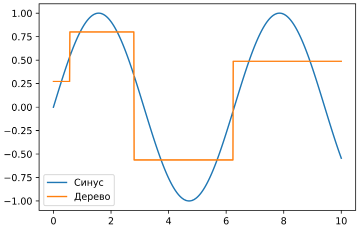


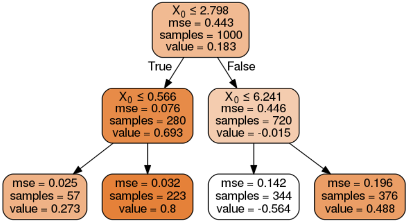

```
def classify(X):
    if X[0] < 0:
        if X[1] < 0:
            return "blue"
        else:
            return "green"
    else:
        if X[1] > 0:
            return "red"
        else:
            return "orange"
```

Решающее дерево

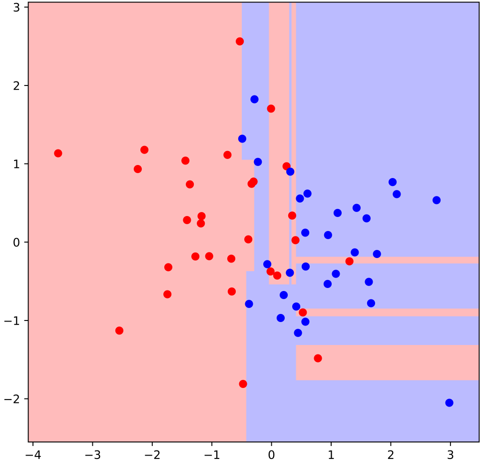


Решающее дерево для регрессии

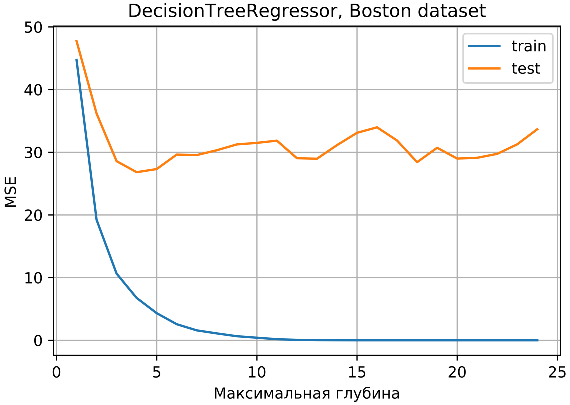




Разделяющая кривая и переобучение



Разделяющая кривая и переобучение



Построение дерева

Определение

Индекс неоднородности - величина, оценивающая неоднородность выборки.

Для регрессии:

- MSE: $H(Y) = \frac{1}{|Y|}\sum_{i=1}^{|Y|}(y_i - \overline{y})^2$

$$\overline{y} = \frac{1}{|Y|}\sum_{i=1}^{|Y|} y_i$$

Построение дерева

Для классификации (P_i - доля класса i в X, L - число классов):

- Энтропия: $H(X) = -\sum_{i=1}^{L} P_i \log P_i$
- Джини: $H(X) = \sum_{i=1}^{L} P_i(1 - P_i)$
- Misclassification: $H(X) = 1 - \max_{1..L} P_i$

Замечание: нужно считать, что $P_i log(P_i) = 0$ при $P_i = 0$

Построение дерева

Для классификации (P_i - доля класса i в X, L - число классов):

- Энтропия: $H(X) = -\sum_{i=1}^{L} P_i \log P_i$
- Джини: $H(X) = \sum_{i=1}^{L} P_i(1 - P_i)$
- Misclassification: $H(X) = 1 - \max_{1..L} P_i$

Замечание: нужно считать, что $P_i log(P_i) = 0$ при $P_i = 0$

Построение дерева

Определение

Уменьшение среднего индекса неоднородности при разбиении: $I(Q, f, v) = H(Q) - \frac{|L|}{|Q|}H(L) - \frac{|R|}{|Q|}H(R)$
Q *- выборка,* f *- признак,* v *- порог,* L *и* R *- соответсвующие им разбиения выборки* Q *на две части.*

Построение дерева

- Будем строить дерево от корня (стартовая вершина) к листьям (вершины, из которых некуда идти)
- В начале в стартовой вершине лежит вся выборка

Построение дерева

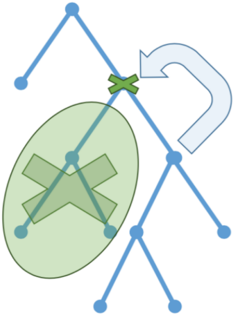
- Если в текущей вершине выполнен критерий останова - ничего не делаем в этой вершине.
- Выбрать f и v так, чтобы $I(Q, f, v)$ было максимально, например, перебрав все признаки и пороги.
- Разделим данную выборку на L и R согласно выбранным f и v, создадим двух потомков текущей вершиы и положим в них L и R соответственно.
- Повторим для каждой дочерней вершины.

Варианты критериев останова

- Максимальная глубина дерева.
- Минимальный размер выборки в вершине.
- Все обьекты в ввершине одного класса
- Требование на функционал качества вида "улучшился на k процентов"

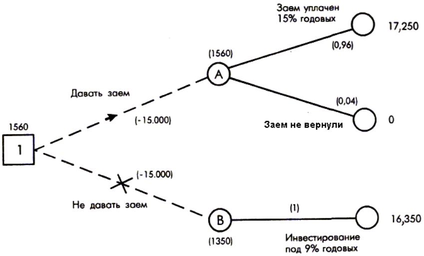
Дополнительно

- Построенные деревья можно уменьшать, пытаясь улучшить качество - "стрижка деревьев". Используется в алгоритмах C4.5 и CART построения деревьев.



Дополнительно

- Деревья малой глубины легко интерпретируемы человеком, из-за чего часто применяются, например, в банковской сфере, т.к. доверять "чёрному ящику"деньги сложнее.



Дополнительно

- Категориальные признаки можно обрабатывать, создав для каждого значения признака по потомку в дереве, а можно закодировать его средним значением целевой переменной среди элементов данного класса (Для бинарной классификации - доля обьектов). Для Джини и энтропийного критерия результат как при полном переборе. [Hastie T., Tibshirani R., Friedman J. (2009). The Elements of Statistical Learning.]

Ансабли деревьев

Пусть есть "слабый классификатор"(угадывающий правильный ответ с вероятностью p, немного большей, чем случайный предсказатель)
Можно ли как-то используя его, сделать "сильный классификатор"?

Ансабли деревьев

Возьмём три таких классификатора, независимо угадывающих с вероятностью $p = 0.55$ и будет относить обьект к тому классу, за который проголосовало большинство. Тогда вероятность угадать класс верно равна
$1 - 3p(1-p)^2 - p^3 = 0.57475 > p$

Ансабли деревьев

Проблема в том, что деревья строятся не случайно (алгоритм, описанный выше детерменирован, что, однако, верно не для всех реализаций) и тем более не независимо. Давайте модифицируем алгоритм и сделаем случайный лес из случайных деревьев.

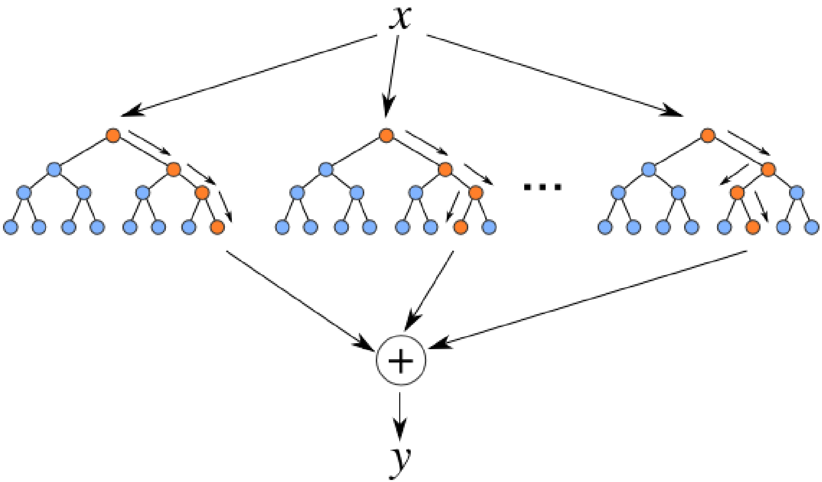
Bootstrap

Определение

Пусть дана выборка X из n обьектов. Выберем несколько раз, например n, равновероятно случайный обьект из X (выбор с повторениями). Выборку составленную из этих обьектов назовём bootstrap-выборкой.

Например, из [1, 2] могут получиться выборки [1, 1], [1, 2], [2, 1], [2, 2].

Случайный лес



Случайный лес

- Строим случайный лес с k деревьями.
- Сгенерируем k bootstrap-выборок из исходной
- Обучим на каждой выборке своё дерево, но при построении дерева, в каждом узле дерева, будем выбирать m случайных признаков и искать оптимальное разделение только по ним (m - заранее фиксировано, например, корень от числа признаков)
- Ответ всего алгоритма - класс, за который проголосовало большинство или среднее для классификации и регрессии соответственно.

Bagging

Определение

В данном подходе мы агрегируем данных алгоритмов, обученных на bootstrap-выборках. В общем случае такой подход называется bagging (bootstrap aggregating).

Принцип (Анны Карениной)

Все счастливые семьи похожи друг на друга, каждая несчастливая семья несчастлива по-своему.

Замечания

- Предлагается строить деревья максимально грубокими, чтобы они могли выделять сложные зависимости, тогда как из-за усреднения их переобученность не будет мешать (меньше variance при усреднении, но тот же bias)
- На выборках, в которых пропорции классов сильно отличаются, могут возникнуть проблемы
- Случайный лес, не переобучается при росте числа деревьев

Замечания

- При построении дерева в конкретной бутстрапной выборке не появилось около 37% всех обьектов, поэтому для фиксированного обьекта можно оценить качество его предсказания, используя деревья, в обучении которых он не участвовал. Усредняя это качество по всем элементам исходной выборки, получим Out-Of-Bag "самооценку"качества дерева.
- Также случайные леса умеют оценивать важность признаков, вычисляя feature importance.

Градиентный бустинг

Определение

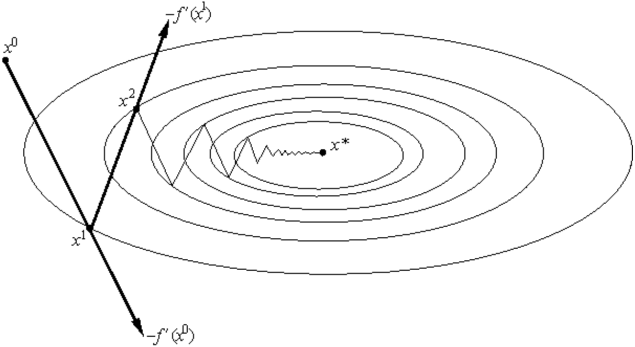
Пусть дана функция $f(x_1 ... x_n)$ нескольких переменных. Её градиентом ∇f называют вектор $(\frac{\partial f(x_1 ... x_n)}{\partial x_1} ... \frac{\partial f(x_1 ... x_n)}{\partial x_n})$ её частных производных.

Градиент является направлением наискорейшего роста, тогда как противоположное направление - антиградиент - является направлением наискорейшего убывания.

Градиентный спуск

Предложим следующий алгоритм численной минимизации функции f: Выберем начальную точку x_0.
После этого на каждой итерации будем двигаться в направлении наискорейшего спуска:
$x_{i+1} = x_i - \lambda_i \nabla f(x_i)$
λ_i - шаг алгоритма, выбирается постоянным или некоторыми другими способами

Градиентный спуск



Градиентный бустинг

Будем строить алгоритм как $A(x) = \sum_{i=0}^{n} b_i(x)$, где

b_i - базовые алгоритмы.

Начальное приближение выбирается произвольно, например, среднее значение целевой переменной.

$b_0(x) = \overline{y}$

Пусть L - функция потерь, непрерывно дифференцируемаяю

Градиентный бустинг

Уже построили $A_{N-1}(x) = \sum\limits_{i=0}^{N-1} b_i(x)$

Задача: $\min\limits_{b_N} \sum\limits_{i=1}^{|X|} L(y_i, \sum_{i=0}^{N-1} b_i(x_i) + b_N(x_i))$

Какой сдвиг b_N в пространстве алгоритмов будет давать наискорейшее убывание функции потерь?

Градиентный бустинг

Ответ: такой что $b_N(x_k) = -\frac{\partial L}{\partial a}(y_1, \sum_{i=0}^{N-1} b_i(x_k))$
Итого: новый алгоритм будем обучать на исходных признаках и целевых значения, указанных выше. Ответ обученного алгоритма на каждом шаге - сумма ответов алгоритмов, полученных на предыдущих шагах.

Градиентный бустинг

Заметим, что мы осуществляем по сути градиентный спуск в пространстве алгоритмов, поэтому, как и алгоритме градиентного спуска полезно добавить множитель λ, чтобы осуществлять шаг не на всю длину градиента, а только в его направлении:

$$A_N(x) = \sum_{i=0}^{N-1} b_i(x) + \lambda b_N(x_i)$$

На практике выбор этого множителя представляет нетривиальную задачу: при большом λ алгоритм не будет сходится, а при малом λ алгоритм будет сходится очень медленно.

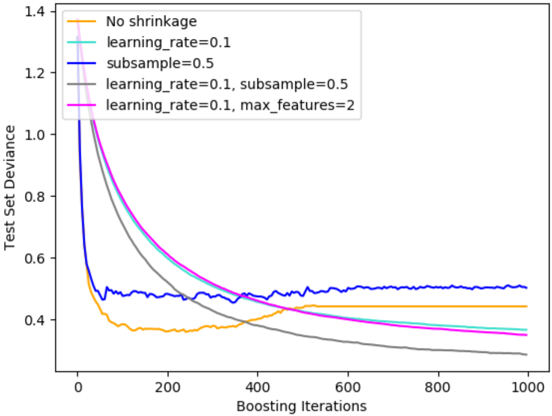
Градиентный бустинг для MSE

Если $L(y, a) = (a - y)^2$, то новые целевые значения, на которые обучается очередной алгоритм, вычислются очень просто: $-\frac{\partial L}{\partial a} = 2(y - a)$ Так как мы ввели шаг алгоритма, то двойку можно убрать и новый алгоритм нужно обучать на вектор ошибок предыдущих алгоритмов: $(y_1 - A_{N-1}(x_1), \ldots, y_{|X|} - A_{N-1}(x_{|X|}))$ с исходными признаками.

Градиентный бустинг

- В качестве базовых алгоритмов предлагается использовать неглубокие решающие деревья, которые можно быстро обучать.
- В отличие от случайного леса, алгоритм тяжело распараллеливается.

Градиентный бустинг



Ссылки

О решающих деревьях (pdf)
Визуализация градиентного бустинга