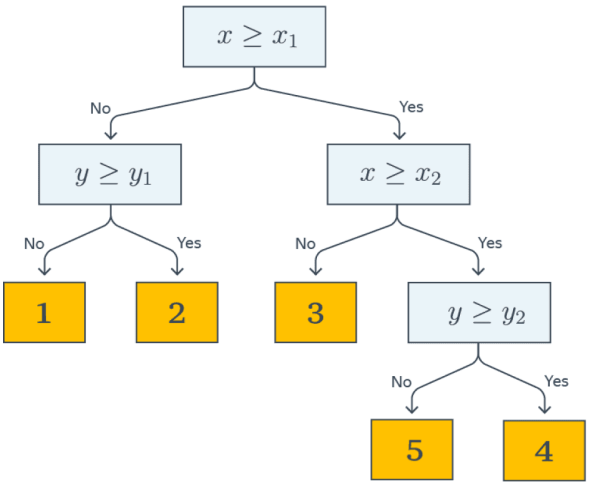
Решающие деревья

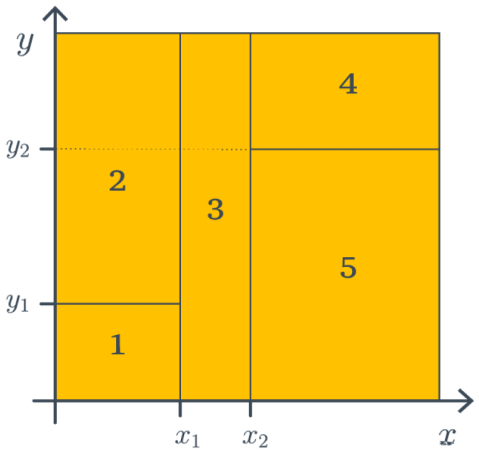
Конспект Яндекс учебника https://education.yandex.ru/handbook/ml

Введение

Решающее дерево (decision tree) предсказывает какое-то значения с помощью последовательного применения простых правил (предикатов). В чистом виде мало где применяются, но часто применяются в составе ансамблей - моделей, которые агрегируют множество других моделей для предсказания.

Пример решающего дерева





Определение решающего дерева

Решающее дерево - бинарное дерево, в котором:

каждой внутренней вершине v приписан предикат $B_v : \mathbb{X} \to 0, 1$;
каждой листовой вершине v приписан прогноз $c_v \in \mathbb{Y}$, где $\mathbb{Y}$ — область значений целевой переменной (в случае классификации листу может быть также приписан вектор вероятностей классов).

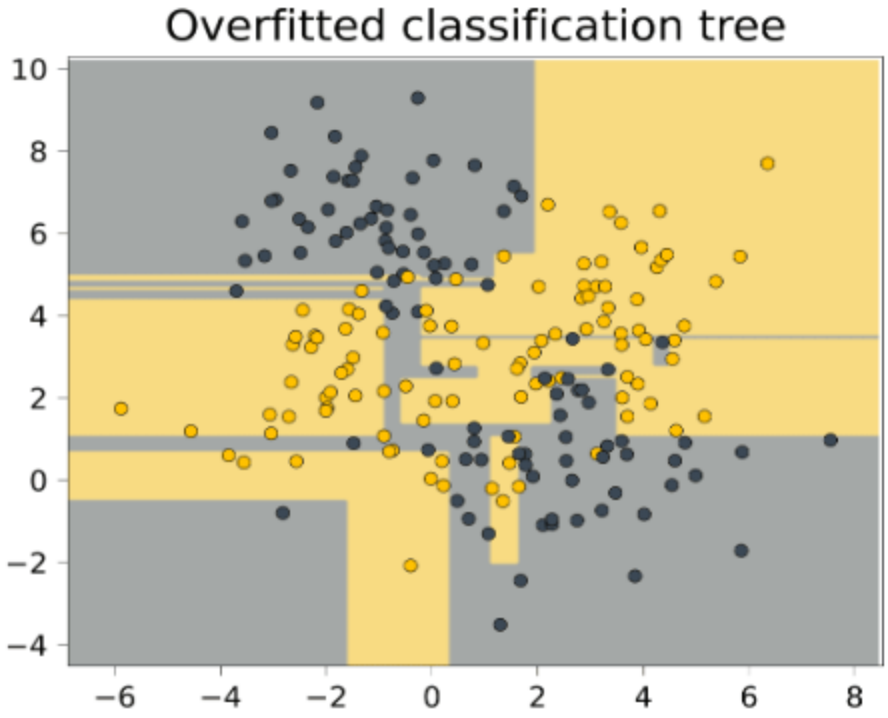
Соответственно в ходе предсказания спускаемся от корня дерева до листа. Вообще, предикатом по факту может являться что угодно, но все используют просто сравнения с каким-то числом.

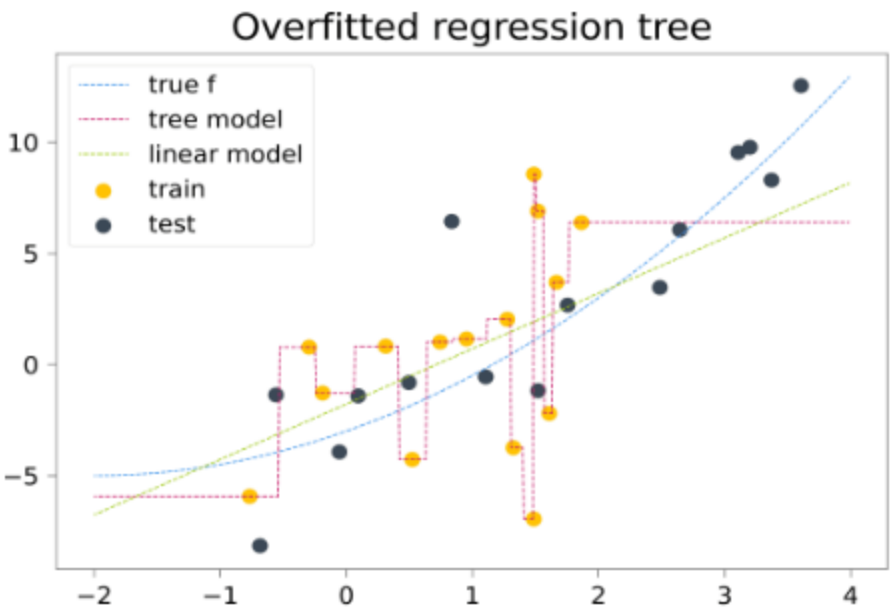
Свойства решающего дерева:

1) Из-за того, что функция получается кусочно-постоянной, то мы не можем использовать градиентные методы

2) Наше предсказание не сможет выйти за границы обучающей выборки

3) Если сделать дерево слишком большим и получится так, что в листах дерева будет очень мало объектов, то, конечно, оно потеряет обобщающую способность (переобучится).





Так как решающие деревья любят подстраиваться под обучающую выборку, то они очень плохо делают предсказания на тестовых выборках, значения которых отличаются от обучающих.

Сложность построения оптимального решающего дерева

Допустим, мы предсказываем какую-то зависимость и у нас есть функция потерь, которую нам нужно оптимизировать. Но так как градиентом мы пользоваться не можем, то будем использовать решающие пни (деревья из одной вершины). Всего существует не более (N-1)*D таких предикатов, тогда алгоритм будет выглядеть таким образом:

```
min_loss = inf
optimal_border = None

for j in range(D):
    for t in X[:, j]:     # Можно брать сами значения признаков в качестве порогов
        loss = calculate_loss(t, j, X, y)
        if loss < min_loss:
           min_loss, optimal_border = loss, (j, t)
```

O(N^2*D)

Это был алгоритм построения идеального дерева из одной вершины, теперь нужно обобщить для произвольной глубины. Если мы рекурсивно будем выполнять предыдущий алгоритм, то мы построим идеальное с точки зрения обучающей выборки дерево, которое будет очень слабым на тестовой. Поэтому мы должны строить дерево минимальное по глубине (чтобы не было переобучения), которое показывает приемлемое качество на тренировочной выборке.

Построение такого оптимального дерева - NP полная задача (не решается за полиномиальное время), поэтому упростим себе задачу:

1) Будем искать не оптимальное дерево, а просто хорошее. Для этого будем использовать жадный алгоритм при построении дерева (об этом дальше).

2) Попытаемся улучшить ассимптотику алгоритма.

Жадный алгоритм построения решающего дерева

Пусть Xm - множество объектов в текущем листе (изначально это вся выборка), тогда:

1. Создаём вершину v.
2. Если выполнен *критерий остановки* $Stop(X_m)$, то останавливаемся, объявляем эту вершину листом и ставим ей в соответствие ответ $Ans(X_m)$, после чего возвращаем её.
3. Иначе: находим предикат (иногда ещё говорят *сплит*) $B_{j,t}$, который определит наилучшее разбиение текущего множества объектов X_m на две подвыборки X_ℓ и X_r, максимизируя *критерий ветвления* $Branch(X_m, j, t)$.
4. Для X_ℓ и X_r рекурсивно повторим процедуру.

Функция Ans может выдавать метку класса в случае классификации или какой-то статистический параметр в случае регрессии.

Функция Stop - может быть какое-то тривиальное правило.

Функция Branch - измеряет насколько хорош предполагаемый сплит.

Оптимального выбора в общем случае нет, но есть хорошо зарекомендовавшие себя практики.

Методы регуляризации решающих деревьев.

Для этого есть разные критерии, обычно используются все сразу:

- ограничение по максимальной глубине дерева;
- ограничение на минимальное количество объектов в листе;
- ограничение на максимальное количество листьев в дереве;
- требование, чтобы функционал качества $Branch$ при делении текущей подвыборки на две улучшался не менее чем на s процентов.

P. S. можно улучшить ассисптотику построения дерева, если запоминать статистические параметры подвыборок.

Mixed integer optimization

Очень много проблем можно избежав представив данные к формату, которые применяют NP решатели, которые приближённо и быстро рих решат.

Актуальность

Практически всё сейчас встроено под капотом современных библиотек.