Введение в машинное обучение для Java-разработчиков

Лекция 2

Ермилов Сергей

education

Деревья решений

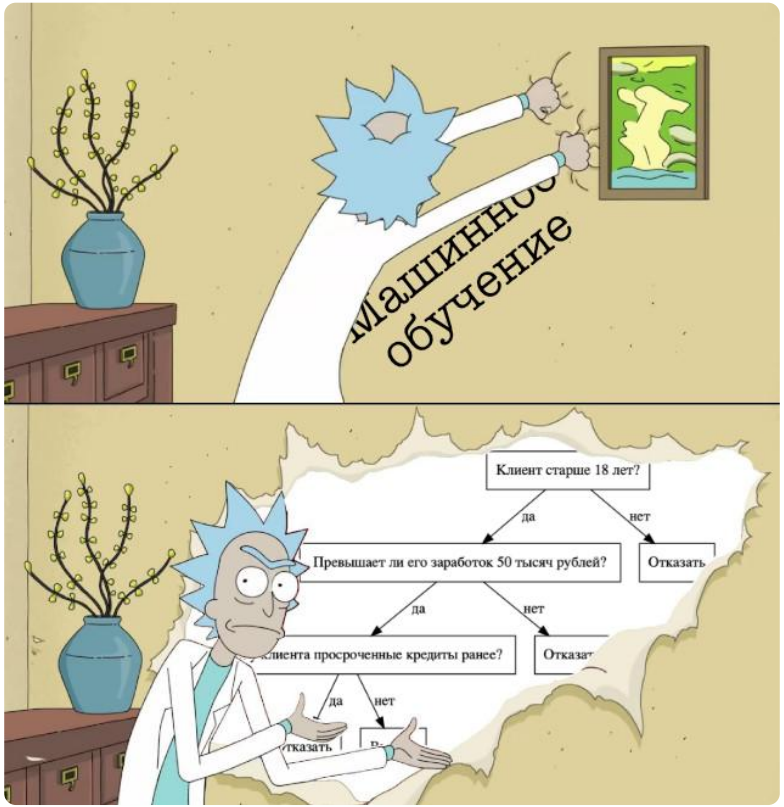
Деревья решений

Деревья решений — семейство алгоритмов значительно отличающееся от линейных моделей, но также используемые в задачах классификации и регрессии.

Метод основан на структуре данных деревьях, которые по сути представляют собой последовательные инструкции с условиями. Например, в задаче кредитного скоринга может быть следующий алгоритм принятия решения:

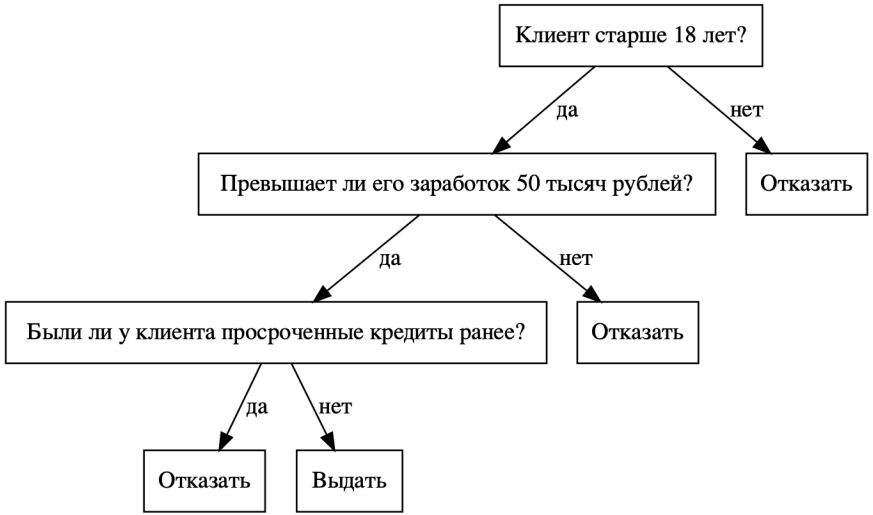
1. Старше ли клиент 18 лет? Если да, то продолжаем, иначе отказываем в кредите.
2. Превышает ли его заработок 50 тысяч рублей? Если да, то продолжаем, иначе отказываем в кредите.
3. Были ли у клиента просроченные кредиты ранее? Если да, отказываем в кредите, иначе выдаем.

Деревья решений



Деревья решений

В листьях (терминальных узлах) деревьев стоят значения целевой функции (прогноз), а в узлах - условия перехода, определяющие, по какому из ребер идти. Если речь идет о бинарных деревьях (каждый узел производит ветвление на две части), обычно, если условие в узле истинно, то происходит переход по левому ребру, если ложно, то по правому.



Деревья решений

В задачах машинного обучения чаще всего в вершинах прописываются максимально простые условия. Обычно это сравнение значения одного из признаков x_j с некоторым заданным порогом t:

$$[x_j \leq t]$$

Если решается задача классификации, конечным прогнозом является класс или распределение вероятностей классов. В случае регрессии прогноз в листе является вещественным числом.

Большим плюсом деревьев является тот факт, что они легко интерпретируемы.

Деревья решений

Деревья обладают отрицательным качеством — они склонны к переобучению. Легко построить дерево, в котором каждый лист будет соответствовать одному объекту обучающей выборки. Оно будет идеально подогнано под обучающую выборку, давать стопроцентный ответ на ней, но при этом не будет восстанавливать оригинальных закономерностей, и качество ответов на новых данных будет неудовлетворительным.

Построение деревьев решений

Построение деревьев решений

В машинном обучении деревья строятся последовательно от корня к листьям ("жадный" способ). Вначале выбирается корень и критерий, по которому выборка разбивается на две. Затем то же самое делается для каждого из потомков этого корня и так далее до достаточного уровня ветвления. Задача состоит в выборе способа разбиения каждого из узлов, то есть в выборе значения порога, с которым будет сравниваться значение одного из признаков в каждом узле.

Построение деревьев решений

Разбиение выбирается с точки зрения некоторого заранее заданного функционала качества $Q(X, j, t)$. Находятся наилучшие значения j и t для создания *предиката* $[x_j < t]$.

Параметры j и t можно выбирать перебором: признаков конечное число, а из всех возможных значений порога t можно рассматривать только те, при которых получаются различные разбиения на две подвыборки, таким образом, различных значений параметра t будет столько же, сколько различных значений признака x_j в обучающей выборке.

Построение деревьев решений

В каждой вершине производится проверка, не выполняется ли некоторое условие останова. Если оно выполняется, то разбиение прекращается, и вершина объявляется листом, который будет содержать прогноз.

В задаче классификации это будет класс, к которому относится большая часть объектов из выборки в листе X_m

$$a_m = argmax_{y \in Y}[\sum_{i \in X_m} y_i = y]$$

или доля объектов определенного класса k, если требуется предсказать вероятности классов

$$a_{mk} = |X_m|^{-1} \sum_{i \in X_m} [y_i = k]$$

В случае регрессии можно в качестве ответа давать средний по выборке в листе

$$a_m = |X_m|^{-1} \sum_{i \in X_m} y_i$$

После построения дерева может проводиться его *стрижка* (pruning) — удаление некоторых вершин согласно некоторому подходу с целью понижения сложности модели и повышения обобщающей способности.

Критерии информативности

Критерий информативности

В случае регрессии разброс будет характеризоваться дисперсией, поэтому критерий информативности будет записан в виде

$$H(X) = X^{-1} \sum_{i \in X} (y_i - y_{avg}(X))^2$$

где $y_{\text{avg}}(X)$ — среднее значение ответа в выборке X:

$$y_{avg}(X) = \ X\ ^{-1} \sum_{i \in X} y_i$$

Критерий информативности

В задаче классификации есть несколько способов определить критерий информативности.

Для начала обозначим через p_k долю объектов класса k в выборке X:

$$p_k = |X|^{-1} \sum_{i \in X} [y_i = k]$$

p_k будет характеризовать вероятность выдачи класса k.

Критерий информативности

Критерий Джини или *индекс Джини* выглядит следующим образом:

$$H(X) = \sum_{k=1}^{K} p_k(1 - p_k),$$

где K – количество классов в наборе данных X.

Его минимум достигается когда все объекты в подмножестве относятся к одному классу, а максимум - при равном содержании объектов всех классов. Критерий информативности Джини можно интерпретировать как вероятность ошибки случайного классификатора.

Критерий информативности

Энтропийный критерий или *энтропия Шеннона*. Записывается как:

$$H(X) = -\sum_{k=1}^{K} p_i log_2 p_i$$

Минимум энтропии также достигается когда все объекты относятся к одному классу, а максимум - при равномерном распределении. Стоит отметить, что в формуле полагается, что $0\log_2 0 = 0$.

Критерии останова

Критерии останова

Критерии останова — это критерии, которые показывают, нужно ли остановить процесс построения дерева. Правильный выбор критериев останова роста дерева может существенно повлиять на его качество. Существует большое количество возможных ограничений

Критерии останова

Критерии останова — это критерии, которые показывают, нужно ли остановить процесс построения дерева. Правильный выбор критериев останова роста дерева может существенно повлиять на его качество. Существует большое количество возможных ограничений:

- Ограничение максимальной глубины дерева. Этот критерий считается достаточно грубым, но хорошо зарекомендовавшим себя в построении композиций деревьев - когда несколько деревьев объединяются в один алгоритм.

Критерии останова

Критерии останова — это критерии, которые показывают, нужно ли остановить процесс построения дерева. Правильный выбор критериев останова роста дерева может существенно повлиять на его качество. Существует большое количество возможных ограничений:

- Ограничение максимальной глубины дерева. Этот критерий считается достаточно грубым, но хорошо зарекомендовавшим себя в построении композиций деревьев - когда несколько деревьев объединяются в один алгоритм.
- Ограничение максимального количества листьев.

Критерии останова

Критерии останова — это критерии, которые показывают, нужно ли остановить процесс построения дерева. Правильный выбор критериев останова роста дерева может существенно повлиять на его качество. Существует большое количество возможных ограничений:

- Ограничение максимальной глубины дерева. Этот критерий считается достаточно грубым, но хорошо зарекомендовавшим себя в построении композиций деревьев - когда несколько деревьев объединяются в один алгоритм.
- Ограничение максимального количества листьев.
- Ограничение минимального количества n объектов в листе. При этом оно должно быть достаточным, чтобы построить надежный прогноз.

Критерии останова

Критерии останова — это критерии, которые показывают, нужно ли остановить процесс построения дерева. Правильный выбор критериев останова роста дерева может существенно повлиять на его качество. Существует большое количество возможных ограничений:

- Ограничение максимальной глубины дерева. Этот критерий считается достаточно грубым, но хорошо зарекомендовавшим себя в построении композиций деревьев - когда несколько деревьев объединяются в один алгоритм.
- Ограничение максимального количества листьев.
- Ограничение минимального количества n объектов в листе. При этом оно должно быть достаточным, чтобы построить надежный прогноз.
- Останов в случае, когда все объекты в листе относятся к одному классу.

Критерии останова

Критерии останова — это критерии, которые показывают, нужно ли остановить процесс построения дерева. Правильный выбор критериев останова роста дерева может существенно повлиять на его качество. Существует большое количество возможных ограничений:

- Ограничение максимальной глубины дерева. Этот критерий считается достаточно грубым, но хорошо зарекомендовавшим себя в построении композиций деревьев - когда несколько деревьев объединяются в один алгоритм.
- Ограничение максимального количества листьев.
- Ограничение минимального количества n объектов в листе. При этом оно должно быть достаточным, чтобы построить надежный прогноз.
- Останов в случае, когда все объекты в листе относятся к одному классу.
- Требование улучшения функционала качества при разбиении на какую-то минимальную величину.

Критерии останова

Критерии останова — это критерии, которые показывают, нужно ли остановить процесс построения дерева. Правильный выбор критериев останова роста дерева может существенно повлиять на его качество. Существует большое количество возможных ограничений:

- Ограничение максимальной глубины дерева. Этот критерий считается достаточно грубым, но хорошо зарекомендовавшим себя в построении композиций деревьев - когда несколько деревьев объединяются в один алгоритм.
- Ограничение максимального количества листьев.
- Ограничение минимального количества n объектов в листе. При этом оно должно быть достаточным, чтобы построить надежный прогноз.
- Останов в случае, когда все объекты в листе относятся к одному классу.
- Требование улучшения функционала качества при разбиении на какую-то минимальную величину.

Подбор оптимальных критериев — сложная задача, которая обычно решается методом кросс-валидации.

Обрезка деревьев

Обрезка деревьев

В случае применения метода *стрижки (обрезки, прунинга)* деревьев использовать критерии останова необязательно, и можно строить переобученные деревья, затем снижая их сложность, удаляя листья по некоторому критерию (например, пока улучшается качество на отложенной выборке). Считается, что стрижка работает лучше, чем критерии останова.

Обрезка деревьев

Одним из методов стрижки является *cost-complexity pruning*. Допустим, мы построили дерево, обозначенное как T_0. В каждом из листьев находятся объекты одного класса, и значение функционала ошибки $R(T)$ при этом будет минимально на T_0. Для борьбы с переобучением к нему добавляют "штраф" за размер дерева (аналогично регуляризации, рассмотренной нами в предыдущих уроках) и получают новый функционал $R_\alpha(T)$:

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

где $|T|$ – число листьев в дереве, α – некоторый параметр регуляризации.

Таким образом если при построении дерева на каком-то этапе построения алгоритма ошибка будет неизменна, а глубина дерева увеличиваться, итоговый функционал, состоящий из их суммы, будет расти.

Обрезка деревьев

Существенным минусом стрижки является трудоемкость процедуры. Например, она может требовать вычисления функционала качества на валидационной выборке на каждом шаге. К тому же, на данный момент одиночные деревья на практике почти не используются, а используются композиции деревьев, и в этом случае стрижка как метод борьбы с переобучением становится еще более сложным подходом. Обычно в такой ситуации достаточно использовать простые критерии останова.

CART

CART

CART (Classification and regression trees) — первый из алгоритмов, состоящий в обычном последовательном построении дерева решений. Придуман в 1983 году. На первой итерации строятся все возможные разбиения исходного пространства на два и выбирается такое, при котором максимально выделен один из классов в одно подпространство. На следующих итерациях выбирается худший лист (с наибольшим разнообразием классов), и на нем проводится та же операция. Так продолжается до достижения одного из критериев останова. В качестве функции оценки качества разбиения используется критерий Джини

CART

Полученное дерево будет подогнано под обучающую выборку (*переобучено*), так что затем требуется его кросс-валидация или обрезка методом cost-complexity pruning.

Ансамбли деревьев

Ансамбли деревьев

Основным недостатком деревьев решений является их склонность к переобучению и тот факт, что даже при небольшом изменении обучающей выборки дерево может значительно измениться. Однако их объединение в *ансамбли* или *композиции* на практике дает очень хорошие результаты. Ансамбли — это методы, сочетающие в себе несколько алгоритмов машинного обучения для получения более мощной модели.

Ансамбли деревьев

В случае задачи регрессии при использовании композиции $a(x)$ из N *базовых алгоритмов* $b_n(x)$ ответом будет считаться среднее значение ответа каждого алгоритма

$$a(x) = N^{-1}\sum^{N}_{n=1} b_n(x),$$

в задачах классификации, соответственно, знак полученного усредненного ответа или (что аналогично) класс определяется путем *голосования*: объект относится к классу, за который "проголосовало" наибольшее число базовых алгоритмов.

Ансамбли деревьев

Одни из самых хорошо зарекомендовавших себя на практике решения задач классификации и регрессии с использованием деревьев решения — это *случайные леса (Random Forest)* и *градиентный бустинг (Gradient Boosting)*.

Случайный лес

Случайный лес

Случайные леса названы так из-за того, что в процесс построения деревьев, из которых они состоят, внесен элемент случайности для обеспечения уникальности каждого из деревьев. Такая рандомизация заключается в обучении базовых алгоритмов на разных подвыборках обучающей выборки. Один из способов построения случайных подвыборок — *бутстрап (bootstrap)*.

Случайный лес

Бутстрэппинг заключается в получении из выборки длины l нескольких разных выборок той же длины l. Для получения бутстрап-выборки из исходной выборки l раз выбирается случайный элемент, причем каждый раз новый элемент выбирается из всей выборки. Таким образом, в полученной в конечном итоге бутстрап-выборке некоторые элементы исходной выборки будут встречаться несколько раз, а некоторые будут вовсе отсутствовать, и при повторении N раз мы получим N разных выборок длиной l. Например, если у нас есть исходная выборка вида [a, b, c, d, e], возможными бутстрап-выборками могут быть [a, b, a, c, b] или [b, e, e, d, b] и т.д.

Случайный лес

При построении случайного леса вначале генерируется количество бутстрап-выборок, равное количеству деревьев в алгоритме. Для уменьшения корреляции базовых алгоритмов рандомизируют сам процесс построения каждого дерева: если в стандартном методе построения деревьев мы в каждом узле выбираем j-й признак и порог t, с которым сравнивается его значение, и потом эти значения оптимизируются с помощью функции ошибки, то в методе случайного леса в каждой вершине j-й признак выбирается не из всего пространства признаков, а из его случайного подмножества размера m, **которое каждый раз выбирается заново** (в этом отличие от метода случайных подпространств, где подпространство выбирается единожды и используется для построения всего дерева).

Случайный лес

Есть некоторые практические рекомендации по построению случайных лесов: в задачах классификации рекомендуется брать $m = d^{1/2}$, где d – общее число признаков, и строить дерево до тех пор, пока в каждом листе не останется по одному объекту, а в задаче регрессии принимать $m = d/3$ и строить дерево, пока в листьях не останется по пять объектов.

Далее построенные деревья объединяются в композицию, и при предсказаниях с его помощью используется усредненный ответ на каждом дереве.

Смещение и разброс

Смещение и разброс

Ошибка алгоритмов складывается из *смещения (bias)* (отклонение среднего ответа обученного алгоритма от ответа идеального алгоритма) и *разброса* или *дисперсии (variance)* (разброс ответов обученных алгоритмов относительно среднего ответа). Также к этому разложению обычно прибавляется *шум*, который характеризует ошибку идеального алгоритма и которым никак нельзя управлять — это характеристика входных данных.

Смещение и разброс

Как правило, простые семейства алгоритмов (например, линейные классификаторы) характеризуются высоким смещением и низким разбросом, а сложные семейства (в т.ч. деревья) наоборот — низким смещением и высоким разбросом. Можно сказать, что разброс характеризует чувствительность метода обучения к выборке, то есть насколько будет изменяться ответ обученного алгоритма в зависимости от изменений в обучающей выборке.

Смещение и разброс

Объединение нескольких деревьев с одинаковым смещением в композицию не будет приводить к увеличению ее смещения, а вот компонента разброса будет снижаться, если базовые алгоритмы независимы, то есть не коррелируют друг с другом: разброс композиции при усреднении ответов будет в N раз меньше разброса одного базового алгоритма. Однако, на практике базовые алгоритмы всегда в той или иной степени скоррелированы, так как обучаются на подвыборках одной выборки, поэтому возникает необходимость уменьшения корреляции получаемых алгоритмов.

Градиентный бустинг

Градиентный бустинг

Рассмотрим пример с задачей регрессии и квадратичной функцией потерь

$$Q(w, x) = \frac{1}{2} \sum_{i=1}^{N} (a(x_i) - y_i)^2 \rightarrow min$$

Обучим дерево и посчитаем насколько сильно отличаются прогнозы данного дерева от истинных значений

$$diff_i^1 = y_i - a_1(x_i)$$

Скорректируем ответы первого дерева с помощью обучения второго дерева

$$\hat{Y} = a_1(X) + a_2(X) = a_1(X) + diff^1$$

Повторим данный трюк K раз.

$$\hat{Y} = a_1(X) + a_2(X) + \ldots + a_k(X) = a_1(X) + diff^1 + \ldots . + diff^k$$

Градиентный бустинг. А где градиенты?

Рассмотрим квадратичную функцию потерь

$$Q(a, x) = \frac{1}{2} \sum_{i=1}^{N} (a(x_i) - y_i)^2 \to min$$

Обозначим $z_{ik} = a_k(x_i)$

Продифференцируем функцию потерь по z

$$Q'(z_{ik}, x) = z_{ik} - y_i = -diff = y_i - a_k(x_i)$$

Для каждого объекта очередное дерево в бустинге обучается предсказывать антиградиент функции потерь по предсказанию уже обученных алгоритмов в композиции.

Градиентный бустинг. А где градиенты?

Рассмотрим квадратичную функцию потерь

$$Q(a, x) = \frac{1}{2} \sum_{i=1}^{N} (a(x_i) - y_i)^2 \rightarrow min$$

Обозначим $z_i = a(x_i)$

Продифференцируем функцию потерь по z

$$Q'(z_i, x) = z_i - y_i = -diff = y_i - a(x_i)$$

Для каждого объекта очередное дерево в бустинге обучается предсказывать антиградиент функции потерь по предсказанию уже обученных алгоритмов в композиции.

Чтобы не переобучаться добавим параметр learning_rate μ

$$\hat{Y} = a_1(X) + \mu a_2(X) + \ldots + \mu a_k(X)$$

Вопросы

education