Семинары по решающим деревьям

Евгений Соколов
sokolov.evg@gmail.com

26 ноября 2015 г.

1 Решающие деревья

Решающие деревья хорошо описывают процесс принятия решения во многих ситуациях. Например, когда клиент приходит в банк и просит выдать ему кредит, то сотрудник банка начинает проверять условия:

1. Какой возраст у клиента? Если меньше 18, то отказываем в кредите, иначе продолжаем.
2. Какая зарплата у клиента? Если меньше 50 тысяч рублей, то переходим к шагу 3, иначе к шагу 4.
3. Какой стаж у клиента? Если меньше 10 лет, то не выдаем кредит, иначе выдаем.
4. Есть ли у клиента другие кредиты? Если есть, то отказываем, иначе выдаем.

Такой алгоритм, как и многие другие, очень хорошо описывается решающим деревом. Это отчасти объясняет их популярность.

Первые работы по использованию решающих деревьев для анализа данных появились в 60-х годах, и с тех пор несколько десятилетий им уделялось очень большое внимание. Несмотря на свою интерпретируемость и высокую выразительную способность, деревья крайне трудны для оптимизации из-за свой дискретной структуры — дерево нельзя продифференцировать по параметрам и найти с помощью градиентного спуска хотя бы локальный оптимум. Более того, даже число параметров у них не является постоянным и может меняться в зависимости от глубины, выбора критериев дробления и прочих деталей. Из-за этого все методы построения решающих деревьев являются жадными и эвристичными.

На сегодняшний день решающие деревья практически не используются как отдельные методы классификации или регрессии. В то же время, как оказалось, они очень хорошо объединяются в композиции — решающие леса, которые являются одними из наиболее сильных и универсальных моделей.

§1.1 Определение

Рассмотрим бинарное дерево, в котором:

- каждой внутренней вершине v приписана функция $\beta_v : \mathbb{X} \to \{0, 1\}$;

- каждой листовой вершине v приписана метка класса $c_v \in Y$.

Рассмотрим теперь алгоритм $a(x)$, который стартует из корневой вершины v_0 и вычисляет значение функции β_{v_0}. Если оно равно нулю, то алгоритм переходит в левую дочернюю вершину, иначе в правую, вычисляет значение предиката в новой вершине и делает переход или влево, или вправо. Процесс продолжается, пока не будет достигнута листовая вершина; алгоритм возвращает тот класс, который приписан этой вершине. Такой алгоритм называется *бинарным решающим деревом*.

На практике в большинстве случаев используются одномерные предикаты β_v, которые сравнивают значение одного из признаков с порогом. Существуют и многомерные предикаты, например:

- линейные $\beta_v(x) = [\langle w, x\rangle < s]$;
- метрические $\beta_v(x) = [\rho(x, x_v) < s]$, где точка x_v является одним из объектов выборки любой точкой признакового пространства.

Многомерные предикаты позволяют строить более сложные разделяющие поверхности, но очень редко используются на практике — например, из-за того, что усиливают и без того выдающиеся способности деревьев к переобучению. Далее мы будем говорить только об одномерных предикатах.

§1.2 Построение деревьев

Легко убедиться, что для любой выборки можно построить решающее дерево, не допускающее на ней ни одной ошибки. Скорее всего, это дерево будет переобученным и не сможет показать хорошее качество на новых данных. Можно было бы поставить задачу поиска дерева, которое является минимальным (с точки зрения количества листьев) среди всех деревьев, не допускающих ошибок на обучении — в этом случае можно было бы надеяться на наличие у дерева обобщающей способности. К сожалению, эта задача является NP-полной, и поэтому приходится ограничиваться жадными алгоритмами построения дерева.

Опишем базовый алгоритм построения бинарного решающего дерева. Начнем со всей обучающей выборки X^ℓ и найдем наилучшее ее разбиение на две части $R_1(j, s) = \{x \,|\, x_j \leqslant s\}$ и $R_2(j, s) = \{x \,|\, x_j > s\}$ с точки зрения заранее заданного критерия $Q(X, j, s)$. Найдя наилучшие значения j и s, создадим корневую вершину дерева, поставив ей в соответствие предикат $[x_j \leqslant s]$. Объекты разобьются на две части — одни попадут в левое поддерево, другие в правое. Для каждой из этих подвыборок повторим процедуру, построив дочерние вершины для корневой, и так далее. В каждой вершине мы проверяем, не выполнилось ли некоторое условие останова — и если выполнилось, то прекращаем рекурсию и объявляем эту вершину листом. Когда дерево построено, каждому листу ставится в соответствие ответ. В случае с классификацией это может быть класс, к которому относится больше всего объектов в листе. Для регрессии это может быть среднее значение, медиана или другая функция от целевых переменных объектов в листе. Выбор конкретной функции зависит от критерия качества.

Решающие деревья могут обрабатывать пропущенные значения — ситуации, в которых для некоторых объектов неизвестны значения одного или нескольких признаков. Для этого необходимо модифицировать процедуру разбиения выборки в вершине, что можно сделать несколькими способами.

После того, как дерево построено, можно провести его *стрижку* (pruning) — удаление некоторых вершин с целью понижения сложности и повышения обобщающей способности. Существует несколько подходов к стрижке, о которых мы немного упомянем ниже.

Таким образом, конкретный метод построения решающего дерева определяется:

1. Видом предикатов в вершинах;
2. Критерием информативности $Q(X, \beta_v)$;
3. Критерием останова;
4. Методом обработки пропущенных значений;
5. Методом стрижки.

Также могут иметь место различные расширения, связанные с учетом весов объектов, работой с категориальными признакам и т.д. Ниже мы обсудим варианты каждого из перечисленных пунктов.

§1.3 Критерии информативности

При построении дерева необходимо задать *критерий информативности* $Q(X, j, s)$, на основе которого осуществляется разбиение выборки на каждом шаге.

Для задач регрессии достаточно просто ввести критерий: чем меньше разброс ответов в вершине, тем эта вершина лучше. В случае классификации все сложнее, поскольку нет однозначного способа охарктеризовать разнообразие классов среди объектов в вершине. Рассмотрим различные способы задания таких критериев..

Пусть R_m — множество объектов обучающей выборки, попавших в вершину m. Через $N_m = |R_m|$ будем обозначать число таких объектов. Обозначим через p_{mk} долю объектов класса k ($k \in \{1, \ldots, K\}$), попавших в вершину m:

$$p_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} [y_i = k],$$

где $N_m = |R_m|$. Через k_m обозначим класс, чьих представителей оказалось больше всего среди объектов, попавших в вершину m: $k_m = \arg\max_k p_{mk}$.

1.3.1 Ошибка классификации

Вычислим долю объектов из R_m, которые были бы неправильно классифицированы, если бы вершина m была листовой и относила все объекты к классу k_m:

$$F_E(R_m) = \frac{1}{N_m} \sum_{x_i \in R_m} [y_i \neq k_m].$$

Критерий информативности при ветвлении вершины m определяется как

$$Q_E(R_m, j, s) = F_E(R_m) - \frac{N_\ell}{N_m} F_E(R_\ell) - \frac{N_r}{N_m} F_E(R_r),$$

где ℓ и r — индексы левой и правой дочерних вершин.

Задача 1.1. *Покажите, что ошибку классификации также можно записать в виде* $F_E(R_m) = 1 - p_{m,k_m}$.

Данный критерий является достаточно грубым, поскольку учитывает частоту p_{m,k_m} лишь одного класса.

1.3.2 Индекс Джини

Функционал имеет вид

$$F_G(R_m) = \sum_{k \neq k'} p_{mk} p_{mk'}.$$

Критерий информативности определяется так же, как и в предыдущем случае:

$$Q_G(R_m, j, s) = F_G(R_m) - \frac{N_\ell}{N_m} F_G(R_\ell) - \frac{N_r}{N_m} F_G(R_r).$$

Задача 1.2. *Покажите, что индекс Джини* $F_G(R_m)$ *также можно записать в виде* $F_G(R_m) = \sum_{k=1}^K p_{mk}(1 - p_{mk}) = 1 - \sum_{k=1}^K p_{mk}^2$.

Решение.

$$\sum_{k \neq k'} p_{mk} p_{mk'} = \sum_{k=1}^K p_{mk} \sum_{k' \neq k} p_{mk'} = \sum_{k=1}^K p_{mk}(1 - p_{mk}).$$

■

Задача 1.3. *Рассмотрим вершину* m *и объекты* R_m*, попавшие в нее. Сопоставим в соответствие вершине* m *алгоритм* $a(x)$*, который выбирает класс случайно, причем класс* k *выбирается с вероятностью* p_{mk}*. Покажите, что матожидание частоты ошибок этого алгоритма на объектах из* R_m *равно индексу Джини.*

Решение.

$$\mathsf{E} \frac{1}{N_m} \sum_{x_i \in R_m} [y_i \neq a(x_i)] = \frac{1}{N_m} \sum_{x_i \in R_m} \mathsf{E}[y_i \neq a(x_i)] = \frac{1}{N_m} \sum_{x_i \in R_m} (1 - p_{m,y_i}) =$$

$$= \sum_{k=1}^K \frac{\sum_{x_i \in R_m} [y_i = k]}{N_m} (1 - p_{mk}) = \sum_{k=1}^K p_{mk}(1 - p_{mk}).$$

■

Выясним теперь, какой смысл имеет максимизация критерия информативности Джини. Сразу выбросим из критерия $F(R_m)$, поскольку данная величина не зависит от j и s. Преобразуем критерий:

$$- \frac{N_\ell}{N_m} F(R_\ell) - \frac{N_r}{N_m} F(R_r) = -\frac{1}{N_m} \left(N_\ell - \sum_{k=1}^K p_{\ell k}^2 N_\ell + N_r - \sum_{k=1}^K p_{rk}^2 N_r \right) =$$

$$= \frac{1}{N_m} \left(\sum_{k=1}^K p_{\ell k}^2 N_\ell + \sum_{k=1}^K p_{rk}^2 N_r - N_m \right) = \{N_m \text{ не зависит от } j \text{ и } s\} =$$

$$= \sum_{k=1}^K p_{\ell k}^2 N_\ell + \sum_{k=1}^K p_{rk}^2 N_r.$$

Запишем теперь в наших обозначениях число таких пар объектов (x_i, x_j), что оба объекта попадают в одно и то же поддерево, и при этом $y_i = y_j$. Число объектов класса k, попавших в поддерево ℓ, равно $p_{\ell k} N_\ell$; соответственно, число пар объектов с одинаковыми метками, попавших в левое поддерево, равно $\sum_{k=1}^K p_{\ell k}^2 N_\ell^2$. Интересующая нас величина равна

$$\sum_{k=1}^K p_{\ell k}^2 N_\ell^2 + \sum_{k=1}^K p_{rk}^2 N_r^2. \tag{1.1}$$

Заметим, что данная величина очень похожа на полученное выше представление для критерия Джини. Таким образом, максимизацию критерия Джини можно условно интерпретировать как максимизацию числа пар объектов одного класса, оказавшихся в одном поддереве. Более того, иногда индекс Джини определяют именно через выражение (1.1).

1.3.3 Энтропийный критерий

Рассмотрим дискретную случайную величину, принимающую K значений с вероятностями $p_1, \ldots, p_K$ соответственно. *Энтропия* этой случайной величины определяется как $H(p) = -\sum_{k=1}^K p_k \log_2 p_k$.

Задача 1.4. *Покажите, что энтропия ограничена сверху и достигает своего максимума на равномерном распределении* $p_1 = \cdots = p_K = 1/K$.

Решение. Нам понадобится неравенство Йенсена: для любой вогнутой функции f выполнено

$$f\left(\sum_{i=1}^n a_i x_i\right) \geqslant \sum_{i=1}^n a_i f(x_i),$$

если $\sum_{i=1}^n a_i = 1$.

Применим его к логарифму в определении энтропии (он является вогнутой функцией):

$$H(p) = \sum_{k=1}^K p_k \log_2 \frac{1}{p_k} \leqslant \log_2 \left(\sum_{k=1}^K p_i \frac{1}{p_i}\right) = \log_2 K.$$

Наконец, найдем энтропию равномерного распределения:

$$-\sum_{k=1}^K \frac{1}{K} \log_2 \frac{1}{K} = -K \frac{1}{K} \log_2 \frac{1}{K} = \log_2 K.$$

■

Энтропия ограничена снизу нулем, причем минимум достигается на вырожденных распределениях ($p_i = 1$, $p_j = 0$ для $i \neq j$).

Энтропийный критерий определяется как

$$Q_H(R_m, j, s) = H(p_m) - \frac{N_\ell}{N_m} H(p_\ell) - \frac{N_r}{N_m} H(p_r),$$

где $p_i = (p_{i1}, \ldots, p_{iK})$ — распределение классов в i-й вершине. Видно, что данный критерий отдает предпочтение более «вырожденным» распределениям классов.

1.3.4 Критерии в задачах регрессии

В задачах регрессии, как правило, в качестве критерия выбирают дисперсию ответов в листе:

$$F(R_m) = \frac{1}{N_m} \sum_{x_i \in R_m} \left(y_i - \frac{1}{N_m} \sum_{x_j \in R_m} y_j \right)^2 .$$

Можно использовать и другие критерии — например, среднее абсолютное отклонение от медианы.

§1.4 Критерии останова

Можно придумать большое количестве критериев останова. Перечислим некоторые ограничения и критерии:

- Ограничение максимальной глубины дерева.
- Ограничение минимального числа объектов в листе.
- Ограничение максимального количества листьев в дереве.
- Останов в случае, если все объекты в листе относятся к одному классу.
- Требование, что функционал качества при дроблении улучшался как минимум на s процентов.

С помощью грамотного выбора подобных критериев и их параметров можно существенно повлиять на качество дерева. Тем не менее, такой подбор является трудозатратным и требует проведения кросс-валидации.

§1.5 Методы стрижки дерева

Стрижка дерева является альтернативой критериям останова, описанным выше. При использовании стрижки сначала строится переобученное дерево (например, до тех пор, пока в каждом листе не окажется по одному объекту), а затем производится оптимизация его структуры с целью улучшения обобщающей способности. Существует ряд исследований, показывающих, что стрижка позволяет достичь лучшего качества по сравнению с ранним остановом построения дерева на основе различных критериев.

Тем не менее, на данный момент методы стрижки редко используются и не реализованы в большинстве библиотек для анализа данных. Причина заключается в том, что деревья сами по себе являются слабыми алгоритмами и не представляют большого интереса, а при использовании в композициях они либо *должны* быть переобучены (в случайных лесах), либо должны иметь очень небольшую глубину (в бустинге), из-за чего необходимость в стрижке отпадает.

Одним из методов стрижки является *cost-complexity pruning*. Обозначим дерево, полученное в результате работы жадного алгоритма, через T_0. Поскольку в каждом из листьев находятся объекты только одного класса, значение функционала $R(T)$ будет минимально на самом дереве T_0 (среди всех поддеревьев). Однако

данный функционал характеризует лишь качество дерева на обучающей выборке, и чрезмерная подгонка под нее может привести к переобучению. Чтобы преодолеть эту проблему, введем новый функционал $R_\alpha(T)$, представляющий собой сумму исходного функционала $R(T)$ и штрафа за размер дерева:

$$R_\alpha(T) = R(T) + \alpha|T|, \tag{1.2}$$

где $|T|$ — число листьев в поддереве T, а $\alpha \geqslant 0$ — параметр. Это один из примеров *регуляризованных* критериев качества, которые ищут баланс между качеством классификации обучающей выборки и сложностью построенной модели. В дальнейшем мы много раз будем сталкиваться с такими критериями.

Можно показать, что существует последовательность вложенных деревьев с одинаковыми корнями:

$$T_K \subset T_{K-1} \subset \cdots \subset T_0,$$

(здесь T_K — тривиальное дерево, состоящее из корня дерева T_0), в которой каждое дерево T_i минимизирует критерий (1.2) для α из интервала $\alpha \in [\alpha_i, \alpha_{i+1})$, причем

$$0 = \alpha_0 < \alpha_1 < \cdots < \alpha_K < \infty.$$

Эту последовательность можно достаточно эффективно найти путем обхода дерева. Далее из нее выбирается оптимальное дерево по отложенной выборке или с помощью кросс-валидации.

§1.6 Обработка пропущенных значений

Одним из основных преимуществ решающих деревьев является возможность работы с пропущенными значениями. Рассмотрим некоторые варианты.

Пусть нам нужно вычислить функционал качества для предиката $\beta(x) = [x_j < s]$, и в выборке R_m для некоторых объектов не известно значение признака j — обозначим их через V_{mj}. В этом случае при вычислении функционала можно просто проигнорировать эти объекты, сделав поправку на потерю информации от этого:

$$Q(R_m, j, s) \approx \frac{|R_m \setminus V_{mj}|}{|R_m|} Q(R_m \setminus V_{mj}, j, s).$$

Затем, если данный предикат окажется лучшим, поместим объекты из V_{mj} как в левое, так и в правое поддерево. Также можно присвоить им при этом веса N_ℓ/N_m в левом поддереве и N_r/N_m в правом. В дальнейшем веса можно учитывать, добавляя их как коэффициенты перед индикаторами $[y_i = k]$ во всех формулах.

На этапе применения дерева необходимо выполнять похожий трюк. Если объект попал в вершину, предикат которой не может быть вычислен из-за пропуска, то прогнозы для него вычисляются в обоих поддеревьях, и затем усредняются с весами, пропорциональными числу обучающих объектов в этих поддеревьях. Иными словами, если прогноз вероятности для класса k в поддереве R_m обозначается через $a_{mk}(x)$, то получаем такую формулу:

$$a_{mk}(x) = \begin{cases} a_{\ell k}(x), & \beta_m(x) = 0; \\ a_{rk}(x), & \beta_m(x) = 1; \\ \frac{N_\ell}{N_m} a_{\ell k}(x) + \frac{N_r}{N_m} a_{rk}(x), & \beta_m(x) \text{ нельзя вычислить.} \end{cases}$$

Другой подход заключается в построении *суррогатных предикатов* в каждой вершине. Так называется предикат, который использует другой признак, но при этом дает разбиение, максимально близкое к данному.

§1.7 Учет категориальных признаков

Самый очевидный способ обработки категориальных признаков — разбивать вершину на столько поддеревьев, сколько имеется возможных значений у признака (multi-way splits). Такой подход может показывать хорошие результаты, но при этом есть риск получения дерева с крайне большим числом листьев.

Рассмотрим подробнее другой подход. Пусть категориальный признак x_j имеет множество значений $Q = \{u_1, \ldots, u_q\}$, $|Q| = q$. Разобьем множество значений на два непересекающихся подмножества: $Q = Q_1 \sqcup Q_2$, и определим предикат как индикатор попадания в первое подмножество: $\beta(x) = [x_j \in Q_1]$. Таким образом, объект будет попадать в левое поддерево, если признак x_j попадает в множество Q_1, и в первое поддерево в противном случае. Основная проблема заключается в том, что для построения оптимального предиката нужно перебрать $2^{q-1} - 1$ вариантов разбиения, что может быть не вполне возможным.

Оказывается, можно обойтись без полного перебора в случаях с бинарной классификацией и регрессией [1]. Обозначим через $R_m(u)$ множество объектов, которые попали в вершину m и у которых j-й признак имеет значение u; через $N_m(u)$ обозначим количество таких объектов.

В случае с бинарной классификацией упорядочим все значения категориального признака на основе того, какая доля объектов с таким значением имеет класс $+1$:

$$\frac{1}{N_m(u_{(1)})} \sum_{x_i \in R_m(u_{(1)})} [y_i = +1] \leqslant \ldots \leqslant \frac{1}{N_m(u_{(q)})} \sum_{x_i \in R_m(u_{(q)})} [y_i = +1],$$

после чего заменим категорию $u_{(i)}$ на число i, и будем искать разбиение как для вещественного признака. Можно показать, что если искать оптимальное разбиение по критерию Джини или энтропийному критерию, то мы получим такое же разбиение, как и при переборе по всем возможным $2^{q-1} - 1$ вариантам.

Для задачи регрессии с MSE-функционалом это тоже будет верно, если упорядочивать значения признака по среднему ответу объектов с таким значением:

$$\frac{1}{N_m(u_{(1)})} \sum_{x_i \in R_m(u_{(1)})} y_i \leqslant \ldots \leqslant \frac{1}{N_m(u_{(q)})} \sum_{x_i \in R_m(u_{(q)})} y_i.$$

Именно такой подход используется в библиотеке Spark MLlib [1].

§1.8 Методы построения деревьев

Существует несколько популярных методов построения деревьев:

- ID3: использует энтропийный критерий. Строит дерево до тех пор, пока в каждом листе не окажутся объекты одного класса, либо пока разбиение вершины дает уменьшение энтропийного критерия.

[1] http://spark.apache.org/docs/latest/mllib-decision-tree.html

- C4.5: использует критерий Gain Ratio (нормированный энтропийный критерий). Критерий останова — ограничение на число объектов в листе. Стрижка производится с помощью метода Error-Based Pruning, который использует оценки обобщающей способности для принятия решения об удалении вершины. Обработка пропущенных значений осуществляется с помощью метода, который игнорирует объекты с пропущенными значениями при вычислении критерия ветвления, а затем переносит такие объекты в оба поддерева с определенными весами.

- CART: использует критерий Джини. Стрижка осуществляется с помощью Cost-Complexity Pruning. Для обработки пропусков используется метод суррогатных предикатов.

Список литературы

[1] *Hastie T., Tibshirani R., Friedman J.* (2009). The Elements of Statistical Learning.