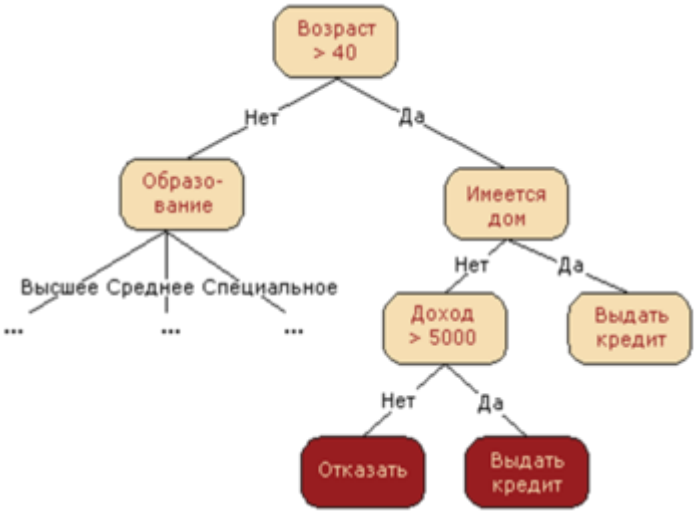
Решающее дерево (Decision tree) — решение задачи обучения с учителем, основанный на том, как решает задачи прогнозирования человек. В общем случае — это k-ичное дерево с *решающими правилами*в нелистовых вершинах (узлах) и некотором заключении о целевой функции в листовых вершинах (*прогнозом*). *Решающее правило* — некоторая функция от объекта, позволяющее определить, в какую из дочерних вершин нужно поместить рассматриваемый объект. В листовых вершинах могут находиться разные объекты: класс, который нужно присвоить попавшему туда объекту (в задаче классификации), вероятности классов (в задаче классификации), непосредственно значение целевой функции (задача регрессии).

Чаще всего на практике используются **двоичные решающие деревья**.



Алгоритм построения

В корне дерева — рассматриваем всю обучающую выборку.

1. Проверить критерий останова алгоритма. Если он выполняется, выбрать для узла выдаваемый прогноз, что можно сделать несколькими способами.

2. Иначе требуется разбить множество на несколько не пересекающихся. В общем случае в вершине tзадаётся решающее правило $Q_t(x)$, принимающее некоторый диапазон значений. Этот диапазон разбивается на R_t непересекающихся множеств объектов, $S_1, S_2, \ldots, S_{R_t}$, где R_t — количество потомков у вершины, а каждое S_i — это множество объектов, попавших в i-го потомка.

3. Множество в узле разбивается согласно выбранному правилу, для каждого узла алгоритм запускается рекурсивно.

Решающие правила

Чаще всего в качестве $Q_t(x)$ берут просто один из признаков, то есть $x^{i(t)}$.

Традиционные разбиения на диапазоны:

- $S_t(j) = \left\{x \in \mathbb{X} : h_j \le x^{i(t)} \le h_{j+1}\right\}$ для выбранных $h_1, \ldots, h_{j+1}$
- $S_t(1) = \{x \in \mathbb{X} : \langle x, v\rangle \le 0\}; S_t(2) = \{x \in \mathbb{X} : \langle x, v\rangle > 0\}$ — по сути проверка угла.
- $S_t(1) = \{x \in \mathbb{X} : \rho(x, x_0) \le h\}; S_t(2) = \{x \in \mathbb{X} : \rho(x, x_0) > h\}$, где расстояние ρопределено в некотором метрическом пространстве (например, $\rho(x, y) = |x - y|$).
- $S_t(1) = \left\{x \in \mathbb{X} : x^{i(t)} \le h\right\}; S_t(2) = \left\{x \in \mathbb{X} : x^{i(t)} > h\right\}$ — **предикаты**, $\langle x, v\rangle$ — скалярное произведение векторов.

В целом, взять можно любые решающие правила, но лучше — интерпретируемые, поскольку их легче настраивать. Особого смысла брать что-то сложнее предикатов нет, так как уже с их помощью можно получить дерево со 100%-й точностью на обучающейся выборке (но при этом и скорее всего переобучиться).

Выбор оптимального решающего правила

Считаем, что решаем задачу многоклассовой классификации или регрессии в R-ичном дереве.

Обычно для построения дерева выбирается целое семейство решающих правил. Чтобы найти среди них оптимальное для каждого конкретного узла, требуется ввести некоторый критерий оптимальности. Для этого вводят некоторую меру $\mathrm{I}(t)$ измерения того, насколько разбросаны объекты (регрессия) или перемешаны классы (классификация) в некотором узле t. Эта мера называется критерием информативности.

Затем для каждого варианта решающего правила подсчитывается мера того, насколько будут разбросаны объекты (регрессия) или перемешаны классы (классификация) при таком разбиении:

$$\Delta \mathrm{I}(X_t, t) = \mathrm{I}(X_t, t) - \sum_{i=1}^{R} \mathrm{I}(X_{t_i}, t_i) \frac{N(t_i)}{N(t)}$$, где R — на сколько узлов разбивается узел, t — текущий узел, $t_1, \ldots, t_R$ — узлы-потомки, получающиеся при выбранном разбиении, $N(t_i)$ — количество объектов обучающей выборки, попадающие в потомок i, $N(t)$ — попавших в текущий узел, X_{t_i} — объекты, попавшие в t_i-ую вершину.

$\Delta \mathrm{I}(X_t, t)$ также называется **Information gain**, то есть сколько информации мы получим при таком разбиении. Для выбора решающего правила требуется взять argmax от неё по всевозможным признакам и параметрам семейства решающих правил. При таком взятии мы как раз и получим оптимальное разбиение множества объектов в текущей вершине.

Критерии останова

Семинары Соколова, стр. 6

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

Стрижка **Править**

Для любой обучающей выборки существует дерево, которое не будет допускать на нём ни одной ошибки. Подобрать правильный критерий останова бывает затруднительно, поэтому прибегают к стрижке — строят дерево целиком, а затем начинают обрубать узлы с листов.

Стрижка по валидационной выборке

Лекции Воронцова, 63-я минута.

Рассмотрим бинарное решающее дерево на примере задачи классификации. Выделим валидационной выборку порядка половины размера всей выборки (или можно, например, разделить выборку на 2 части следующим образом: $\frac{2}{3}$

всей выборки оставить в качестве тренировочной, $\frac{1}{3}$— в качестве валидационной)

Построим дерево по тренировочной выборке. Пропустим через построенное дерево валидационную выборку и рассмотрим любую внутреннюю вершину t и её левую и правую вершины L_t, R_t. Если до t не дошло ни одного объекта из валидационной выборки, то говорим, что эта вершина (и все её поддеревья) незначимые и делаем из t листовую (ставим в качестве предиката мажоритарный класс в этой вершине по тренировочной выборке). Если до t дошли объекты из валидационной выборки, то рассмотрим следующие 3 величины:

1. Число ошибок классификации поддеревом из вершины t
2. Число ошибок классификации поддеревом из вершины L_t
3. Число ошибок классификации поддеревом из вершины R_t

Если в 1) Число ошибок классификации поддеревом из вершины t равно 0, то значит делаем из tлистовую с соответствующим прогнозом для класса.

Выберем минимум из трёх вышеприведенных пунктов. В зависимости от того, какое из них минимально, сделаем соответственно следующие действия:

1. Ничего не делать
2. Заменить дерево из вершины t деревом из вершины L_t
3. Заменить дерево из вершины t деревом из вершины R_t

Cost-complexity pruning

Семинары Соколова, стр. 6 – 7.

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:

$$R_\alpha(T) = R(T) + \alpha|T| \quad (*)$$

где $|T|$ — число листьев в поддереве T, а $\alpha > 0$ — 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: $T_K \subset T_{K-1} \subset \ldots \subset T_0$, (здесь T_K — тривиальное дерево, состоящее из корня дерева T_0), в которой каждое дерево T_i минимизирует критерий $(*)$ для α из интервала $\alpha \in [\alpha_i, \alpha_{i+1})$, причем $0 = \alpha_0 < \alpha_1 < \ldots < \alpha_K < +\infty$. Эту последовательность можно достаточно эффективно найти путем обхода дерева. Далее из нее выбирается оптимальное дерево по отложенной выборке или с помощью кросс-валидации.

Выбор прогноза в листе

- Самый простой способ — взять самый часто встречающийся класс среди объектов обучающей выборки, попавших в этот лист, для классификации или среднее целевых функций этих объектов для регрессии.
- В задачах классификации с k классами в листе t также можно хранить вероятности классов — например, по классической вероятности: $\mathbb{P}(y = y_i|x) = \frac{\#X_t}{N(t)}$, где $\mathbb{Y} = \{y_1, y_2, \ldots, y_k\}$ — классы. Более подробно, как в таком случае выбирать класс, описано в обработке пропусков
- Если задана матрица штрафов, то есть в случае несимметричных потерь, можно минимизировать штраф и взять в качестве класса в листе $\hat{y}$:

$\hat{y} = \underset{y \in \mathbb{Y}}{\operatorname{argmin}} \sum_i \lambda_{y_i y}$, где $\lambda_{y_i y}$ — элементы матрицы штрафов.

- В регрессии — использовать функцию потерь, то есть:

$$\hat{y} = \underset{y \in \mathbb{Y}}{\operatorname{argmin}} \sum_i L(y, y_i)$$

Выбор функции потерь в общем случае является параметром алгоритма.

Прогнозирование

Для случая отсутствия пропущенных значений. Алгоритм: начиная с корня, применить к новому объекту решающее правило. Таким образом определяется, в какой потомок объект должен "попасть", и рекурсивно запустить этот процесс для него.

Сложность прогнозирования для одного объекта для полностью построенного дерева — $O(h)$, где h — высота дерева. Вообще говоря, дерево может быть несбалансированным, поэтому оценка в $O(\log N)$, где N — размер тренировочной выборки, не гарантируется.

Метод обработки пропущенных значений

Пропуск пропущенных значений

Лекции Воронцова, 59-я минута.
При обучении дерева объекты с пропущенными значениями у признака, по которому идет разбиение,
игнорируются:

$$\Delta \mathrm{I}(X_t, t) \approx \frac{\#\{x \in X_t : x_{i(t)} \text{not missing}\}}{N(t)} \cdot \Delta \mathrm{I}(\{x \in X_t : x_{i(t)} \text{not missing}\}, t)$$

При построении прогноза при необходимости разбить подвыборку в вершине t по отсутствующему признаку происходит следующая процедура: будем как бы предполагать, что этот признак принимает случайное значение. Определим по обучающей выборке вероятности, с которой новый объект попадёт к каждому потомку — $w_1, w_2, \ldots, w_R, w_i = \frac{N(t_i)}{N(t)}$. Затем отправим объект независимо к каждому потомку, получим прогнозы $y_1, y_2, \ldots, y_R$. В случае регрессии это будут непосредственно значения целевой функции, в случае классификации — вероятности принадлежности какому-то зафиксированному классу y(который, как будет видно ниже, надо перебирать): $y_i = \mathbb{P}(y|x, t_i)$

Дальше можно поступать образом, схожим с выбором прогноза в листе: так, для регрессии можно просто объединить отклики с вероятностями в качестве весов, $\hat{y} = w_1 y_1 + \ldots + w_R y_R$.

Пусть t_0 — корень дерева. В случае классификации $$\hat{y} = \underset{y \in \mathbb{Y}}{\operatorname{argmax}} \mathbb{P}(y|x, t_0) \overset{\text{def}}{=} \underset{y \in \mathbb{Y}}{\operatorname{argmax}} \sum_{i=1}^{R} w_i \mathbb{P}(y|x, t_i)$$

Такой алгоритм работы с пропущенными значениями используется в ID3, C4.5.

Суррогатные разбиения

Находим другой признак, по которому разбиение будет максимально похожим. Выкидываем те объекты, по которому по данному признаку есть пропущенные значения, делаем разбиение. Ищем другой признак (видимо, предполагаем, что по нему нет пропущенных значений, либо сразу выкидывать объекты с пропущенными значениями и по первому приведенному признаку и по второму), берем максимально похожее разбиение на изначальное. Например, можно сравнивать как можно большее пересечение левых и правых поддеревьев (в случае бинарного дерева).

Такой алгоритм работы с пропущенными значениями используется в CART

One-hot кодирование

Из вершины t делаем столько детей, сколько уникальных значений у категориального признака.

При таком подходе размер дерева увеличивается $\Rightarrow$ увеличивается риск переобучения.

Перевод категориальных в вещественные

Семинары Соколова, стр. 8

Пусть категориальный признак x_j имеет множество значений $Q = \{u_1, \ldots, u_q\}, |Q| = q$. Разобьем множество значений на два непересекающихся подмножества: $Q = Q_1 \bigsqcup Q_2$, и определим предикат как индикатор попадания в первое подмножество: $\beta(x) = \mathbb{I}[x_j \in Q_1]$. Таким образом, объект будет попадать в левое поддерево, если признак x_j попадает в множество Q_1, и в правое 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. 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 $+1$:

$$\frac{1}{N_m(u_{(1)})} \sum_{x_i \in R_m(u_{(1)})} \mathbb{I}[y_i = +1] \leq \cdots \leq \frac{1}{N_m(u_{(q)})} \sum_{x_i \in R_m(u_{(q)})} \mathbb{I}[y_i = +1]$$

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:

$$\frac{1}{N_m(u_{(1)})} \sum_{x_i \in R_m(u_{(1)})} y_i \leq \cdots \leq \frac{1}{N_m(u_{(q)})} \sum_{x_i \in R_m(u_{(q)})} y_i$$

Источник: http://ru.learnmachinelearning.wikia.com

Видео-урок по решающим деревьям:

https://ru.coursera.org/learn/vvedenie-mashinnoe-obuchenie/lecture/kVz6T/rieshaiushchiie-dieriev-ia