2022

Лекция 11. Дерево решений

МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ

ПАПУЛИН С.Ю. (PAPULIN.STUDY@YANDEX.RU)

Дополнительные материалы доступны по ссылке: https://github.com/mlmethods

Содержание

1. Деревья решений 2
2. CART (Classification and Regression Trees) 2
   2.1. Регрессия 2
   2.2. Классификация 5
   2.3. Подрезка дерева решений 8
   2.4. Деревья решений и линейные модели 9
3. Комбинации методов 10
   3.1. Бэггинг 10
   3.2. Случайный лес 12
   3.3. Бустинг 14
   3.4. Стекинг 16
Список литературы 17

1. Деревья решений

Особенности дерева решений

- Может быть использована как для регрессии, так и для классификации
- Относится к непараметрическим методам
- Пространство признаков (предикторов) разбивается на ряд простых регионов
- В качестве предсказания для x, принадлежащего некоторому региону, используется среднее значение или мода обучающих наблюдений того же региона для задачи регрессии и принцип большинства для задачи классификации
- Результат процесса разделения пространства признаков на регионы представляется в виде дерева
- Деревья решений просты для интерпретации
- Однако могут давать результаты хуже, чем ранее рассмотренные методы

Комбинации/ансамбли деревьев

- Для улучшения качества предсказания используются комбинации деревьев, такие как бэггинг, случайный лес, сверхслучайные деревья, бустинг
- Каждый из этих подходов использует множество деревьев, которые комбинируются для получения единого предсказания
- Комбинация большого количества деревьев часто приводит к существенному улучшению качества предсказания за счет некоторой потери в интерпретируемости

2. CART (Classification and Regression Trees)

2.1. Регрессия

Пример

Предсказание зарплаты игрока в зависимости от количества хитов и лет в профессиональной лиге.

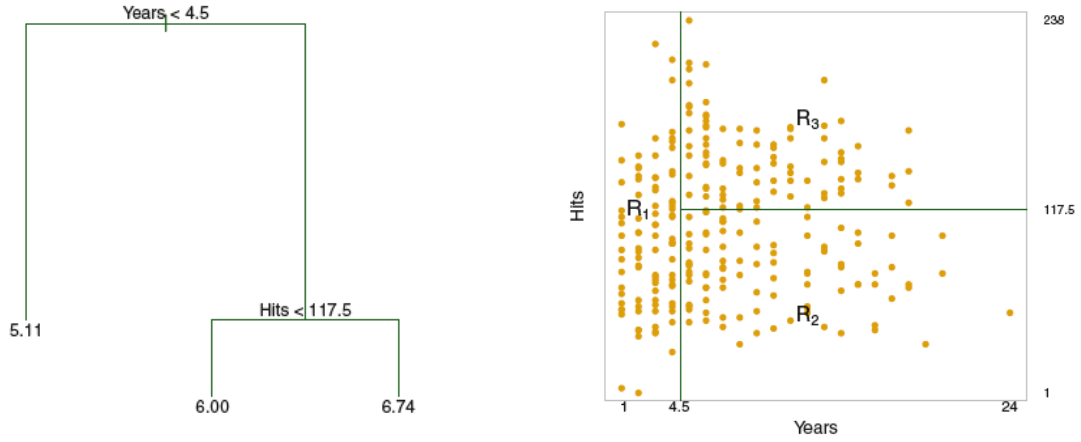

Рисунок – Пример построения дерева решений [1]

Терминология:

- Дерево строится сверху вниз
- R_1, R_2, R_3 соответствуют терминальным узлам или листьям дерева
- Точки вдоль дерева, где происходит деление пространства признаков, соответствуют внутренним узлам
- Ветви – сегменты дерева, которые соединяют узлы

Данное дерево сегментировало игроков на три региона пространства признаков:

- Игроки, кто играет 4 и менее лет
- Игроки, кто играет 5 и более лет и сделал менее 118 хитов в прошлом году
- Игроки, кто играет 5 и более лет и сделал по крайней мере 118 хитов в прошлом году

Эти три региона можно записать как

$$R_1 = \{X|Years < 4.5\}$$

$$R_2 = \{X|Years \geq 4.5, Hits < 117.5\}$$

$$R_3 = \{X|Years \geq 4.5, Hits \geq 117.5\}$$

Интерпретация дерева:

- Количество лет – наиболее важный признак для определения размера зарплаты. Игроки с меньшим опытом зарабатывают меньше, чем более опытные
- Если игрок менее опытный, то его количество хитов не сильно влияет на его зарплату
- Для игроков, кто провел 5 и более лет в профессиональной лиге, количество хитов в прошлом году влияет на зарплату. Игроки с большим количеством хитов, как правило, имеют более высокую зарплату
- Дерево решений значительно упрощает действительную взаимосвязь между хитами, количеством лет и зарплатой, но при этом дает простую интерпретацию полученного результат.

<u>Обучение и предсказание</u>

1. Обучение: разбиваем пространство признаков (предикторов) – множество возможных значений для $x_1, x_2, \dots, x_p$ – на J уникальных непересекающихся регионов $R_1, R_2, \dots, R_J$
2. Предсказание: для каждого наблюдения, которое попадает в регион R_j, мы предсказывает одно и то же значение, которое соответствует среднему целевому значению для обучающих наблюдений из R_j

Обучение

Общая задача

В теории форма регионов может быть любой, однако используются гиперпрямоугольники (многомерные прямоугольники). Цель найти такие области $R_1, R_2, \ldots, R_J$, которые минимизируют RSS, т.е.

$$\sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j} \right)^2 \to \min,$$

где $\hat{y}_{R_j}$ – среднее целевое значение (отклик) наблюдений из обучающего набора в j-ой области.

Однако невозможно перебрать все варианты деления пространства признаков на J областей.

Для решения данной проблемы используется жадный подход известный как рекурсивное бинарное деление (recursive binary splitting)

Он начинается с вершины дерева, когда все наблюдения принадлежат одному региону, и затем последовательно делит пространство признаков. Каждое деление приводит к появлению двух новых ветвей.

Подход называется жадным из-за того, что на каждом шаге в процессе построения дерева, лучшее деление выбирается исключительно на основе текущих данных (даже если это деление приведет к более плохому результату в дальнейшем), то есть не оцениваются последующие шаги.

Построение дерева

Для выполнения бинарного деления выбирается предиктор j и точка деления s такие, что при делении пространства признаков на регионы $\{x | x_j < s\}$ и $\{x | x_j \geq s\}$ ведет к наибольшему уменьшению RSS.

Таким образом, рассматриваются все признаки от 1 до p и все возможные значения точки деления s для каждого признака, после чего выбирается признак и деление по нему такие, которые соответствуют наименьшему RSS.

Замечание

Как правило, точка деления s по признаку j выбирается как среднее двух соседних значений признаков на координатной оси

Для признака j и точки деления s получаем:

$$R_1(j, s) = \{x | x_j < s\}$$

$$R_2(j, s) = \{x | x_j \geq s\}$$

Необходимо минимизировать выражение:

$$\sum_{i: x_i \in R_{\text{left}}(j,s)} \left(y_i - \hat{y}_{R_{\text{left}}} \right)^2 + \sum_{i: x_i \in R_{\text{right}}(j,s)} \left(y_i - \hat{y}_{R_{\text{right}}} \right)^2,$$

где $\hat{y}_{R_{\text{left}}}$ и $\hat{y}_{R_{\text{right}}}$ – средние целевые значения наблюдений из обучающего множества в регионах $R_{\text{left}}(j,s)$ и $R_{\text{right}}(j,s)$, соответственно.

В общем виде минимизацию можно записать следующим образом

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\left[RSS_{\text{left}} + RSS_{\text{right}}\right]$$

или

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\left[\sum_{i:x_i \in R_{\text{left}}(j,s)} \left(y_i - \hat{y}_{R_{\text{left}}}\right)^2 + \sum_{i:x_i \in R_{\text{right}}(j,s)} \left(y_i - \hat{y}_{R_{\text{right}}}\right)^2\right]$$

где $\theta = (j,s)$.

Процесс повторяется при дальнейшем делении пространства признаков, однако в этом случае делится один из ранее полученных регионов. В качестве критерия остановки используются, например, достижение заданного значения глубины дерева или минимальное количество наблюдений в терминальном узле.

Поиск минимизирующих признака и точки деления осуществляется достаточно быстро, особенно когда количество признаков p не слишком большое.

Предсказание

После разделения пространства признаков на регионы $R_1, R_2, \ldots, R_J$, в качестве предсказания отклика на тестовое наблюдение используется среднее значение обучающих наблюдений в регионе, которому принадлежит тестовое наблюдение.

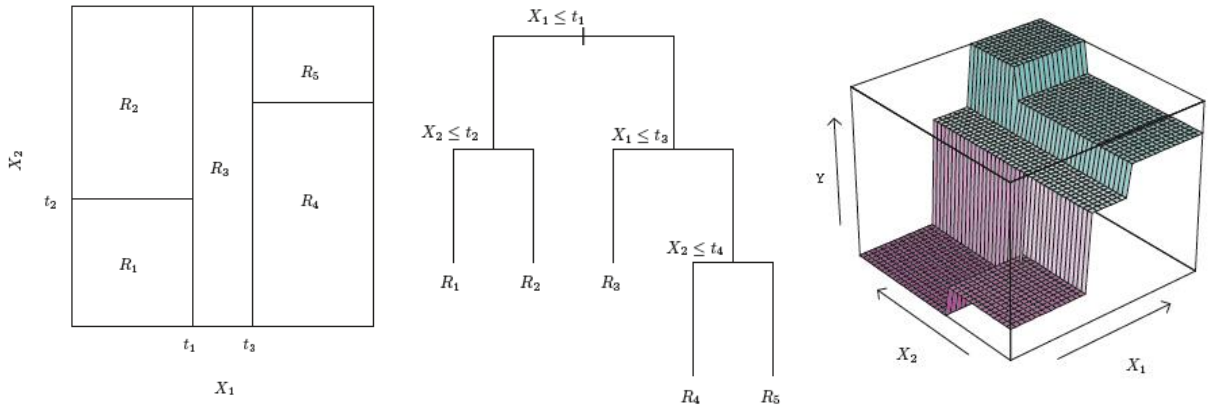

Рисунок – Пример функции предсказания дерева решений [1]

2.2. Классификация

Обучение

Дерево решений для задачи классификации строится аналогично как было рассмотрено ранее. Отличие заключается в том, что в данном случае оптимизации по RSS не подходит, так как у нас категориальные (дискретные) целевые значения.

В качестве альтернативы RSS для задачи классификации используются следующие:

- Ошибка классификации
- Gini-индекс
- Энтропия

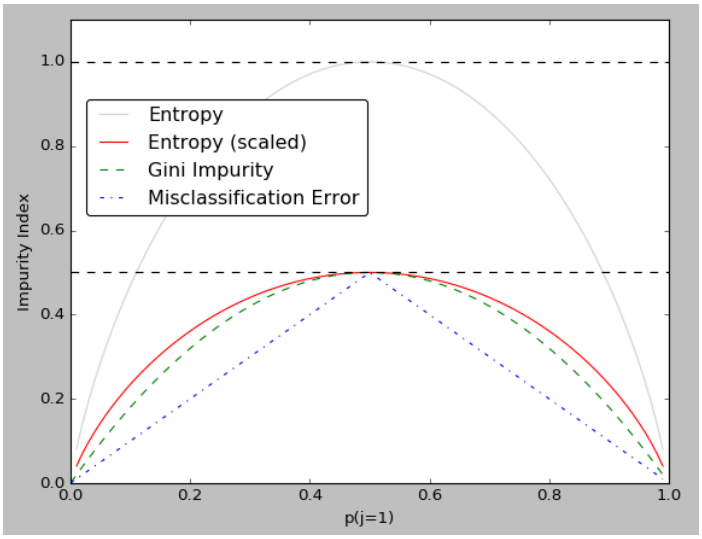

Рисунок – Ошибка классификации, Gini-индекс, энтропия [link]

Ошибка классификации

Ошибка классификации – доля обучающих наблюдений в регионе, которые не принадлежат классу с наибольшим количество элементов в этом регионе:

$$E_m^{CE} = 1 - \max_k \hat{p}_{mk},$$

где

$$\hat{p}_{mk} = \frac{1}{n_m} \sum_{i:x_i \in R_m(j,s)} I(y_i = k),$$

где $\hat{p}_{mk}$ – доля обучающих наблюдений k-го класса в регионе m.

Однако ошибка классификации склонна к преждевременному завершению деления дерева. Поэтому на практике отдают предпочтение таким критериям, как Gini-индекс и энтропия.

Gini-индекс

Gini-индекс вычисляется как сумма дисперсий по каждому классу (распределение Бернулли)

$$E_m^{Gini} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

Gini-индекс принимает небольшое значение, если все значения $\hat{p}_{mk}$ близки к 0 или 1.

Данная мера определяет чистоту (purity) региона. Небольшое значение свидетельствует, что в регионе превалируют наблюдения одного класса.

Энтропия

Энтропия используется в качестве альтернативы Gini-индексу, вычисляется как

$$E_m^{Entropy} = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

Так же, как и при использовании Gini-индекса, энтропия дает небольшое значение, если $\hat{p}_{mk}$ близки к 0 или 1, определяя таким образом чистоту региона.

При построении дерева классификации, как правило, используется Gini-индекс или энтропию для оценки качества деления, так как они более чувствительны к чистоте, чем ошибка классификации. При этом Gini-индекс предпочтительнее с вычислительной точки зрения.

Принятие решения

Для каждого признака и точки деления вычисляется улучшение неопределенности (или прирост информации в терминах энтропии):

$$\text{impurity\_improvement} = \frac{n_c}{n}\left[E_c - \left(\frac{n_{\text{left}}}{n_c}E_{\text{left}} + \frac{n_{\text{right}}}{n_c}E_{\text{right}}\right)\right]$$

Задача заключается в том, чтобы найти такой признак j и точку деление s, которые дадут наибольшую определенность (улучшение неопределенности) после разделения:

$$\hat{\theta} = \underset{\theta}{\text{argmax}}\frac{n_c}{n}\left[E_c - \underbrace{\left(\frac{n_{\text{left}}}{n_c}E_{\text{left}} + \frac{n_{\text{right}}}{n_c}E_{\text{right}}\right)}_{\text{минимизация}}\right]$$

где $\theta = (j, s)$.

Когда приходим к состоянию, при котором $\text{impurity\_improvement}$ равно нулю, полагаем, что алгоритм далее не может улучшить текущее положение и останавливаемся.

Предсказание

Для предсказания используется принцип большинства вместо среднего значения, т. е. предсказывается тот класс, который имеет большинство наблюдений обучающего множества в регионе для заданного тестового наблюдения.

Проблемы

Дерево решений может давать хорошие предсказания на обучающем множестве, но алгоритм склонен к переобучению, что ведет к плохим результатам на тестовом множестве.

Решение:

- Подрезка (см. далее)
- Комбинация деревьев (см. раздел 3)

2.3. Подрезка дерева решений

Подрезка необходима для того, чтобы избежать переобучения при построении модели предсказания на базе дерева решений. Можно выделить два подхода:

- Подрезка с предварительными ограничениями (pre-pruning). В данном случае регулируются, например, такие гиперпараметры, как глубина дерева, минимальное количество элементов в листе, и дерево строится не более чем до того момента, пока не будут достигнуты заранее установленные значения. Гиперпараметры могут быть выбраны с использованием отложенной выборки или кросс-валидации;
- Подрезка после построения дерева (post-pruning). В этом случае строится полное дерево, а затем выбирается поддерево.

Рассмотрим более подробно последний вариант подрезки. Как было сказано ранее, результирующее дерево может быть очень сложным, что часто приводит к переобучению. Дерево же с небольшим количеством регионов может уменьшить разброс за счет небольшого увеличения смещения. Наша задача получить поддерево, которое будет давать наименьшую ошибку на новых данных. Эту ошибку можно оценить посредством отложенной выборки или кросс-валидации. Однако оценить все возможные варианты поддеревьев может быть очень сложной задачей.

Чтобы уменьшить количество варианты используется техника на основе функции оценки сложности – CCP (Cost Complexity Pruning). Вместо всех поддеревьев мы рассматриваем только те, которые соответствуют некоторому неотрицательному параметру α.

Для каждого α существует поддерево $T \subset T_0$, такое что выражение

$$\sum_{m=1}^{|T|} \sum_{i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2 + \alpha |T|$$

принимает наименьшее значение, где $|T|$ – количество листов (терминальных узлов) дерева T; R_m – регион, соответствующий терминальному узлу m. Параметр α контролирует сложность поддерева. Когда $\alpha = 0$, получаем исходное дерево T_0. С увеличением α количество терминальных узлов сокращается.

Общий алгоритм построения дерева решений с подрезкой на основе функции CCP:

1. Для построения полного бинарного дерева используем рекурсивный алгоритм на обучающем множестве, останавливаемся только когда терминальные узлы имеют количество наблюдений меньшее чем установленное число
2. Применяем CCP к полному дереву, чтобы получить последовательность лучших поддеревьев как функция от α

3. Используем кросс-валидацию (или отложенную выборку) для выбора α. Для каждого сплита
   - Повторяем шаг 1 и 2 на обучающей части сплита
   - Рассчитываем качество предсказания на проверочной части сплита как функцию от α

   Усредняем результаты для каждого α и выбираем α с минимальной средней ошибкой
4. Возвращаем поддерево, которое соответствует выбранному значению α

Более подробно с приведенной выше техникой для задачи классификации можно ознакомиться здесь [2, 3].

2.4. Деревья решений и линейные модели

Линейная регрессия:

$$f(x) = \theta_0 + \sum_{j=1}^{p} x_j \, \theta_j$$

Дерево решений:

$$f(x) = \sum_{m=1}^{M} c_m \, 1(x \in R_m)$$

где c_m – значение предсказания для региона m; $R_1, \dots, R_m$ – регионы пространства признаков.

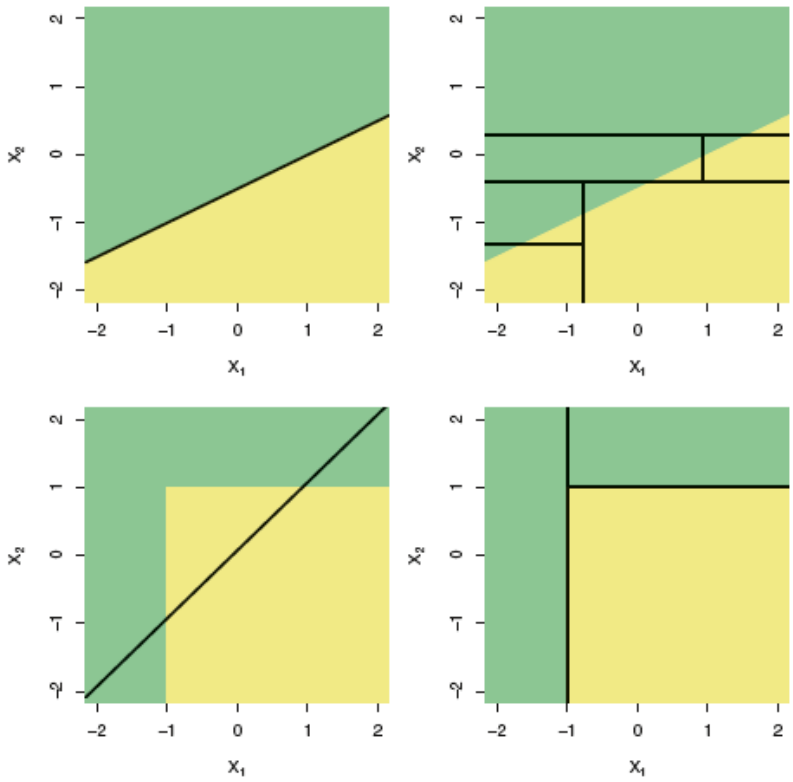

Рисунок – Линейная модель и дерево решений [1]

Преимущества

- Интерпретируемость
- Деревья могут обрабатывать категориальные данных без необходимости перевода их в числовые переменные
- Подходят для нелинейных взаимосвязей между признаками и целевыми значениями. В этом случае деревья решений могут превосходить ранее рассмотренные подходы.

Недостатки

- Небольшие изменения в обучающих данных могут стать причиной больших изменений в структуре дерева и как следствие в предсказаниях. Это же определяет склонность деревьев решений к переобучению.

Агрегирования множества деревьев решений такими способами, как бэггинг, случайный лес, сверхслучайные деревья и бустинг, позволяет значительно улучшить качество предсказания.

3. Комбинации методов

Цель использования комбинирующих методов заключается в том, чтобы улучшить обобщающую способность предсказаний за счет комбинации множества базовых однотипных моделей.

Можно выделить два подхода:

- Усредняющие методы: основной принцип – построить независимо нескольких однотипных моделей и затем усреднить их предсказание. В среднем агрегированное предсказание лучше, чем предсказание одной базовой модели, так как при комбинации множества базовых моделей уменьшается дисперсия (variance). Примеры: бэггинг, случайный лес, сверхслучайные деревья
- Бустинг методы: базовые однотипные модели обучаются последовательно, стараясь уменьшить смещение (bias) агрегированной модели. Таким образом, используются несколько слабых моделей для получения одной более качественной. Примеры: AdaBoost, Gradient Tree Boosting

3.1. Бэггинг

Для регрессии

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{b}(x),$$

где B – количество базовых моделей (например, деревьев решений), обученных на B различных обучающих множествах.

Обучающие множества формируются посредством техники бутстреп из одного исходного обучающего множества.

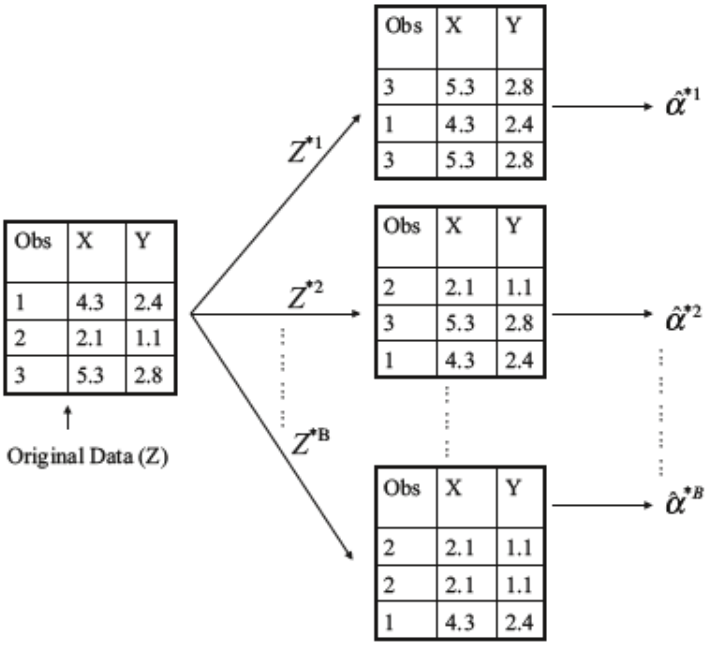

Рисунок – Пример формирования наборов данных посредством бутстрепа [1]

Каждое индивидуальное дерево имеет высокую дисперсию (variance), но низкое смещение (bias). Усредняя B деревьев, уменьшается дисперсия. B соответствует 100–1000.

Для задачи классификации используется принцип большинства.

Ошибка OOB (Out-of-Bag)

Ошибка OOB позволяет оценить тестовую ошибку модели без необходимости использования отложенной выборки или кросс-валидации. Так как модели обучаются на данных, полученных посредством бутстрепа (выборка из исходного множества с возвратом), то часть наблюдений исходного множества не будет участвовать при обучении. В среднем таких наблюдений будет около $1/3$ для каждого набора данных, полученного бутстрепом.

Пояснение

Вероятность взять наблюдение i для обучения из исходного набора данных:

$$p = \frac{1}{n}$$

Вероятность не взять наблюдение i:

$$\bar{p} = 1 - p = 1 - \frac{1}{n}$$

Вероятность не взять n раз наблюдение i при выборке с возвратом есть

$$\bar{p}_n = \prod_{i=1}^{n} \bar{p} = \bar{p}^n$$

При $n \to \infty$

$$\lim_{n\to\infty} \bar{p}_n = \lim_{n\to\infty} \left(1 - \frac{1}{n}\right)^n = e^{-1} \approx 0.37$$

Почему так? Вспоминаем экспоненциальную функцию в виде предела

$$e^x = \lim_{n\to\infty}\left(1+\frac{x}{n}\right)^n$$

Пример

Пусть $n = 100$, тогда вероятность

$$\bar{p_n} \approx 0.37$$

В результате эти наблюдения можно использовать как проверочные для оценки тестовой ошибки.

Это работает следующим образом. Для каждого наблюдения будет около $B/3$ моделей, в которых оно не участвовало в обучении. Соответственно, получаем $B/3$ предсказаний от этих моделей для этого наблюдения. Далее для получения итогового предсказания в случае регрессии усредняем полученные предсказания или используем принцип большинства при классификации. Таким образом, мы получим предсказания для каждого наблюдения.

Данный подход с применением OOB для оценки тестовой ошибки похож на кросс-валидацию и может быть использован, когда имеется большой набор данных, для которых выполнение кросс-валидации проблематично с вычислительной точки зрения.

Важность переменных

Бэггинг, как правило, ведет к улучшению точности по сравнению с предсказанием посредством одного дерева решений. В то же время при бэггинге сложно интерпретировать результат, так как для предсказания используется множество деревьев и неясно какие признаки являются наиболее важными. Таким образом, бэггинг улучшает точность предсказания за счет интерпретируемости дерева.

Однако мы можем получить общую важность для каждого предиктора с использованием RSS для регрессии или, например, Gini для классификации.

В случае с регрессией мы можем вычислить суммарное значение, на которое уменьшается RSS в связи с выбором сплитов по рассматриваемому признаку и затем найти среднее по всем B деревьям для этого признака. Большое значение указывает на важность предиктора.

Подобным образом вычисляется важность признаков для классификации. Мы подсчитываем суммарное значение, на которое уменьшается Gini-индекс за счет сплитов по заданному признаку и затем вычисляем среднее по всем B деревьям для этого признака.

3.2. Случайный лес

Случайный лес позволяет получить лучший результат по сравнению с бэггингом посредством небольших изменений, которые декоррелируют деревья. Так же, как и раньше, мы строим множество деревьев на обучающих выборках, полученных посредством бутстрепа из исходного обучающего набора. Однако каждый раз, когда необходимо выполнить сплит, из всех p предикторов выбираются случайным образом m кандидатов. В итоге сплит выполняется по одному из m признаков. Для каждого нового сплита выбирается новый набор из m признаков. Как правило,

$$m \approx \sqrt{p}.$$

Другими словами, при построении случайного леса для каждого сплита в дереве алгоритм рассматривает только m признаков и не позволяет использовать большинство имеющихся признаков.

Данную особенность можно объяснить следующим образом: если в наборе данных существует очень сильный предиктор при других менее сильных предикторах, то в коллекции деревьев большинство или все из них будут использовать сильный предиктор на верхнем сплите. Следовательно все деревья будут похоже между собой.

Поэтому предсказания деревьев будут обладать высокой корреляцией. При усреднении множества величин с высокой корреляцией не приводит к такому же сокращению дисперсии, как при усреднении множества некоррелируемых величин.

В частности, в данном случае это означает, что бэггинг не ведет к существенному сокращению дисперсии по сравнению с использованием одного дерева. Случайный лес преодолевает данную проблему за счет того, что для каждого сплита рассматривается только подмножества признаков.

В среднем $(p-m)/p$ сплитов даже не будут рассматривать сильный предиктор, поэтому другие предикторы получают больше шансов быть выбранными.

Данный процесс рассматривается как декорреляция деревьев, в основе которого лежит получение менее изменчивых средних значений деревьев (с меньшей дисперсией) и, следовательно, более надежных результатов.

Использование небольшого значения m при построении случайного леса, как правило, бывает полезным, когда у нас большое количество коррелируемых предикторов.

Так же как и при бэггинге, в случае со случайным лесом увеличение числа деревьев B не ведет к переобучению. Поэтому на практике, как правило, используется достаточно большое значение B для стабилизации ошибки предсказаний.

Пример

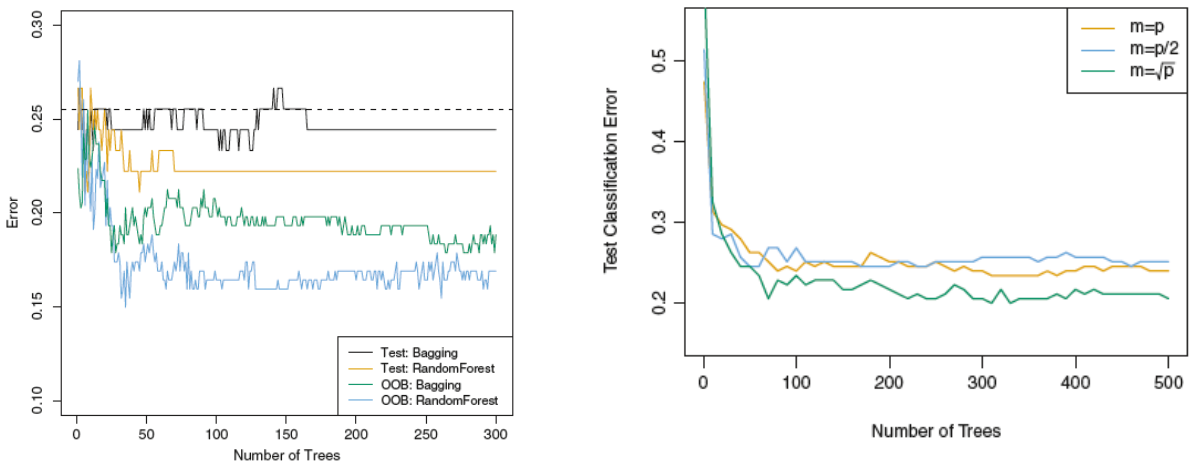

Рисунок – Сравнение ошибки предсказания беггинга и случайного леса в зависимости от количества деревьев [1]

3.3. Бустинг

Общий принцип

Бустинг – техника последовательного обучения множества слабых базовых моделей для построения более качественной общей модели предсказания. В этом случае используется одно обучающее множество. Первая модель в последовательности обучается предсказывать y, а последующие стремятся уменьшить ошибки предсказания предыдущих. В итоге получаем аддитивную модель как взвешенную последовательность множества базовых.

Бустинг является общим подходом, который может быть применен к различным методам регрессии и классификации.

В качестве примеров можно выделить AdaBoost, Gradient Boosting, XGBoost. Рассмотрим в общем виде Gradient Boosting для регрессии и AdaBoost для задачи классификации.

Регрессия. Gradient Boosting

В данном случае в качестве базовой модели возьмем дерево решений. Так же как и при бэггинге, бустинг использует комбинацию большого количества деревьев, $\hat{f}^1, \hat{f}^2, \dots, \hat{f}^B$. Деревья строятся последовательно, при этом каждое последующее дерево использует информацию от предыдущих деревьев. Общий вид алгоритма приведен ниже.

Алгоритм

1. Устанавливаем $\hat{f}(x) = 0$ и $r_i = y_i$ для всех i в обучающем множестве
2. Для $b = 1,2, \dots, B$ повторяем:
   - а. Обучаем дерево $\hat{f}^b$ с d сплитами ($d + 1$ терминальных узлов) на обучающем множестве (X, r)
   - б. Обновляем $\hat{f}$
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$
   - в. Обновляем остатки
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$$
3. Возвращаем модель
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

Бустинг имеет три настраиваемых гиперпараметра:

- Количество деревьев B. В отличии от предыдущих подходов бустинг может привести к переобучению при слишком большом значении B. Можно использовать кросс-валидацию для выбора B.
- Параметр регуляризации λ – небольшое положительное число (например, 0.01 или 0.001), которое контролирует скорость обучения. При очень малом значении λ необходимо использовать очень большое значение B, чтобы получить приемлемый результат.
- Количество сплитов d в каждом дереве. Параметр определяет сложность ансамбля. Часто при $d = 1$ бустинг дает хороший результат. В этом случае деревья будут состоять из одного

сплита. В целом, как правило, нескольких сплитов для построения деревьев вполне достаточно. При этом увеличивает интерпретируемость. Так, например, при одном сплите мы получим аддитивную модель, в которой каждое слагаемое будет соответствовать одному признаку.

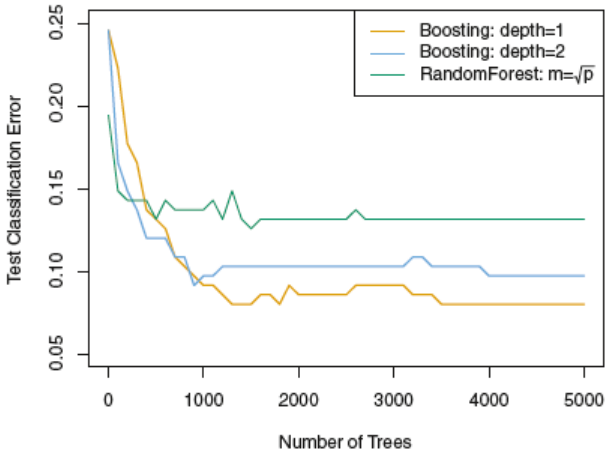

Рисунок – Сравнение ошибки предсказания случайного леса и бустинга в зависимости от количества деревьев [1]

Классификация. AdaBoost

Общая идея заключается в следующем – последовательно строим модель на базе слабых классификаторов, при этом уменьшаем ошибку классификатора на предыдущем шаге. Если на обучающем множестве были случаи неправильной классификации, то для этих наблюдений увеличивается вес. Следующий классификатор при обучении использует эти веса, чтобы сделать акцент на случаях, где была сделана ошибка. После этого обученный классификатор может сам давать ошибочные классификации, что ведет к изменению весов наблюдений для последующего классификатора. Каждый классификатор в последовательности имеет свой вес. Таким образом, получаем конечный классификатор как линейную комбинацию последовательно обученных классификаторов.

В качестве базового классификатора выступает слабая модель. Как правило, это решающее дерево с глубиной 1. Однако в общем виде базовым классификатором может быть любым.

Важным моментом является то, как учитывать вес наблюдений при обучении слабых классификаторов. Существует два подхода: если сам классификатор способен учитывать вес, то используем такую возможность; если нет, то можно выполнить повторную выборку исходного размера с возвратом с учетом весов наблюдений. В последнем случае элементы с большим весом будут иметь больше шансов быть выбранными и, следовательно, могут появиться несколько раз, а элементы с малым весов могут вообще не участвовать в обучении.

В случае с деревом решений вес наблюдений можно учитывать при расчете вероятностей следующим образом:

$$p_k = \frac{\sum_{i=1}^{n} w_i \cdot 1(y_i = k)}{\sum_{j=1}^{n} w_j}.$$

Ниже приведен алгоритм AdaBoost для многоклассовой классификации (SAMME) [4]

1 Инициализация весов наблюдений

$$w_i = \frac{1}{n},$$

где $i = 1,2,\dots,n$

2 Цикл $m = 1$ до M

а Обучение классификатора $T_m(x)$ на обучающих данным с весами w_i

б Вычисление ошибки

$$err_m = \frac{\sum_{i=1}^{n} w_i \cdot 1\left(y_i \neq T_m(x_i)\right)}{\sum_{j=1}^{n} w_j}$$

в Вычисление веса классификатора $G_m(x)$

$$\alpha_m = \log\left[\frac{1 - err_m}{err_m}\right] + \log(K - 1),$$

где K – количество классов

г Обновление весов

$$w_i \leftarrow w_i \cdot \exp\left(\alpha_m \cdot 1\left(y_i \neq T_m(x_i)\right)\right)$$

д Нормализация весов w_i

3

$$C(x) = \underset{k}{\operatorname{argmax}}\left[\sum_{m=1}^{M} \alpha_m \cdot 1(T_m(x) = k)\right]$$

3.4. Стекинг

Ранее мы рассмотрели подходы с обучением однотипных моделей на множестве наборов данных, полученных посредством бутстрепа, такие как бэггинг, случайный лес и пр. При этом для итогового предсказания в задаче регрессии использовалось усреднение предсказаний, полученных от каждой обученной модели, а для классификации – принцип большинства.

В стекинге для итогового предсказания используется метамодель. Как правило, если решается задача регрессии, то в качестве метамодели выбирается линейная регрессия, а при классификации – логистическая регрессия.

В общем виде стекинг состоит из двух видов моделей:

- Базовые модели (две и более): линейная регрессия, метод опорных векторов, деревья решений и пр.
- Метамодель (одна): как правило, линейная регрессия или логистическая регрессия

При этом входными значениями для метамодели являются предсказания от базовых моделей, выполняя роль входных признаков.

Процесс обучения можно представить в виде двух стадий:

- Обучение базовых моделей на обучающем множестве
- Обучение метамодели на предсказаниях базовых моделей

Ниже на рисунках показан процесс обучения с использованием отложенной выборки и предсказания на новых данных

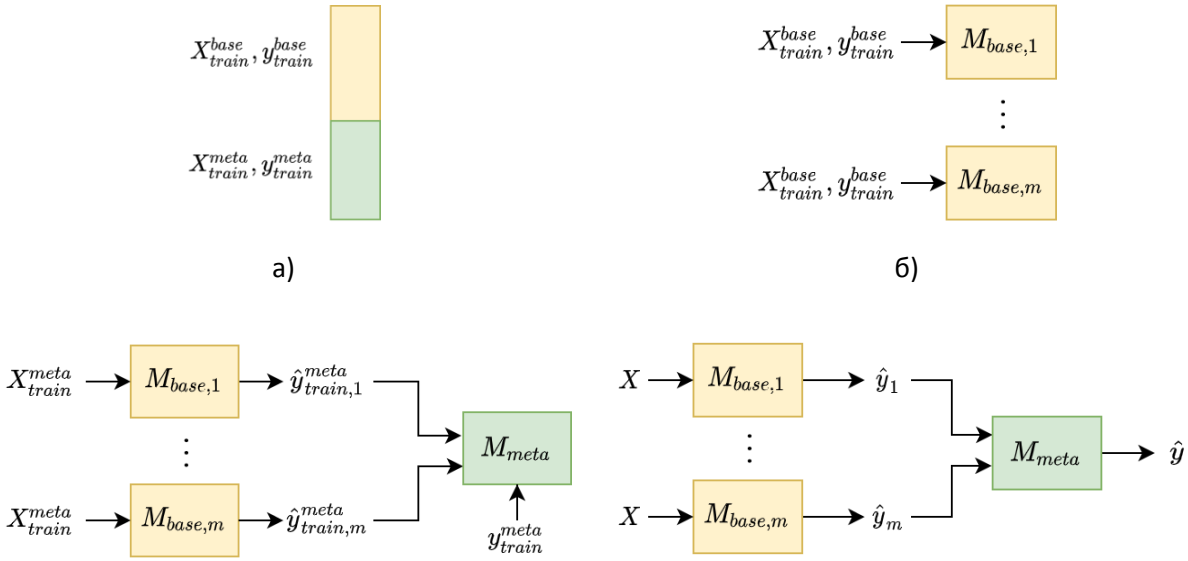

Рисунок – Обучение и предсказание в стекинге: а) обучающее множество для базовых моделей и метамодели; б) обучение базовых моделей; в) обучение метамодели; г) предсказание на новых данных

Следует подчеркнуть, что обучения метамодели происходит на данных, которые не использовались для обучения базовых моделей.

Список литературы

1. An Introduction to Statistical Learning by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshir. URL: https://www.statlearning.com/
2. Minimal Cost-Complexity Pruning / Applied Data Mining and Statistical Learning. URL: https://online.stat.psu.edu/stat508/lesson/11/11.8/11.8.2
3. Cost-Complexity Pruning / ML Wiki. URL: http://mlwiki.org/index.php/Cost-Complexity_Pruning
4. Zhu, H. Zou, S. Rosset, T. Hastie, “Multi-class AdaBoost”, 2009. URL: https://hastie.su.domains/Papers/samme.pdf