Алгоритмы, основанные на решающих деревьях

Штыков Павел[1]

[1]Московский государственный университет им. М. В. Ломоносова
shtykov.pa@gmail.com

2 января 2022

1 Введение

Модели, основанные на решающих деревьях, получили широкое распространение в машинном обучении и являются базовым алгоритмом для многих задач. В этой работе мы рассмотрим базовые алгоритмы построения решающих деревьев и методы их ансамблирования. Также мы выделили современные области применения моделей и проанализировали их популярность. В работе рассмотрены следующие модели: решающее дерево, случайный лес и градиентный бустинг.

2 Происхождение

Впервые алгоритм *решающего дерева* был представлен британским исследователем Уильямом Белсоном в 1959 году [6] и использовался в биологии для задач классификации.

В самом простом исполнении модель представляет собой *бинарное дерево*, в каждой нелистовой вершине которого находится некоторое *правило* (*предикат*), по которому происходит разбиение выборки на *две части*. Одна часть уходит в *левое поддерево*, другая – в *правое*. Всем элементам выборки, которые дошли до листовой вершины, присваивается *метка класса*, соответствующая *этому листу*. Простое *решающее дерево* представлено на рисунке 1.

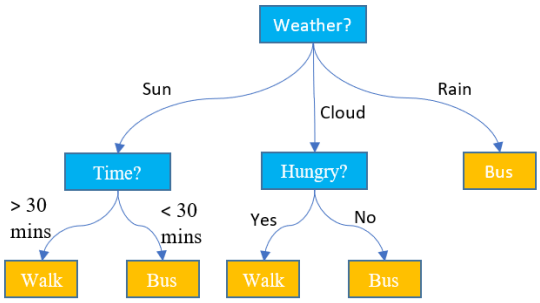

Рис. 1: Простое решающие дерево

Данный алгоритм *интуитивен* и обладает *высокой интерпретируемостью*. Во многих областях жизни люди пользуются схожими правилами для принятия решений (например, врач при диагностике задает несколько *простых вопросов* пациенту и, основываясь на своем *опыте*, ставит предварительный диагноз).

3 Базовый алгоритм построения решающего дерева

Приведем алгоритм построения решающего дерева, взяв за основу стандартный алгоритм ID3[1](Iterative Dichotomiser)[14].

Общие предположения и обозначения:

- Во всех задачах, если не оговорено другого, решается задача классификации, где Y — множество классов
- F — матрица объектов-признаков, $f_j(x): X \to \mathbb{R}$ — j-тый признак объекта x
- $X^\ell = \left(x_i, y_i\right)_{i=1}^{\ell}$ — обучающая выборка, где $y_i \in Y$ — метки классов
- $a(x): X \to Y$ — алгоритм классификации из некоторого семейства $\mathscr{A}$

Алгоритм ID3 строит дерево жадно сверху вниз. Для каждой вершины v алгоритм подбирает дискретный признак $f_v(x)$, разделяющий подвыборку U, дошедшую до вершины v, на k частей. Для каждой части создается дочерняя вершина, куда перенаправляется соответствующая подвыборка. Далее для каждой вновь созданной вершины алгоритм продолжается рекурсивно.

Хотя формально k может быть любым, на практике чаще всего используют $k = 2$ (то есть *бинарное дерево*).

Пример простого порогового правила классификации:

$$f_v(x) = I[f_j(x) \geq a],$$

где $I[x]$ — индикаторная функция и a — некоторая константа.

Выбор признака f_v в каждой вершине происходит за счет максимизации некоторой функции $\mathrm{Gain}(f, U)$:

$$f_v := \arg\max_{f \in F} \mathrm{Gain}(f, U)$$

Функция $\mathrm{Gain}(f, U)$ измеряет выигрыш от ветвления вершины v по некоторому признаку f, т. е. показывает, насколько хорошо подвыборка U разделяется с помощью правила f. Максимизация этой величины позволяет нам найти такой признак f_v, после разбиения которым подвыборки в дочерних вершинах будут в целом более «однородными», чем в U. Как меру «однородности» часто используют информационную энтропию или коэффициент неопределенности Джини (Gini impurity).

Ветвление заканчивается, когда функция $\mathrm{Gain}(f, U)$ становится меньше некоторого порога G_0. Вершина, для которой выполняется $\mathrm{Gain}(f, U) < G_0$, объявляется листовой и наделяется меткой класса, преобладающего среди подвыборки U, дошедшей до этой вершины.

4 Ансамбли над решающими деревьями

Ансамблирование — важный механизм машинного обучения. Хотя ансамбли и можно строить почти над любыми семействами алгоритмов, особенно важную роль ансамбли сыграли именно в отношении решающих деревьев. В данной части мы рассмотрим несколько наиболее известных (и широко применяемых на практике) алгоритмов, таких как Random Forest, Gradient Boosting и др.

4.1 Общая постановка задачи

Пусть имеется набор из T базовых алгоритмов $a_t : X \to Y,\ t = 1, \ldots, T$. Потребуем, чтобы каждый алгоритм был представим в виде:

$$a_i(x) = C(b_i(x)),$$

где C — некоторое решающее правило, $b_i(x)$ — модель дающая некоторую численную оценку результата. Такая формальность необходима для использования более простого пространства оценок вида

$$a_i : X \xrightarrow{b_i} \mathbb{R} \xrightarrow{C} Y.$$

[1]Позже алгоритм был улучшен до версии C4.5[15]. В частности, были реализованы обработка пропущенных значений и механизм прунинга («подстрижка» деревьев, аналог регуляризации). Также важной реализацией является алгоритм CART[12]. В своей сути он крайне похож на C4.5, но в нем дополнительно была решена задача регрессии.

Главная идея ансамблирования — с помощью усреднения голосов базовых алгоритмов сделать предсказание более точным и надежным. То есть ансамбль есть некоторая функция от набора базовых алгоритмов:

$$a(x) = C\big[F\big(b_1(x), \ldots, b_T(x)\big)\big],$$

где F — мета-алгоритм (или алгоритм ансамблирования). Именно в мета-алгоритме и заключается вся сложность и уникальность конкретного ансамбля.

4.2 Random Forest

Алгоритм случайного леса (random forest или RF)[7], представляет собой простое усреднение по предсказаниям нескольких деревьев, но с небольшими изменениями в процессе их обучения:

- Для каждого дерева генерируется своя обучающая выборка с помощью бутстрапа (то есть выборка генерируется из базовой поэлементным извлечением с повторением);
- В каждой вершине дерева признак разбиения $f_v(x)$ выбирается не из всего множества признаков, а из некоторого подмножества размером m (где m — гиперпараметр);
- Деревья строятся либо до полного исчерпания выборки, либо до некоторой максимальной глубины (обычно около 30). Построенные деревья *не* подвергаются процедуре прунига (в отличие от алгоритмов C4.5[15] и CART[12])

Заметим, что из-за независимости процесса обучения деревьев Random Forest хорошо параллелится.

На рис. 2 показан процесс сглаживания разделяющей гиперплоскости с ростом количества деревьев в ансамбле (источник [5]).

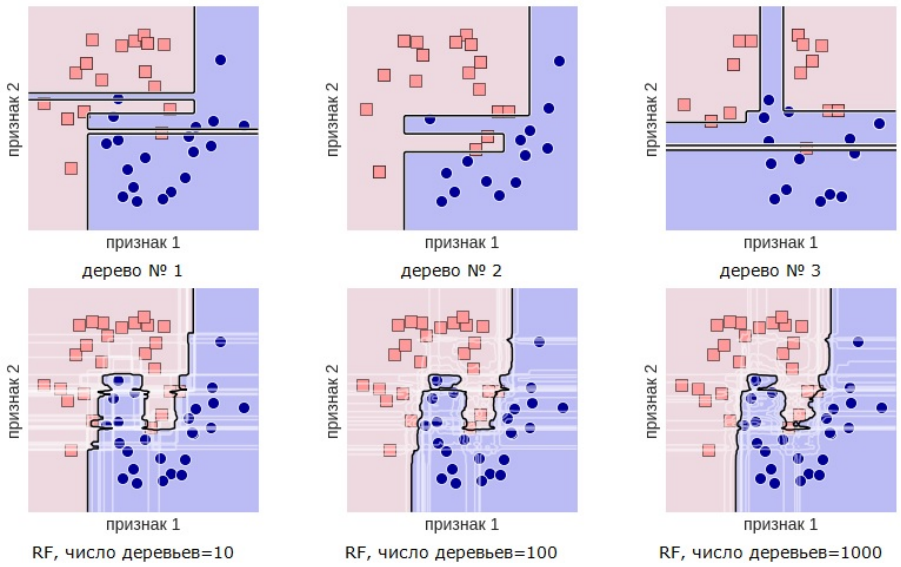

Рис. 2: Изменение формы разделяющей гиперплоскости в зависимости от количества деревьев в ансамбле

4.3 Gradient Boosting

В данной части будет разобран более сложный метод ансамблирования — градиентный бустинг [2](gradient boosting или GB)[11]. В данном методе ансамбль строиться *последовательно*: каждый последующий базовый алгоритм пытается *приблизить антиградиент* функции ошибки предыдущих базовых алгоритмов.

Опишем алгоритм построения. Пусть уже построен ансамбль из $T-1$ базовых алгоритмов:

$$f_{T-1} = \sum_{t=1}^{T-1} \beta_t a_t(x), \quad a_t : X \to Y, \quad \beta_t \in \mathbb{R}_+.$$

[2]Исторически первым изобретенным бустингом был AdaBoost[10]. Однако градиентный бустинг является законченным обобщением метода для любых функций потерь и сейчас пользуется большей популярностью

Тогда на шаге T (зафиксировав f_{T-1}) нам нужно найти такие коэффициент обучения β_T и модель $a_T(x)$, которые минимизируют следующий функционал ошибки:

$$Q(f, y) = \sum_{i=1}^{\ell} \mathscr{L}(\underbrace{f_{T-1,i} + \beta_T a_T(x_i)}_{f_{T,i}}, y_i) \to \min_{\beta_T, a_T}.$$

Для поиска a_T обучим его на антиградиент функции ошибки (используя MSE как функцию ошибки при обучении):

$$a_T := \arg\min_{a \in \mathscr{A}} \sum_{i=1}^{\ell} \left(-\frac{\partial \mathscr{L}(f(x_i), y_i)}{\partial f(x_i)}\bigg|_{f(x)=f_{T-1}(x)} - a(x_i) \right)^2.$$

По сути мы совершаем одну итерацию градиентного спуска в пространстве функционалов ошибки $Q(f, y)$. Но так как обучить базовую модель точно восстанавливать антиградиент не получится, мы используем приближение по MSE. β_T найдем, решив задачу одномерной оптимизации:

$$\beta_T := \arg\min_{\beta_T > 0} \sum_{i=1}^{\ell} \mathscr{L}(f_{T-1,i} + \beta_T a_T(x_i), y_i).$$

После чего обновим $f_T := f_{T-1} + \beta_T a_T(x)$ и продолжим алгоритм итеративно (заметим, что β_T можно домножить на некоторый дополнительный коэффициент $\alpha \in (0, 1)$, вращающий степень доверия выбранному направлению градиентного шага).

К этому моменту мы никак не конкретизировали класс базовых алгоритмов $\mathscr{A}$. Так стандартным классом считается именно семейство решающих деревьев (обычно используется некоторая модификация CART[12], притом лучше подходят неглубокие деревья). Поэтому, когда речь идет о градиентном бустинге, подразумевается обычно бустинг над деревьями.

4.3.1 XGBoost

XGBoost[8] — популярная (особенно среди участников Kaggle[1]) и быстрая реализация градиентного бустинга над решающими деревьями.

Основное отличие XGBoost от обычного GB в том, что XGBoost использует для оптимизации в пространстве функционалов ошибки $Q(f, y)$ метод Ньютона-Рафсона, а не градиентный спуск. То есть обучение T-ой базовой модели выглядит так:

$$a_T := \arg\min_{a \in \mathscr{A}} \sum_{i=1}^{\ell} \frac{1}{2} h_{T-1}(x_i) \left(-\frac{g_{T-1}(x_i)}{h_{T-1}(x_i)} - a(x_i) \right)^2,$$

где $g_T(x_i)$ и $h_T(x_i)$ соответственно градиент и гессиан функции потерь $\mathscr{L}(f(x_i), y_i)$:

$$g_{T-1}(x_i) = \frac{\partial \mathscr{L}(f(x_i), y_i)}{\partial f(x_i)}\bigg|_{f(x)=f_{T-1}(x)}, \qquad h_{T-1}(x_i) = \frac{\partial^2 \mathscr{L}(f(x_i), y_i)}{\partial f(x_i)^2}\bigg|_{f(x)=f_{T-1}(x)}.$$

Далее алгоритм аналогичен градиентному бустингу.

4.3.2 CatBoost

CatBoost[13] — еще один популярный вариант реализации градиентного бустинга от компании Яндекс. В нем реализовано множество разнообразных механизмов и эвристик, позволяющих исправить недостатки обычного градиентного бустинга. Приведем некоторые из них:

Переобучение в градиентах При приближении градиентов базовыми алгоритмами мы считаем их в точках x_i, в которых ансамбль f_{T-1} учился аппроксимировать y_i, следовательно происходить утечка. Авторы CatBoost решили эту проблему с помощью аналога Out-Of-Bag: градиент на каждом шаге вычисляется на выборке сгенерированной некоторой перестановкой, не включающей текущий x_i.

Категориальные признаки CatBoost имеет встроенные методы обработки категориальных признаков. Для этого используется статистика по целевому признаку (target statistics, TS), но с учетом наличия различных перестановок для обучающей выборки. Это позволяет бороться с переобучением TS.

Небрежные деревья В отличие от GB или XGBoost в CatBoost используются не обычные CART деревья, а небрежные (oblivious) деревья. В таких деревьях в левом и правом потомке каждой вершины используется один и тот же признак разбиения $f_v(x)$, следовательно дерево получается сбалансированным (рис. 3). Это позволяет ускорить обучение дерева.

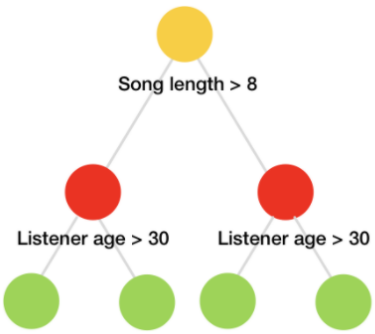

Рис. 3: Пример небрежного дерева

5 Применение

В данной части мы исследуем популярность и современные области применения алгоритмов, основанных на решающих деревьях, и сравним их с другими методами машинного обучения.

Заметим, что во многих статьях и исследованиях на тему популярности алгоритмов машинного обучения решающие деревья объединяются в одну группу с ансамблями (RF, GB и тд.). Это не позволит нам точно выявить популярность одиночных решающих деревьев.

Начнем с популярности запросов в Google. Так, основываясь на показателях Google Trends (рис. 4), можно заметить, что популярность GB и RF за последние 10 лет заметно выросла. Однако она все еще значительно меньше, чем у SVM и нейронных сетей.

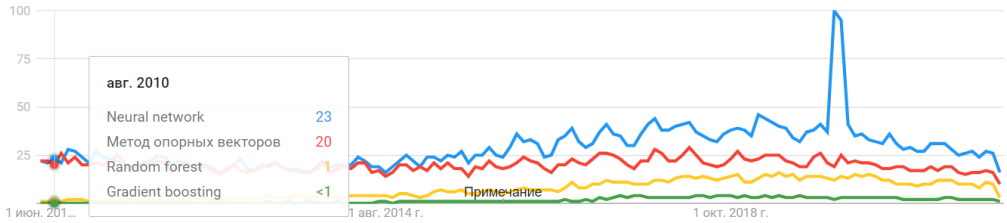

Рис. 4: **TODO**: *Перерисовать в нормальный график!* Сравнение популярности тем в поисковых запросах Google

Ранее мы отмечали популярность XGBoost среди решений победителей Kaggle[1]. Кроме этого Kaggle проводит большие ежегодные опросы среди участников платформы. На рисунке 5 представлен сводный график популярности алгоритмов с 2017 по 2020 года среди респондентов категории Data Scientists (график взят из статьи [4]).

Можно заметить, что RF (вместе с обычными решающими деревьями) и GB, занимают верхние позиции рейтинга, вплотную подбираясь к линейной регрессии и обгоняя столь популярные сейчас нейронные сети.

Изучим популярность алгоритмов, основанных на решающих деревьях, среди крупных компаний. Так, по данным аналитикового агентства HG Insights[3], открытую библиотеку с реализацией XGBoost на январь 2022 года используют 5962 компании, среди них Nvidia, Bank of America, Amazon и др.

CatBoost[13], аналог, представленный Яндексом в 2017 году, уже сейчас используется во многих сервисах компании[2]. Например, при выборе ответа на запрос пользователя в голосовом помощнике Алиса или для защиты от спама. Так как библиотека распространяется открыто, то ее используют и другие разработчики. Например, CatBoost использовался в экспериментах по обнаружению частиц в CERN[9] и показал результаты значительно лучше, чем у предыдущих моделей.

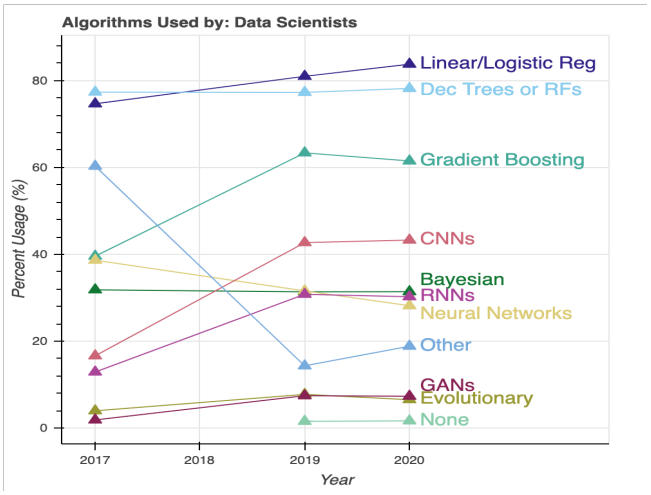

Рис. 5: Популярность алгоритмов машинного обучения среди пользователей Kaggle

6 Заключение

Исследование показало, что наиболее широкое распространение получили ансамбли решающих деревьев. Во многих областях они не уступают нейронным сетям по частоте использования. Однако остается открытым вопрос популярности и области применения одиночных решающих деревьев.

Список литературы

[1] Awesome xgboost. `https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions`. Доступ: 27.01.2022.

[2] Catboost: от 0 до 1.0.0. `https://www.youtube.com/watch?v=0bzrXjqWcTY&t=5639s`. Доступ: 05.01.2022.

[3] Companies currently using xgboost. `CompaniesCurrentlyUsingXGBoost`. Доступ: 27.01.2022.

[4] Data science trends based on 4 years of kaggle surveys. `https://towardsdatascience.com/data-science-trends-based-on-4-years-of-kaggle-surveys-60878d68551f`. Доступ: 05.01.2022.

[5] Ансамбли в машинном обучении. https://dyakonov.org/2019/04/19/ансамбли-в-машинном-обучении. Доступ: 27.01.2022.

[6] William A. Belson. Matching and Prediction on the Principle of Biological Classification. *Journal of the Royal Statistical Society Series C*, 8(2):65–75, June 1959.

[7] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[8] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[9] Denis Derkach, Mikhail Hushchyn, Tatiana Likhomanenko, Alex Rogozhnikov, Nikita Kazeev, Victoria Chekalina, Radoslav Neychev, Stanislav Kirillov, and Fedor Ratnikov and. Machine-learning-based global particle-identification algorithms at the LHCb experiment. *Journal of Physics: Conference Series*, 1085:042038, sep 2018.

[10] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[11] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232, 2001.

[12] Richard A. Olshen Charles J. Stone. Leo Breiman, Jerome H. Friedman. *Classification and regression trees.* Brooks/Cole Publishing, 1984.

[13] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features, 2019.

[14] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[15] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.