Шпаргалка по Градиентному бустингу / Концепции Cheatsheet (XeLaTeX)

Краткий справочник

Содержание

Определение Градиентного Бустинга (GBM)

Градиентный Бустинг (Gradient Boosting Machine, GBM) — это мощный ансамблевый метод машинного обучения, который строит модели **последовательно**, где каждая новая модель исправляет ошибки предыдущей. Считается одним из наиболее эффективных алгоритмов для табличных данных.

Аналогия: Лепка Скульптуры

Процесс бустинга можно сравнить с лепкой скульптуры:

- Начинаем с грубой основы (первая простая модель).
- Замечаем недочеты (ошибки модели).
- Добавляем "кусочек глины" там, где нужно (обучаем новую модель на ошибках).
- Повторяем добавление "кусочков" (новых моделей), пока результат не станет удовлетворительным.

Каждое добавление "кусочка глины" — это новая слабая модель в ансамбле бустинга.

1 Идея Бустинга (Boosting)

Отличие от Бэггинга

В отличие от **бэггинга** (например, Random Forest), где модели обучаются независимо и параллельно на разных подвыборках данных, в **бустинге** модели строятся строго **последовательно**.

Алгоритм Последовательного Исправления Ошибок

Общая схема бустинга выглядит так:

1. Обучается первая (обычно простая) модель F_0 на исходных данных.
2. Вычисляются ошибки (или остатки) $e_1 = y - F_0(x)$ этой модели.
3. Следующая модель h_1 обучается предсказывать эти ошибки e_1.
4. Предсказание ансамбля обновляется: $F_1(x) = F_0(x) + \nu \cdot h_1(x)$ (где ν - темп обучения).
5. Вычисляются новые ошибки $e_2 = y - F_1(x)$.
6. Обучается следующая модель h_2 на ошибках e_2.
7. Ансамбль обновляется: $F_2(x) = F_1(x) + \nu \cdot h_2(x)$.
8. Шаги повторяются M раз (заданное число моделей) или до остановки по критерию.

Ключевая Идея

Ансамбль постепенно "учится" на своих ошибках. Каждая последующая модель фокусируется на тех объектах или аспектах данных, где предыдущие модели ошибались больше всего, тем самым улучшая общее предсказание.

2 Градиентный Спуск в Пространстве Функций

Минимизация Функции Потерь

GBM обобщает идею бустинга, используя **градиентный спуск** для минимизации произвольной дифференцируемой **функции потерь (Loss Function)** $L(y, F(x))$. Здесь y - истинное значение, $F(x)$ - текущее предсказание ансамбля.
Ключевое отличие от стандартного градиентного спуска: оптимизация происходит не в пространстве параметров модели, а в **пространстве функций**.

Шаги Градиентного Бустинга

Процесс обучения на шаге m (для $m = 1, \ldots, M$):

1. **Инициализация:** Начинаем с простого предсказания $F_0(x)$, обычно константы, минимизирующей loss (например, среднее y для MSE, медиана для MAE, логарифм шансов для LogLoss).

2. **Вычисление Псевдо-остатков:** Для каждого объекта i вычисляется **отрицательный градиент** функции потерь по пред-

сказанию ансамбля на предыдущем шаге $F_{m-1}(x)$:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

Эти r_{im} называются **псевдо-остатками** и показывают, в каком "направлении" нужно изменить предсказание $F_{m-1}(x_i)$, чтобы уменьшить ошибку L.

3. **Обучение Слабой Модели:** Новая слабая модель $h_m(x)$ (обычно неглубокое дерево решений) обучается аппроксимировать псевдо-остатки $\{(x_i, r_{im})\}_{i=1}^N$.

4. **Поиск Оптимального Шага (опционально):** Для дерева решений часто находят оптимальные значения γ_{jm} в листьях j дерева h_m.

5. **Обновление Ансамбля:** Предсказание ансамбля обновляется:

$$F_m(x) = F_{m-1}(x) + \nu \cdot h_m(x)$$

где ν (nu) — это **темп обучения (learning rate)**, коэффициент $0 < \nu \leq 1$ (обычно маленький, e.g., 0.01-0.1), который масштабирует вклад каждой новой модели. Он помогает предотвратить переобучение и делает сходимость более плавной.

Аналогия: Спуск с Холма

Представьте функцию потерь как ландшафт, где высота — это ошибка.

- Ваше текущее положение — предсказание ансамбля $F_{m-1}(x)$.
- Градиент $\frac{\partial L}{\partial F}$ показывает направление самого крутого подъема.
- Псевдо-остатки ($-\frac{\partial L}{\partial F}$) указывают в сторону спуска (антиградиент).
- Обучение $h_m(x)$ на псевдо-остатках — это попытка найти "шаг" в направлении спуска.
- Learning rate ν — это размер этого шага. Маленькие шаги помогают не "проскочить" долину (минимум ошибки).

3 Основные Функции Потерь (Loss Functions)

Зависимость от Задачи

Выбор функции потерь $L(y, F)$ критически важен и зависит от решаемой задачи (регрессия или классификация) и специфики данных (например, наличие выбросов).

3.1. A Функции Потерь для Регрессии

Регрессионные Loss-функции

- **MSE (Mean Squared Error) / L2 Loss:**

$$L(y, F) = \frac{1}{2}(y - F)^2$$

  Псевдо-остатки: $r = y - F$. Стандартный выбор, но чувствителен к выбросам из-за квадратичной ошибки.
- **MAE (Mean Absolute Error) / L1 Loss:**

$$L(y, F) = |y - F|$$

  Псевдо-остатки: $r = \text{sign}(y - F)$. Менее чувствительна к выбросам, чем MSE.
- **Huber Loss:**

$$L(y, F) = \begin{cases} \frac{1}{2}(y - F)^2 & \text{if } |y - F| \leq \delta \\ \delta(|y - F| - \frac{1}{2}\delta) & \text{if } |y - F| > \delta \end{cases}$$

  Комбинирует свойства MSE (для малых ошибок) и MAE (для больших ошибок), что делает ее робастной к выбросам. Параметр δ контролирует порог переключения.
- **Quantile Loss:** Используется для предсказания квантилей распределения целевой переменной.

3.2. B Функции Потерь для Бинарной Классификации

Бинарные Классификационные Loss-функции

Здесь $y \in \{0, 1\}$ или $y \in \{-1, 1\}$, а F обычно представляет логит вероятности p, т.е. $F = \log(\frac{p}{1-p})$.

- **LogLoss (Логистическая / Бинарная Кросс-Энтропия):**

  $L(y, F) = \log(1+e^{-F})$ (для $y = 1$) или $L(y, F) = \log(1+e^{F})$

  Или общая форма для $y \in \{0, 1\}$: $L(y, p) = -[y \log(p) + (1 - y) \log(1 - p)]$, где $p = \sigma(F) = \frac{1}{1+e^{-F}}$. Стандартный и наиболее распространенный выбор для задач классификации. Псевдо-остатки: $r = y - p$.
- **Exponential Loss (Экспоненциальная):**

$$L(y, F) = e^{-yF} \quad (\text{для } y \in \{-1, 1\})$$

  Используется в алгоритме AdaBoost. Сильнее штрафует за неверные предсказания, может быть менее робастна к шуму/выбросам, чем LogLoss.

3.3. C Функции Потерь для Многоклассовой Классификации

Многоклассовые Классификационные Loss-функции

- **Multinomial LogLoss (Категориальная Кросс-Энтропия):** Обобщение бинарной LogLoss на случай $K > 2$ классов. Ансамбль предсказывает вектор логитов $F = (F_1, \ldots, F_K)$, вероятности получаются через Softmax: $p_k = \frac{e^{F_k}}{\sum_{j=1}^K e^{F_j}}$.

$$L(y, p) = -\sum_{k=1}^{K} y_k \log(p_k)$$

  где y - one-hot вектор истинного класса.

4 Популярные Библиотеки GBM

Зачем Нужны Продвинутые Реализации?

Стандартный алгоритм GBM имеет ряд ограничений (например, склонность к переобучению, не самая высокая скорость). На практике почти всегда используют его улучшенные реализации, такие как XGBoost, LightGBM и CatBoost, которые включают множество оптимизаций и дополнительных возможностей.

4.1. A XGBoost (eXtreme Gradient Boosting)

Ключевые Особенности XGBoost

- **Регуляризация:** В функцию потерь при построении дерева добавляются штрафы L1 (Lasso) и L2 (Ridge) на веса листьев. Это контролирует сложность моделей и эффективно борется с переобучением.
- **Улучшенный Поиск Сплитов:** Использует информацию о второй производной функции потерь (Гессиан) для более точного построения деревьев.
- **Обработка Пропусков (NaN):** Имеет встроенный механизм

для работы с пропущенными значениями: при построении дерева алгоритм "учится", в какую ветку (левую или правую) лучше направлять объекты с NaN для каждого признака.
- **Оптимизации Скорости:**
  - Параллельные вычисления на уровне построения дерева (по признакам).
  - Приближенные алгоритмы поиска сплитов (quantile approximation) для больших данных.
  - Кэширование градиентов и гессианов.
  - Блочная структура данных для эффективного доступа к памяти.
- **Кросс-валидация:** Встроенная функция для кросс-валидации при подборе числа деревьев.

Позиционирование: XGBoost долгое время был "золотым стандартом" и де-факто выбором №1 для соревнований и промышленных задач на табличных данных.

4.2. B LightGBM (Light Gradient Boosting Machine)

Ключевые Особенности LightGBM

- **Высокая Скорость и Низкое Потребление Памяти:** Основное преимущество, особенно на больших датасетах. Достигается за счет нескольких техник.
- **Leaf-wise Рост Деревьев:** Вместо роста по уровням (level-wise, как в XGBoost), LightGBM выбирает для расщепления тот лист, который даст максимальное уменьшение функции потерь (gain). Это позволяет строить более глубокие и асимметричные деревья, что часто эффективнее, но требует контроля глубины (max_depth) для предотвращения переобучения на малых данных.
- **GOSS (Gradient-based One-Side Sampling):** Для ускорения обучения на каждой итерации используются не все данные. Алгоритм сохраняет все объекты с большими градиентами (на которых модель сильно ошибается) и случайно отбирает долю объектов с малыми градиентами. Это позволяет сфокусироваться на "сложных" объектах без большого смещения оценки градиента.
- **EFB (Exclusive Feature Bundling):** Техника для уменьшения числа признаков путем объединения "взаимоисключающих" признаков (тех, которые редко принимают ненулевые значения одновременно, например, one-hot кодированные).
- **Оптимизированная Обработка Категориальных Признаков:** Поддерживает передачу категориальных признаков напрямую (без OHE), используя специальные алгоритмы сплита (Fisher).

Позиционирование: Отличный выбор для очень больших датасетов, где скорость обучения и потребление памяти критичны.

4.3. C CatBoost (Categorical Boosting)

Ключевые Особенности CatBoost

- **Лучшая Обработка Категориальных Признаков:** Главная "фишка". Использует продвинутые методы кодирования категорий "на лету" во время обучения:
  - **Ordered Target Statistics (TS):** Вычисляет статистики целевой переменной (например, среднее значение y) для каждой категории, но делает это хитро, используя "исторические" данные (только объекты, идущие до текущего в некоторой случайной перестановке), чтобы избежать утечки целевой переменной (target leakage) и переобучения.
  - Комбинации категориальных признаков генерируются автоматически.
  - Не требует предварительной обработки категорий (как OHE), что упрощает пайплайн и часто дает лучшее качество.
- **Ordered Boosting:** Модификация градиентного бустинга, которая также борется с target leakage и prediction shift, обучая модель на остатках, полученных на данных, не включающих текущий объект (для TS).
- **Симметричные (Oblivious) Деревья:** Все узлы на одном уровне дерева используют одно и то же условие (признак и порог) для сплита. Это:
  - Действует как неявная регуляризация.
  - Значительно ускоряет предсказание модели (особенно на CPU).
  - Упрощает структуру модели.
- **Меньше Настройки Гиперпараметров:** Часто показывает хорошие результаты "из коробки" с параметрами по умолчанию.
- **Хорошая Визуализация:** Встроенные инструменты для анализа модели и процесса обучения.

Позиционирование: Идеален для задач с большим количеством категориальных признаков. Часто дает высокое качество с минимальной настройкой и имеет очень быстрое время предсказания.

5 Важность Признаков (Feature Importance) в GBM

Методы Оценки Важности

GBM, как и другие ансамбли деревьев, позволяет оценить вклад каждого признака в итоговое предсказание. Основные методы:

- **Gain (Прирост / Feature Importance):** Среднее уменьшение функции потерь (или другого критерия, например, Gini impurity/Variance reduction) при использовании признака для сплита во всех деревьях ансамбля. Суммарный gain по всем сплитам признака делится на общее число сплитов по этому признаку (или просто суммируется, зависит от реализации). **Считается наиболее надежным методом.**
- **Split Count / Frequency (Частота Использования / Weight):** Просто подсчитывает, сколько раз признак был выбран для разделения узла во всех деревьях. Простой метод, но не учитывает, насколько *полезным* был каждый сплит. Может переоценивать важность числовых признаков с большим количеством потенциальных порогов.
- **Coverage (Покрытие):** Среднее количество объектов в обучающей выборке, которые проходят через сплиты по данному признаку (иногда взвешенное по gain или другим метрикам). Показывает, какую долю данных "затрагивает" признак. Предоставляется не всеми библиотеками (есть в XGBoost, LightGBM).

Использование Информации о Важности Признаков

Знание важности признаков полезно для:

- **Интерпретации модели:** Понять, какие факторы наиболее сильно влияют на предсказание (хотя GBM остается моделью "черного ящика" по сравнению с линейными моделями).
- **Отбора признаков (Feature Selection):** Исключить неважные или малозначимые признаки, что может упростить модель, ускорить обучение/предсказание и иногда даже улучшить качество за счет уменьшения шума.
- **Генерации новых признаков (Feature Engineering):** Сосредоточить усилия на создании признаков на основе наиболее важных существующих.

Предостережение

Следует с осторожностью относиться к результатам оценки важности:

- Методы (особенно 'Split Count') могут быть **смещены** в сторону числовых признаков с большим количеством уникальных значений (high cardinality numerical features) или категориальных признаков с большим числом категорий (если используется OHE или простые методы кодирования), так как у них больше потенциальных точек для сплита. CatBoost с его обработкой категорий менее подвержен этой проблеме для категориальных данных.
- **Коррелирующие признаки:** Если два признака сильно коррелируют и оба полезны, модель может использовать для сплитов то один, то другой. В результате их важность может быть "размазана" между ними, и каждый по отдельности будет выглядеть менее важным, чем он есть на самом деле.