Шпаргалка по Градиентному бустингу / Концепции Cheatsheet (XeLaTeX)

Краткий справочник

April 2, 2025

Contents

Что такое Градиентный Бустинг?

Градиентный Бустинг (Gradient Boosting Machine, GBM) — это мощный ансамблевый метод машинного обучения, который строит модели **последовательно**, где каждая новая модель исправляет ошибки предыдущей. Это один из самых эффективных алгоритмов для табличных данных.
Аналогия: Представь, что ты лепишь скульптуру. Вместо того чтобы сразу создать шедевр (что сложно), ты сначала делаешь грубую основу, потом видишь недочеты и добавляешь кусочек глины там, где нужно, потом еще и еще, пока скульптура не станет идеальной. Каждое добавление "кусочка глины" — это новая слабая модель в бустинге.

1 Идея Бустинга (Boosting)

Последовательное Исправление Ошибок

В отличие от **бэггинга** (как в Random Forest), где модели обучаются независимо и параллельно, в **бустинге** модели строятся одна за другой:

1. Обучается первая (обычно простая) модель на исходных данных.
2. Вычисляются ошибки (остатки) этой модели.
3. Следующая модель обучается предсказывать эти ошибки (или что-то, связанное с ними).
4. Предсказания новой модели добавляются к предсказаниям ансамбля (с некоторым весом), чтобы уменьшить общую ошибку.
5. Шаги 2-4 повторяются много раз, пока ошибка не перестанет уменьшаться или не будет достигнуто заданное число моделей.

Ключевая идея: ансамбль "учится" на своих ошибках, постепенно улучшая предсказание. Каждая следующая модель фокусируется на тех данных, где предыдущие модели ошибались больше всего.

2 Градиентный Спуск на Функциях

Как Бустинг "Учится" Ошибкам?

GBM использует идею **градиентного спуска**, но не в пространстве параметров (как в нейросетях), а в **пространстве функций**.

- Мы хотим минимизировать некоторую **функцию потерь (Loss Function)**, $L(y, F(x))$, где y - истинное значение, $F(x)$ - текущее предсказание ансамбля.
- **Инициализация:** Начальное предсказание $F_0(x)$ обычно простое: константа (например, среднее значение y для регрессии или логарифм шансов для классификации).
- На каждом шаге m мы вычисляем **псевдо-остатки** (pseudo-residuals) — это **отрицательный градиент** функции потерь по предсказанию ансамбля на предыдущем шаге ($F_{m-1}(x)$):

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

  где i - номер объекта.
- Новая слабая модель $h_m(x)$ (обычно дерево решений) обучается предсказывать эти псевдо-остатки r_{im}.
- Ансамбль обновляется: $F_m(x) = F_{m-1}(x) + \nu \cdot h_m(x)$, где ν (nu) — это **темп обучения (learning rate)**, маленький коэффициент (например, 0.01-0.1), который уменьшает вклад каждой новой модели и делает обучение более робастным.

Аналогия: Представь, что ты стоишь на холме (твоя текущая ошибка) и хочешь спуститься вниз (минимизировать ошибку). Градиент показывает направление "самого крутого подъема". Ты делаешь шаг в **противоположном** направлении (анти-градиент) — это и есть обучение новой модели на псевдо-остатках. Learning rate — это размер твоего шага. Маленькие шаги помогают не проскочить минимум.

3 Основные Функции Потерь (Loss Functions)

Выбор Функции Потерь

Выбор функции потерь зависит от задачи:

- **Регрессия:**
  - **MSE (Mean Squared Error):** $L(y, F) = \frac{1}{2}(y - F)^2$. Псевдо-остатки — это просто обычные остатки $(y - F)$. Чувствительна к выбросам.
  - **MAE (Mean Absolute Error):** $L(y, F) = |y - F|$. Псевдо-остатки $-\text{sign}(y - F)$. Менее чувствительна к выбросам.
  - **Huber Loss:** Комбинация MSE и MAE, робастна к выбросам.
- **Бинарная Классификация:**
  - **LogLoss (Логистическая функция потерь / Бинарная Кросс-Энтропия):** $L(y, F) = y \log(1 + e^{-F}) + (1 - y) \log(1 + e^{F})$ (для $y \in \{0, 1\}$). Стандартный выбор. Предсказание F здесь — это логит вероятности.
  - **Exponential Loss (Экспоненциальная):** $L(y, F) = e^{-yF}$ (для $y \in \{-1, 1\}$). Используется в классическом AdaBoost. Более агрессивно наказывает за ошибки.
- **Многоклассовая Классификация:** Обычно используется **Multinomial LogLoss**.

4 Популярные Библиотеки: XGBoost, LightGBM, CatBoost

Ключевые Отличия и Фишки (Концептуально)

Хотя базовый GBM существует, на практике почти всегда используют его продвинутые реализации. Вот их главные "фишки":

- **XGBoost (eXtreme Gradient Boosting):**
  - **Регуляризация:** Включает L1 и L2 регуляризацию на веса листьев деревьев, что помогает бороться с переобучением.
  - **Обработка пропусков:** Встроенный механизм для работы с NaN (учится, в какую ветку направлять NaN при сплите).
  - **Оптимизации скорости:** Параллельные вычисления, приближенные алгоритмы поиска сплитов (quantile approximation), кэширование градиентов.
  - *Фишка:* Первым предложил многие из этих улучшений, стал "золотым стандартом".
- **LightGBM (Light Gradient Boosting Machine):**
  - **Скорость и память:** Часто быстрее XGBoost и потребляет меньше памяти.
  - **Leaf-wise рост деревьев:** Строит дерево не по уровням (level-wise), а выбирая лист, который даст наибольшее уменьшение ошибки (leaf-wise). Это эффективнее, но может привести к переобучению на малых данных.
  - **GOSS (Gradient-based One-Side Sampling):** Сохраняет объекты с большими градиентами (те, на которых модель сильно ошибается) и случайно отбрасывает часть объектов с малыми градиентами для ускорения обучения.
  - **EFB (Exclusive Feature Bundling):** Объединяет взаимоисключающие признаки (те, что редко одновременно ненулевые, как one-hot encoding) для уменьшения размерности.
  - *Фишка:* Скорость и эффективность на больших датасетах.
- **CatBoost (Categorical Boosting):**
  - **Обработка категориальных признаков:** Главная фишка! Использует продвинутые методы (Ordered Target Statistics) для кодирования категорий "на лету" без предварительной обработки вроде One-Hot Encoding, что часто дает лучший результат и предотвращает переобучение.
  - **Симметричные деревья (Oblivious Trees):** Все узлы на одном уровне дерева используют одно и то же условие для сплита. Это ускоряет предсказание и действует как регуляризация.
  - **Меньше тюнинга:** Часто дает хорошие результаты с параметрами по умолчанию.
  - **Ordered Boosting:** Вариация бустинга, помогающая бороться со сдвигом предсказаний (prediction shift) из-за использования целевой переменной при кодировании категорий.
  - *Фишка:* Лучшая (из коробки) работа с категориальными данными и быстрая скорость предсказания.

5 Важность Признаков (Feature Importance)

Важность Признаков

Градиентный бустинг, как и другие древовидные модели, позволяет оценить важность признаков:

- **Gain (Прирост):** Среднее уменьшение ошибки (loss), которое дает сплит по данному признаку во всех деревьях ансамбля. Чем больше признак уменьшает ошибку, тем он важнее. **Часто считается наиболее информативным методом.**
- **Split Count / Frequency (Частота Использования):** Сколько раз признак использовался для сплита во всех деревьях. Проще, но менее точно, так как не учитывает, насколько *полезным* был сплит.
- **Coverage (Покрытие):** Среднее количество объектов, проходящих через сплиты по данному признаку (взвешенное по уровню ошибки). Не все библиотеки предоставляют.

Знание важности признаков помогает:

- Понять, на какие данные модель опирается больше всего.
- Провести отбор признаков (Feature Selection).
- Интерпретировать модель (хотя GBM все еще сложнее интерпретировать, чем линейные модели).

Предостережение: Важно помнить, что методы оценки важности признаков (особенно 'Split Count') могут быть смещены в сторону признаков с большим количеством уникальных значений (high cardinality numerical features) или категориальных признаков с большим числом категорий (если они не обработаны CatBoost-ом).