Машинное обучение

DS-поток

Лекция 8

Ансамбли моделей

История

1. **AdaBoost** — 1990

   Появился в процессе исследования вопроса о том,
   можно ли из большого количества относительно слабых
   и простых моделей получить одну сильную.

2. **Gradient Boosting** — 1999

3. **XGBoost** — март 2014

   Стартовал как исследовательский проект Тяньцзи Чена.
   После победы в Higgs Machine Learning Challenge,
   он стал хорошо известным и популярным в сообществе Kaggle.

4. **LightGBM** — январь 2017

   Microsoft разработал "легкую" версию бустинга.

5. **CatBoost** — июль 2017

   Яндекс выложил свою версию бустинга
   с различными оптимизациями.

Ансамбли моделей

XGBoost

LightGBM

CatBoost

Повторение градиентного бустинга

Функционал качества $Q(Y, \widehat{y}) = \sum_{i=1}^{n} \mathcal{L}(Y_i, \widehat{y}(x_i)) \longrightarrow \min_{\widehat{y}}$,

где $\mathcal{L}(y, z)$ — кусочно дифф. функция потерь.

Строим композицию в виде $\widehat{y}_T(x) = \sum_{t=0}^{T} \gamma_t b_t(x)$.

Построение новой модели шаге t:

1. Вычислить градиенты по обучающей выборке

$$\widetilde{g}^t = \left(\nabla_s \mathcal{L}(Y_i, s)|_{s=\widehat{y}_{t-1}(x_i)} \right)_{i=1}^{n}$$

2. Обучить новую модель по MSE по выборке $(x_1, -\widetilde{g}_1^t), \ldots, (x_n, -\widetilde{g}_n^t)$.

$$b_t(x) = \arg\min_{b \in \mathcal{F}} \sum_{i=1}^{n} \left(b(x_i) + \widetilde{g}_i^t \right)^2.$$

2.3 Подобрать коэффициент при b_t

$$\widetilde{\gamma}_t = \arg\min_{\gamma \in \mathbb{R}} \sum_{i=1}^{n} \mathcal{L}(Y_i, \widehat{y}_{t-1}(x_i) + \gamma b_t(x_i)).$$

2.4 Добавить модель к композиции

$$\widehat{y}_t(x) = \widehat{y}_{t-1}(x) + \eta \widetilde{\gamma}_t b_t(x).$$

Доска

Далее на слайдах только основные моменты

eXtreme Gradient Boosting (XGBoost)

Обычный градиентый бустинг:

$$\sum_{i=1}^{n} \left(\widetilde{g}_i b(x_i) + \frac{1}{2} b^2(x_i) \right) \to \min_{b \in \mathcal{F}}$$

XGBoost:

$$\sum_{i=1}^{n} \left(\widetilde{g}_i b(x_i) + \frac{1}{2} \widetilde{h}_i b^2(x_i) \right) \to \min_{b}$$

Вывод:

В обычном градиентном бустинге используем аппроксимацию второго порядка и отбрасываем информацию о вторых производных. Как будто функция имеет одну кривизну по всем направлениям.

Регуляризация в XGBoost

Дерево $b(x)$ можно описать формулой

$$b(x) = \sum_{k=1}^{\ell} \gamma_k \cdot I\{x \in R_k\}$$

где $k = 1, \ldots, \ell$ — индексы листьев,

R_k — соответствующие области разбиения: $\bigsqcup_{k=1}^{\ell} R_k = \mathscr{X}$,

γ_k — значения в листьях.

Сложность $b(x)$ зависит от:

1. Число листьев ℓ.
   Чем больше листьев, тем сложнее разделяющая поверхность.
2. Норма коэффициентов в листьях $\|b\|_2^2 = \sum_{k=1}^{\ell} \gamma_k^2$.
   Чем дальше коэффициенты от нуля, тем сильнее b влияет на ответ.

Добавим регуляризаторы, штрафующие за оба вида сложности:

$$\sum_{i=1}^{n} \left(\widetilde{g}_i b(x_i) + \frac{1}{2} \widetilde{h}_i b^2(x_i) \right) + \mu\ell + \frac{\lambda}{2} \sum_{k=1}^{\ell} \gamma_k^2 \to \min_b$$

Переподбор в листьях дерева

Оптимальные коэффициенты в листьях

$$\gamma_k = \frac{-G_k}{H_k + \lambda},$$

$$G_k = \sum_{i \in R_k} \widetilde{g}_i \qquad H_k = \sum_{i \in R_k} \widetilde{h}_i$$

Ошибка дерева с оптимальными коэффициентами в листьях

$$\widetilde{Q}(Y, b) = \frac{1}{2} \sum_{k=1}^{\ell} \frac{-G_k^2}{H_k + \lambda} + \mu\ell$$

Обучение дерева в XGBoost

Функционал $\widetilde{Q}(Y, b)$ для заданной структуры дерева b вычисляет мин. ошибку, которую можно получить, подобрав ответы в листьях.

$$\widetilde{Q}(Y, b) = \sum_{k=1}^{\ell} \left(-\frac{1}{2} \frac{G_k^2}{H_k + \lambda} + \mu \right) = \sum_{k=1}^{\ell} H(R_j)$$

Заметим, что $H(X)$ подходит на роль критерия информативности:

- $H(X)$ зависит лишь от выборки X, попавшей в вершину.
- $H(X)$ тем меньше, чем лучше объекты X предсказываются константой γ при оптимальном выборе этой константы.

  $H(R_j)$ показывает ошибку в листе R_j при предсказании γ_j.

$\Rightarrow$ $H(X)$ можно использовать как критерий информативности при построении дерева.

Обучение дерева в XGBoost

Выберем разбиение $I\{x_j < t\}$ в вершине X_m из задачи:

$$Q(X_m, j, t) = H(X_m) - H(X_\ell) - H(X_r) \to \max_{j,t}$$

где информативность вычисляется по формуле:

$$H(X) = -\frac{1}{2}\left(\sum_{i \in X} \widetilde{g}_i\right)^2 \Bigg/ \left(\sum_{i \in X} \widetilde{h}_i + \lambda\right) + \mu.$$

Тем самым выбираем структуру дерева исходя
из минимизации исходной функции потерь.

Критерий остановки:

Вершину объявим листом, если даже лучшее из разбиений приводит к отрицательному значению Q, то есть к росту ошибки $\widetilde{Q}(b)$.

Замечение: В Q нет нормировки на размеры дочерних вершин, т.к. здесь $H(X)$ зависит от размера классов, а ранее было $H(X) = \min_{c \in Y} \frac{1}{|X|} \sum_{x_i \in X} \mathcal{L}(Y_i, c)$,.

Особенности XGBoost

1. Использование вторых производных
   Базовая модель приближает направление,
   посчитанное с учетом вторых производных функции потерь.

2. Регуляризация
   Добавляются штрафы за количество листьев
   и за норму коэффициентов.

3. Другой критерий информативности
   При построении дерева используется критерий информативности,
   зависящий от оптимального вектора сдвига.

4. Другой критерий останова
   Критерий останова при обучении дерева также зависит
   от оптимального сдвига.

5. И много другого
   Обработка пропущенных значений, аппроксимация порога t, ...

Ансамбли моделей

XGBoost

LightGBM

CatBoost

LightGBM

Основная идея

Если антиградиент $\widetilde{g}_i$ на объекте большой, то этот объект важен и данный $\widetilde{g}_i$ внесет большой вклад в уменьшение итоговой ошибки.

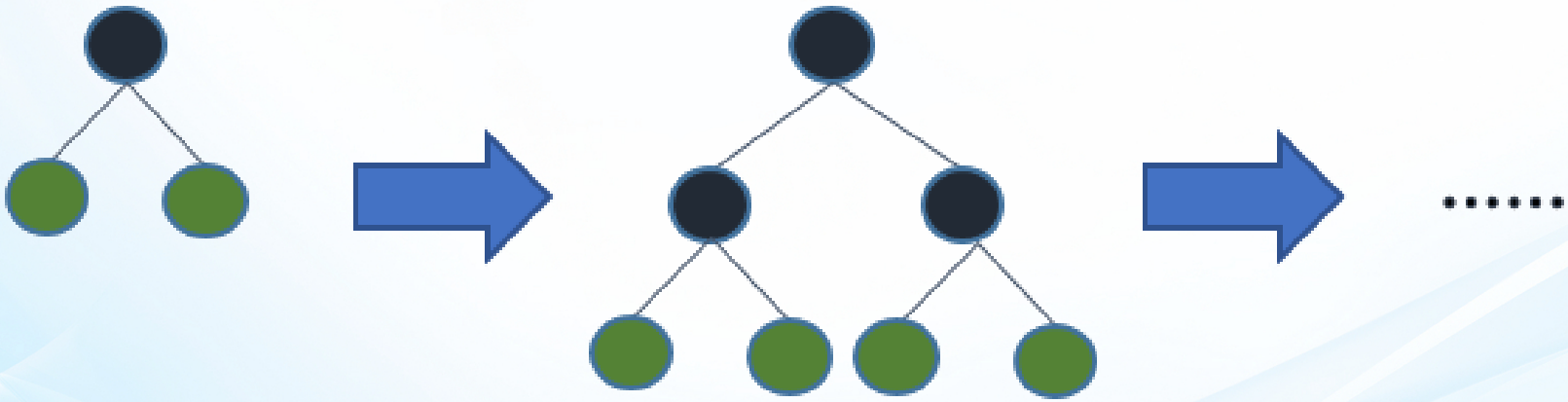
$\Rightarrow$ будем использовать $\widetilde{g}_i$ как вес объекта x_i.

Поиск разбиения в вершине дерева:

1. Берем объекты с большим $\widetilde{g}_i$.
2. Добавим к ним случайно выбранное множество объектов из объектов с маленьким $\widetilde{g}_i$.
3. Будем использовать только эти объекты для нахождения оптимального разбиения.

Структура обычного решающего дерева

Обычное дерево



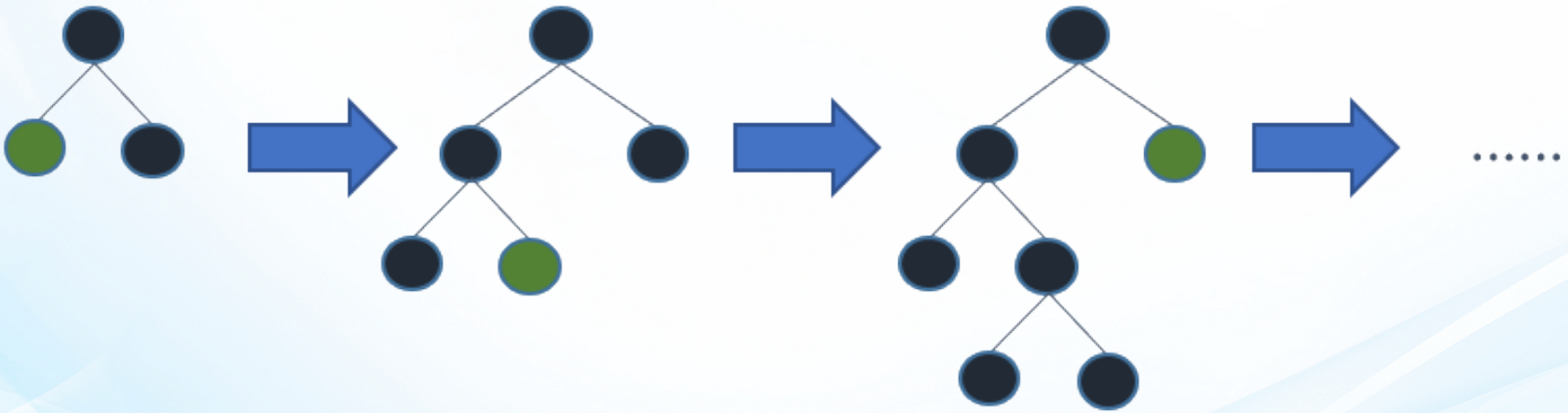
Level-wise tree growth

Дерево строится по уровням.

Для каждой вершины в уровне находим оптимальное разбиение.

Структура дерева в LightGBM

LightGBM



Leaf-wise tree growth

Среди всех листов дерева находим лист,
который оптимальнее всего разделить на этой итерации.

Полученнное дерево может быть очень несбалансированным.

Особенности LightGBM

1. Другая структура дерева.

2. Отбор объектов для построения дерева по градиентам.

3. Exclusive Feature Bundling

   Признаки, которые редко бывают одновременно ненулевыми на одном объекте, заменяются на один признак.

4. Histogram-based поиск оптимального порога (как и в XGBoost).

   Возможные значения признака разбиваются на бины.
   В качестве порогов рассматриваются только граница бинов.

Ансамбли моделей

XGBoost

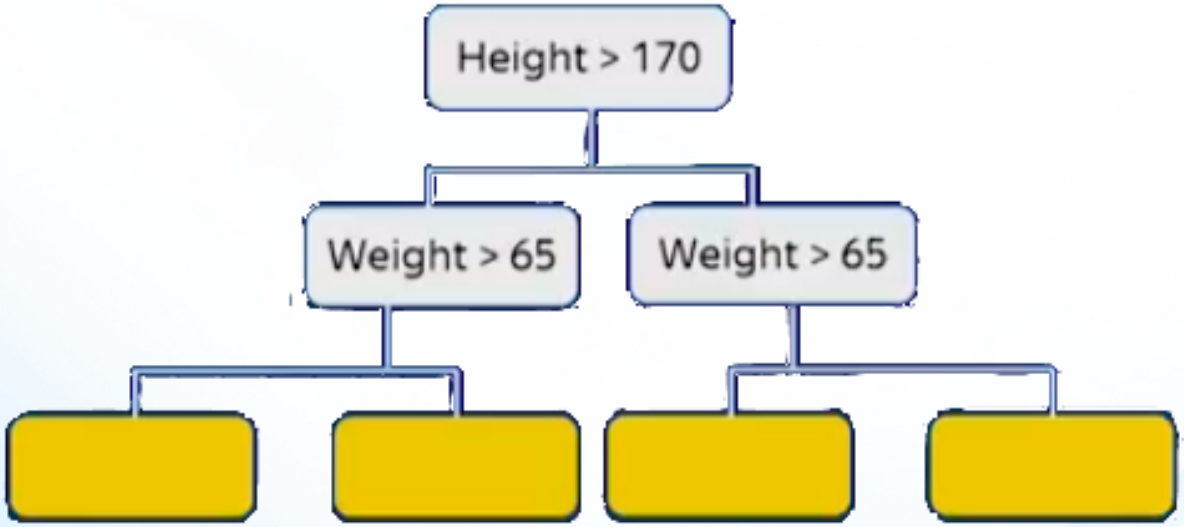
LightGBM

CatBoost

CatBoost: Структура дерева

Дерево строится по уровням.

Для всех вершин одного уровня предикат одинаковый.



- Сильно снижается время обучения и предсказания.
- Уменьшается требуемая память.
- Является неплохой регуляризацией.

Особенности CatBoost

1. Другая структура дерева.

2. Minimum Variance Sampling.

   Признаки для построения дерева выбираются на основе градиента.
   Сэмлирование производится по величине градиента.

3. Mean encoding с регуляризацией Expanding mean.

   Обработка категориальных признаков. Об этом на след. лекции.

4. Histogram-based поиск оптимального порога (как и в XGBoost).

   Возможные значения признака разбиваются на бины.
   В качестве порогов рассматриваются только граница бинов.

5. И многое другое.


ВСЁ!