Лекция 10
Градиентный бустинг

Е. А. Соколов
ФКН ВШЭ

23 ноября 2020 г.

Мы уже разобрались с двумя типами методов построения композиций — бустингом и бэггингом, и познакомились с градиентным бустингом и случайным лесом, которые являются наиболее яркими представителями этих классов. На практике реализация градиентного бустинга оказывается очень непростой задачей, в которой успех зависит от множества тонких моментов. Мы рассмотрим конкретную реализацию градиентного бустинга — пакет XGBoost [1], который считается одним из лучших на сегодняшний день. Это подтверждается, например, активным его использованием в соревнованиях по анализу данных на `kaggle.com`.

1 Extreme Gradient Boosting (XGBoost)

§1.1 Градиентный бустинг

Вспомним, что на каждой итерации градиентного бустинга вычисляется вектор сдвигов s, который показывает, как нужно скорректировать ответы композиции на обучающей выборке, чтобы как можно сильнее уменьшить ошибку:

$$s = \left(-\frac{\partial L}{\partial z}\Big|_{z=a_{N-1}(x_i)} \right)_{i=1}^{\ell} = -\nabla_z \sum_{i=1}^{\ell} L(y_i, z_i)\Big|_{z_i=a_{N-1}(x_i)} \tag{1.1}$$

После этого новый базовый алгоритм обучается путем минимизации среднеквадратичного отклонения от вектора сдвигов s:

$$b_N(x) = \arg\min_{b \in \mathcal{A}} \sum_{i=1}^{\ell} (b(x_i) - s_i)^2$$

§1.2 Альтернативный подход

На прошлой лекции мы аргументировали использование среднеквадратичной функции потерь тем, что она наиболее проста для оптимизации. Попробуем найти более адекватное обоснование этому выбору.

Мы хотим найти алгоритм $b(x)$, решающий следующую задачу:

$$\sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + b(x_i)) \to \min_b$$

Разложим функцию L в каждом слагаемом в ряд Тейлора до второго члена с центром в ответе композиции $a_{N-1}(x_i)$:

$$\sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + b(x_i)) \approx$$
$$\approx \sum_{i=1}^{\ell} \left(L(y_i, a_{N-1}(x_i)) - s_i b(x_i) + \frac{1}{2} h_i b^2(x_i) \right),$$

где через h_i обозначены вторые производные по сдвигам:

$$h_i = \frac{\partial^2}{\partial z^2} L(y_i, z) \Big|_{a_{N-1}(x_i)}$$

Первое слагаемое не зависит от нового базового алгоритма, и поэтому его можно выкинуть. Получаем функционал

$$\sum_{i=1}^{\ell} \left(-s_i b(x_i) + \frac{1}{2} h_i b^2(x_i) \right) \to \min_b \tag{1.2}$$

Покажем, что он очень похож на среднеквадратичный из формулы (1.1). Преобразуем его:

$$\sum_{i=1}^{\ell} (b(x_i) - s_i)^2$$
$$= \sum_{i=1}^{\ell} \left(b^2(x_i) - 2 s_i b(x_i) + s_i^2 \right) = \{\text{последнее слагаемое не зависит от } b\}$$
$$= \sum_{i=1}^{\ell} \left(b^2(x_i) - 2 s_i b(x_i) \right)$$
$$= 2 \sum_{i=1}^{\ell} \left(-s_i b(x_i) + \frac{1}{2} b^2(x_i) \right) \to \min_b$$

Видно, что последняя формула совпадает с (1.2) с точностью до константы, если положить $h_i = 1$. Таким образом, в обычном градиентном бустинге мы используем аппроксимацию второго порядка при обучении очередного базового алгоритма, и при этом отбрасываем информацию о вторых производных (то есть считаем, что функция имеет одинаковую кривизну по всем направлениям).

§1.3 Регуляризация

Будем далее работать с функционалом (1.2). Он измеряет лишь ошибку композиции после добавления нового алгоритма, никак при этом не штрафуя за излишнюю сложность этого алгоритма. Ранее мы решали проблему переобучения путем ограничения глубины деревьев, но можно подойти к вопросу и более гибко. Мы выясняли, что дерево $b(x)$ можно описать формулой

$$b(x) = \sum_{j=1}^{J} b_j [x \in R_j]$$

Его сложность зависит от двух показателей:

1. Число листьев J. Чем больше листьев имеет дерево, тем сложнее его разделяющая поверхность, тем больше у него параметров и тем выше риск переобучения.

2. Норма коэффициентов в листьях $\|b\|_2^2 = \sum_{j=1}^{J} b_j^2$. Чем сильнее коэффициенты отличаются от нуля, тем сильнее данный базовый алгоритм будет влиять на итоговый ответ композиции.

Добавляя регуляризаторы, штрафующие за оба этих вида сложности, получаем следующую задачу:

$$\sum_{i=1}^{\ell} \left(-s_i b(x_i) + \frac{1}{2} h_i b^2(x_i) \right) + \gamma J + \frac{\lambda}{2} \sum_{j=1}^{J} b_j^2 \to \min_b$$

Если вспомнить, что дерево $b(x)$ дает одинаковые ответы на объектах, попадающих в один лист, то можно упростить функционал:

$$\sum_{j=1}^{J} \left\{ \underbrace{\left(-\sum_{i \in R_j} s_i \right)}_{=-S_j} b_j + \frac{1}{2} \left(\lambda + \underbrace{\sum_{i \in R_j} h_i}_{=H_j} \right) b_j^2 + \gamma \right\} \to \min_b$$

Каждое слагаемое здесь можно минимизировать по b_j независимо. Заметим, что отдельное слагаемое представляет собой параболу относительно b_j, благодаря чему можно аналитически найти оптимальные коэффициенты в листьях:

$$b_j = \frac{S_j}{H_j + \lambda}$$

Подставляя данное выражение обратно в функционал, получаем, что ошибка дерева с оптимальными коэффициентами в листьях вычисляется по формуле

$$H(b) = -\frac{1}{2} \sum_{j=1}^{J} \frac{S_j^2}{H_j + \lambda} + \gamma J \tag{1.3}$$

§1.4 Обучение решающего дерева

Мы получили функционал $H(b)$, который для заданной структуры дерева вычисляет минимальное значение ошибки (1.2), которую можно получить путем подбора коэффициентов в листьях. Заметим, что он прекрасно подходит на роль критерия информативности — с его помощью можно принимать решение, какое разбиение вершины является наилучшим! Значит, с его помощью мы можем строить дерево. Будем выбирать разбиение $[x_j < t]$ в вершине R так, чтобы оно решало следующую задачу максимизации:

$$Q = H(R) - H(R_\ell) - H(R_r) \to \max,$$

где информативность вычисляется по формуле

$$H(R) = -\frac{1}{2}\left(\sum_{(h_i,s_i)\in R} s_j\right)^2 \Bigg/ \left(\sum_{(h_i,s_i)\in R} h_j + \lambda\right) + \gamma.$$

За счет этого мы будем выбирать структуру дерева так, чтобы оно как можно лучше решало задачу минимизации исходной функции потерь. При этом можно ввести вполне логичный критерий останова: вершину нужно объявить листом, если даже лучшее из разбиений приводит к отрицательному значению функционала Q.

§1.5 Заключение

Итак, градиентный бустинг в XGBoost имеет ряд важных особенностей.

1. Базовый алгоритм приближает направление, посчитанное с учетом вторых производных функции потерь.

2. Функционал регуляризуется — добавляются штрафы за количество листьев и за норму коэффициентов.

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

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

2 Стекинг

Разумеется, существуют способы построения композиций помимо бустинга и бэггинга. Большую популярность имеет *стекинг*, в котором прогнозы алгоритмов объявляются новыми признаками, и поверх них обучается ещё один алгоритм (который иногда называют мета-алгоритмом). Стекинг очень популярен в соревнованиях по анализу данных, поскольку позволяет агрегировать разные модели (различные композиции, линейные модели, нейросети и т.д.; иногда в качестве базовых алгоритмов могут выступать результаты градиентного бустинга с разными значениями гиперпараметров).

Допустим, мы независимо обучили N базовых алгоритмов $b_1(x), \ldots, b_N(x)$ на выборке X, и теперь хотим обучить на их прогнозах мета-алгоритм $a(x)$. Самым простым вариантом будет обучить его на этой же выборке:

$$\sum_{i=1}^{\ell} L(y_i, a(b_1(x_i), \ldots, b_N(x_i))) \to \min_a$$

При таком подходе $a(x)$ будет отдавать предпочтение тем базовым алгоритмам, которые сильнее всех подогнались под целевую переменную на обучении (поскольку по их прогнозам лучше всего восстанавливаются истинные ответы). Если среди базовых алгоритмов будет идеально переобученный (то есть запомнивший ответы на всей обучающей выборке), то мета-алгоритму будет выгодно использовать только прогнозы данного переобученного базового алгоритма, поскольку это позволит добиться лучших результатов с точки зрения записанного функционала. При этом такой мета-алгоритм, конечно, будет показывать очень низкое качество на новых данных.

Чтобы избежать таких проблем, следует обучать базовые алгоритмы и мета-алгоритм на разных выборках. Разобьём нашу обучающую выборку на K блоков $X_1, \ldots, X_K$, и обозначим через $b_j^{-k}(x)$ базовый алгоритм $b_j(x)$, обученный по всем блокам, кроме k-го. Тогда функционал для обучения мета-алгоритма можно записать как

$$\sum_{k=1}^{K} \sum_{(x_i, y_i) \in X_k} L\left(y_i, a(b_1^{-k}(x_i), \ldots, b_N^{-k}(x_i))\right) \to \min_a$$

В данном случае при вычислении ошибки мета-алгоритма на объекте x_i используются базовые алгоритмы, которые не видели этот объект при обучении, и поэтому мета-алгоритм не может переобучиться на их прогнозах.

Блендинг. Частным случаем стекинга является блендинг, в котором мета-алгоритм является линейным:

$$a(x) = \sum_{n=1}^{N} w_n b_n(x).$$

Это самый простой способ объединить несколько алгоритмов в композицию. Иногда даже блендинг без обучения весов (то есть вариант с $w_1 = \cdots = w_N = 1/N$) позволяет улучшить качество по сравнению с отдельными базовыми алгоритмами.

Категориальные и текстовые признаки. Категориальные и текстовые признаки могут быть серьёзной помехой для использования композиций над деревьями. Стандартным способом кодирования является бинаризация для категориальных признаков и TF-IDF для текстовых, что приводит к очень большой размерности признакового пространства. Случайный лес на таком наборе признаков будет обучаться долго из-за большой глубины деревьев, а градиентный бустинг может показать слишком плохие результаты из-за небольшой глубины базовых деревьев (например, глубина 4 позволяет учитывать лишь зависимость целевой переменной от наборов из 4-х признаков; в случае с текстами ответы могут зависеть от существенно более крупных наборов слов).

Одним из решений этой проблемы может стать стекинг, в котором градиентным бустингом обучается мета-алгоритм $a(x)$, а каждый категориальный и текстовый признак схлопывается в одно число соответствующим базовым алгоритмом. Для категориальных признаков базовый алгоритм, например, может вычислять счётчики — при этом обратим внимание, что мы уже отмечали важность разделения обучающих выборок для счётчиков и настраиваемых поверх них моделей. Также популярным выбором для базовых алгоритмов являются линейные модели.

Список литературы

[1] *Tianqi Chen, Carlos Guestrin* (2016). XGBoost: A Scalable Tree Boosting System. // http://arxiv.org/abs/1603.02754