Семинары по композиционным методам

Евгений Соколов
sokolov.evg@gmail.com

18 марта 2016 г.

4 Композиционные методы машинного обучения

Мы уже разобрались с двумя классами 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`.

§4.1 Extreme Gradient Boosting (XGBoost)

4.1.1 Градиентный бустинг

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

$$s = \left(-\frac{\partial L}{\partial z}\bigg|_{z=a_{N-1}(x_i)} \right)_{i=1}^{\ell} = -\nabla_s \sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + s_i)$$

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

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

Также мы обсуждали возможность вычисления шага с помощью методов второго порядка. В этом случае задача поиска базового алгоритма примет вид

$$b_N(x) = \arg\min_{b \in \mathcal{A}} \sum_{i=1}^{\ell} \left(b(x_i) - \frac{s_i}{h_i} \right)^2, \tag{4.1}$$

где через h_i мы обозначили вторые производные функции потерь:

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

4.1.2 Альтернативный подход

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 $a_{N-1}(x_i)$:

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

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

$$\sum_{i=1}^{\ell} \left(-s_i b(x_i) + \frac{1}{2} h_i b^2(x_i) \right) \tag{4.2}$$

Покажем, что он очень похож на среднеквадратичный из формулы (4.1). Преобразуем его:

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

Видно, что последняя формула совпадает с (4.2) с точностью до коэффициентов $\frac{2}{h_i}$. Можно считать, что в (4.2) они отброшены для большей устойчивости функционала — из-за них может произойти деление на ноль в окрестности оптимума.

4.1.3 Регуляризация

Будем далее работать с функционалом (4.2). Он измеряет лишь ошибку композиции после добавления нового алгоритма, никак при этом не штрафуя за излишнюю сложность модели. Ранее мы решали проблему переобучения путем ограничения глубины деревьев, 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 $\|b\|^2 = \sum_{j=1}^{J} b_j^2$. 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:

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

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

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

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:

$$b_j = \frac{S_i}{H_j + \mu}$$

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

$$H(b) = \frac{1}{2} \sum_{j=1}^{J} \frac{S_j^2}{H_j + \mu} + \lambda J \tag{4.3}$$

4.1.4 Обучение решающего дерева

Мы получили функционал $H(b)$, который для заданной структуры дерева вычисляет минимальную ошибку, 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 так, чтобы оно решало следующую задачу максимизации:

$$Q = H(b_l) + H(b_r) - H(b) - \lambda \to \max$$

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

4.1.5 Заключение

Итак, градиентный бустинг в XGBoost имеет ряд важных особенностей.

1. Базовый алгоритм приближает направление, посчитанное с учетом вторых производных функции потерь.
2. Отклонение направления, построенного базовым алгоритмом, измеряется с помощью модифицированного функционала — из него удалено деление на вторую производную, за счет чего избегаются численные проблемы.
3. Функционал регуляризуется — добавляются штрафы за количество листьев и за норму коэффициентов.
4. При построении дерева используется критерий информативности, зависящий от оптимального вектора сдвига.
5. Критерий останова при обучении дерева также зависит от оптимального сдвига.

Список литературы

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