Семинары по композиционным методам

Евгений Соколов
sokolov.evg@gmail.com

7 марта 2014 г.

1 Композиционные методы машинного обучения

§1.1 Бутстрэппинг

Рассмотрим простой пример построения композиции алгоритмов. Пусть дана конечная выборка X^ℓ и вещественные ответы на ней Y^ℓ. Будем решать задачу линейной регрессии. Сгенерируем подвыборку с помощью *бустрэппинга*. Равномерно возьмем из выборки ℓ объектов с возвращением. Из-за возвращения среди них окажутся повторы; оставив по одной копии каждого объекта, мы получим подвыборку меньшего размера X_1. Повторив процедуру n раз, сгенерируем n подвыборок $X_1, \ldots, X_n$. Обучим по каждой из них линейную функцию регрессии, получив алгоритмы $a_1(x), \ldots, a_n(x)$.

Предположим, что существует истинная функция ответа для всех объектов $y(x)$, а также задано распределение на объектах $p(x)$. В этом случае мы можем записать ошибку каждой функции регрессии

$$\varepsilon_i(x) = a_i(x) - y(x), \qquad i = 1, \ldots, n,$$

и записать матожидание среднеквадратичной ошибки

$$\mathbb{E}_x(a_i(x) - y(x))^2 = \mathbb{E}_x\varepsilon_i^2(x).$$

Средняя ошибка построенных функций регрессии имеет вид

$$E_1 = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_x\varepsilon_i^2(x).$$

Предположим, что ошибки несмещены и некоррелированы:

$$\mathbb{E}_x\varepsilon_i(x) = 0;$$
$$\mathbb{E}_x\varepsilon_i(x)\varepsilon_j(x) = 0, \quad i \neq j.$$

Построим теперь новую функцию регрессии, которая будет усреднять ответы построенных нами функций:

$$a(x) = \frac{1}{n}\sum_{i=1}^{n}a_i(x).$$

Найдем ее среднеквадратичную ошибку:

$$\begin{aligned}
E_n &= \mathbb{E}_x \left(\frac{1}{n} \sum_{i=1}^{n} a_i(x) - y(x) \right)^2 = \\
&= \mathbb{E}_x \left(\frac{1}{n} \sum_{i=1}^{n} \varepsilon_i(x) \right)^2 = \\
&= \frac{1}{n^2} \mathbb{E}_x \left(\sum_{i=1}^{n} \varepsilon_i^2(x) + \underbrace{\sum_{i \neq j} \varepsilon_i(x) \varepsilon_j(x)}_{=0} \right) = \\
&= \frac{1}{n} E_1.
\end{aligned}$$

Таким образом, усреднение ответов позволило уменьшить ошибку в n раз!

Следует отметить, что рассмотренный нами пример не очень применим на практике, поскольку мы сделали предположение о некоррелированности ошибок, что редко выполняется. Если это предположение неверно, то уменьшение ошибки оказывается не таким значительным. Ниже мы рассмотрим более сложные методы объединения алгоритмов в композицию, которые позволяют добиться высокого качества в реальных задачах.

§1.2 Адаптивный бустинг

Алгоритм AdaBoost (Adaptive Boosting) — одна из первых реализаций идеи о том, что путем объединения алгоритмов можно улучшить их качество.

Рассмотрим задачу классификации на два класса: $\mathbb{Y} = \{-1, +1\}$. Нашей задачей является поиск классификатора, минимизирующего число ошибок на обучении:

$$Q(a, X^\ell) = \sum_{i=1}^{\ell} [y_i a(x_i) < 0] \to \min_a . \tag{1.1}$$

Данный функционал является дискретным, что затрудняет его оптимизацию. Чтобы обойти эту проблему, перейдем к его верхней оценке. Заметим, что индикатор можно оценить сверху экспонентой:

$$[z < 0] \leqslant \exp(-z).$$

Применив эту оценку, получим новую задачу:

$$\tilde{Q}(a, X^\ell) = \sum_{i=1}^{\ell} \exp(-y_i a(x_i)) \to \min_a . \tag{1.2}$$

Пусть дано некоторое семейство базовых классификаторов $\mathcal{A}$, каждый алгоритм из которого является отображением из множества объектов в $\{-1, +1\}$. В AdaBoost предлагается строить итоговый классификатор в виде суммы классификаторов из данного семейства:

$$a(x) = \operatorname{sign} \sum_{n=1}^{N} \gamma_n a_n(x).$$

Сумма строится последовательно. Каждый новый классификатор и вес выбираются так, чтобы минимизировать функционал (1.2). После добавления классификатор и его вес уже не меняются, т.е. процедура является жадной.

Пусть мы уже построили сумму из $(N-1)$-го классификатора.

Задача 1.1. *Выведите формулы для $a_N(x)$ и γ_N.*

Решение. Мы хотим получить классификатор вида

$$a(x) = \sum_{n=1}^{N} \gamma_n a_n(x).$$

Все слагаемые с 1-го по $(N-1)$-й фиксированы, и требуется найти лишь последний базовый классификатор и вес при нем. Запишем функционал:

$$\begin{aligned}
\tilde{Q}(a, X^\ell) &= \sum_{i=1}^{\ell} \exp\left(-y_i \sum_{n=1}^{N} \gamma_n a_n(x_i)\right) = \\
&= \sum_{i=1}^{\ell} \underbrace{\exp\left(-y_i \sum_{n=1}^{N-1} \gamma_n a_n(x_i)\right)}_{=w_i} \exp\left(-y_i \gamma_N a_N(x_i)\right) = \\
&= \sum_{i=1}^{\ell} w_i \exp\left(-y_i \gamma_N a_N(x_i)\right).
\end{aligned}$$

Выделим отдельно слагаемые, отвечающие за верно классифицированные объекты:

$$\begin{aligned}
\tilde{Q}(a, X^\ell) &= \sum_{i=1}^{\ell} [y_i a_N(x_i) = 1] w_i \exp\left(-y_i \gamma_N a_N(x_i)\right) + \\
&\sum_{i=1}^{\ell} [y_i a_N(x_i) = -1] w_i \exp\left(-y_i \gamma_N a_N(x_i)\right) = \\
&= e^{-\gamma_N} \sum_{i=1}^{\ell} [y_i a_N(x_i) = 1] w_i + e^{\gamma_N} \sum_{i=1}^{\ell} [y_i a_N(x_i) = -1] w_i = \\
&= (e^{\gamma_N} - e^{-\gamma_N}) \sum_{i=1}^{\ell} [y_i a_N(x_i) = -1] w_i + e^{-\gamma_N} \sum_{i=1}^{\ell} w_i.
\end{aligned}$$

Все дальнейшие рассуждение получатся более стройными и интерпретируемыми, если мы отнормируем веса:

$$\tilde{Q}(a, X^\ell) = \left((e^{\gamma_N} - e^{-\gamma_N}) \sum_{i=1}^{\ell} [y_i a_N(x_i) = -1] \tilde{w}_i + e^{-\gamma_N} \underbrace{\sum_{i=1}^{\ell} \tilde{w}_i}_{=1} \right) \sum_{i=1}^{\ell} w_i,$$

где $\tilde{w}_i = w_i / \sum_i w_i$. Множитель $\sum_i w_i$ из функционала можно вычеркнуть, поскольку он не зависит от $a_N(x)$ и γ_N. Получаем задачу

$$\tilde{Q}(a, X^\ell) = (e^{\gamma_N} - e^{-\gamma_N}) \varepsilon_N + e^{-\gamma_N} \to \min, \tag{1.3}$$

где через ε_N мы обозначили взвешенную ошибку

$$\varepsilon_N = \sum_{i=1}^{\ell} [y_i a_N(x_i) = -1]\tilde{w}_i.$$

Выясним сначала, как искать оптимальный базовый классификатор $a_N(x)$. Заметим, что в функционале (1.3) от $a_N(x)$ зависит лишь ε_N. Таким образом, выбирается классификатор, минимизирующий взвешенную ошибку:

$$a_N(x) = \arg\min \sum_{i=1}^{\ell} [y_i a_N(x_i) = -1]\tilde{w}_i.$$

Перейдем теперь к коэффициенту γ_N. Продифференцируем по нему функционал и приравняем к нулю:

$$\frac{\partial \tilde{Q}}{\partial \gamma_N} = (e^{\gamma_N} + e^{-\gamma_N})\varepsilon_N - e^{-\gamma_N} = 0.$$

Преобразуя, получаем

$$\gamma_N = \frac{1}{N} \log \frac{1 - \varepsilon_N}{\varepsilon_N}.$$

■

Заметим, что вес w_i будет большим, если отступ классификатора $a(x)$ на объекте x_i будет большим и отрицательным. Таким образом, AdaBoost пытается уменьшать ошибку на тех объектах, которые хуже всего получилось классифицировать на предыдущей итерации.