

Семинары по композиционным методам

Евгений Соколов
sokolov.evg@gmail.com

18 февраля 2015 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 *бутстрэппинга*. Равномерно возьмем из выборки ℓ объектов с возвращением. Отметим, что из-за возвращения среди них окажутся повторы. Обозначим новую выборку через X_1 . Повторив процедуру n раз, сгенерируем n подвыборок X_1, \dots, X_n . Обучим по каждой из них линейную функцию регрессии, получив *базовые алгоритмы* $b_1(x), \dots, b_n(x)$.

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

$$\varepsilon_i(x) = b_i(x) - y(x), \quad i = 1, \dots, n,$$

и записать матожидание среднеквадратичной ошибки

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

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:

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

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

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

Найдем ее среднеквадратичную ошибку:

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

Таким образом, усреднение ответов позволило уменьшить средний квадрат ошибки 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:

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

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

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

Применив эту оценку, получим новую задачу:

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

Пусть дано некоторое семейство базовых классификаторов \mathcal{A} , каждый алгоритм из которого является отображением из пространства объектов в $\{-1, +1\}$. В

AdaBoost требуется, чтобы в данном семействе для любого набора весов w_1, \dots, w_ℓ нашелся алгоритм, взвешенная ошибка которого будет меньше $1/2$:

$$\exists a \in \mathcal{A} : \sum_{i=1}^{\ell} w_i [a(x_i) \neq y_i] < \frac{1}{2}. \quad (1.3)$$

Это свойство называется *слабой обучаемостью*.

Предлагается строить итоговый классификатор в виде суммы *базовых классификаторов* (*weak classifiers*) из данного семейства с положительными коэффициентами:

$$a(x) = \text{sign} \sum_{n=1}^N \gamma_n a_n(x), \quad \gamma_n > 0.$$

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

Допустим, мы уже построили $(N - 1)$ базовый алгоритм b_1, \dots, b_{N-1} . Перед построением очередного классификатора в AdaBoost вычисляются веса объектов по формуле

$$w_i = \exp \left(-y_i \sum_{n=1}^{N-1} \gamma_n b_n(x_i) \right),$$

которые затем нормируются:

$$\tilde{w}_i = \frac{\exp \left(-y_i \sum_{n=1}^{N-1} \gamma_n b_n(x_i) \right)}{\sum_{j=1}^{\ell} \exp \left(-y_j \sum_{n=1}^{N-1} \gamma_n b_n(x_j) \right)}.$$

Заметим, что если у объекта большой положительный отступ, то его вес будет близок к нулю. По мере уменьшения отступа вес будет увеличиваться. Новый базовый классификатор выбирается так, чтобы минимизировать взвешенную ошибку на обучающей выборке:

$$b_N(x) = \arg \min_{b_N \in \mathcal{A}} \sum_{i=1}^{\ell} [y_i b_N(x_i) = -1] \tilde{w}_i = \arg \min_{b_N \in \mathcal{A}} \varepsilon_N. \quad (1.4)$$

Видно, что AdaBoost настраивает следующий базовый алгоритм только на те объекты, с которыми не справляется уже построенная композиция. При этом даже если выборка уже верно классифицируется, бустинг будет пытаться увеличивать отступы объектов, что согласуется с принципом максимизации зазора и приводит к уменьшению переобучения.

Коэффициент при новом алгоритме также выражается через взвешенную ошибку:

$$\gamma_N = \frac{1}{2} \log \frac{1 - \varepsilon_N}{\varepsilon_N}. \quad (1.5)$$

Задача 1.1. Обозначим через $\tilde{w}^{(n)}$ веса на n -й итерации. Покажите, что взвешенная ошибка базового классификатора b_N относительно весов со следующего шага $\tilde{w}^{(N+1)}$ равна $1/2$.

Решение.

$$\begin{aligned}
& \sum_{i=1}^{\ell} \tilde{w}_i^{(N+1)} [b_N(x_i) \neq y_i] = \\
&= \sum_{i=1}^{\ell} \frac{w_i^N \exp(-y_i \gamma_N b_N(x_i))}{\sum_{j=1}^N w_j^N \exp(-y_j \gamma_N b_N(x_j))} [b_N(x_i) \neq y_i] = \\
&= \frac{\sum_{i=1}^{\ell} w_i^N \exp(-y_i \gamma_N b_N(x_i)) [b_N(x_i) \neq y_i]}{\sum_{j=1}^N w_j^N \exp(-y_j \gamma_N b_N(x_j))} = \\
&= \frac{e^{\gamma_N} \sum_{i=1}^{\ell} w_i^N [b_N(x_i) \neq y_i]}{e^{\gamma_N} \sum_{j=1}^N w_j^N [b_N(x_j) \neq y_j] + e^{-\gamma_N} \sum_{j=1}^N w_j^N [b_N(x_j) = y_j]} = \\
&= \left\{ \gamma_N = \frac{1}{2} \log \left(\frac{1 - \varepsilon_N}{\varepsilon_N} \right) \right\} = \\
&= \frac{\sqrt{\frac{1 - \varepsilon_N}{\varepsilon_N}} \sum_{i=1}^{\ell} w_i^N [b_N(x_i) \neq y_i]}{\sqrt{\frac{1 - \varepsilon_N}{\varepsilon_N}} \sum_{j=1}^N w_j^N [b_N(x_j) \neq y_j] + \sqrt{\frac{\varepsilon_N}{1 - \varepsilon_N}} \sum_{j=1}^N w_j^N [b_N(x_j) = y_j]} = \\
&= \frac{\sqrt{\frac{1 - \varepsilon_N}{\varepsilon_N}} \varepsilon_N}{\sqrt{\frac{1 - \varepsilon_N}{\varepsilon_N}} \varepsilon_N + \sqrt{\frac{\varepsilon_N}{1 - \varepsilon_N}} (1 - \varepsilon_N)} = \\
&= \frac{\sqrt{\varepsilon_N (1 - \varepsilon_N)}}{\sqrt{\varepsilon_N (1 - \varepsilon_N)} + \sqrt{\varepsilon_N (1 - \varepsilon_N)}} = \\
&= \frac{1}{2}.
\end{aligned}$$

■

Данный результат означает, что распределение на новой итерации подбирается так, что классификатору с предыдущего шага сложнее всего справиться с ним.

Скорость сходимости. Выясним, с какой скоростью уменьшается ошибка композиции на обучающей выборке. Нетрудно показать, что функционал $\tilde{Q}(a, X^\ell)$ представим в виде

$$\tilde{Q}(a, X^\ell) = \left((e^{\gamma_N} - e^{-\gamma_N}) \varepsilon_N + e^{-\gamma_N} \right) \sum_{i=1}^{\ell} w_i.$$

Заметим, что

$$\sum_{i=1}^{\ell} w_i = \sum_{i=1}^{\ell} \exp \left(-y_i \sum_{n=1}^{N-1} \gamma_n b_n(x_i) \right),$$

то есть сумма весов представляет собой ошибку композиции, состоящей из $(N-1)$ -го алгоритма; обозначим ее через \tilde{Q}_{N-1} . Получаем, что ошибка на текущей итерации выражается через ошибку на предыдущей:

$$\tilde{Q}_N = \left((e^{\gamma_N} - e^{-\gamma_N})\varepsilon_N + e^{-\gamma_N} \right) \tilde{Q}_{N-1}.$$

Подставим оптимальный коэффициент (1.5) в данное уравнение:

$$\begin{aligned} \tilde{Q}_N &= \left(\left(\sqrt{\frac{1-\varepsilon_N}{\varepsilon_N}} - \sqrt{\frac{\varepsilon_N}{1-\varepsilon_N}} \right) \varepsilon_N + \sqrt{\frac{\varepsilon_N}{1-\varepsilon_N}} \right) \tilde{Q}_{N-1} = \\ &= \left(\sqrt{\varepsilon_N(1-\varepsilon_N)} + (1-\varepsilon_N)\sqrt{\frac{\varepsilon_N}{1-\varepsilon_N}} \right) \tilde{Q}_{N-1} = \\ &= 2\sqrt{\varepsilon_N(1-\varepsilon_N)}\tilde{Q}_{N-1}. \end{aligned}$$

Ранее мы потребовали, чтобы семейство классификаторов \mathcal{A} обладало свойством слабой обучаемости; из него следует, что мы найдем такой классификатор a_N , что его взвешенная ошибка будет меньше $1/2$: $\varepsilon_N < 1/2$. Предположим, что на каждой итерации мы будем находить классификатор, ошибка которого отделена от $1/2$ константой α :

$$\varepsilon_n \leq \frac{1}{2} - \alpha, \quad \alpha > 0.$$

Учитывая это условие, оценим ошибку композиции на N -й итерации:

$$\begin{aligned} \tilde{Q}_N &= 2\sqrt{\varepsilon_N(1-\varepsilon_N)}\tilde{Q}_{N-1} \leq 2\sqrt{\left(\frac{1}{2}-\alpha\right)\left(\frac{1}{2}+\alpha\right)}\tilde{Q}_{N-1} = 2\sqrt{\frac{1}{4}-\alpha^2}\tilde{Q}_{N-1} = \\ &= \left(\sqrt{1-4\alpha^2}\right)\tilde{Q}_{N-1} \leq (1-4\alpha^2)^{N/2}\tilde{Q}_1. \end{aligned}$$

Таким образом, ошибка убывает со скоростью геометрической прогрессии с коэффициентом $\sqrt{1-4\alpha^2}$.

§1.3 Семейства базовых алгоритмов

Итак, в AdaBoost задача построения базового классификатора $b_N(x)$ сводится к поиску классификатора, минимизирующего взвешенную ошибку

$$\sum_{i=1}^{\ell} \tilde{w}_i [b_N(x_i) \neq y_i] \rightarrow \min_{b_N \in \mathcal{A}}. \quad (1.6)$$

Рассмотрим некоторые семейства базовых алгоритмов \mathcal{A} и способы решения задачи (1.6).

Решающие пни. Одними из самых популярных базовых классификаторов являются *решающие пни*. Это деревья, состоящие из одной вершины:

$$b(x; j, t) = \begin{cases} +1, & x_j < t; \\ -1, & \text{иначе.} \end{cases}$$

Их параметрами являются номер признака j и порог t . Для такого семейства задача (1.6) решается путем полного перебора по всем параметрам j и t (в качестве вариантов для порога t можно рассматривать точки ровно между соседними значениями признака j , а также по точке слева от минимального и справа от максимального значения этого признака).

Решающие деревья. Многие критерии качества, используемые при построении решающих деревьев, выражаются через распределение объектов по классам. Допустим, мы сейчас рассматриваем вершину R_m , в которую попало N_m объектов. Тогда доля объектов k -го класса обозначается как

$$p_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} [y_i = k].$$

Используя данные величины, легко определить, например, энтропийный критерий качества:

$$H(p_m) = - \sum_{k=1}^K p_{mk} \log p_{mk}.$$

В случае, когда объекты имеют веса, можно немного модифицировать определение доли объектов k -го класса:

$$\tilde{p}_{mk} = \sum_{x_i \in R_m} \tilde{w}_i [y_i = k].$$

Легко проверить, что $\sum_{k=1}^K \tilde{p}_{mk} = 1$. Определения всех критериев не меняются, в них лишь используются модифицированные распределения \tilde{p}_{mk} .

Линейные классификаторы. Композиции линейных классификаторов имеют вид

$$a_N(x) = \sum_{n=1}^N \gamma_n \text{sign}\langle w_n, x \rangle.$$

Рассмотрим способы настройки функционала (1.6). Допустим, мы выбрали некоторую верхнюю гладкую оценку для него:

$$\tilde{Q}(w) = \sum_{i=1}^{\ell} \tilde{w}_i \mathcal{L}(y_i, \langle w, x_i \rangle).$$

Если мы будем оптимизировать $\tilde{Q}(w)$ с помощью полного градиентного спуска, то никаких специальных трюков не требуется — веса будут уже учтены в градиенте. Если же мы выберем метод стохастического градиента, то можно учесть веса, выбирая на каждой итерации объект x_i с вероятностью \tilde{w}_i вместо $1/\ell$.

Произвольный метод. Учет весов объектов можно встроить в любой метод обучения для любого семейства базовых классификаторов. Для этого достаточно производить обучение базовых алгоритмов по подвыборке, в которую объект x_i выбирается с вероятностью \tilde{w}_i .

§1.4 Многоклассовый AdaBoost

Рассмотрим обобщение AdaBoost на многоклассовый случай, предложенное в работе [1].

Пусть $X^\ell = \{x_1, \dots, x_\ell\} \subset \mathbb{X}$ — выборка с K возможными ответами: $\mathbb{Y} = \{1, \dots, K\}$. Ответ на i -м объекте обозначим через y_i . Мы воспользуемся K -мерным кодированием ответов: вместо номера класса c возьмем вектор $\vec{y} = (y_1, \dots, y_K)$, в котором c -я координата равна единице, а все остальные равны $-\frac{1}{K-1}$:

$$y_k = \begin{cases} 1, & k = c, \\ -\frac{1}{K-1}, & k \neq c. \end{cases} \quad (1.7)$$

Каждый алгоритм теперь будет представлять собой K -мерный вектор $\vec{a}(x) = (a_1(x), \dots, a_K(x))$, удовлетворяющий требованию $a_1(x) + \dots + a_K(x) = 0$ ¹. В качестве ответа выдается тот класс, чья компонента максимальна:

$$a(x) = \arg \max_{k=1, \dots, K} a_k(x).$$

Рассмотрим следующее обобщение экспоненциальной функции потерь на наш случай:

$$q(\vec{a}, x, \vec{y}) = \exp\left(-\frac{1}{K}(y_1 a_1(x) + \dots + y_K a_K(x))\right) = \exp\left(-\frac{1}{K}\langle \vec{y}, \vec{a}(x) \rangle\right).$$

Данный функционал штрафует за отрицательную корреляцию вектора ответов классификатора с истинным вектором ответов. Выбор именно такого функционала обусловлен тем, что минимум его математического ожидания достигается на алгоритмах $a_i(x)$, выдающих апостериорные вероятности классов (аналогичным свойством обладает классический AdaBoost).

Заметим, что в случае двух классов мы получим привычную нам постановку задачи AdaBoost. Действительно, если $K = 2$, то в любом векторе ответов \vec{y} одна из компонент будет равна -1 , а другая $+1$; для любого классификатора $\vec{a}(x) = (a_1(x), a_2(x))$ будет выполнено $a_1(x) = -a_2(x)$. Функция потерь примет вид

$$\begin{aligned} q(\vec{a}, x, \vec{y}) &= \exp\left(-\frac{1}{2}(y_1 a_1(x) + y_2 a_2(x))\right) = \\ &= \exp\left(-\frac{1}{2}(y_1 a_1(x) + (-y_1)(-a_1(x)))\right) = \\ &= \exp(-y_1 a_1(x)), \end{aligned}$$

¹ Если не ввести это требование, то классификаторы будут определены неоднозначно: прибавление константы ко всем компонентам $a_1(x), \dots, a_K(x)$ никак не изменит сам алгоритм.

то есть перейдет в стандартную экспоненциальную функцию потерь.

Пусть \mathcal{A} — семейство базовых классификаторов. Потребуем, чтобы каждый из них возвращал вектор вида (1.7), то есть чтобы одна из компонент ответа была равна единице, а все остальные $-\frac{1}{K-1}$. Каждый базовый классификатор $\vec{b}(x)$ возвращает вектор, но ему можно однозначно поставить в соответствие многоклассовый классификатор $B : \mathbb{X} \rightarrow \{1, \dots, K\}$ по правилу

$$B(x) = k \iff b_k(x) = 1,$$

и наоборот,

$$b_k(x) = \begin{cases} 1, & \text{если } B(x) = k, \\ -\frac{1}{K-1}, & \text{если } B(x) \neq k. \end{cases}$$

Будем строить композицию вида

$$\vec{a}_N(x) = \sum_{n=1}^N \gamma_n \vec{b}_n(x), \quad b_n \in \mathcal{A}.$$

Композицию будем наращивать последовательно, оптимизируя экспоненциальную функцию потерь:

$$\begin{aligned} Q(\vec{a}_N) &= \sum_{i=1}^{\ell} \exp \left(-\frac{1}{K} \langle \vec{y}_i, \vec{a}_{N-1}(x_i) + \gamma_N \vec{b}_N(x_i) \rangle \right) = \\ &= \sum_{i=1}^{\ell} \underbrace{\exp \left(-\frac{1}{K} \langle \vec{y}_i, \vec{a}_{N-1}(x_i) \rangle \right)}_{=w_i^{(N)}} \exp \left(-\frac{1}{K} \gamma_N \langle \vec{y}_i, \vec{b}_N(x_i) \rangle \right) \end{aligned}$$

Нормируя веса, получаем задачу

$$\sum_{i=1}^{\ell} \tilde{w}_i^{(N)} \exp \left(-\frac{1}{K} \gamma_N \langle \vec{y}_i, \vec{b}_N(x_i) \rangle \right) \rightarrow \min_{\gamma_N, \vec{b}_N}$$

Перейдем к многоклассовому классификатору $B_N(x)$, соответствующему $\vec{b}_N(x)$, и распишем в функционале скалярное произведение $\langle \vec{y}_i, \vec{b}_N(x_i) \rangle$:

$$\begin{aligned}
& \sum_{i=1}^{\ell} \tilde{w}_i^{(N)} \exp \left(-\frac{1}{K} \gamma_N \langle \vec{y}_i, \vec{b}_N(x_i) \rangle \right) = \\
& = \sum_{i: y_i = B_N(x_i)} \tilde{w}_i^{(N)} \exp \left(-\frac{1}{K} \gamma_N \left((K-1) \frac{1}{(K-1)^2} + 1 \right) \right) + \\
& + \sum_{i: y_i \neq B_N(x_i)} \tilde{w}_i^{(N)} \exp \left(-\frac{1}{K} \gamma_N \left((K-2) \frac{1}{(K-1)^2} - \frac{2}{K-1} \right) \right) = \\
& = \sum_{i: y_i = B_N(x_i)} \tilde{w}_i^{(N)} e^{-\frac{\gamma_N}{K-1}} + \sum_{i: y_i \neq B_N(x_i)} \tilde{w}_i^{(N)} e^{\frac{\gamma_N}{(K-1)^2}} = \\
& = e^{-\frac{\gamma_N}{K-1}} \underbrace{\sum_{i=1}^{\ell} \tilde{w}_i^{(N)}}_{=1} + \left(e^{\frac{\gamma_N}{(K-1)^2}} - e^{-\frac{\gamma_N}{K-1}} \right) \sum_{i=1}^{\ell} \tilde{w}_i^{(N)} [y_i \neq B_N(x_i)] = \\
& = e^{-\frac{\gamma_N}{K-1}} + \left(e^{\frac{\gamma_N}{(K-1)^2}} - e^{-\frac{\gamma_N}{K-1}} \right) \varepsilon_N.
\end{aligned}$$

Если $\gamma_N > 0$, то получаем следующую задачу:

$$B_N^* = \arg \min \sum_{i=1}^{\ell} \tilde{w}_i^{(N)} [y_i \neq B_N(x_i)].$$

Таким образом, мы свели построение базового классификатора к минимизации взвешенного числа ошибок на обучающей выборке.

Найдем теперь γ_N . Продифференцируем функционал и приравняем к нулю:

$$\frac{\partial Q}{\partial \gamma_N} = -\frac{1}{K-1} e^{-\frac{\gamma_N}{K-1}} + \left(\frac{1}{(K-1)^2} e^{\frac{\gamma_N}{(K-1)^2}} + \frac{1}{K-1} e^{-\frac{\gamma_N}{K-1}} \right) \varepsilon_N = 0.$$

Решая уравнение, получаем

$$\gamma_N^* = \frac{(K-1)^2}{K} \left(\log \left(\frac{1 - \varepsilon_N}{\varepsilon_N} \right) + \log(K-1) \right).$$

Список литературы

- [1] *Zhu, Ji et al.* (2009). Multi-class AdaBoost. // *Statistics and Its Interface*, 2, p. 177-186.