Теория статистического обучения

Н. К. Животовский
nikita.zhivotovskiy@phystech.edu

10 апреля 2017 г.

Материал находится в стадии разработки, может содержать ошибки и неточности. Автор будет благодарен за любые замечания и предложения, направленные по указанному адресу

1 Схемы сжатия выборок.

Схемы сжатия выборок естественным образом обобщают понятие опорных векторов для произвольных классов классификаторов. Суть этого подхода заключается в том, что для хорошей обобщающей способности достаточно уметь выделять конечное число точек выборки, с помощью которых затем можно восстановить правильные классы всех элементов выборки.

Схема сжатия представляет из себя набор из двух функций. Первая из них (на самом деле это последовательность функций) $\kappa_n : (\mathcal{X} \times \mathcal{Y})^n \to (\mathcal{X} \times \mathcal{Y})^k$ — так называемая функция сжатия. От нее требуется чтобы по выборке она всегда выбирала ее подвыборку: для $n \geq k$ выполнено $\kappa_n((x_i, y_i)_{i=1}^n) \subseteq (x_i, y_i)_{i=1}^n$. Функция восстановления $\rho : (\mathcal{X} \times \mathcal{Y})^k \to \{1, -1\}^{\mathcal{X}}$. Число k называется *размером* схемы сжатия выборки. В этом разделе мы рассматриваем задачу бинарной классификации и индикаторную функцию потерь. Далее мы считаем, что функции ρ и κ_n инвариантны относительно перестановок аргументов.

Теорема 1.1. *Пусть для класса $\mathcal{F}$ существует схема сжатия κ_n, ρ размера k, такая что для любой функции $f \in \mathcal{F}$ и любой выборки $(x_i, f(x_i))_{i=1}^n$ выполнено $\hat{g}(x_i) = f(x_i)$ для всех $i = 1, \ldots, n$, где $\hat{g} = \rho(\kappa_n((x_i, f(x_i))_{i=1}^n))$. Тогда с вероятностью $1 - \delta$,*

$$L(\hat{g}) \leq \frac{k \log(\frac{en}{k})}{n - k} + \frac{\log(\frac{1}{\delta})}{n - k}.$$

Доказательство.

Напомним, что $L(\hat{g}) = P(\hat{g}(X) \neq Y)$. Для начала зафиксируем некоторое k элементное подмножество выборки $(X_i, Y_i)_{i=1}^n$. Без ограничения общности будем считать, что оно состоит из первых элементов выборки. Обозначим его A и обозначим $\hat{g}_A = \rho(A)$. Оценим для $\varepsilon > 0$ вероятность следующего события:

$$\begin{aligned}
&P(L(\hat{g}_A) \geq \varepsilon \cap \forall i = 1, \ldots, n : \hat{g}_A(X_i) = Y_i) \\
&\leq P(L(\hat{g}_A) \geq \varepsilon \cap \forall i = k+1, \ldots, n : \hat{g}_A(X_i) = Y_i) \\
&\leq (1 - \varepsilon)^{n-k} \\
&\leq \exp(-\varepsilon(n - k)).
\end{aligned}$$

Это событие заключается в том, что для некоторого фиксированного подмножества A функция имеет вероятность ошибки большую или равную ε, но при этом правильно восстанавливает всю выборку. Чтобы учесть произвольное k-элементное подмножество (таким образом учтется произвольная функция κ_n) мы оценим

$$\begin{aligned} &P(\exists A \subseteq (X_i, Y_i)_{i=1}^n, \mid A \mid = k : L(\hat{g}_A) \geq \varepsilon \cap \forall i = 1, \ldots, n : \hat{g}_A(X_i) = Y_i) \\ &\leq C_n^k \exp(-\varepsilon(n-k)) \\ &\leq \left(\frac{en}{k}\right)^k \exp(-\varepsilon(n-k)). \end{aligned}$$

Из условия теоремы следует, что это событие не пустое. Обозначая вероятность этого 'плохого' события за δ и решая относительно ε, мы получаем утверждение теоремы. ■

Часто условие точного восстановления всех элементов выборки включается в определение схем сжатия. Однако, на практике требование правильного восстановления всей выборки является очень сильным. Оказывается, что можно легко обобщить последний результат на случай, когда выборка восстанавливается с небольшим числом ошибок.

Теорема 1.2. *Пусть схема сжатия κ_n, ρ размера k гарантированно правильно восстанавливает только объекты из $\kappa_n((X_i, Y_i)_{i=1}^n)$. Тогда с вероятностью $1 - \delta$*

$$L(\hat{g}) \leq \frac{n}{n-k} L_n(\hat{g}) + \sqrt{\frac{k \log(\frac{en}{k}) + \log(\frac{1}{\delta})}{n-k}},$$

где $\hat{g} = \rho(\kappa_n((X_i, Y_i)_{i=1}^n))$

Упр. 1.1. Используя неравенство Хеффдинга, доказать Теорему.

Таким образом, получаем, что существование конечной схемы сжатия размера в бесшумном случае (Теорема 1.1) или существование в общем конечной случае схемы сжатия выборок с дополнительным свойством $\frac{n}{n-k} L_n(\hat{g}) \to 0$ при $n \to \infty$ (Теорема 1.2) влечет обучаемость.

2 Онлайн обучаемость.

Самая простая модель онлайн обучения состоит в том, что последовательно получаются точки x_t из некоторого абстрактного пространства $\mathcal{X}$. На очередном шаге нужно сделать предсказание класса точки x_t. Очередное предсказание будем обозначать символом p_t. После очередного раунда мы получаем правильное предсказание y_t. Затем эти правильные ответы можно использовать для построения p_{t+1}. Наша цель сделать как можно меньше неверных предсказаний на конечной выборке.

Предположим, что существует известный класс функций $\mathcal{F}$, такой что для некоторой $f^* \in \mathcal{F}$ выполнено $y_t = f^*(x_t)$ для всех $t = 1, \ldots, T$. Такой задачу онлайн обучения будем называть *бесшумной* (realizable case)

Опр. 2.1 (Онлайн обучаемость в бесшумном случае). *Класс $\mathcal{F}$ является онлайн обучаемым в бесшумном случае, если существует такой алгоритм онлайн предсказания A, что для любой конечной последовательности точек, для любой функции $f^* \in \mathcal{F}$ число ошибок на ней ограничено величиной $M(\mathcal{F}) < \infty$, зависящей только от $\mathcal{F}$.*

Определим число ошибок алгоритма за T шагов $M_T(\mathcal{F})$ в худшем случае по выбору последовательности точек и функции $f^* \in \mathcal{F}$. Тогда обучаемость можно определить стремление к нулю $\frac{M_T(\mathcal{F})}{T} \to 0$.

Пример 2.1 (Обучаемость конечных классов голосованием большинства). Докажите, что любой конечный класс обучается следующим простым алгоритмом:

$V_1 = \mathcal{F}$,
for $t := 1, 2, \ldots$
получаем x_t,
$p_t = \text{sign}(\text{card}(\{h \in V_t : h(x_t) = +1\}) - \text{card}(\{h \in V_t : h(x_t) = -1\}))$,
в случае равенства мощностей $p_t = +1$.
получаем $y_t = f^*(x_t)$,
обновляем $V_{t+1} = \{f \in V_t : f(x_t) = y_t\}$.

Допускает не более $\log_2(|\ \mathcal{F}\ |)$ ошибок на любой конечной выборке.

Ставится вопрос об обучаемости бесконечных классов. Рассмотрим полное бинарное дерево глубины T с вершинами из множества $\mathcal{X}$. Такое дерево имеет $2^{T+1} - 1$ вершину. Обозначим вершины $x_1, \ldots, x_{2^{T+1}-1}$. Вершина x_1 является корнем этого дерева. Говорят, что класс $\mathcal{F}$ *разбивает* дерево глубины T, если для любого вектора $a \in \{+1, -1\}^T$ найдется функция $f \in \mathcal{F}$ и путь из вершины в один из листьев дерева (задаваемый вершинами $x'_1(a), x'_2(a) \ldots, x'_T(a)$) для которого выполнено $x'_i(a) = a_i$.

Дерево отвечает за возможность среды (выбирающей последовательность точек и $f^* \in F$) заставлять любой наш алгоритм допускать большое число ошибок в худшем случае. Дерево глубины T соответствует T ошибкам.

Опр. 2.2. *Размерностью Литтлстона класса $\mathcal{F}$ называется число $Ldim(\mathcal{F})$ равное наибольшей глубине разбиваемого классом $\mathcal{F}$ дерева. Если такого числа не существует, полагаем $Ldim(\mathcal{F}) = +\infty$.*

Теорема 2.1 (Стандартный оптимальный алгоритм (SOA)). *Следующий алгоритм делает не более $Ldim(\mathcal{F})$ ошибок на любой конечной выборке.*

$V_1 = \mathcal{F}$,
for $t := 1, 2, \ldots$
получаем x_t,
$p_t = \text{sign}(Ldim(\{h \in V_t : h(x_t) = +1\}) - Ldim(\{h \in V_t : h(x_t) = -1\}))$,
в случае равенства размерностей $p_t = +1$.
получаем $y_t = f^*(x_t)$,
обновляем $V_{t+1} = \{f \in V_t : f(x_t) = y_t\}$.

Доказательство.

Идея доказательства состоит в том, что после каждой ошибки размерность Литтлстоуна уменьшается хотя бы на 1. Значит число ошибок не более $Ldim(\mathcal{F})$. Обозначим $V_t^+ = \{h \in V_t : h(x_t) = +1\}$ и $V_t^- = \{h \in V_t : h(x_t) = -1\}$. Предполагая обратное, считаем что на некотором x_t произошла ошибка, но при этом $Ldim(V_{t+1}) = Ldim(V_t)$. Одновременно из выражения для pt легко следует, что $Ldim(V_t^+) = Ldim(V_t^-) = Ldim(V_{t+1}) = Ldim(V_t)$. Получаем противоречие, так как можно построить дерево глубины $Ldim(V_{t+1}) + 1$ для V_t. ∎

Лемма 2.2. *Любой онлайн алгоритм в худшем случае сделает хотя бы* $\min(Ldim(\mathcal{F}), T)$ *ошибок на выборке длины* T.

Упр. 2.1. Докажите лемму.

Очевидным следствием является следующий результат.

Теорема 2.3 (Онлайн обучаемость). *Класс* $\mathcal{F}$ *является онлайн обучаемым в бесшумном случае тогда и только тогда, когда* $Ldim(\mathcal{F}) < \infty$.

Пример 2.2. Для семейства односторонних пороговых классификаторов $Ldim(\mathcal{F}) = \infty$. Заметим, что размерность Вапника–Червоненкиса данного класса равна 1.

Пример 2.3. Легко видеть, что размерность Вапника-Червоненкиса не превосходит размерность Литтлстоуна.

Пример 2.4. Для конечного $\mathcal{F}$ выполнено $Ldim(\mathcal{F}) \leq \lfloor \log_2(|\mathcal{F}|) \rfloor$.

3 Онлайн алгоритмы и схемы сжатия

Опр. 3.1. *Онлайн алгоритм называется консервативным, если предсказание* p_t *на шаге* t *зависит только от тех точек, где ранее алгоритм допустил ошибку. Другими словами, консервативные алгоритмы не учитывают все те пары* (x_i, y_i) *для* $i < t$ *на которых не произошла ошибка.*

Теорема 3.1. *Для любого класса* $\mathcal{F}$ *с* $Ldim(\mathcal{F}) < \infty$ *существует схема сжатия в* $Ldim(\mathcal{F})$ *точек. В частности, для любого конечного класса существует схема сжатия в не более чем* $\lfloor \log_2(|\mathcal{F}|) \rfloor$ *точек.*

Доказательство.

Будем считать, что на всем множестве $\mathcal{X}$ введено отношение порядка. Переделаем алгоритм SOA так чтобы он был консервативным. Для этого будем производить обновление множества V_{t+1}, только если на текущем шаге произошла ошибка. Легко видеть, что верхняя оценка на ошибку в таком случае не изменится: на любой выборке алгоритм сделает не более чем $Ldim(\mathcal{F})$ ошибок.

Сжатие. Функция сжатия в данном случае будет работать следующим образом: получив выборку $(x_i, y_i)_i^n$ упорядочим пары согласно порядку на множестве $\mathcal{X}$ и применим последовательно консервативный алгоритм SOA к упорядоченной выборке. В качестве подвыборки размером не более $Ldim(\mathcal{F})$ выбираем множество

всех тех точек, на которых SOA допустил ошибку. Обозначим эту подвыборку $C = (x_{i_j}, y_{i_j})_{j=1}^{k}$.

Восстановление. Если новая точка x находится среди точек во множестве, полученном на этапе сжатия, то есть $x = x_{i_m}$ для некоторого m, то классифицируем ее как y_{i_m}. Иначе, добавляем точку x к множеству $(x_{i_j})_{j=1}^{k}$, упорядочиваем его согласно порядку на множестве $\mathcal{X}$ и запускаем модифицированный SOA на всех этих упорядоченных точках. Точку x классифицируем согласно тому, как ее классифицирует SOA. Элементарно убеждаемся, что все точки исходной выборки будут восстановлены правильно. ■

Список литературы

[1] *Shalev-Shwartz S., Ben-David S.* Understanding Machine Learning: From Theory to Algorithms // Cambridge University Press, 2014

[2] *Floyd S., Warmuth M.* Sample Compression, Learnability, and the Vapnik–Chervonenkis Dimension // Machine Learning, 1995.