Exercise 5. Task 1: Весовые обновления AdaBoost

Daniil Koveh

2025-11-06

Содержание

1. Теория

Цель задачи — аккуратно вывести формулы обновления весов и коэффициента
$beta_m$ в AdaBoost. Алгоритм на m-м шаге минимизирует экспоненциальную потерю \[L_m(\beta, G) = \sum_{i=1}^N w_i^{(m)} \exp\{-\beta \, y_i \, G(x_i)\}, \] где y_i
in
$-1, +1$ — знаковые ответы, $G(x)$ — базовый классификатор (выдаёт
$-1, +1$), а веса \[w_i^{(m)} = \exp\left(- y_i f_{m-1}(x_i)\right) \] переносят в новую итерацию всю информацию о предыдущих ошибках.

Нужно показать, что оптимизация распадается на два простых шага:

1. Сначала при фиксированном
   $beta > 0$ выбираем классификатор G_m, минимизируя взвешенную долю ошибок.
2. Затем подставляем найденный G_m и аналитически вычисляем
   $beta_m$, получая выражение через найденную ошибку
   $mathrmerr_m$.

2. Жизненный пример

Представим, что мы классифицируем письма как «спам/не спам». Бустинг повышает веса именно тех писем, на которых предыдущий ансамбль промахнулся, и в новой итерации «просит» базовый классификатор сосредоточиться на сложных случаях. Если базовый классификатор снова ошибается, его коэффициент
$beta_m$ становится меньшим, а значит, вклад промахивающегося алгоритма в финальное решение будет ограничен. Именно поэтому шаг с минимизацией взвешенной ошибки жизненно важен: он говорит, какие письма сейчас критичны, а формула для
$beta_m$ превращает процент ошибок в число, управляющее вкладом в итоговый логит $f_m(x)$.

3. Академическое решение

3.1. Шаг 1: Минимизация по G

Берём произвольное фиксированное значение
$beta > 0$ и рассматриваем \[L_m(\beta, G) = \sum_{i=1}^N w_i^{(m)} \exp\{-\beta \, y_i \, G(x_i)\}. \] Здесь $G(x_i)$
in
$-1, +1$. Введём индикатор ошибки $I_i =$
$mathbb1$
$y_ineqG(x_i)$. Тогда $y_iG(x_i) = +1$ при верной классификации и -1 при ошибке. Значит, \[\exp\{-\beta \, y_i \, G(x_i)\} = \begin{cases} e^{-\beta}, & y_i = G(x_i);\\ e^{\beta}, & y_i \neq G(x_i). \end{cases} \] Перепишем сумму: \[L_m(\beta, G) = e^{-\beta} \sum_{i=1}^N w_i^{(m)} (1 - I_i) + e^{\beta} \sum_{i=1}^N w_i^{(m)} I_i. \] Так как e^{-beta} и e^{beta} — положительные константы, выбор G влияет только на выражение
$sum_iw_i^{(m)}I_i$, то есть на взвешенную ошибку. Следовательно, \[G_m = \arg\min_G \sum_{i=1}^N w_i^{(m)} I\{y_i \neq G(x_i)\}. \] Это и есть классификатор с минимальной взвешенной ошибкой, что совпадает с первым пунктом из условия.

3.2. Шаг 2: Вычисление $beta_m$

Обозначим взвешенную ошибку аддитивно: \[\mathrm{err}*m = \frac{\sum*{i=1}^N w_i^{(m)} I\{y_i \neq G_m(x_i)\}}{\sum_{i=1}^N w_i^{(m)}}. \] Отдельно выделим лучшую взвешенную точность: \[1 - \mathrm{err}*m = \frac{\sum*{i=1}^N w_i^{(m)} I\{y_i = G_m(x_i)\}}{\sum_{i=1}^N w_i^{(m)}}. \] Подставляем найденный G_m обратно в критерий: \[L_m(\beta, G_m) = e^{-\beta} \sum_{i: y_i = G_m(x_i)} w_i^{(m)} + e^{\beta} \sum_{i: y_i \neq G_m(x_i)} w_i^{(m)}. \] Вынесем общий знаменатель: \[L_m(\beta, G_m) = \sum_{i=1}^N w_i^{(m)} \Big[e^{-\beta} (1 - I_i) + e^{\beta} I_i \Big] = S_m \left[e^{-\beta} (1 - \mathrm{err}_m) + e^{\beta} \mathrm{err}_m \right], \] где $S_m =$
$sum_{i=1}^Nw_i^{(m)}$ — сумма весов (не зависит от
$beta$). Минимизируем правую часть по
$beta$: \[g(\beta) = e^{-\beta} (1 - \mathrm{err}_m) + e^{\beta} \mathrm{err}_m. \] Берём производную и приравниваем её нулю: \[g'(\beta) = -e^{-\beta} (1 - \mathrm{err}_m) + e^{\beta} \mathrm{err}_m = 0. \] Отсюда получаем \[e^{2\beta} = \frac{1 - \mathrm{err}_m}{\mathrm{err}_m} \quad \Rightarrow \quad \beta_m = \tfrac{1}{2} \log \frac{1 - \mathrm{err}_m}{\mathrm{err}_m}, \] что совпадает с формулой в условии. Минимум единственный, потому что $g($
$beta)$ — выпуклая функция.

3.3. Итог

- Минимизация экспоненциальной потери сводится к выбору базового классификатора, который даёт минимальную взвешенную ошибку (учитывая веса текущей итерации).
- После выбора G_m оптимальный коэффициент слагаемого в ансамбле получаем в замкнутой форме через

  $mathrmerr_m$.
- Таким образом, AdaBoost — это последовательность двух простых шагов: «найди лучший классификатор под текущие веса» и «преобразуй его ошибку в коэффициент
  $beta_m$».

3.4. Что запомнить

- Экспоненциальная потеря переводит ошибки классификации в экспоненты e^{pmbeta}: это и заставляет веса «взлетать» на трудных объектах.
- Формула
  $beta_m =$

$frac12$
log
$frac1 - mathrmerr_m mathrmerr_m$ делает алгоритм устойчивым: чем больше ошибок, тем меньше вклад очередного классификатора.

- Ключ к пониманию AdaBoost — видеть, как смена весов превращает простую жадную итерацию в ансамбль, который концентрируется на сложных точках и не переоценивает слабые классификаторы.