УДК 004.8

МАЙТАК Р. В., студент гр. ИИм-231 (КузГТУ)
Научный руководитель БАУМГАРТЭН М.И., к.ф.-м.н., доцент (КузГТУ)
г. Кемерово

АЛГОРИТМ ADABOOST КАК ИНСТРУМЕНТ СОВРЕМЕННОГО ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

За последние десятилетия в области информационных технологий сформировался мощный инструмент прогноза – *машинное обучение*, в основе которого лежит использование больших массивов данных. Своё название данный инструмент получил благодаря своей возможности обучения компьютерных технологий без явного вмешательства человека и его создания им нового программного кода. В качестве основы машинного обучения выступают уже ранее созданные алгоритмы анализа данных, с помощью которых можно смоделировать или спрогнозировать какой-либо исход или результат для определённо заданных критериев.

Дословно с английского языка понятие *бустинг* (англ. boosting) означает увеличение. Соответственно, применительно к машинному обучению бустинг будет представлять собой алгоритм по увеличению эффективности процессов анализа и прогнозирования других алгоритмов. Принцип действия бустинга довольно прост: алгоритмы с низкой эффективностью, которые ещё называются «слабыми алгоритмами», на каждой новой итерации объединяются в комбинации. В результате таких объединений каждая вновь сформированная комбинация алгоритмов обучается на основе полученных данных об ошибках на предыдущих итерациях, в результате чего повышает свою эффективность [1]. Одним из первых видов бустингов является созданный ещё в 1995 году *адаптивный бустинг* или *AdaBoost*. Как и другие виды бустинга, *Adaptive Boosting* представляет собой алгоритм, суть которого заключается в построении композиции из базовых алгоритмов с целью повышения их эффективности [2]. Отличительная сторона AdaBoost заключается в распределении в рамках этого алгоритма весовых коэффициентов, которые изначально одинаковы в присвоении каждому объекту данных. Далее, на следующих итерациях, коэффициенты автоматически корректируются, а неверно классифицированным элементам присваивается уже больший вес. Этот процесс аналогичным образом повторяется до момента, пока не будет устранена критическая ошибка или пока разница между прогнозируемыми и фактическими значениями не достигнет интервала допустимых значений.

Основными *достоинствами* алгоритма AdaBoost являются: 1) Простота принципа работы и реализации алгоритма; 2) Гибкость и универсальность, так как AdaBoost можно комбинировать практически с любим алгоритмом машинного обучения; кроме того, алгоритм способен работать как с числовым, так и текстовым набором данных; 3) Обширная область применения; 4) Минимальная потребность в ресурсных и временных затратах; 5) Способность определять объекты, которые могут являться шумовыми помехами.

К главным *недостаткам* алгоритма AdaBoost относятся следующие факторы: 1) Обучающие выборки должны обладать обширной размерностью; 2) Возможность начала процесса переобучения существует лишь в случае, если шумовые помехи не были распознаны или вовремя устранены; 3) Вероятность построения обширных композиций из базовых алгоритмов может вызвать потребность в наличии большого объема свободной памяти [3].

Несмотря на то, что AdaBoost является одним из первых созданных бустингов, благодаря своей простоте и обширной области применения он до сих пор остаётся популярным и востребованным у ИТ-специалистов во всем мире. Данный алгоритм можно использовать для любых данных, которые можно классифицировать на несколько типов при помощи заданных критериев. Это может быть система распознавания лиц, система учета студентов вуза или система по определению предпочтений пользователей онлайн-кинотеатров или музыкальных сервисов. В качестве простого примера может выступать такой критерий как «да/нет»: если к набору данных можно применить утверждение «да» или «нет», то к таким данным можно будет применить алгоритм AdaBoost.

Рассмотрим подробнее особенности математического аппарата алгоритма AdaBoost. Наилучшим образом он взаимодействует со слабыми алгоритмами, поэтому именно вместе с ними и целесообразно использовать AdaBoost, если перед нами стоит цель решения задач классификации. В качестве слабых алгоритмов, взаимодействующие с AdaBoost, обычно используют алгоритмы деревьев решений. По этой причине мы разберем математический аппарат работы алгоритма AdaBoost именно для решения задач классификации.

Прежде всего, поскольку AdaBoost является алгоритмом бустинга, для возможности его применения должны выполняться следующие правила:

$$a(x) = \sum_{t=1}^{T} a_t * b_t(x); \quad (1),$$

где, a_t – весовой коэффициент (число должно быть неотрицательным), $b_t(x)$ – базовые алгоритмы.

Для задач классификации вещественным значением является величина отступа (M), которая рассчитывается следующим образом:

$$M = \langle \omega, x \rangle = \omega^T * x; \quad (2)$$

Таким образом, для задач бинарной классификации ($y = \{-1, +1\}$, исходя из особенностей алгоритма AdaBoost, описанных выше, уравнение примет следующий вид:

$$Q_T = \sum_{i=1}^{i} [M_i < 0] \leq \sum_{i=1}^{i} \exp(-M_i) = \widetilde{Q_T}\ ; \quad (3)$$

Поскольку задачей AdaBoost является поиск лучшей композиции алгоритмов $b_t(x)$ и перераспределение весовых коэффициентов, то при фиксации данных предыдущих композиций $(b_1(x), \dots, b_T(x)$ и $a_1, \dots, a_T)$ можно расписать уравнение следующим образом:

$$Q_T \le \widetilde{Q_T} = \sum_{i=1}^{i} \exp\left(-y_i * \sum_{t=1}^{T-i} a_t b_t(x_i)\right) * \exp\left(-y_i * a_T * b_T(x_i)\right); \quad (4)$$

В рамках этого выражения были отдельно обособлены весовые коэффициента и выделен множитель:

$$\omega_i = exp(-y_i * \sum_{t=1}^{T-i} a_t b_t(x_i)); \quad (5)$$

Теперь можно сделать вывод, что эти коэффициенты рассчитываются для каждого объекта выборки, а множитель при этом увеличивается для объектов, которые плохо поддаются классификации. Результатом такого увеличения станет больший учет алгоритма при обучении данных объектов. При поиске каждой новой композиции алгоритмов весовые коэффициенты нормируются по следующей формуле:

$$\widetilde{\omega_\iota} = \frac{\omega_i}{\sum_{j=1}^{i} \omega_j}, i = 1,,,l; \quad (6)$$

Это делается для того, чтобы в сумме нормированные весовые коэффициенты были равны 1, что позволит принимать им адекватные значения:

$$\sum_{i-1}^{i} \widetilde{\omega_\iota} = 1; \quad (7)$$

В результате всех этих операций теперь можно подсчитать долю неверно классифицированных объектов текущего алгоритма:

$$N(b) = \sum_{i=1}^{i} \widetilde{\omega_\iota} * [b(x_i) \neq y_i]; \quad (8)$$

Тогда доля верных классификаций для текущего алгоритма рассчитывается как:

$$P(b) = \sum_{i=1}^{i} \widetilde{\omega_\iota} * [b(x_i) \neq y_i] = 1 - N(b); \quad (9)$$

Наилучший вариант композиции алгоритмов определяется на основе минимизации доли неверно классифицированных объектов:

$$b_t = \arg\min_b N(b); \quad (10)$$

Оптимальное значения весовых коэффициентов для этих алгоритмов, в свою очередь, рассчитывается по следующей формуле:

$$a_t = 1/2 \ln \frac{1 - N(b_t)}{N(b_t)}; \quad (11)$$

Именно так выглядит в математическом виде алгоритм AdaBoost. На каждой новой итерации (от 1 до T) происходит обучение текущего алгоритма при помощи объектов обучающей выборки, каждый из которых обладает весом [2]. Работа алгоритма AdaBoost предполагает наличие входных данных, т.е. последовательности определённых действий, выполнение которых позволит получить искомые выходные данные. В качестве входных данных выступают обучающая выборка X^l и количество алгоритмов, сформированных в композиции, — T. На выходе алгоритма должны быть определены: набор базовых алгоритмов $(b_1(x), \dots, b_T(x))$ и весовые коэффициенты $(a_1, \dots, a_T)$.

Алгоритм AdaBoost подразумевает следующий набор действий:

1) Определение начальных значений весовых коэффициентов объектов. Согласно принципу алгоритма веса определяются как равные между собой.

$$\omega_i = \frac{1}{l}, i = 1, \dots, l; \quad (12)$$

2) На втором этапе запускается цикл итераций: $t = 1, \dots, T$.

3) Теперь на каждой итерации начинается поиск наилучшего алгоритма, то есть такого, который определяет минимум доли неверной классификации заданной нами выборки по уже ранее представленной формуле:

$$b_t = \arg\min_b N(b); \quad (13)$$

4) Для найденного алгоритма необходимо определить оптимальный весовой коэффициент:

$$a_t = 1/2 \ln \frac{1 - N(b_t)}{N(b_t)}; \quad (14)$$

5) После этого происходит пересчёт весовых коэффициентов всех объектов:

$$\omega_i = \omega_i * \exp\left(-a_t y_i b_t(x_i)\right), i = 1, \dots, l; \quad (15)$$

6) На шестом этапе необходимо нормировать обновлённые весовые коэффициенты:

$$\omega_{sion} = \sum_{j=1}^{l} \omega_j; \ \omega_i = \omega_i / \omega_{sum}; \ i = 1, \dots, l; \quad (16)$$

После 6-го шага цикл завершается, после чего происходит возврат к шагу 2, что означает начало новой итерации. Все итерации выполняются аналогично до того момента, пока не будет получена «правильная» комбинация базовых алгоритмов и не будет достигнуто исключение ошибки.

Таким образом, простота и универсальность алгоритма AdaBoost позволяют реализовать его на базе практически любого языка программирования. Среди наиболее популярных можно выделить Java, Scala, R и Python. В частности, у Python существует специальная библиотека для машинного обучения – Scikit-learn, широкие возможности которой позволяют реализовать в том числе и алгоритм AdaBoost [1, 3].

Список литературы:

1. Жерон, О. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow / О. Жерон. – М.: O’Reilly Media. 2018. – 662 с.
2. Колец, Д. Классические задачи Computer Science на языке Python / Д. Колец. – СПб.: Питер. 2020 – 256 с.
3. Кормен, Т., Лейзерсон,Ч. Алгоритмы: построение и анализ / Т. Кормен. – М.: ООО И.Д. Вильямс. 2013. – 1328 с.