Бустинг [M.192]

По сравнению с бэггингом бустинг (boosting) несколько более сложная процедура, но во многих случаях работает эффективнее. Как и бэггинг, бустинг использует неустойчивость алгоритмов обучения и начинает создание ансамбля на основе единственного исходного множества. Но если в бэггинге модели строятся параллельно и независимо друг от друга, то в бустинге каждая новая модель строится на основе результатов ранее построенных моделей, то есть модели создаются последовательно. Бустинг создает новые модели таким образом, чтобы они дополняли ранее построенные, выполняли ту работу, которую другие модели сделать не смогли на предыдущих шагах. И наконец, последнее отличие бустинга от бэггинга заключается в том, что всем построенным моделям в зависимости от их точности присваиваются веса (бэггинг, напомним, использует взвешенное голосование или усреднение).

В настоящее время разработано большое количество различных модификаций бустинга. Рассмотрим один из наиболее популярных алгоритмов — **AdaBoost.M1**, который предназначен для решения задач классификации. Будем использовать обозначения, ранее введенные для бэггинга.

Вместо извлечения выборок из исходного множества данных бустинг в качестве «возмущающего» фактора применяет взвешивание примеров. Вес каждого примера устанавливается в соответствии с его влиянием на обучение классификатора. На каждой итерации вектор весов подстраивается таким образом, чтобы отражать эффективность данного классификатора. В результате вес неправильно классифицированных примеров увеличивается. Итоговый классификатор также агрегирует обученные классификаторы путем голосования, но теперь голос классификатора является функцией его точности.

Обозначим вес примера x на итерации t как w_k^1, при этом вес на первой итерации задается как $w_x^1 = 1/N$ для каждого x (N — число примеров). На каждой итерации $t = 1, 2, \ldots, T$ классификатор C_t конструируется из данных примеров в соответствии с распределением их весов wt (то есть как будто вес w_k^1 отражает вероятность появления примера x). Ошибка ε_t классификатора t также измеряется относительно весов и представляет собой сумму весов примеров, которые были классифицированы неправильно. Когда ε_t становится больше $0{,}5$, итерации прекращаются, последний классификатор удаляется и T изменяется на $t-1$. Наоборот, если $\varepsilon_t = 0$ (классификатор C_t правильно классифицировал все примеры), итерации останавливаются, и $T = t$. В остальных случаях вектор весов $w_t + 1$ для следующей итерации генерируется путем умножения текущих весов примеров, которые были правильно распознаны классификатором C_t, на коэффициент $\beta_t = \varepsilon_t/(1-\varepsilon_t)$, а затем нормируется так, чтобы $\sum_x w_x^{t+1} = 1$. Тогда

$$w_{t+1} = w_t \cdot \frac{\varepsilon^t}{\left(1-\varepsilon^t\right)}. \qquad (1)$$

Итоговый классификатор C^* получается путем суммирования голосов классификаторов C^1, $C^2, \ldots, C^T$, где голос классификатора C^t определяется как $\log(1/\beta^t)$ единиц.

Было показано, что если ошибка отдельного классификатора ε_t всегда меньше $0{,}5$, то значение ошибки итогового классификатора C^* экспоненциально стремится к 0 с увеличением числа итераций t. Последовательность слабых классификаторов C^1, $C^2, \ldots, C^T$ может быть усилена до классификатора C^*, который обычно получается более точным, чем отдельные классификаторы. Конечно, при этом нельзя гарантировать высокую обобщающую способность C^*.

Процедура бустинга включает следующие шаги.

1. Для всех примеров исходного множества данных устанавливаются равные начальные веса w_0.
2. На основе взвешенного набора примеров строится классификатор C_t, вычисляется и запоминается выходная ошибка данного классификатора ε_t.
3. Рассчитывается коррекция весов примеров обучающего множества, и веса корректируются по формуле (1).
4. Если ошибка $\varepsilon_t = 0$ или $\varepsilon_t \geq 0{,}5$, то классификатор C_t удаляется и процедура бустинга останавливается.
5. В противном случае осуществляется переход на шаг 2 и начинается следующая итерация.

Процедура поясняется с помощью блок-схемы на рисунке 1.

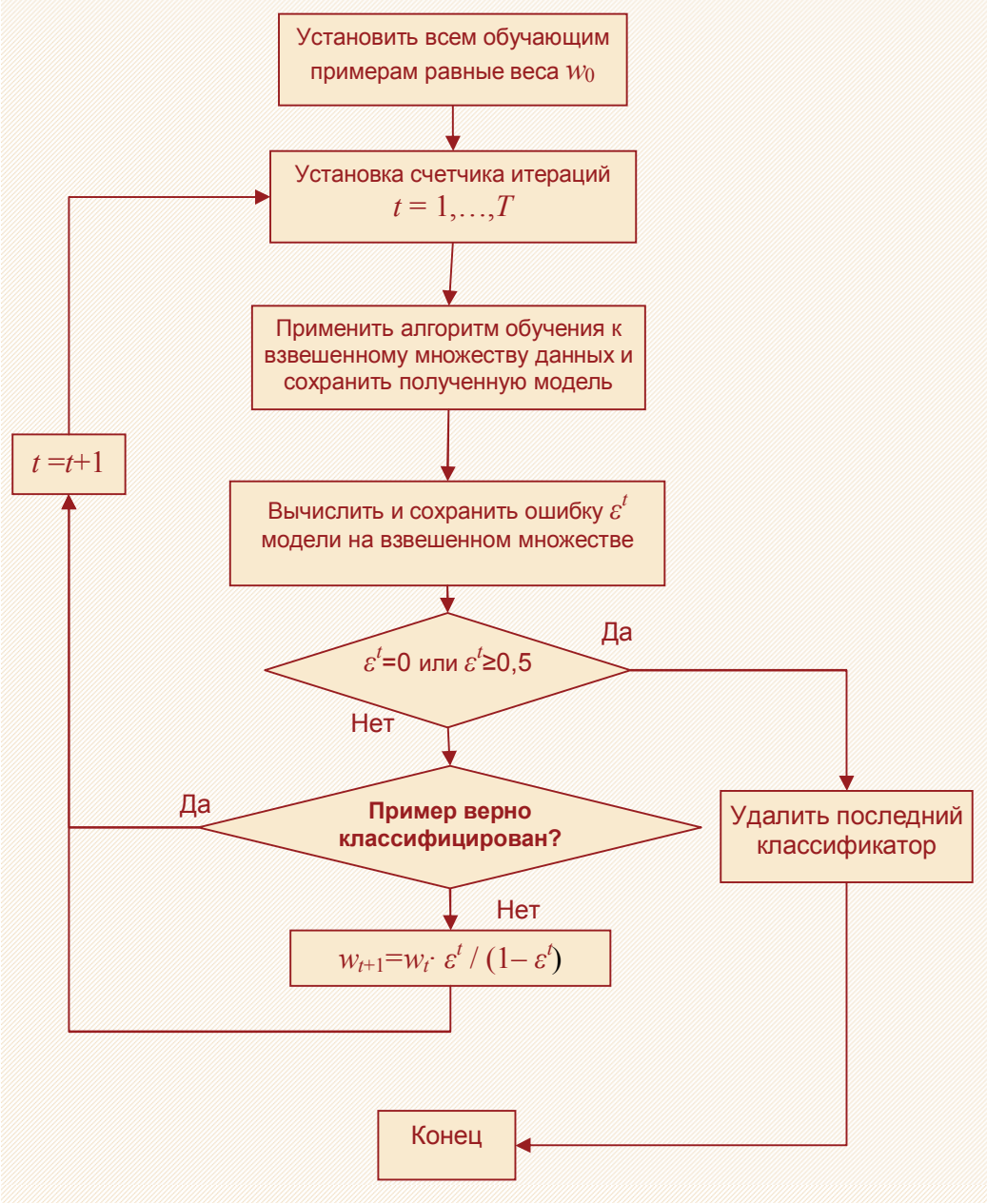

Рисунок 1 – Блок-схема алгоритма AdaBoost.M1

Таким образом, параметрами, настраиваемыми на каждой итерации, являются веса примеров. При этом чем больше раз пример был неправильно распознан предыдущими моделями, тем выше его вес. Вес можно рассматривать как вероятность попадания примера на следующую итерацию.

В построении первой модели будут участвовать все примеры, в построении второй — только те, которые были неправильно распознаны первой, в построении третьей — примеры, неправильно распознанные двумя предыдущими моделями, и т. д. Модели будут дополнять друг друга — работать с теми примерами, от которых «отказались» другие модели. Из теории машинного обучения известно, что именно трудные для распознавания примеры толкают процесс обучения вперед, заставляя модель классифицировать их, в то время как простые примеры распознаются быстро и для дальнейшей тренировки бесполезны.

Веса примеров увеличиваются или уменьшаются в зависимости от выхода каждого нового классификатора. В результате некоторые трудные примеры могут стать еще труднее, а легкие — еще легче, и наоборот. После каждой итерации веса отражают, насколько часто тот или иной пример классифицировался неправильно.

Генерация классификаторов

Теперь рассмотрим, как процедура бустинга генерирует последовательность классификаторов.

Выход ансамбля комбинируется процедурой взвешенного голосования. При определении весов классификатор, который хорошо работает на обучающей выборке, должен получить больший вес. Классификатор, который работает хуже (ошибка ближе к $0{,}5$), получает более низкий вес. Формальная запись имеет вид:

$$W^t = -\log\left(\frac{1}{1-\varepsilon^t}\right).$$

В данном случае как W_t мы обозначили вес классификатора, построенного на итерации t, который будет использоваться при взвешенном голосовании. Вес — это положительное число в интервале от нуля до бесконечности. Кроме того, формула объясняет, почему классификаторы с нулевой ошибкой должны быть удалены: при $\varepsilon_t = 0$ выражение для весов окажется неопределенным из-за того, что аргумент логарифма равен 1. При комбинировании выходов ансамбля веса всех классификаторов, которые «голосуют» за определенный класс (то есть имеют этот класс на выходе), суммируются, и выбирается класс, набравший наибольшую сумму голосов. Иными словами, определяющую роль играют не столько выходы моделей, сколько их веса.

Последовательность генерации моделей в бустинге и их участие в комбинировании результатов методом голосования представлены на рисунке 2.

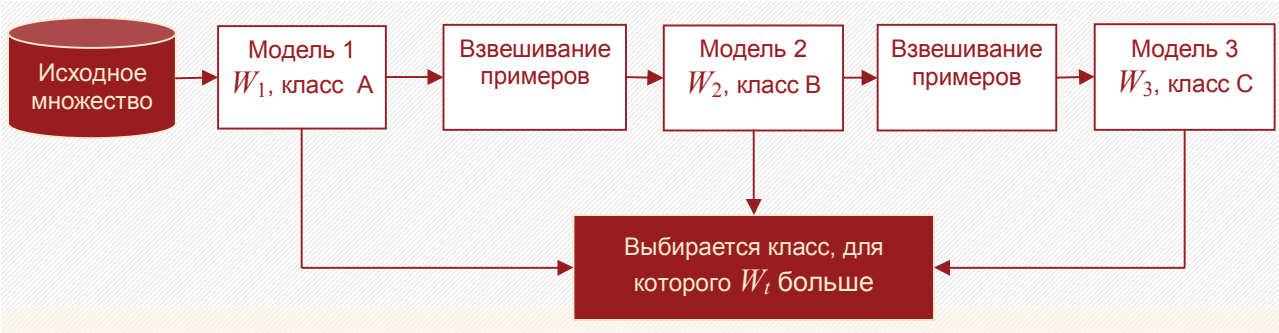

Рисунок 2 – Схема бустинга

Недостатки бустинга

Основным недостатком бустинга является то, что примеры с низкими весами не попадают на следующую итерацию, поэтому часть информации, полезной для обучения, может быть потеряна. Кроме того, в процессе бустинга из участия в обучении постепенно исключаются самые простые примеры, что приводит к так называемому вырождению обучающих выборок, на основе которых строятся последние модели. В первую очередь распознаются наиболее типичные примеры, хорошо отражающие предметную область. Образцы, трудные для распознавания, часто являются нетипичными, аномальными и пр. Обучение классификаторов на поздних итерациях может почти полностью вестись на таких примерах, что, в конечном итоге, приводит к переобучению. В силу вышесказанного бустинг более склонен к переобучению, чем бэггинг.

Чтобы избежать переобучения, количество итераций T должно быть настолько малым, насколько это возможно без потери точности. Алгоритм останавливается, когда на некотором C_t ошибка приближается к 0, невзирая на то что C^* может распознать все обучающие примеры, а C_t — нет. Дальнейшие итерации в данной ситуации, скорее всего, не приведут к приросту точности. Вместо этого увеличится сложность C^*, но его эффективность не повысится.

Наконец, общей проблемой бустинга и бэггинга является низкая прозрачность модели для аналитика и сложность интерпретации результатов, что в целом характерно для всех ансамблей моделей.