Лекции по алгоритмическим композициям
(черновик)

К. В. Воронцов

19 мая 2005 г.

Содержание

1 Композиции алгоритмов

При решении сложных задач классификации, регрессии и прогнозирования часто оказывается, что ни один из имеющихся эвристических алгоритмов не даёт желаемого качества обучения. В таких случаях имеет смысл строить композицию алгоритмов, в которой недостатки одних алгоритмов компенсировались бы достоинствами других. При определённых условиях качество композиции может оказаться заметно лучше, чем у отдельных составляющих её алгоритмов.

Напомним основные обозначения. Рассматривается задача обучения по прецедентам $\langle X, Y, y^*, X^\ell \rangle$, где

X — пространство объектов;

Y — множество ответов;

$y^* : X \to Y$ — неизвестная целевая зависимость;

$X^\ell = (x_1, \ldots, x_\ell)$ — обучающая выборка;

$Y^\ell = (y_1, \ldots, y_\ell)$ — вектор ответов на обучающих объектах, $y_i = y^*(x_i)$.

Задача заключается в том, чтобы построить алгоритм $a : X \to Y$, аппроксимирующий целевую зависимость.

Наиболее общее определение алгоритмической композиции даётся в алгебраическом подходе Журавлёва [5]. Вводится множество R, называемое *пространством оценок*, и рассматриваются алгоритмы, имеющие вид суперпозиции $a(x) = C\big(b(x)\big)$, где функция $b\colon X \to R$ называется *алгоритмическим оператором*, а функция $C\colon R \to Y$ — *решающим правилом*. Данное предположение о структуре алгоритмов не является хоть сколько-нибудь стесняющим ограничением. Во-первых, многие алгоритмы классификации именно так и устроены: сначала вычисляется оценка принадлежности объекта x заданному классу, затем решающее правило $C(b)$ огрубляет эту оценку, переводя её в номер класса. В частности, значение $b(x)$ может быть расстоянием от объекта x до разделяющей поверхности, либо оценкой вероятности того, что объект x принадлежит выделенному классу, либо даже вектором оценок принадлежности объекта x каждому из классов. Отметим, что строение множества R пока никак не фиксируется. Во-вторых, всегда остаётся возможность отказаться от использования решающего правила, положив $R = Y$ и взяв тождественное отображение $C(b) \equiv b$. В регрессионных задачах обычно так и поступают.

Опр. 1.1 *Алгоритмической композицией, составленной из алгоритмических операторов $b_t\colon X \to R$, $t = 1, \ldots, T$, корректирующей операции $F\colon R^T \to R$ и решающего правила $C\colon R \to Y$ называется алгоритм $a\colon X \to Y$ вида*

$$a(x) = C\big(F(b_1(x), \ldots, b_T(x))\big), \quad x \in X. \tag{1.1}$$

Использование пространства оценок R значительно расширяет множество допустимых корректирующих операций, что особенно актуально для задач классификации. Когда множество Y конечно, семейства «хороших» корректирующих операций вида $F\colon Y^T \to Y$ оказываются слишком бедными, практически не оставляя возможностей для оптимизации [5]. Положение меняется, если корректирующие операции применять не к самим алгоритмам, а к их алгоритмическим операторам. В этом случае корректирующие операции получают на входе неогрублённую информацию, что способствует повышению качества классификации.

В англоязычной литературе в основном рассматривается случай $R = \mathbb{R}$, и алгоритмические операторы называются *вещественнозначными классификаторами* (real-valued classifiers). В общем случае операторы b_t, составляющие алгоритмическую композицию, называют также *базовыми алгоритмами* (base algorithms).

Основные свойства алгоритмических композиций, отличающие их от обычных эвристических алгоритмов.

- Композиция объединяет базовые алгоритмы, способные самостоятельно решать ту же исходную задачу.
- Композиция не знает внутреннего устройства базовых алгоритмов. Для неё это «чёрные ящики», имеющие только две функции: обучения по заданной выборке и вычисления ответа для заданного объекта. Это свойство очень удобно с технологической точки зрения: для настройки базовых алгоритмов можно задействовать богатый арсенал стандартных методов обучения.
- Композиция позволяет получать высокое качество обучения, недостижимое для отдельных базовых алгоритмов.

Два основных принципа построения алгоритмических композиций.

- *Специализация.* Пространство объектов делится на области, в каждой из которых строится свой алгоритм, специализирующийся на объектах только этой области. Исходная задача разбивается на более простые подзадачи по принципу «разделяй и властвуй». К таким методам относятся комитеты старшинства [8], решающие деревья и смеси экспертов [25].
- *Усреднение.* В этом случае корректирующая операция не получает информации о том, в какой области пространства находится объект, и работает только с ответами, выданными базовыми алгоритмами. Если базовые алгоритмы достаточно различны, то в результате усреднения их погрешности компенсируют друг друга. Причём усреднение следует понимать в обобщённом смысле, это не обязательно среднее арифметическое, и даже не обязательно линейная операция. На идее усреднения основаны комитеты большинства, бустинг [21], баггинг [17], монотонная коррекция [2].

Основные стратегии построения алгоритмических композиций.

- *Последовательная оптимизация.* Базовые алгоритмы строятся по очереди, и каждый следующий старается компенсировать недостатки предыдущих. Это жадная стратегия. Она не гарантирует построения наилучшей композиции, но на практике оказывается наиболее удобной. К таким методам относятся комитеты, бустинг, монотонная коррекция.
- *Глобальная оптимизация.* Базовые алгоритмы перестраиваются по очереди с помощью итерационного процесса, называемого EM-алгоритмом [25]. Фактически, это та же последовательная оптимизация, но повторяющаяся итерационно. «Настоящая» глобальная оптимизация всех базовых алгоритмов является

тяжёлой многоэкстремальной задачей и требует знания их внутреннего устройства, что затрудняет применение стандартных методов обучения.

- *Независимая оптимизация.* Базовые алгоритмы настраиваются независимо друг от друга. Чтобы они не получались слишком похожими, настройка производится по различным частям обучающей выборки, либо по различным частям признакового описания, либо при различных начальных приближениях. Типичным представителем этого подхода является баггинг.

- *Алгебраический подход*, описанный в теоретических работах Журавлёва и его учеников [5, 6], позволяет строить корректные алгоритмические композиции чисто алгебраическими методами, не прибегая к оптимизации. Этот подход крайне продуктивен при исследовании вопросов полноты моделей алгоритмов вида (1.1). Однако он плохо приспособлен для практического построения алгоритмов, так как не позволяет управлять сложностью композиции и склонен к переобучению. Более практичные схемы, разработанные в рамках алгебраического подхода, используют упомянутые выше стратегии оптимизации [1, 2].

Пример 1.1 Простейшим примером корректирующей операции является среднее арифметическое ответов, выданных базовыми алгоритмами:

$$F\big(b_1(x),\ldots,b_T(x)\big) = \frac{1}{T}\sum_{t=1}^{T} b_t(x), \quad x \in X. \tag{1.2}$$

Если X — вероятностное пространство, b_t — независимые случайные величины, то ошибка (дисперсия) результата коррекции уменьшается по мере увеличения числа базовых алгоритмов со скоростью $O(T^{1/2})$. На практике предполагать независимость, конечно же, нельзя, поскольку базовые алгоритмы настраиваются на решение одной и той же задачи. В общем случае, когда независимости нет, ошибки могут уменьшаться существенно медленнее или даже увеличиваться с ростом T.

1.1 Последовательное обучение базовых алгоритмов

Функционалы качества Введём функционал качества алгоритма $a\colon X \to Y$ на выборке X^ℓ с заданными на ней ответами Y^ℓ и весами объектов $W^\ell = (w_1,\ldots,w_\ell)$:

$$Q(a; X^\ell, Y^\ell, W^\ell) = \sum_{i=1}^{\ell} w_i L\big(a(x_i), y_i\big),$$

где $L\colon Y \times Y \to \mathbb{R}_+$ — функция потерь. Ради упрощения обозначений некоторые аргументы функционала Q будем иногда опускать. Будем полагать, что

$L(y, y^*) = [y \neq y^*]$ в задачах классификации;
$L(y, y^*) = (y - y^*)^2$ в задачах регрессии.

При фиксированном решающем правиле C определим функцию потерь $\widetilde{L}\colon R \times \times Y \to \mathbb{R}_+$ по формуле $\widetilde{L}(b, y^*) = L(C(b), y^*)$. Тогда появится возможность оценивать

качество не только самих алгоритмов, но и алгоритмических операторов — отображений вида $b\colon X \to R$:

$$Q(b; X^\ell, Y^\ell, W^\ell) = \sum_{i=1}^{\ell} w_i \widetilde{L}\big(b(x_i), y_i\big).$$

Будем полагать, что в нашем распоряжении имеется некоторый стандартный *метод обучения* базовых алгоритмов μ, решающий задачу минимизации функционала качества $Q(b)$ в заранее заданном семействе алгоритмов B:

$$\mu(X^\ell, Y^\ell, W^\ell) = \arg\min_{b \in B} Q(b; X^\ell, Y^\ell, W^\ell).$$

Процесс последовательного обучения базовых алгоритмов заключается в следующем. На первом шаге с помощью стандартного метода обучения строится первый базовый алгоритм b_1. Если его качество удовлетворяет, то дальнейшее построение композиции лишено смысла и процесс на этом заканчивается. В противном случае алгоритм b_1 фиксируется и строится второй алгоритм b_2, при одновременной оптимизации корректирующей операции F. На t-м шаге базовый алгоритм b_t и корректирующая операция F оптимизируются при фиксированных $b_1, \ldots, b_{t-1}$:

$$b_1 = \arg\min_{b} Q(b, X^\ell, Y^\ell); \tag{1.3}$$

$$b_2 = \arg\min_{b,F} Q(F(b_1, b), X^\ell, Y^\ell);$$

$$\ldots$$

$$b_t = \arg\min_{b,F} Q(F(b_1, \ldots, b_{t-1}, b), X^\ell, Y^\ell). \tag{1.4}$$

Во многих случаях для решения задачи (1.4) удаётся приспособить стандартные методы обучения, решающие более простую задачу (1.3). Для этого на вход метода обучения $\mu(X^\ell, Y^\ell, W^\ell)$ подаётся модифицированный вектор весов W^ℓ и/или модифицированный вектор ответов Y^ℓ.

Общий алгоритм 1.1, реализующий эту идею, требует уточнения формул пересчёта весов и ответов. Для каждого конкретного типа корректирующей операции получение таких формул является отдельной математической задачей.

Забегая немного вперёд, заметим, что модификация весов, как правило, сводится к увеличению весов у наиболее «трудных» объектов, на которых чаще ошибались предыдущие базовые алгоритмы, а модификация ответов — к аппроксимации невязки $y_i - a(x_i)$ вместо исходных ответов y_i. Точное сведение задачи вида (1.4) к задаче вида (1.3) далеко не всегда возможно, и в некоторых случаях приходится прибегать к адекватной подмене функционала качества.

Критерии останова могут использоваться различные, в зависимости от специфики задачи; возможно также применение нескольких критериев одновременно:

- Построено заданное количество базовых алгоритмов T.

Алгоритм 1.1 Построение алгоритмической композиции путём последовательного обучения базовых алгоритмов

Вход:
выборка X^ℓ;
метод обучения базовых алгоритмов μ;
Выход:
алгоритмическая композиция $F(b_1, \ldots, b_T)$;

1: **для всех** $i = 1, \ldots, \ell$
2: инициализировать веса $w_i := 1$;
3: **для всех** $t = 1, \ldots, T$, пока не выполнен критерий останова
4: $b_t := \mu(X^\ell, Y^\ell, W^\ell)$;
5: вычислить ошибки алгоритма $a = C\big(F(b_1, \ldots, b_t)\big)$ на выборке X^ℓ, при возможности без построения корректирующей операции и самого алгоритма;
6: модифицировать веса W^ℓ и/или ответы Y^ℓ с учётом ошибок, допущенных базовым алгоритмом b_t;

- Достигнута заданная точность на обучающей выборке:

$$Q(F(b_1, \ldots, b_t); X^\ell, Y^\ell) \leqslant \varepsilon.$$

- Достигнутую точность на контрольной выборке X^k не удаётся улучшить на протяжении последних d шагов: $t - t^* \geqslant d$, где

$$t^* = \arg\min_{s=1,\ldots,t} Q(F(b_1, \ldots, b_s); X^k, Y^k).$$

Выполнение этого критерия считается признаком переобучения, и процесс наращивания композиции обрывается сразу после t^*-го базового алгоритма.

1.2 Метод комитетов

Процесс последовательного построения базовых алгоритмов проще всего реализуется в том случае, когда корректирующая операция не имеет собственных настраиваемых параметров. К таким случаям относятся описанные ниже методы построения комитетов большинства и старшинства.

1.2.1 Комитеты большинства

Рассмотрим задачу классификации на два непересекающихся класса. Для удобства положим $Y = \{-1, 1\}$, $R = \mathbb{R}$ и допустим, что базовые алгоритмы возвращают только два ответа -1 и 1. Зафиксируем решающее правило

$$C(b) = \operatorname{sign}(b) = \begin{cases} 1, & b \geqslant 0; \\ -1, & b < 0; \end{cases}$$

Опр. 1.2 *Комитетом большинства называется алгоритмическая композиция $a(x)$ вида (1.1), с корректирующей операцией (1.2) и решающим правилом* $\operatorname{sign}(b)$*:*

$$a(x) = \operatorname{sign}\left(\sum_{t=1}^{T} b_t(x)\right).$$

Алгоритм $a(x)$ относит объект x к тому классу, к которому его относят большинство базовых алгоритмов. В этом смысле алгоритм $a(x)$ подобен комитету экспертов, принимающих решения большинством голосов.

Введём функционал качества композиции, равный числу ошибок, допускаемых алгоритмом a на обучающей выборке:

$$Q(a) = \sum_{i=1}^{\ell} \big[y_i a(x_i) < 0\big] = \sum_{i=1}^{\ell} \big[S_{iT} < 0\big],$$

где $S_{it} = y_i b_1(x_i) + \cdots + y_i b_t(x_i)$. Очевидно, S_{it} может принимать только целые значения от $-t$ до t. Для вычисления S_{it} удобно пользоваться рекуррентной формулой:

$$S_{i0} = 0;$$
$$S_{it} = S_{i,t-1} + y_i b_t(x_i), \quad t = 1, 2, \ldots$$

Минимум функционала $Q(a)$ на t-м шаге достигается, когда базовый алгоритм b_t правильно классифицирует те объекты x_i, для которых $S_{i,t-1}$ принимает значение либо 0, либо -1. Какими бы ни были ответы алгоритма b_t на остальных объектах, они не повлияют на знак суммы S_{it}, а значит, и на ответ алгоритма a. Отсюда следует, что для минимизации функционала $Q(a)$ по базовому алгоритму b_t достаточно применить стандартный метод обучения к подвыборке обучающих объектов $\{x_i \in X^\ell \mid -1 \leqslant S_{i,t-1} \leqslant 0\}$. Или, что то же самое, минимизировать функционал $Q(b_t; W^\ell)$ с весами объектов

$$w_i = \big[-1 \leqslant S_{i,t-1} \leqslant 0\big].$$

Легко понять, что такая стратегия пересчёта весов весьма недальновидна. Если на t-м шаге окажется $S_{it} \leqslant -2$, то алгоритм a допустит ошибку на объекте x_i, и последующие базовые алгоритмы не будут стремиться её исправить. Гораздо разумнее настраивать алгоритм b_t на тех объектах, на которых большинство предыдущих допустили ошибку

$$w_i = \big[S_{i,t-1} \leqslant 0\big].$$

Данная стратегия тоже не лишена изъяна. Обучающих объектов с ненулевыми весами может оказаться слишком мало, и тогда базовые алгоритмы, построенные по сильно урезанным подвыборкам, будут обладать слишком низким качеством. Чтобы этого не происходило, можно пополнять обучающие подвыборки дополнительными объектами, причём в первую очередь брать объекты x_i с наименьшими значениями $S_{i,t-1}$.

Для реализации этой идеи введём параметр ℓ_0 — длину обучающих подвыборок, по которым будут обучаться базовые алгоритмы. Упорядочим объекты выборки X^ℓ по возрастанию значений $S_{i,t-1}$. Объекты с одинаковыми значениями $S_{i,t-1}$ договоримся располагать в случайном порядке. Возьмём первые ℓ_0 объектов упорядоченной выборки, и положим для них $w_i = 1$, для остальных объектов положим $w_i = 0$. Именно этот вариант и описан в алгоритме 1.2.

Обратим внимание, что в описании алгоритма вместо S_{it} фигурирует S_i, так как при реализации нет необходимости хранить двумерный массив, и можно записывать S_{it} на место, занимаемое $S_{i,t-1}$.

Алгоритм 1.2 Построение комитета большинства

для всех $i = 1, \ldots, \ell$
  $w_i := 1$; $S_i := 0$;
для всех $t = 1, \ldots, T$, пока не выполнен критерий останова
  $b_t := \mu(X^\ell, Y^\ell, W^\ell)$;
  для всех $i = 1, \ldots, \ell$
    $S_i := S_i + y_i b_t(x_i)$;
  упорядочить выборку X^ℓ по возрастанию значений $(S_i + r_i)$,
  где r_i — случайные числа в интервале $\left(-\frac{1}{2}, \frac{1}{2}\right)$;
  для всех $i = 1, \ldots, \ell$
    $w_i := [i \leqslant \ell_0]$;

Параметр ℓ_0 можно задавать априори или подбирать по критерию минимума ошибок на скользящем контроле. Увеличение ℓ_0 повышает качество базовых алгоритмов, но уменьшает их различность. В предельном случае, когда $\ell_0 = \ell$, все базовые алгоритмы становятся одинаковыми и применение коррекции вообще теряет смысл. Таким образом, оптимизация параметра ℓ_0 позволяет найти компромисс между качеством и различностью базовых алгоритмов.

Ещё одна альтернативная эвристика заключается в том, чтобы увеличивать веса тех объектов, на которых часто ошибались предыдущие алгоритмы, и, наоборот, уменьшать веса объектов, на которых ошибок не было. Например, можно положить $w_i = \gamma^{-S_{i,t-1}}$, где γ — параметр метода настройки. Однако, в таком случае накладывается дополнительное ограничение на метод обучения базовых алгоритмов — он должен уметь решать задачу минимизации функционала $Q(b_t; W^\ell)$ с произвольными вещественными весами объектов. Далеко не все стандартные методы обучения способны учитывать веса объектов, хотя соответствующее обобщение в большинстве случаев строится несложно. Варианты алгоритма, использующие только бинарные веса, совместимы с более широким классом методов обучения.

1.2.2 Комитеты старшинства

Рассмотрим задачу классификации с произвольным числом классов, $|Y| = M$. Возьмём базовые алгоритмы, осуществляющие классификацию на 2 непересекающихся класса, $b_t \colon X \to \{0, 1\}$. Положим $R = \{0, 1\}$. Решающие правила использовать не будем.

Опр. 1.3 *Комитетом старшинства называется алгоритмическая композиция*

$$a(x) = F\big(b_1(x), \ldots, b_T(x)\big),$$

в которой корректирующая операция F имеет параметры $c_1, \ldots, c_T \in Y$, $c_0 \notin Y$ и реализуется следующим образом:

Алгоритм 1.3 Классификация объекта $x \in X$ комитетом старшинства

для всех $t = 1, \ldots, T$
  если $b_t(x) = 1$ **то**
    вернуть c_t;
вернуть c_0.

Алгоритм $a(x)$ подобен комитету экспертов с логикой старшинства. Каждый эксперт b_t имеет свою «область компетентности» $\{x \in X \mid b_t(x) = 1\}$ и либо голосует за свой класс c_t, либо признаёт свою некомпетентность, и тогда право решения переходит к следующему эксперту, в порядке убывания старшинства.

Заметим также, что понятие комитета старшинства в точности совпадает с понятиями *решающего списка* [29] и *машины покрывающих множеств* [26].

Когда $b_t(x) = 1$, говорят, что b_t *покрывает* объект x.

Ответ c_0 означает отказ комитета от классификации объекта. Положим штраф за ошибку равным 1, штраф за отказ λ. Введём функционал качества композиции, равный суммарному штрафу за ошибки и отказы:

$$Q(a) = \sum_{i=1}^{\ell} [a(x_i) \neq c_0]\,[a(x_i) \neq y_i] + \lambda \sum_{i=1}^{\ell} [a(x_i) = c_0].$$

Рассмотрим, как изменятся множества ошибок и отказов на t-м шаге, при добавлении базового алгоритма b_t, относящего объекты к классу c_t.

Во-первых, алгоритм b_t не влияет на классификацию объектов, покрытых предыдущими алгоритмами $b_1, \ldots, b_{t-1}$. Обозначим множество индексов непокрытых объектов через I_{t-1}:

$$I_{t-1} = \{i \mid b_1(x_i) = \cdots = b_{t-1}(x_i) = 0\}.$$

Во-вторых, добавление базового алгоритма b_t влечёт ошибки при $b_t(x_i) = 1$ и $c_t \neq y_i$, и отказы при $b_t(x_i) = 0$, для всех $i \in I_{t-1}$. Однако штрафовать за отказ имеет смысл только на объектах класса c_t, так как в противном случае не-отказ приводит к ошибке.

Таким образом, минимизация функционала $Q(a) \equiv Q\big(F(b_1, \ldots, b_t)\big)$ по базовому алгоритму b_t эквивалентна минимизации функционала

$$Q(b_t) = \sum_{i \in I_{t-1}} [b_t(x_i) = 1]\,[c_t \neq y_i] + \lambda \sum_{i \in I_{t-1}} [b_t(x_i) = 0]\,[c_t = y_i],$$

который легко приводится к стандартному виду — взвешенной сумме потерь:

$$Q(b_t) = \sum_{i=1}^{\ell} \underbrace{[i \in I_{t-1}]\big([y_i \neq c_t] + \lambda[y_i = c_t]\big)}_{w_i} \big[b_t(x_i) \neq \underbrace{[y_i = c_t]}_{z_i}\big].$$

Для минимизации данного функционала достаточно применить произвольный стандартный метод обучения: $b_t = \mu(X^\ell, Z^\ell, W^\ell)$, подставив в него веса объектов $W^\ell = (w_1, \ldots, w_\ell)$ и бинарные ответы $Z^\ell = (z_1, \ldots, z_\ell)$.

Полученные формулы пересчёта весов w_i и ответов z_i имеют прозрачный смысл. Базовый алгоритм b_t настраивается таким образом, чтобы покрыть как можно больше объектов «своего» класса c_t и как можно меньше объектов «чужих» классов. Параметр λ позволяет найти компромисс между этими, вообще говоря, противоречивыми, критериями настройки. Представляется разумным брать $\lambda \ll 1$, поскольку покрытие чужого объекта гарантирует ошибку, а не-покрытие своего объекта может быть скомпенсировано последующими базовыми алгоритмами.

Алгоритм 1.4 описывает процесс построения комитета старшинства. В этом алгоритме преднамеренно не фиксирована стратегия выбора класса c_t на шаге 4. Она существенно зависит от специфики предметной области и конкретной задачи.

Алгоритм 1.4 Построение комитета старшинства

1: **для всех** $i = 1, \ldots, \ell$
2: $w_i := 1$;
3: **для всех** $t = 1, \ldots, T$, пока не выполнен критерий останова
4: выбрать класс c_t;
5: **для всех** $i = 1, \ldots, \ell$
6: $z_i := [y_i = c_t]$;
7: **если** $w_i \neq 0$ **то** $w_i := 1 - z_i + \lambda z_i$;
8: $b_t := \mu(X^\ell, Z^\ell, W^\ell)$;
9: **для всех** $i = 1, \ldots, \ell$
10: **если** $b_t(x_i) = 1$ **то** $w_i := 0$;

Возможные стратегии формирования последовательности $c_1, \ldots, c_T$

- Выбирается тот класс, в котором осталось больше объектов с ненулевыми весами w_i.
- Выбирается тот класс, для которого удаётся получить лучшее значение функционала качества.
- Задаётся приоритетный порядок классов $Y = \{v_1, \ldots, v_M\}$, разбивающий последовательность базовых алгоритмов на блоки по T_m алгоритмов для каждого из классов v_m, $m = 1, \ldots, M$:

$$\underbrace{b_{v_1 1}, \ldots, b_{v_1 T_1}}_{\text{класс } v_1}, \quad \underbrace{b_{v_2 1}, \ldots, b_{v_2 T_2}}_{\text{класс } v_2}, \quad \ldots \quad \underbrace{b_{v_M 1}, \ldots, b_{v_M T_M}}_{\text{класс } v_M},$$

- Задаётся приоритетный порядок классов $Y = \{v_1, \ldots, v_M\}$, как в предыдущем случае. Однако теперь для каждого класса строится независимый *подкомитет*, способный самостоятельно отделять объекты данного класса от объектов всех остальных классов. Для этого перед началом каждого блока все объекты с нулевыми весами возвращаются в выборку. Соответственно, модифицируется правило пересчёта весов на шаге 7:

  если $w_i \neq 0$ **или** $c_t \neq c_{t-1}$ **то** $w_i := 1 - z_i + \lambda z_i$;

  В результате каждый класс получает своё отдельное описание. Решения такого комитета, составленного из независимых подкомитетов, легче поддаются содержательной интерпретации,

Стратегия выбора параметра λ При выборе параметра λ можно руководствоваться следующими соображениями.

- С точки зрения функционала $Q(b_t)$ покрытие одного чужого объекта стоит столько же, сколько не-покрытие λ^{-1} своих объектов. Достаточно разумными представляются значения λ в диапазоне от 0.01 до 0.5.

- Уменьшение λ приводит к сокращению количества ошибок на чужих объектах, но и к уменьшению общего числа покрываемых объектов. Это способствует росту числа членов комитета и в конечном итоге ведёт к переобучению. С другой стороны, увеличение λ приводит к стремлению покрыть как можно больше объектов, не обращая внимания на допускаемые ошибки. Это способствует чрезмерному упрощению комитета. Таким образом, параметр λ позволяет управлять сложностью и обобщающей способностью алгоритмической композиции. Для практического определения оптимального значения λ рекомендуется использовать скользящий контроль.

Кусочно-линейные разделяющие поверхности Если базовые алгоритмы строят линейные разделяющие поверхности, то комитет старшинства является кусочно-линейной разделяющей поверхностью.

Пример 1.2 Задача на плоскости: $X = \mathbb{R}^2$, $Y = \{\circ, \times\}$, $b_t(x)$ — полуплоскости.

TODO:
вставить картинки

1. Разделимость одной полуплоскостью. Вывод: итерации могут закончиться после первого шага.

2. Разделимость одной полуплоскостью не достигается, двумя — достигается. Вывод: процесс последовательного наращивания позволяет построить корректный алгоритм.

3. Разделимость тремя полуплоскостями. Если после построения b_2 вернуться к оптимизации b_1, то третьего базового алгоритма уже не нужно. Выводы: жадное наращивание не оптимально; иногда имеет смысл вернуться назад и перенастроить имеющиеся базовые алгоритмы, чтобы обнаружить и отбросить некоторые лишние алгоритмы.

1.3 Линейная коррекция

Опр. 1.4 *Линейная корректирующая операция есть функция* $F : \mathbb{R}^T \to \mathbb{R}$ *вида*

$$F(b_1, \ldots, b_T) = \frac{1}{T} \sum_{t=1}^{T} \alpha_t b_t, \tag{1.5}$$

где α_t *— числовые параметры.*

Линейные корректирующие операции успешно применяются для решения задач классификации и регрессии. В случае классификации конструкцию (1.5) называют также *взвешенным голосованием классификаторов.*

1.3.1 Бустинг в задачах классификации

Рассмотрим задачу классификации на два класса, $Y = \{-1, 1\}$. Допустим, что базовые алгоритмы также возвращают только два ответа -1 и 1, и решающее правило фиксировано: $C(b) = \text{sign}(b)$.

Искомая алгоритмическая композиция имеет вид:

$$a(x) = C\big(F(b_1(x), \ldots, b_T(x))\big) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t b_t(x)\right), \quad x \in X.$$

Определим функционал качества композиции как число ошибок, допускаемых ею на обучающей выборке:

$$Q_T = \sum_{i=1}^{\ell} \Big[y_i \sum_{t=1}^{T} \alpha_t b_t(x_i) < 0 \Big].$$

Для упрощения задачи минимизации функционала Q_T введём две эвристики.

Эвристика 1 При добавлении в композицию слагаемого $\alpha_t b_t(x)$ будем оптимизировать только базовый алгоритм b_t и коэффициент при нём α_t, не трогая предыдущих слагаемых $\alpha_j b_j(x)$, $j = 1, \ldots, t-1$.

Эвристика 2 Чтобы решить задачу оптимизации параметра α_t аналитически, заменим в функционале качества пороговую функцию потерь какой-нибудь её непрерывно дифференцируемой оценкой сверху.

Процесс последовательного обучения базовых алгоритмов в совокупности с указанными эвристиками приводит к алгоритмам *бустинга* (boosting), пользующихся большой популярностью благодаря своей простоте и высокой эффективности. В частности, экспоненциальная оценка $[b < 0] \leqslant e^{-b}$, $b \in \mathbb{R}$, существенно упрощает выкладки и приводит к знаменитому алгоритму AdaBoost [31].

Рассмотрим экспоненциальную верхнюю оценку функционала Q_T:

$$\begin{aligned} Q_T \leqslant \bar{Q}_T &= \sum_{i=1}^{\ell} \exp\left(-y_i \sum_{t=1}^{T} \alpha_t b_t(x_i) \right) = \\ &= \sum_{i=1}^{\ell} \underbrace{\exp\left(-y_i \sum_{t=1}^{T-1} \alpha_t b_t(x_i) \right)}_{w_i} \exp(-y_T \alpha_T b_T(x_i)). \end{aligned}$$

Заметим, что введённые здесь величины w_i не зависят от α_T и b_T, следовательно, они могут быть вычислены перед построением базового алгоритма b_T. Введём также нормированный вектор $\widetilde{W}^\ell = \big(\widetilde{w}_1, \ldots, \widetilde{w}_\ell\big)$, где $\widetilde{w}_i = w_i \,/\, \sum_{j=1}^{\ell} w_j$.

Теорема 1.1 *Пусть Q — стандартный функционал качества алгоритма классификации b на обучающей выборке X^ℓ, Y^ℓ с нормированным вектором весов объектов W^ℓ:*

$$Q(b; X^\ell, Y^\ell, W^\ell) = \sum_{i=1}^{\ell} w_i \big[y_i b(x_i) < 0 \big].$$

Пусть $\min\limits_b Q(b; W^\ell) < 1/2$ для любого нормированного вектора весов W^ℓ. Тогда минимум функционала $\bar{Q}_T$ достигается при

$$b_T = \arg\min_b Q(b; \widetilde{W}^\ell);$$

$$\alpha_T = \frac{1}{2} \ln \frac{1 - Q(b_T; \widetilde{W}^\ell)}{Q(b_T; \widetilde{W}^\ell)}.$$

Доказательство.

Выразим $\bar{Q}_T$ через функционал качества базового алгоритма. Это легко сделать, если воспользоваться очевидным разложением $e^{-\alpha\beta} = e^{-\alpha}[\beta = 1] + e^{\alpha}[\beta = -1]$, справедливым для любых $\alpha \in \mathbb{R}$ и $\beta \in \{-1, 1\}$:

$$\bar{Q}_T = \Bigg(e^{-\alpha_T} \underbrace{\sum_{i=1}^{\ell} \widetilde{w}_i[b_T(x_i) = y_i]}_{1-Q} + e^{\alpha_T} \underbrace{\sum_{i=1}^{\ell} \widetilde{w}_i[b_T(x_i) \neq y_i]}_{Q} \Bigg) \underbrace{\sum_{i=1}^{\ell} w_i}_{\bar{Q}_{T-1}} =$$

$$= \big(e^{-\alpha_T}(1-Q) + e^{\alpha_T} Q\big)\bar{Q}_{T-1},$$

где введено краткое обозначение $Q = Q(b_T; \widetilde{W}^{\ell})$. Дифференцируя функционал $\bar{Q}_T$ по параметру α_T и приравнивая нулю производную, получим оптимальное значение параметра α_T:

$$\alpha_T = \frac{1}{2} \ln \frac{1-Q}{Q}.$$

Подставляя это значение обратно в $\bar{Q}_T$, получим

$$\bar{Q}_T = \bar{Q}_{T-1} \sqrt{4Q(1-Q)}. \tag{1.6}$$

Поскольку $\bar{Q}_{T-1}$ фиксировано, минимизация $\bar{Q}_T$ эквивалентна минимизации Q при $Q < 1/2$ и максимизации Q при $Q > 1/2$, однако второй случай не проходит по условию теоремы. ■

Требование $Q(b_t; \widetilde{W}^{\ell}) < 1/2$ означает, что метод обучения должен обеспечивать построение алгоритма $b_t(x)$, доля ошибок которого на обучающей выборке хотя бы немного меньше $1/2$. Иными словами, от каждого базового алгоритма требуется, чтобы он классифицировал объекты хотя бы немного лучше, чем наугад. Это достаточно слабое требование, и на практике оно, как правило, выполняется. Более того, этого условия оказывается достаточно, чтобы гарантировать сходимость алгоритма AdaBoost за конечное число шагов.

Теорема 1.2 *Если на каждом шаге метод обучения обеспечивает построение базового алгоритма b_t такого, что $Q(b_t; \widetilde{W}^{\ell}) < 1/2 - \gamma$ при некотором $\gamma > 0$, то AdaBoost гарантирует построение корректного алгоритма $a(x)$ за конечное число шагов.*

Доказательство.

Непосредственно из (1.6) вытекает сходимость функционала Q_T к нулю со скоростью геометрической прогрессии: $Q_{T+1} \leqslant \bar{Q}_{T+1} \leqslant \bar{Q}_1(1 - 4\gamma^2)^{\frac{T}{2}}$. Наступит момент, когда значение $\bar{Q}_T$ окажется меньше 1. Но тогда функционал Q_T обратится в нуль, поскольку он может принимать только целые неотрицательные значения. ■

При реализации алгоритма веса объектов удобно пересчитывать после построения каждого базового алгоритма b_t по рекуррентной формуле

$$w_i := w_i \exp\big(-\alpha_t y_i b_t(x_i)\big).$$

Вес объекта увеличивается, когда b_t допускает на нём ошибку. Таким образом, при настройке каждого базового алгоритма больший вес получают те объекты, которые оказались наиболее трудными для всех предыдущих алгоритмов.

Алгоритм 1.5 AdaBoost — построение линейной комбинации классификаторов

1: **для всех** $i = 1, \ldots, \ell$
2: $\quad w_i := 1/\ell$;
3: **для всех** $t = 1, \ldots, T$, пока не выполнен критерий останова
4: $\quad b_t := \mu(X^\ell, Y^\ell, W^\ell)$ — решение задачи минимизации $Q(b; W^\ell) \to \min\limits_b$;
5: $\quad \alpha_t := \dfrac{1}{2} \ln \dfrac{1 - Q(b_t; W^\ell)}{Q(b_t; W^\ell)}$;
6: $\quad$ **для всех** $i = 1, \ldots, \ell$
7: $\quad\quad w_i := w_i \cdot \begin{cases} \exp(\alpha_t), & b_t(x_i) \neq y_i; \\ \exp(-\alpha_t), & b_t(x_i) = y_i; \end{cases}$
8: $\quad$ нормировка: $w_i := w_i \,/\, \sum_{j=1}^{\ell} w_j$, для всех $i = 1, \ldots, \ell$;

Обобщающая способность бустинга В течение последних 10 лет бустинг остаётся одним из наиболее популярных методов машинного обучения, наряду с нейронными сетями и машинами опорных векторов. Основные причины — простота, универсальность, гибкость (возможность построения различных модификаций), и, главное, неожиданно высокая обобщающая способность.

Во многих экспериментах наблюдалось практически неограниченное улучшение качества обучения (частоты правильной классификации на независимой тестовой выборке) при наращивании числа алгоритмов в композиции. Более того, качество на тестовой выборке часто продолжало улучшаться даже после достижения безошибочного распознавания всей обучающей выборки [21]. Это перевернуло существовавшие долгое время представления о том, что для повышения обобщающей способности необходимо ограничивать сложность алгоритмов. На примере бустинга стало очевидно, что хорошим качеством могут обладать сколь угодно сложные алгоритмические композиции при условии их «правильной» настройки.

Впоследствии феномен бустинга получил теоретическое обоснование. Оказалось, что взвешенное голосование не увеличивает сложность алгоритма, а лишь сглаживает ответы базовых алгоритмов. Количественные оценки обобщающей способности бустинга формулируются в терминах отступа [16]. *Отступом* (margin) объекта x_i называется величина $M_i = y_i a(x_i)$. В некотором смысле отступ — это расстояние от объекта до границы классов. Если объект относится алгоритмом к чужому классу, то его отступ отрицателен. Чем больше в обучающей выборке объектов с большим отступом, тем лучше разделяются классы, тем надёжнее может быть классификация. Эффективность бустинга объясняется тем, что по мере добавления базовых алгоритмов увеличиваются отступы обучающих объектов. Причём бустинг продолжает «раздвигать» классы даже после достижения безошибочной классификации обучающей выборки.

К сожалению, полученные в [16] количественные оценки обобщающей способности дают лишь качественное обоснование феноменам бустинга. Хотя они существенно точнее сложностных оценок теории Вапника-Червоненкиса [35], всё же они сильно завышены и требуют обучающих выборок длиной порядка 10^4–10^6.

Более основательные эксперименты показали, что иногда бустинг всё же переобучается [27, 28]. По мнению автора статьи [23] причины эффективности бустинга до конца ещё не поняты и его дальнейшее улучшение остаётся открытой проблемой.

Достоинства AdaBoost

- Хорошая обобщающая способность. В реальных задачах удаётся строить композиции, превосходящие по качеству составляющие их алгоритмы.
- Возможность идентифицировать объекты, являющиеся шумовыми выбросами. Это наиболее «трудные» объекты x_i, для которых по окончании процесса веса w_i оказались максимальными.
- Простота реализации.

Недостатки AdaBoost

- AdaBoost склонен к переобучению при наличии значительного уровня шума в данных. Экспоненциальная функция потерь слишком сильно увеличивает веса «ниболее трудных» объектов, на которых ошибаются многие базовые алгоритмы. Однако именно эти объекты чаще всего оказываются шумовыми выбросами. В результате AdaBoost начинает настраиваться на шум, что ведёт к переобучению. Проблема решается путём удаления выбросов или применения менее «агрессивных» функций потерь. В частности, алгоритм LogitBoost использует логистическую функцию [22].
- AdaBoost требует достаточно длинных обучающих выборок. Другие методы линейной коррекции, в частности, баггинг, способны строить алгоритмы сопоставимого качества по меньшим выборкам данных.
- Бустинг часто приводит к построению громоздких композиций, состоящих из сотен алгоритмов. Такие композиции исключают возможность содержательной интерпретации, требуют больших объёмов памяти для хранения базовых алгоритмов и существенных затрат времени на вычисление классификаций.

TODO:
Связь AdaBoost с градиентным методом минимизации.
Связь AdaBoost с логистической регрессией.
Обобщения бустинга. Большое число классов. Вещественнозначные базовые классификаторы. Базовые классификаторы, допускающие отказ от классификации — совпадает с алгоритмом взвешенного голосования по логическим закономерностям.

1.3.2 Бустинг в задачах восстановления регрессии

1.3.3 Баггинг

Метод баггинга (bagging) был предложен Л. Брейманом в 1996 году [17, 18, 19]. Основная его идея заключается в том, чтобы целенаправленно увеличивать различия между базовыми алгоритмами, делая их, по возможности, более независимыми. Один из способов достижения этой цели — обучать базовые алгоритмы не по всей выборке, а по различным её частям. Другой способ — обучать их на разных частях признакового описания. Выделение подмножеств объектов и/или признаков производится, как правило, случайным образом. Третий способ — использовать при оптимизации

параметров алгоритма различные начальные приближения, либо применять стохастические алгоритмы оптимизации. Построенные базовые алгоритмы объединяются в композицию с помощью простого или взвешенного голосования.

Результаты эмпирического сравнения бустинга и баггинга [32].

- Есть задачи, в которых существенно лучшие результаты показывает бустинг, и есть задачи, в которых лучшим оказывается баггинг.
- Бустинг работает лучше на больших обучающих выборках, баггинг — на малых. При увеличении длины выборки баггинг уменьшает разнообразие классификаторов, а бустинг увеличивает.
- Бустинг лучше воспроизводит границы классов сложной формы.
- Баггинг направлен исключительно на уменьшение дисперсии (variance) ответов. Бустинг способствует уменьшению как дисперсии, так и смещения (bias).

1.4 Квазилинейная коррекция (смеси алгоритмов)

Линейные корректирующие операции естественным образом обобщаются на тот случай, когда веса базовых алгоритмов не постоянны и зависят от положения объекта x в пространстве X.

Поставим каждому базовому алгоритму $b_t(x)$ в соответствие весовую функцию $\omega_t(x)$, принимающую значения из отрезка $[0,1]$. Определим композицию так, чтобы ответ базового алгоритма $b_t(x)$ на объекте x учитывался с весом $\omega_t(x)$. Будем говорить, что весовая функция $\omega_t(x)$ описывает *область компетентности* базового алгоритма $b_t(x)$. Понятие области компетентности введено Растригиным в [9].

Опр. 1.5 *Квазилинейная корректирующая операция есть функция* $F\colon \mathbb{R}^{2T} \to \mathbb{R}$,

$$F\big(b_1,\ldots,b_T,\omega_1,\ldots,\omega_T\big) = \sum_{t=1}^{T} \omega_t b_t. \tag{1.7}$$

В задачах восстановления регрессии естественным дополнительным требованием является условие нормировки:

$$\sum_{t=1}^{T} \omega_t(x) = 1, \quad \text{для всех } x \in X.$$

Если оно выполнено, то для любого объекта x коррекция сводится к усреднению ответов базовых алгоритмов с некоторыми весами (зависящими от x), так что результат не выходит за пределы отрезка $\big[\min\limits_t b_t(x), \max\limits_t b_t(x)\big]$.

В задачах классификации с пороговым решающим правилом $C(b) = [b > 0]$ или $C(b) = \operatorname{sign}(b)$ нормировку делать не обязательно, поскольку важен только знак выражения (1.7).

Более того, при отсутствии нормировки появляется возможность оценить компетентность всей совокупности базовых алгоритмов на классифицируемом объекте x.

Если значение $\max\{\omega_1(x),\ldots,\omega_T(x)\}$ оказывается меньше некоторого порога, можно считать, что объект x попадает в «область неуверенности», где все алгоритмы некомпетентны. Обычно это связано с тем, что в данной области обучающих прецедентов просто не было. Таким образом, попутно решается задача *обнаружения новизны* (novelty detection).

В зарубежной литературе композиции вида (1.7) принято называть *смесями экспертов* (Mixture of Experts, ME), базовые алгоритмы — *экспертами*, весовые функции — входами или *гейтами* (gates) [15]. По-русски «смесь экспертов» звучит не очень удачно, поэтому будем говорить о «смесях алгоритмов».

1.4.1 Смесь двух алгоритмов

Рассмотрим самый простой случай, когда композиция состоит из двух базовых алгоритмов, весовые функции удовлетворяют требованию нормировки, решающее правило фиксировано:

$$a(x) = C\big(\omega_1(x)b_1(x) + \omega_2(x)b_2(x)\big), \quad \omega_1(x) + \omega_2(x) = 1, \quad x \in X.$$

Рассмотрим функционал качества композиции с весами объектов $W^\ell = (w_1,\ldots,w_\ell)$:

$$Q(a; W^\ell) = \sum_{i=1}^{\ell} w_i L\big(a(x_i), y_i\big).$$

Гипотеза 1.1 *Функция потерь $\widetilde{L}(b, y^*) = L(C(b), y^*)$ является выпуклой по b.*

Условие выпуклости имеет прозрачную интерпретацию: потери растут быстрее, чем величина отклонения от правильного ответа y^*. Во многих задачах требование выпуклости является естественным. В частности, ему удовлетворяет квадратичная функция $(b - y^*)^2$, применяемая в регрессионных задачах, а также экспоненциальная функция $\exp(-by^*)$, применяемая в задачах классификации. Однако пороговая функция $[C(b) \neq y^*]$ выпуклой уже не является.

Гипотеза выпуклости позволяет оценить сверху функционал $Q(a)$ суммой двух функционалов стандартного вида:

$$Q(a) \leqslant \bar{Q}(a) = \underbrace{\sum_{i=1}^{\ell} \omega_1(x_i) w_i \widetilde{L}\big(b_1(x_i), y_i\big)}_{Q(b_1)} + \underbrace{\sum_{i=1}^{\ell} \omega_2(x_i) w_i \widetilde{L}\big(b_2(x_i), y_i\big)}_{Q(b_2)}.$$

Если весовые функции $\omega_1(x)$ и $\omega_2(x)$ известны, то минимизация функционалов $Q(b_1)$ и $Q(b_2)$ может быть выполнена порознь. Для этого достаточно применить выбранный стандартный метод обучения базовых алгоритмов:

$$b_1 = \mu(X^\ell, Y^\ell, W^\ell \otimes \Omega_1^\ell);$$
$$b_2 = \mu(X^\ell, Y^\ell, W^\ell \otimes \Omega_2^\ell);$$

где $\otimes$ — операция покомпонентного умножения векторов, $\Omega_1^\ell = \big(\omega_1(x_1),\ldots,\omega_1(x_\ell)\big)$, $\Omega_2^\ell = \big(\omega_2(x_1),\ldots,\omega_2(x_\ell)\big)$.

Алгоритм 1.6 M2E — построение смеси двух базовых алгоритмов

Вход:

X^ℓ, Y^ℓ — обучающая выборка;

$W^\ell = (w_1, \ldots, w_\ell)$ — исходный вектор весов объектов;

$b_0(x)$ — начальное приближение одного из базовых алгоритмов;

Выход:

алгоритмическая композиция: $a(x) = b_1(x)\omega_1(x) + b_2(x)\omega_2(x)$;

1: $\omega_1 := \arg\min\limits_{\omega} \sum\limits_{i=1}^{\ell} \omega(x_i) w_i \widetilde{L}(b_0(x_i), y_i)$;

2: **повторять**

3: $\quad \Omega_1^\ell := \big(\omega_1(x_1), \ldots, \omega_1(x_\ell)\big)$;

4: $\quad b_1 := \mu(X^\ell, Y^\ell, W^\ell \otimes \Omega_1^\ell)$;

5: $\quad b_2 := \mu(X^\ell, Y^\ell, W^\ell \otimes (1 - \Omega_1^\ell))$;

6: $\quad d_i := w_i\big(\widetilde{L}(b_1(x_i), y_i) - \widetilde{L}(b_2(x_i), y_i)\big)$, $\quad$ для всех $i = 1, \ldots, \ell$;

7: $\quad \omega_1 := \arg\min\limits_{\omega} \sum\limits_{i=1}^{\ell} \omega(x_i) d_i$;

8: **пока** не стабилизируется значение $Q(a; W^\ell)$;

Верно и обратное: если известны базовые алгоритмы $b_1(x)$ и $b_2(x)$, то можно оценить весовые функции $\omega_1(x)$ и $\omega_2(x)$. В силу условия нормировки $\omega_2(x) = 1 - \omega_1(x)$. Поэтому достаточно найти только функцию $\omega_1(x)$, минимизирующую

$$\bar{\bar{Q}}(\omega_1) = \sum_{i=1}^{\ell} \omega_1(x_i) \underbrace{w_i\big(\widetilde{L}(b_1(x_i), y_i) - \widetilde{L}(b_2(x_i), y_i)\big)}_{d_i = \text{const}(\omega_1)}. \tag{1.8}$$

Если известен параметрический вид весовой функции $\omega_1(x)$, то эта задача легко решается градиентными методами, аналогичными методам обучения нейронных сетей.

Итак, знание областей компетентности позволяет настроить оба базовых алгоритма, а знание базовых алгоритмов позволяет аппроксимировать форму областей компетентности. Остаётся только запустить итерационный процесс, в котором эти две задачи решаются поочерёдно. Итерации можно начинать с построения начального приближения $b_0(x)$ для одного из базовых алгоритмов. Алгоритм 1.6 описывает эту процедуру подробно.

Выбор семейства весовых функций определяется спецификой предметной области. Например, в задаче может иметься априорная информация о том, что при больших и малых значениях некоторого признака $f(x)$ поведение целевой зависимости принципиально различно. Тогда имеет смысл использовать весовую функцию вида $\omega(x; \alpha, \beta) = \big(1 + \exp(\alpha f(x) - \beta)\big)^{-1}$ с настраиваемыми параметрами $\alpha, \beta \in \mathbb{R}$.

Если подобной априорной информации нет, то в качестве «универсальных» областей компетентности берут области простой формы. Например, в пространстве $X = \mathbb{R}^n$ можно взять полуплоскости $\omega(x, \alpha) = \big(1 + \exp(-x^\intercal \alpha)\big)^{-1}$ или сферические гауссианы $\omega(x, \alpha, \beta) = \exp\big(-\beta\|x - \alpha\|^2\big)$ с параметрами $\alpha \in \mathbb{R}^n$, $\beta \in \mathbb{R}$. В этих

двух случаях для решения задачи (1.8) становятся непосредственно применимы известные методы настройки нейронных сетей, в частности, однослойный персептрон с сигмоидной или гауссовской функцией активации. По этой причине весовые функции называют также *входными сетями* (gating networks).

1.4.2 Иерархические смеси алгоритмов

Метод M2E легко приспособить для построения сколь угодно сложных смесей, состоящих из произвольного числа алгоритмов.

Идея заключается в следующем. Сначала строится смесь двух алгоритмов и анализируется качество работы каждого из них на своей области компетентности. Если качество алгоритма недостаточно, то он заменяется смесью двух новых алгоритмов, разбивающих его область $\omega(x)$ на две новые области с весовыми функциями $\omega(x)\omega_1(x)$ и $\omega(x)\omega_2(x)$, причём $\omega_1(x)+\omega_2(x)=1$, чтобы по-прежнему выполнялось условие нормировки. Процесс дробления областей компетентности продолжается до тех пор, пока в смеси остаются алгоритмы неудовлетворительного качества. В результате получается бинарное дерево алгоритмов, называемое *иерархической смесью алгоритмов* (hierarchical mixture of experts, HME). Алгоритм 1.7 более подробно описывает детали рекурсивной реализации этой идеи.

Способ построения HME напоминает синтез бинарного решающего дерева. Главное отличие в том, что весовые функции производят нечёткое разделение пространства объектов на области. Ещё одно существенное отличие — в применяемом критерии ветвления.

Критерии ветвления Базовый алгоритм расщепляется на два, когда качество его работы оказывается неудовлетворительным. Существуют различные способы оценить качество алгоритма, но мы рассмотри только два (оба условия записаны для алгоритма $b_1(x)$, но точно так же они записываются и для $b_2(x)$).

- Относительная точность на обучающей выборке X^ℓ хуже заданного порога ε:

$$Q(b_1; X^\ell, Y^\ell, W^\ell \otimes \Omega_1^\ell) > \varepsilon \sum_{i=1}^{\ell} w_i \omega_1(x_i).$$

  Недостатком этого критерия является необходимость задания параметра ε.

- Точность на заранее выделенной контрольной выборке X^k хуже, чем у родительского алгоритма:

$$Q(b_1; X^k, Y^k, W^\ell \otimes \Omega_1^\ell) > Q(b_0; X^k, Y^k, W^\ell \otimes \Omega_1^\ell).$$

  Данный критерий оценивает склонность рассматриваемой ветки иерархии к переобучению. Недостатком этого критерия является необходимость выделения части объектов в контрольную выборку.

Алгоритм 1.7 Рекурсивный алгоритм построения иерархической смеси алгоритмов

Вход:
X^ℓ, Y^ℓ — обучающая выборка;

ПРОЦЕДУРА `Разветвить` (W^ℓ, b_0);
$W^\ell = (w_1, \ldots, w_\ell)$ — веса обучающих объектов;
b_0 — алгоритм, заменяемый смесью $a(x) = b_1(x)\omega_1(x) + b_2(x)\omega_2(x)$;
$b_1, \omega_1, b_2, \omega_2 := M2E(W^\ell, b_0)$;
если выполнен критерий ветвления для b_1 **то**
  `Разветвить` $(W^\ell \otimes \Omega_1^\ell, b_1)$;
если выполнен критерий ветвления для b_2 **то**
  `Разветвить` $(W^\ell \otimes \Omega_2^\ell, b_2)$;

Основной алгоритм
$W^\ell := (1/\ell, \ldots, 1/\ell)$;
$b_0 := \mu(X^\ell, Y^\ell, W^\ell)$;
`Разветвить` (W^ℓ, b_0);

1.5 Монотонная коррекция

Опр. 1.6 *Пусть R и Y — частично упорядоченные множества. Монотонные функции вида $F : R^T \to Y$ будем называть монотонными корректирующими операциями.*

Монотонная коррекция является ещё одним естественным обобщением линейной. Любая линейная корректирующая операция является монотонной, обратное в общем случае неверно. При линейной коррекции вес каждого базового алгоритма остается постоянным на всем пространстве объектов, что представляется не вполне обоснованной эвристикой. Монотонная коррекция обладает более богатыми возможностями для настройки.

1.5.1 Вывод формул пересчета весов и ответов

1.5.2 Монотонизация выборки

1.5.3 Бинарная монотонная корректирующая операция

1.5.4 Кусочно-непрерывная монотонная корректирующая операция

1.6 Алгебраический подход

Идея алгебраического синтеза корректных алгоритмов распознавания была впервые опубликована Журавлёвым в 1976 году [3, 4]. До этого были известны только наиболее простые методы комбинирования алгоритмов, такие как взвешенное голосование и комитетные системы [7]. Многие из этих методов не обладали возможностью настройки корректирующей операции. Фактически, они рассматривались наравне с обычными методами распознавания и не носили характера систематической научной теории. Большое разнообразие методов комбинирования алгоритмов появилось существенно позже.

Методы, основанные на идее областей компетентности, были предложены Растригиным [9] и позже Джорданом и Джакобсом и др. в [15]. В 1990 была предло-

жена идея бустинга, позволяющего путём взвешенного голосования некорректных (слабых, weak) классификаторов построить корректный алгоритм [30]. Эффективный алгоритм AdaBoost на основе этой идеи был разработан лишь пятью годами позже [20]. Аналогичные методы, но с несколько иной стратегией оптимизации базовых операторов, были предложены Уолпертом [36] и Брейманом [17]. Следующим обобщением было введение нелинейных весовых функций в [34].

По отношению к алгебраическому подходу перечисленные методы являются частными случаями, отличаясь, главным образом, видом корректирующей операции. Алгебраический подход позволяет «порождать» такого рода методы с общих теоретических позиций. Наиболее абстрактным разделом алгебраического подхода является теория универсальных и локальных ограничений Рудакова. Она даёт критерии полноты, позволяющие строить семейства базовых алгоритмов и корректирующих операций минимальной достаточной сложности [12, 10, 11, 13].

Методы построения корректных алгоритмов, используемые для доказательства теорем существования в теоретических работах по алгебраическому подходу, плохо приспособлены для практического применения. Они не позволяют управлять сложностью композиции и приводят к построению громоздких алгоритмов, склонных к переобучению. На практике применяются проблемно-ориентированные методы алгебраического подхода, основанные на стратегии последовательной оптимизации базовых алгоритмов [1, 2, 14].

В современных обзорах по методам распознавания [24] и алгоритмическим композициям [33] подчеркивается, что построение композиций, в которых одни алгоритмы компенсируют недостатки других, является одним из наиболее перспективных направлений машинного обучения.

1.6.1 Понятия разрешимости, регулярности и полноты

1.6.2 Полнота линейных замыканий

Список литературы

[1] *Воронцов К. В.* О проблемно-ориентированной оптимизации базисов задач распознавания // *ЖВМ и МФ.* — 1998. — Т. 38, № 5. — С. 870–880.
`http:// www.ccas.ru/frc/papers/voron98jvm.pdf`.

[2] *Воронцов К. В.* Оптимизационные методы линейной и монотонной коррекции в алгебраическом подходе к проблеме распознавания // *ЖВМ и МФ.* — 2000. — Т. 40, № 1. — С. 166–176.
`http:// www.ccas.ru/frc/papers/voron00jvm.pdf`.

[3] *Журавлёв Ю. И.* Непараметрические задачи распознавания образов // *Кибернетика.* — 1976. — № 6.

[4] *Журавлёв Ю. И.* Экстремальные алгоритмы в математических моделях для задач распознавания и классификации // *Доклады АН СССР. Математика.* — 1976. — Т. 231, № 3.

[5] *Журавлёв Ю. И.* Об алгебраическом подходе к решению задач распознавания или классификации // *Проблемы кибернетики.* — 1978. — Т. 33. — С. 5–68.

[6] *Журавлёв Ю. И., Рудаков К. В.* Об алгебраической коррекции процедур обработки (преобразования) информации // *Проблемы прикладной математики и информатики.* — 1987. — С. 187–198.
http:// www.ccas.ru/frc/papers/zhurrud87correct.pdf.

[7] *Мазуров В. Д.* Комитеты системы неравенств и задача распознавания // *Кибернетика.* — 1971. — № 3.

[8] *Мазуров В. Д.* Метод комитетов в задачах оптимизации и классификации. — М.: Наука, 1990.

[9] *Растригин Л. А., Эренштейн Р. Х.* Коллективные правила распознавания. — М.: Энергия, 1981. — Р. 244.

[10] *Рудаков К. В.* Полнота и универсальные ограничения в проблеме коррекции эвристических алгоритмов классификации // *Кибернетика.* — 1987. — № 3. — С. 106–109.

[11] *Рудаков К. В.* Симметрические и функциональные ограничения в проблеме коррекции эвристических алгоритмов классификации // *Кибернетика.* — 1987. — № 4. — С. 73–77.
http:// www.ccas.ru/frc/papers/rudakov87symmetr.pdf.

[12] *Рудаков К. В.* Универсальные и локальные ограничения в проблеме коррекции эвристических алгоритмов // *Кибернетика.* — 1987. — № 2. — С. 30–35.
http:// www.ccas.ru/frc/papers/rudakov87universal.pdf.

[13] *Рудаков К. В.* О применении универсальных ограничений при исследовании алгоритмов классификации // *Кибернетика.* — 1988. — № 1. — С. 1–5.
http:// www.ccas.ru/frc/papers/rudakov88universal.pdf.

[14] *Рудаков К. В., Воронцов К. В.* О методах оптимизации и монотонной коррекции в алгебраическом подходе к проблеме распознавания // *Докл. РАН.* — 1999. — Т. 367, № 3. — С. 314–317.
http:// www.ccas.ru/frc/papers/rudvoron99dan.pdf.

[15] Adaptive mixtures of local experts / R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton // *Neural Computation.* — 1991. — no. 3. — Pp. 79–87.

[16] Boosting the margin: a new explanation for the effectiveness of voting methods / R. E. Schapire, Y. Freund, W. S. Lee, P. Bartlett // *Annals of Statistics.* — 1998. — Vol. 26, no. 5. — Pp. 1651–1686.
http:// citeseer.ist.psu.edu/article/schapire98boosting.html.

[17] *Breiman L.* Bagging predictors // *Machine Learning.* — 1996. — Vol. 24, no. 2. — Pp. 123–140.
http:// citeseer.ist.psu.edu/breiman96bagging.html.

[18] *Breiman L.* Bias, variance, and arcing classifiers: Tech. Rep. 460: Statistics Department, University of California, 1996.
http:// citeseer.ist.psu.edu/breiman96bias.html.

[19] *Breiman L.* Arcing classifiers // *The Annals of Statistics.* — 1998. — Vol. 26, no. 3. — Pp. 801–849.
http:// citeseer.ist.psu.edu/breiman98arcing.html.

[20] *Freund Y., Schapire R. E.* A decision-theoretic generalization of on-line learning and an application to boosting // European Conference on Computational Learning Theory. — 1995. — Pp. 23–37.
http:// citeseer.ist.psu.edu/article/freund95decisiontheoretic.html.

[21] *Freund Y., Schapire R. E.* Experiments with a new boosting algorithm // International Conference on Machine Learning. — 1996. — Pp. 148–156.
http:// citeseer.ist.psu.edu/freund96experiments.html.

[22] *Friedman J., Hastie T., Tibshirani R.* Additive logistic regression: a statistical view of boosting: Tech. rep.: Dept. of Statistics, Stanford University Technical Report, 1998.
http:// citeseer.ist.psu.edu/friedman98additive.html.

[23] *Grove A. J., Schuurmans D.* Boosting in the limit: Maximizing the margin of learned ensembles // AAAI/IAAI. — 1998. — Pp. 692–699.
http:// citeseer.ist.psu.edu/grove98boosting.html.

[24] *Jain A. K., Duin R. P. W., Mao J.* Statistical pattern recognition: A review // *IEEE Transactions on Pattern Analysis and Machine Intelligence.* — 2000. — Vol. 22, no. 1. — Pp. 4–37.
http:// citeseer.ist.psu.edu/article/jain00statistical.html.

[25] *Jordan M. I., Jacobs R. A.* Hierarchical mixtures of experts and the EM algorithm // *Neural Computation.* — 1994. — no. 6. — Pp. 181–214.
http:// citeseer.ist.psu.edu/article/jordan94hierarchical.html.

[26] *Marchand M., Shawe-Taylor J.* Learning with the set covering machine // Proc. 18th International Conf. on Machine Learning. — Morgan Kaufmann, San Francisco, CA, 2001. — Pp. 345–352.
http:// citeseer.ist.psu.edu/452556.html.

[27] *Ratsch G., Onoda T., Muller K.-R.* Soft margins for AdaBoost // *Machine Learning.* — 2001. — Vol. 42, no. 3. — Pp. 287–320.
http:// citeseer.ist.psu.edu/ratsch00soft.html.

[28] *Ratsch G., Onoda T., Muller K. R.* An improvement of adaboost to avoid overfitting.
http:// citeseer.ist.psu.edu/6344.html.

[29] *Rivest R. L.* Learning decision lists // *Machine Learning.* — 1987. — Vol. 2, no. 3. — Pp. 229–246.
http:// citeseer.ist.psu.edu/rivest87learning.html.

[30] *Schapire R. E.* The strength of weak learnability // *Machine Learning.* — 1990. — Vol. 5. — Pp. 197–227.
http:// citeseer.ist.psu.edu/schapire90strength.html.

[31] *Schapire R.* The boosting approach to machine learning: An overview // MSRI Workshop on Nonlinear Estimation and Classification, Berkeley, CA. — 2001.
http:// citeseer.ist.psu.edu/schapire02boosting.html.

[32] *Skurichina M., Kuncheva L., Duin R.* Bagging and boosting for the nearest mean classifier: Effects of sample size on diversity and accuracy // Multiple Classifier Systems (Proc. Third International Workshop MCS, Cagliari, Italy) / Ed. by J. K. F. Roli. — Vol. 2364. — Springer, Berlin, 2002. — Pp. 62–71.
http:// citeseer.ist.psu.edu/539135.html.

[33] *Tresp V.* Committee machines // Handbook for Neural Network Signal Processing / Ed. by Y. H. Hu, J.-N. Hwang. — CRC Press, 2001.
http:// citeseer.ist.psu.edu/tresp01committee.html.

[34] *Tresp V., Taniguchi M.* Combining estimators using non-constant weighting functions // Advances in Neural Information Processing Systems / Ed. by G. Tesauro, D. Touretzky, T. Leen. — Vol. 7. — The MIT Press, 1995. — Pp. 419–426.
http:// citeseer.ist.psu.edu/tresp95combining.html.

[35] *Vapnik V.* Statistical Learning Theory. — Wiley, New York, 1998.

[36] *Wolpert D. H.* Stacked generalization // *Neural Networks*. — 1992. — no. 5. — Pp. 241–259.
http:// citeseer.ist.psu.edu/wolpert92stacked.html.