УДК 519.7:004.855.5

КОЭВОЛЮЦИОННЫЙ МЕТОД ОБУЧЕНИЯ АЛГОРИТМИЧЕСКИХ КОМПОЗИЦИЙ

К.В. Воронцов, Д.Ю. Каневский

Вычислительный центр имени А. А. Дородницына Российской академии наук
Москва, ГСП-1, ул. Вавилова, 40, 119991
e-mail: *voron@ccas.ru*, *kanevskiy@forecsys.ru*

Abstract

The new method for ensemble learning is proposed — Cooperative Coevolution Ensemble Learner (CCEL), based on special genetic algorithm. It can be used as a wrapper over any kind of weak algorithms, learning procedures and fusion functions. Experiments on 12 real-world problems from UCI repository show that CCEL has a fairly high generalization performance and constructs ensembles of much smaller size than boosting, bagging and random subspaces method.

Введение

При решении сложных задач обучения по прецедентам (классификации, регрессии, прогнозирования) часто оказывается, что ни один из имеющихся эвристических алгоритмов не даёт желаемого качества обучения. В таких случаях имеет смысл строить композиции алгоритмов, которые позволяют компенсировать недостатки одних алгоритмов достоинствами других. При определённых условиях качество композиции оказывается заметно лучше, чем качество отдельных составляющих её алгоритмов. Построение алгоритмических композиций является одним из наиболее перспективных направлений машинного обучения [3, 20, 26].

Наиболее общей теорией алгоритмических композиций является алгебраический подход к проблеме распознавания, развиваемый научной школой академика РАН Ю. И. Журавлёва [4, 5, 6, 7]. В этом подходе искомый алгоритм строится в виде композиции некоторого количества алгоритмических операторов, корректирующей операции и решающего правила (в зарубежных исследованиях алгоритмические операторы принято называть *базовыми алгоритмами*).

В рамках алгебраического подхода разработаны методы построения последовательностей базовых алгоритмов — так называемых локальных базисов [11]. Каждый базовый алгоритм настраивается таким образом, чтобы наилучшим образом компенсировать совокупную погрешность предыдущих алгоритмов. Для решения данной задачи оказывается возможным приспособить стандартные методы обучения базовых алгоритмов, многократно подавая им на вход одну и ту же обучающую выборку, но модифицируя каждый раз вектор весов объектов и/или вектор ответов. Формулы пересчёта весов и ответов су-

щественно зависят от типа корректирующей операции. Конкретные формулы были получены для линейных, полиномиальных и монотонных корректирующих операций [11].

В современных зарубежных исследованиях наиболее близкими к алгебраическому подходу являются алгоритмы *бустинга* (boosting) [25], которые можно рассматривать как частный случай построения локального базиса. В бустинге всегда применяется линейная корректирующая операция, в то время как алгебраический подход свободен от такого рода ограничений. Наиболее известен алгоритм AdaBoost, который предложили в 1995 году американские учёные Р. Френд и И. Шапир [25]. В ряде экспериментов на реальных данных этот алгоритм показал практически неограниченное улучшение обобщающей способности (частоты ошибок на тестовой выборке) по мере наращивании числа алгоритмов в композиции. Причём во многих случаях обобщающая способность продолжала улучшаться даже после достижения безошибочного распознавания обучающей выборки [16]. Процесс последовательного наращивания композиции удобен тем, что для построения базовых алгоритмов можно применять стандартные семейства алгоритмов и стандартные методы их обучения. Благодаря высокой эффективности, универсальности а также простоте реализации алгоритм AdaBoost быстро завоевал широкую известность.

Характерной особенностью бустинга является применение гладкой аппроксимации пороговой функции потерь. Этот приём позволяет аналитически вывести формулы для вычисления весов объектов и коэффициентов при базовых алгоритмах. Однако неявная подмена оптимизируемого функционала приводит к снижению качества отдельных базовых алгоритмов и избыточному увеличению их количества. Другой недостаток бустинга заключается в том, что жадная стратегия последовательного наращивания композиции в общем случае строит неоптимальный набор базовых алгоритмов. Роль каждого алгоритма в композиции ограничивается компенсацией ошибок предыдущих алгоритмов, построенных раньше него. При этом никак не учитываются последующие алгоритмы. В результате каждый базовый алгоритм оказывается неоптимальным. Это также приводит к увеличению сложности композиции и, как следствие, ухудшению её обобщающей способности. В работах [17, 23, 24] экспериментально показано, что бустинг начинает переобучаться, если в композицию включено слишком много базовых алгоритмов — порядка 10^3. При этом в отдельных задачах переобучение наступает гораздо раньше.

Метод бэггинга (bagging — акроним от «bootstrap aggregation») был предложен Л. Брейманом в 1996 году [14]. Основная его идея заключается в том, чтобы строить базовые алгоритмы независимо друг от друга, но так,

чтобы они получались существенно различными. Один из способов добиться различности — обучать базовые алгоритмы не по всей выборке, а по различным её частям. Для этого применяется процедура *бутстреппинга* (bootstrapping): из множества объектов обучающей выборки формируются *бутстреп-выборки* — случайные подмножества объектов, как правило, с повторами. Затем настроенные на этих подвыборках базовые алгоритмы объединяются (aggregate) путём простого или взвешенного голосования.

Другой способ повышения различности (diversity) базовых алгоритмов — обучение на разных частях признакового описания. В методе *случайных подпространств* (Random Subspace Method, RSM) выделение подмножеств признаков производится случайным образом [19]. Построенные алгоритмы объединяются в композицию также путём простого или взвешенного голосования. Известны и другие методы настройки по подмножествам признаков, так или иначе использующие стохастические приёмы [27, 28, 29].

К сожалению, стратегии независимого построения базовых алгоритмов также могут приводить к построению неоптимальных, чрезмерно сложных композиций. В то же время, задача минимизации сложности композиции при одновременном обучении базовых алгоритмов и корректирующей операции является чрезвычайно трудоемкой, многопараметрической и многоэкстремальной оптимизационной проблемой.

При практическом решении задач глобальной оптимизации неплохо зарекомендовали себя *генетические алгоритмы*, основанные на принципах дарвиновской эволюции. В общем случае генетический алгоритм формирует популяцию индивидов, каждый из которых представляет собой потенциальное решение поставленной задачи. Для таких индивидов определяются генетические операции, позволяющие порождать новые поколения индивидов. Обычно это бинарная операция рекомбинации и унарная — мутации. Затем запускается эволюционный процесс смены поколений, в ходе которого производится селекция индивидов, наилучших с точки зрения качества решения задачи.

Известны различные способы применения генетических алгоритмов в распознавании. Во-первых, для решении задачи отбора признаков (features selection), которая заключается в том, чтобы выбрать подмножество наиболее информативных признаков, позволяющих построить классификатор наилучшего качества. В этом случае индивидами являются подмножества признаков, а рекомбинация — это обмен элементами между двумя подмножествами. Близким к генетическому алгоритму является случайный поиск с адаптацией [9]. Во-вторых, генетические алгоритмы применяются непосредственно для оптимизации структуры классификатора. Например, в [12] генетический алгоритм используется для синтеза логических правил

классификации, имеющих вид конъюнкций. Известны также методы построения решающих деревьев, когда рекомбинация реализуется как обмен поддеревьями у пары решающих деревьев. Во всех подобных случаях генетический алгоритм обязан «знать» внутреннее устройство алгоритма классификации. Фактически, он используется как метод настройки конкретной модели алгоритмов.

В данной работе развивается другой подход, при котором генетический алгоритм «не знает» внутреннего устройства базовых алгоритмов, и выступает лишь в роли надстройки (wrapper) над стандартными методами обучения. В этом варианте генетический алгоритм обобщает идеи бэггинга и RSM. Но, в отличие от них, предлагаемый алгоритм оптимизирует все базовые алгоритмы и корректирующую операцию одновременно. Для этого применяется особая модель эволюции, в которой несколько популяций алгоритмов развиваются параллельно и независимо друг от друга. От каждой популяции выделяется по одному индивиду (базовому алгоритму) для образования композиции. Качество каждого члена композиции оценивается тем, насколько хороша вся композиция в целом. В результате индивиды обучаются решению исходной задачи не по-отдельности, а в симбиозе друг с другом. Такая генетическая модель известна под названием *кооперативной коэволюции* [22].

Анализ последних достижений и публикаций, посвященных проблеме обучения алгоритмических композиций позволяет сделать вывод, что существующие методы (алгебраический подход, бустинг, бэггинг, RSM) могут приводить к построению чрезмерно сложных композиций, склонных к переобучению. *Нерешенным является вопрос о том, каким образом на практике строить композиции, состоящие из минимального или близкого к минимальному числа базовых алгоритмов.*

Целью настоящей работы является разработка и исследование коэволюционного метода обучения алгоритмических композиций, применимого для любых семейств алгоритмов и корректирующих операций, и позволяющего строить композиции высокого качества и малой мощности.

В разделе 1 даётся общая постановка задачи обучения по прецедентам и вводятся основные понятия. В разделе 2 определяется понятие алгоритмической композиции и рассматриваются известные методы построения композиций. В разделе 3 предлагается коэволюционный метод построения композиций CCEL (Cooperative Coevolution Ensemble Learner). В разделе 4 проводится эмпирическое сравнение предлагаемого метода с методами бустинга, бэггинга и RSM на множестве реальных задач классификации из репозитария UCI. В заключении отмечаются достоинства и недостатки нового метода, перечисляются основные направления дальнейших исследований.

1. Задача обучения по прецедентам

Имеется множество *объектов* X, множество *ответов* Y и множество $\mathfrak{A}$ отображений из X в Y, элементы которого будем называть алгоритмами, имея в виду, что они являются эффективно вычислимыми функциями. Предполагается, что существует отображение $y^*: X \to Y$, не обязательно принадлежащее $\mathfrak{A}$, называемое *восстанавливаемой зависимостью*. Задано конечное множество объектов $X^\ell = \{x_1, \ldots, x_\ell\}$, называемое *обучающей выборкой*. Значения отображения $y^*(x)$ известны только на этих объектах, $y_i = y^*(x_i)$.

Задача обучения по прецедентам заключается в том, чтобы построить алгоритм $a \in \mathfrak{A}$, выдающий на объектах обучающей выборки заданные ответы: $a(x_i) = y_i$, $i = 1, \ldots, \ell$. Кроме того, искомый алгоритм a должен обладать *способностью к обобщению*, то есть приближать восстанавливаемую зависимость y^* не только на объектах обучающей выборки, но и на всём множестве X.

Функционал *средней ошибки* алгоритма a на конечной выборке $X' \subset X^\ell$ есть

$$Q(a, X') = \frac{1}{|X'|} \sum_{x_i \in X'} L(a(x_i), y_i),$$

где $L: Y \times Y \to \mathbb{R}_+$ — *функция потерь*. В задачах классификации обычно полагают $L(y, y') = [y \neq y']$, тогда $Q(a, X')$ — это частота ошибок алгоритма a на выборке X'. В общем случае предполагается, что $L(y, y') = 0$ при $y = y'$.

Будем рассматривать задачи, в которых объекты описываются конечным набором признаков $G = \{g_j: X \to D_j \mid j = 1, \ldots, n\}$, где D_j — множество значений j-го признака.

Методом обучения называется отображение $\mu: 2^F \times 2^X \to \mathfrak{A}$, которое произвольному подмножеству признаков $G' \subseteq G$ и произвольной конечной обучающей выборке $X' \subset X$ ставит в соответствие определённый алгоритм. Говорят также, что алгоритм $a = \mu(G', X')$ *обучен* (или *настроен*) по выборке X' на подмножестве признаков G'. При этом часто предполагается, что метод обучения минимизирует эмпирический риск в заданном семействе алгоритмов $A \subset \mathfrak{A}$:

$$\mu(G', X') = \arg \min_{a \in A(G')} Q(a, X'), \tag{1}$$

где $A(G') \subseteq A$ — подсемейство алгоритмов, использующих только признаки подмножества G'. Однако мы не будем требовать, чтобы μ обязательно был методом минимизации эмпирического риска.

Методом обучения с весами будем называть отображение $\mu(G', X', W)$, которое дополнительно на входе принимает вектор весов объектов W, как правило, неотрицательных. Аналогично (1), в стандартном случае

предполагается, что метод обучения с весами минимизирует взвешенную среднюю ошибку:

$$\mu(G', X', W) = \arg\min_{a \in A(G')} \sum_{x_i \in X'} w_i L(a(x_i), y_i),$$

где w_i — вес или степень важности объекта x_i.

2. Алгоритмические композиции

Наиболее общее определение алгоритмической композиции даётся в алгебраическом подходе Ю. И. Журавлёва [7]. 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*.

Алгоритмической композицией, составленной из алгоритмических операторов $b_j\colon X \to R$, $j = 1, \ldots, p$, *корректирующей операции* $F\colon R^p \to R$ и решающего правила $C\colon R \to Y$ называется алгоритм $a\colon X \to Y$ вида

$$a(x) = C\big(F(b_1(x), \ldots, b_p(x))\big), \quad x \in X. \tag{2}$$

На практике решающее правило часто полагают фиксированным. Ориентируясь только на этот случай, отойдём от классического определения, и будем рассматривать композиции вида

$$a(x) = F(b_1(x), \ldots, b_p(x)),$$

где корректирующая операция F действует непосредственно в пространство ответов, $F\colon R^p \to Y$.

В англоязычной литературе алгоритмические операторы b_j принято называть *вещественнозначными классификаторами* (real-valued classifiers), если речь идёт о задачах классификации, а в более общем случае — *базовыми алгоритмами* (base algorithms).

Известные методы построения композиций (локальные базисы, бустинг, бэггинг, RSM) допускают запись в виде обобщённого Алгоритма 1.

При построении локальных базисов веса объектов W и ответы Y^ℓ пересчитываются на шаге 7 по специальным формулам [1, 2].

В методе бустинга на шаге 7 пересчитываются только веса объектов [25].

В методе бэггинга вектор весов является бинарным, то есть некоторые объекты исключаются из обучения, как правило, случайным образом [14].

В методе RSM пересчитываются (также случайным образом) только подмножества признаков G' на шаге 3; веса W и ответы Y^ℓ не изменяются [19].

Критерием останова в простейшем случае является построение заданного числа базовых алгоритмов p. Используются также условия стагнации функционала Q_j, либо увеличения (перехода через минимум) функционала

Алгоритм 1. Построение алгоритмической композиции путём последовательного обучения базовых алгоритмов.

Вход:

выборка объектов X^ℓ и соответствующие им ответы $Y^\ell = \{y_i\}_{i=1}^\ell$;
набор признаков $G = \{g_1, \ldots, g_n\}$;
метод обучения базовых алгоритмов μ;

Выход:

алгоритмическая композиция $F(b_1, \ldots, b_p)$;

1: **для всех** $i = 1, \ldots, \ell$
2: инициализировать вес $w_i := 1$;
3: **для всех** $j = 1, \ldots, p$
4: сформировать подмножество признаков $G' \subseteq G$;
5: настроить j-й базовый алгоритм:
$b_j := \mu(G', X^\ell, W)$;
6: вычислить среднюю ошибку композиции на выборке X^ℓ:
$Q_j := Q\big(F(b_1, \ldots, b_j), X^\ell\big)$;
7: модифицировать веса W^ℓ и/или ответы Y^ℓ, используя значение средней ошибки $Q(b_j, X^\ell)$ и/или $Q(b_j, X^\ell, W)$ и/или Q_j;
8: **если** выполнен критерий останова **то**
9: прервать построение композиции;

средней ошибки $Q\big(F(b_1, \ldots, b_j), X^q\big)$ на заранее выделенной тестовой выборке X^q.

Общим недостатком этих методов является неоптимальность базовых алгоритмов с точки зрения композиции, что приводит к чрезмерному усложнению композиции. Кроме того, последовательные методы (локальные базисы и бустинг) требуют, чтобы метод обучения учитывал веса объектов. Для многих стандартных алгоритмов это ограничение является обременительным. С другой стороны, методы независимого построения базовых алгоритмов (бэггинг и RSM), хотя и не требуют обучения с весами, используют слишком примитивные стратегии отбора объектов и признаков для настройки базовых алгоритмов.

3. Метод кооперативной коэволюции CCEL

Итак, построение алгоритмической композиции сводится к решению оптимизационной задачи

$$Q\big(F(b_1, \ldots, b_p), X^\ell\big) \to \min,$$

где минимизация должна производиться одновременно по всем базовым алгоритмам $b_1, \ldots, b_p$ и корректирующей операции F. Рассмотренные выше методы пытаются свести данную задачу к поочерёдному построению базовых алгоритмов, при этом результат может оказаться весьма далёким от оптимального.

Рассмотрим другой подход к поиску глобального минимума, основанный на специальном генетическом алгоритме, называемом *кооперативной коэволюцией* [22]. Как всякий эволюционный алгоритм, он представляет собой итерационный процесс смены поколений, см. Алгоритм 2.

Инициализация нулевого поколения производится случайным образом.

На t-м поколении имеется $p(t)$ популяций $\Pi_1(t), \ldots, \Pi_{p(t)}(t)$. Каждая популяция $\Pi_j(t)$ представляет собой множество индивидов. Каждый индивид кодируется $(\ell + n)$-мерным бинарным вектором, составленным из характеристических векторов подмножества объектов $X' \subseteq \{x_1, \ldots, x_\ell\}$ и подмножества признаков $G' \subseteq \{g_1, \ldots, g_n\}$. Таким образом, каждый индивид v_j из популяции $\Pi_j(t)$ соответствует некоторой паре подмножеств (G', X'), к которой можно применить метод обучения и получить базовый алгоритм $b_j = \mu(G', X') \equiv \mu(v_j)$.

Для оценки *адаптивности* (fitness) индивида v_j вычисляется оценка качества композиции, в которой на j-м месте стоит алгоритм b_j, а на всех остальных местах — наиболее адаптивные базовые алгоритмы b_s^*, взятые по одному из каждой популяции $\Pi_s(t)$, $s \neq j$:

$$\varphi(v_j) = Q\big(F(b_1^*, \ldots, b_{j-1}^*, \mu(v_j), b_{j+1}^*, \ldots, b_{p(t)}^*), X^\ell\big),$$

Адаптивность оценивается для каждого индивида в каждой популяции.

Затем производится селекция индивидов. В каждой популяции отбирается ограниченное количество наиболее адаптивных индивидов. К ним применяются генетические операции рекомбинации (crossover), мутации и селекции, в результате которых порождается новое поколение $\Pi_j(t+1)$, $j = 1, \ldots, p(t)$. Из каждого поколения t отбирается и запоминается лучшая композиция вида $F(b_1, \ldots, b_{p(t)})$.

В момент смены поколений может приниматься решение об увеличении или уменьшении числа популяций. Если популяция даёт слишком малый вклад в композицию в течение некоторого числа поколений, то она удаляется. Если процесс эволюции вошел в состояние стагнации, и качество лучших композиций перестало заметно изменяться, то создаётся новая популяция. Процесс эволюции прекращается при выполнении заданного критерия останова.

Основное отличие кооперативной коэволюции от стандартных генетических алгоритмов в том, что функция адаптивности оценивает не качество алгоритма b_j в отдельности, а его полезность для композиции. Таким

Алгоритм 2. Коэволюционный метод обучения алгоритмических композиций CCEL.

Вход:

X^ℓ, Y^ℓ, G, μ — аналогично Алгоритму 1;
$T_{\max}$ — максимальное число поколений;
N_0 — размер основной популяции;
N_1 — размер промежуточной популяции;
N_2 — размер элиты, переходящей в следующее поколение без изменений;

Выход:

алгоритмическая композиция $F(b_1, \ldots, b_p)$;

начальное число популяций: $p(1) := 1$;
создать популяцию: $\Pi_1(1) := \texttt{Инициализация}(N_0)$;
для всех поколений $t := 1, \ldots, T_{\max}$
  для всех популяций $\Pi_j(t)$, $j := 1, \ldots, p(t)$
    породить промежуточную популяцию:
      $\Pi'_j := \texttt{Рекомбинация}(\Pi_j(t), N_1)$;
      $\Pi''_j := \texttt{Мутация}(\Pi'_j) \cup \texttt{Селекция}(\Pi_j(t), N_2)$;
    для всех $v_j \in \Pi''_j$
      оценить адаптивность индивида $\varphi(v_j)$;
      если $\varphi(v_j) < Q_t$ **то**
        $Q_t := \varphi(v_j)$;
      запомнить лучшего индивида в популяции:
        $v_j^* := \arg\min\limits_{v \in \Pi''_j} \varphi(v)$; $b_j^* := \mu(v_j^*)$;
    отобрать лучших индивидов в следующее поколение:
      $\Pi_j(t+1) := \texttt{Селекция}(\Pi''_j, N_0)$;
    если $\texttt{Вклад}(\Pi_j) < \delta$ **то**
      удалить популяцию Π_j;
      $p(t+1) := p(t) - 1$;
  если $\texttt{Стагнация}(Q, t)$ **то**
    $p(t+1) := p(t) + 1$;
    добавить популяцию $\Pi_{p(t+1)}(t+1) := \texttt{Инициализация}(N_0)$;
  если $\texttt{Останов}(Q, t)$ **то**
    выход;
вернуть композицию $F\left(b_1^*, \ldots, b_{p(t)}^*\right)$, на которой достигается $\min\limits_t Q_t$.

образом, в ходе эволюции базовые алгоритмы обучаются кооперировать друг с другом с целью наилучшего решения поставленной задачи. При этом каждая

популяция (следовательно, каждый базовый алгоритм) специализируется в своей области объектов и в своём подпространстве признаков.

Описанный коэволюционный метод обучения композиций получил название CCEL — Cooperative Coevolution Ensemble Learner. Как и всякий эволюционный алгоритм, он имеет большое число параметров и большую свободу выбора различных эвристик. Рассмотрим некоторые из них более подробно.

`Инициализация`(N_0) — процедура, создающая N_0 индивидов. Каждый индивид формируется случайным образом. При этом должны задаваться дополнительные параметры алгоритма — вероятность включения объекта q_X и вероятность включения признака q_G.

`Селекция`(Π, N) — генетическая операция, отбирающая N наиболее адаптивных индивидов популяции Π. В методе CCEL она используется дважды. На шаге 5 — для переноса лучших индивидов (*элиты*) из основной популяции родителей в промежуточную популяцию потомков. На шаге 11 — для естественного отбора, при котором из N_1 потомков только N_0 лучших переносятся в основную популяцию следующего поколения.

`Рекомбинация`(Π, N_1) — генетическая операция, порождающая N_1 новых индивидов путём попарного скрещивания индивидов популяции Π. Родительские пары выбираются случайным образом, возможно, с поощрением наиболее адаптивных индивидов. Для формирования потомка в методе CCEL используется так называемый *равномерный кроссинговер* (uniform crossover) — каждый бит потомка равновероятно выбирается у одного из родителей.

`Мутация`(Π) — генетическая операция, производящая случайные изменения в индивидах популяции Π. В качестве дополнительного параметра алгоритма может задаваться вероятность инверсии бита в бинарном коде индивида.

`Вклад`(Π_j) — функция, оценивающая вклад популяции Π_j в композицию. При использовании *проверки изъятием* строятся две композиции — с участием и без участия j-го базового алгоритма. Соответственно, вычисляются оценки их качества: Q_{jt} и $\bar{Q}_{jt}$. Вклад j-й популяции вычисляется как средняя разность этих оценок за последние d итераций: $\frac{1}{d}\sum_{\tau=t-d+1}^{t}\bigl(\bar{Q}_{j\tau} - Q_{j\tau}\bigr)$. Если вклад слишком мал, популяция целиком удаляется. Метод проверки изъятием — надёжный, но ресурсоёмкий. В данном исследовании использовался другой приём: поскольку строилась линейная корректирующая операция (см. ниже), вклад оценивался как среднее значение веса j-го базового алгоритма за последние d итераций: $\frac{1}{d}\sum_{\tau=t-d+1}^{t}\alpha_j(\tau)$. Если это значение оказывалось меньше некоторого малого положительного δ, популяция целиком удалялась.

`Стагнация`(Q, t) — критерий, проверяющий наступление стагнации в последовательности $Q = \{Q_t\}$ в момент времени t. Например, это может быть отсутствие заметных улучшений качества на протяжении последних d поколений: $Q_{t-d} - Q_t < \varepsilon$, где d, ε — параметры алгоритма. При наступлении

стагнации создаётся новая популяция, в надежде на то, что увеличение сложности композиции сможет привести к улучшению её качества.

`Останов`(Q,t) — критерий останова, проверяющий наступление длительной стагнации. Например, можно проверять условие $Q_{t-D} - Q_t < \varepsilon$, где $D \gg d$ — параметр алгоритма. Должно состояться по меньшей мере несколько безуспешных попыток добавить ещё одну популяцию, прежде чем станет ясно, что дальнейшее изменение структуры композиции не способно её улучшить.

4. Экспериментальное исследование метода CCEL

В качестве полигона для экспериментов с методом CCEL были выбраны задачи классификации с двумя классами, $Y = \{-1, +1\}$.

Для построения базовых алгоритмов $b_j \colon X \to Y$ использовался *«наивный» байесовский классификатор* (naive bayes) основанный на предположении, что признаки $g_1(x), \ldots, g_n(x)$ являются независимыми случайными величинами. Одно только это предположение позволяет выписать алгоритм классификации в явном виде [18]. Основным преимуществом алгоритма является вычислительная эффективность: обучение производится за время, линейное по длине выборки, что крайне существенно для ресурсоёмких генетических алгоритмов. В то же время, качество этого алгоритма весьма посредственно, поскольку предположение о независимости признаков, как правило, не выполняется в реальных задачах классификации.

В качестве корректирующей операции применялось взвешенное голосование:

$$F(b_1, \ldots, b_p) = \operatorname{sgn}\left(\sum_{j=1}^{p} \alpha_j b_j\right), \quad \alpha_j \geqslant 0, \quad j = 1, \ldots, p. \tag{3}$$

Известны различные методы настройки весов α_j — от простого голосования, когда $\alpha_j \equiv 1$ и настройка фактически отсутствует, до применения машины опорных векторов. В данном исследовании веса оценивались по формуле, основанной на «наивном» предположении о независимости базовых алгоритмов $b_1(x), \ldots, b_p(x)$. При этом вес алгоритма оказывается тем меньше, чем больше ошибок он допускает [18]:

$$\alpha_j = \ln \frac{|X'| - E_j + 1}{E_j + 1},$$

где E_j — число ошибок базового алгоритма b_j на выборке X'. Поскольку в CCEL каждый базовый алгоритм настраивается по своей обучающей подвыборке $X'_j \subset X^\ell$, возникает альтернатива — какую совокупность объектов X' использовать для оценки весов: всю исходную выборку X^ℓ, обучающую подвыборку X'_j или её дополнение $X^\ell \setminus X'_j$. В данном исследовании был выбран третий вариант, поскольку в этом случае вес базового алгоритма

оценивается по контрольной выборке и характеризует его обобщающую способность. При $E_j \geqslant \frac{1}{2}|X'|$ полагается $\alpha_j = 0$, то есть алгоритм b_j исключается из композиции.

Отметим, что для взвешенного голосования не имеет смысла строить композиции мощности 2. Поэтому шаг 15 в методе CCEL был модифицирован таким образом, чтобы наращивание происходило с одного алгоритма сразу до трёх.

Функционал качества $Q(a, X')$ обычно определяют как величину средней ошибки композиции a на выборке X'. В то же время, известно, что обобщающая способность линейных композиций улучшается при явной максимизации отступов — расстояний от объектов обучения до границы классов [21]. Для композиции вида (3) *отступ* (margin) объекта $x_i \in X^\ell$ определяется как

$$M_i = y_i \sum_{j=1}^{p} \alpha_j b_j(x_i).$$

Отступ принимает отрицательные значения тогда и только тогда, когда композиция (3) ошибается на объекте x_i. Чем дальше объект отстоит от границы классов, тем больше абсолютное значение отступа. В данном исследовании максимизировался функционал качества $\widetilde{Q}$, равный среднему отступу объектов обучения:

$$\widetilde{Q}(a, X') = \frac{1}{|X'|} \sum_{x_i \in X'} M_i.$$

Для эмпирического оценивания обобщающей способности метода обучения M применялась процедура *скользящего контроля* (cross-validation, CV). Исходная выборка N раз случайным образом разделялась на обучающую и контрольную подвыборки: $X^\ell = X_n^t \cup X_n^c$, $n = 1, \ldots, N$. При этом отношение числа обучающих и контрольных объектов t/c соблюдалось как можно точнее внутри каждого класса. Итоговая оценка качества $\mathrm{CV}(M)$ вычислялась путём усреднения частоты ошибок на контроле по всем разбиениям:

$$\mathrm{CV}(M) = \frac{1}{N} \sum_{n=1}^{N} Q\big(M(X_n^t), X_n^c\big).$$

В данном исследовании было выбрано $N = 100$ и $t/c = 7/3$. Различные методы тестировались на одних и тех же разбиениях.

В экспериментах производилось сравнение метода CCEL с базовым алгоритмом, бустингом AdaBoost, бэггингом и RSM. Для них размер

Задача	Байес	CCEL	Бустинг	Бэггинг	RSM
Cancer	5.48	**4.0** (3.95)	4.44	5.63	5.05
Credit-a-1	15.30	**13.37** (4.52)	23.96	15.28	14.78
Credit-a-2	15.87	**14.79** (5.23)	18.88	15.86	15.95
Credit-g	32.11	**25.42** (2.25)	30.01	31.60	32.29
DBC	11.43	**5.61** (5.0)	23.61	11.35	11.44
Heart	19.52	**17.59** (3.65)	22.79	19.53	19.26
Hepatitis	**16.51**	17.64 (3.08)	20.17	16.60	16.57
Liver	37.06	35.59 (1.9)	**34.40**	37.03	36.84
Diabetes	31.80	**23.65** (3.38)	31.62	31.66	30.87
Survival	30.99	**26.45** (2.66)	27.11	29.52	28.02
Tic-tac-toe	28.92	**20.29** (1.95)	33.17	28.96	27.78
Voting	6.51	**5.17** (1.77)	7.1	6.55	6.33

ТАБЛИЦА 1. Результаты расчетов, произведенных для 12 задач из репозитория UCI. Приведены проценты ошибок на тестовой выборке. Для коэволюционного метода в скобках указано среднее число алгоритмов в композиции.

композиции $p = 250$ был фиксирован [14]. Для проведения экспериментов были наугад выбраны 12 задач с двумя классами из репозитория UCI [13]. В таблице 1 собраны результаты расчетов для указанных задач. Таблица содержит значения функционала CV в процентах. Для метода CCEL в скобках указано среднее число алгоритмов в композиции.

Из таблицы видно, что при выбранных методах обучения базовых алгоритмов и корректирующей операции коэволюционный метод заметно превосходит остальные. В 10 задачах из 12 он показал наилучшие результаты.

5. Выводы и направления дальнейшей работы

Основным результатом данной статьи является эмпирическое подтверждение того факта, что генетические методы глобальной оптимизации позволяют строить алгоритмические композиции низкой сложности и достаточно высокого качества.

Предложен коэволюционный алгоритм обучения алгоритмических композиций CCEL, обладающий следующими достоинствами:

- Небольшое число базовых алгоритмов в композиции. Благодаря коэволюции каждый алгоритм чётко специализируется в своей области объектов и в своём подпространстве признаков.
- Относительно высокая обобщающая способность по сравнению со стандартными методами построения композиций.

- Совместимость с любыми методами настройки базовых алгоритмов и корректирующих операций. Метод CCEL можно использовать в качестве надстройки над стандартными библиотеками алгоритмов.
- Невысокая требовательность к качеству базовых алгоритмов. Неплохие результаты удаётся получать даже для таких примитивных методов настройки, как «наивные» байесовские.

К недостаткам CCEL следует отнести:

- Низкую скорость обучения композиции.
- Отсутствие доказательств сходимости. В общем случае кооперативная коэволюция не гарантирует достижения глобального оптимума.

Представляется перспективным дальнейшее развитие коэволюционных методов построения алгоритмических композиций, в том числе:

- Повышение скорости работы алгоритма, возможно, за счёт замены генетического алгоритма на более эффективные методы стохастического поиска — микро-генетический алгоритм [15], случайный поиск с адаптацией [8], и др.
- Применение метода CCEL для различных моделей базовых алгоритмов и корректирующих операций. В частности, есть основания полагать, что коэволюционный алгоритм подходит для совместной настройки базовых алгоритмов и областей их компетентности [10].
- Включение в генетический алгоритм дополнительных эвристик, обеспечивающих построение композиций, корректных на обучающей выборке.

Работа выполнена при поддержке Российского фонда фундаментальных исследований (проекты №05-01-00877 и №04-07-90290) и программы ОМН РАН «Алгебраические и комбинаторные методы математической кибернетики».

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. *Воронцов К. В.* Комбинаторный подход к оценке качества обучаемых алгоритмов // *Математические вопросы кибернетики.* — 2004. — № 13. — С. 5–36.
`http://www.ccas.ru/frc/papers/voron04mpc.pdf`.
4. *Журавлёв Ю. И.* Корректные алгебры над множествами некорректных (эвристических) алгоритмов. Часть I // *Кибернетика.* — 1977. — № 4. — С. 5–17.

5. *Журавлёв Ю. И.* Корректные алгебры над множествами некорректных (эвристических) алгоритмов. Часть II // *Кибернетика.* — 1977. — № 6. — С. 21–27.
6. *Журавлёв Ю. И.* Корректные алгебры над множествами некорректных (эвристических) алгоритмов. Часть III // *Кибернетика.* — 1978. — № 2. — С. 35–43.
7. *Журавлёв Ю. И.* Об алгебраическом подходе к решению задач распознавания или классификации // *Проблемы кибернетики.* — 1978. — Т. 33. — С. 5–68.
8. *Загоруйко Н. Г., Ёлкина В. Н., Лбов Г. С.* Алгоритмы обнаружения эмпирических закономерностей. — Новосибирск: Наука, 1985.
9. *Лбов Г. С.* Методы обработки разнотипных экспериментальных данных. — Новосибирск: Наука, 1981.
10. *Растригин Л. А., Эренштейн Р. Х.* Коллективные правила распознавания. — М.: Энергия, 1981. — Р. 244.
11. *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`.
12. *Au W.-H., Chan K. C. C., Yao X.* A novel evolutionary data mining algorithm with applications to churn prediction // *IEEE Trans. Evolutionary Computation.* — 2003. — Vol. 7, no. 6. — Pp. 532–545.
`http://dblp.uni-trier.de`.
13. *Blake C., Merz C.* UCI repository of machine learning databases: Tech. rep.: Department of Information and Computer Science, University of California, Irvine, CA, 1998.
`http://www.ics.uci.edu/~mlearn/MLRepository.html`.
14. *Breiman L.* Arcing classifiers // *The Annals of Statistics.* — 1998. — Vol. 26, no. 3. — Pp. 801–849.
`http://citeseer.ist.psu.edu/breiman98arcing.html`.
15. *Coello Coello C. A., Toscano Pulido G.* A Micro-Genetic Algorithm for Multiobjective Optimization // First International Conference on Evolutionary Multi-Criterion Optimization / Ed. by E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, D. Corne. — Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001. — Pp. 126–140.
`http://citeseer.ist.psu.edu/444668.html`.
16. *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`.
17. *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`.
18. *Hastie T., Tibshirani R., Friedman J.* The Elements of Statistical Learning. — Springer, 2001.
19. *Ho T. K.* The random subspace method for constructing decision forests // *IEEE Transactions on Pattern Analysis and Machine Intelligence.* — 1998. — Vol. 20, no. 8. — Pp. 832–844.
`http://citeseer.ist.psu.edu/ho98random.html`.
20. *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`.

21. *Mason L., Bartlett P., Baxter J.* Direct optimization of margins improves generalization in combined classifiers: Tech. rep.: Deparment of Systems Engineering, Australian National University, 1998.
http://citeseer.ist.psu.edu/mason98direct.html.
22. *Potter M. A., De Jong K. A.* Cooperative coevolution: An architecture for evolving coadapted subcomponents // *Evolutionary Computation.* — 2000. — Vol. 8, no. 1. — Pp. 1–29.
http://citeseer.ist.psu.edu/potter00cooperative.html.
23. *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.
24. *Ratsch G., Onoda T., Muller K. R.* An improvement of adaboost to avoid overfitting.
http://citeseer.ist.psu.edu/6344.html.
25. *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.
26. *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.
27. *Tsymbal A., Puuronen S.* Ensemble feature selection with the simple bayesian classification in medical diagnostics // 15 IEEE Symp. on Computer-Based Medical Systems CBMS. — 2002. — P. 225.
http://citeseer.ist.psu.edu/623033.html.
28. *Zheng Z., Webb G. I.* Multi strategy ensemble learning: Reducing error by combining ensemble learning techniques // *IEEE transactions on knowledge and data engineering.* — 2004. — Vol. 16, no. 8. — Pp. 980–991.
http://www.csse.monash.edu.au/~webb/Files/WebbZheng03.pdf.
29. *Zheng Z., Webb G., Ting K.* Integrating boosting and stochastic attribute selection committees for further improving the performance of decision tree learning // Proceedings of the 10th IEEE International Conference on Tools with Artificial Intelligence. — IEEE Computer Society Press, 1998. — Pp. 216–223.
http://http://citeseer.ist.psu.edu/article/zheng98integrating.html.