# ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 004.852

## ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА ГРАДИЕНТНОГО БУСТИНГА ДЕРЕВЬЕВ РЕШЕНИЙ

© 2011 г. ***П.Н. Дружков, Н.Ю. Золотых, А.Н. Половинкин***

Нижегородский госуниверситет им. Н.И. Лобачевского

nikolai.zolotykh@gmail.com

*Поступила в редакцию 06.07.2010*

Описана программная реализация алгоритма градиентного бустинга деревьев решений. Приводятся результаты вычислительного эксперимента, показавшего конкурентоспособность предлагаемой программной реализации.

*Ключевые слова:* машинное обучение, бустинг, градиентный бустинг, деревья решений.

### Введение

Одной из задач, изучаемых в машинном обучении, является *задача обучения с учителем*. В рамках этой задачи дано некоторое множество *объектов X*. Каждому объекту $x \in X$ поставлена в соответствие величина *y*, называемая *выходом*, или *ответом*, принадлежащая множеству допустимых ответов *Y*. Упорядоченная пара «объект–ответ» $(x, y)$, где $x \in X$, $y \in Y$, называется *прецедентом*. Требуется восстановить зависимость между входом и выходом, основываясь на данных о конечном наборе прецедентов, называемом *обучающей выборкой*: $\{(x_i, y_i) \mid x_i \in X, y_i \in Y, i = \overline{1,N}\}$. Другими словами, задача состоит в построении *модели* (функции) *f*, которая, получив на вход *x*, предсказала бы значение ответа *y*. Процесс нахождения *f* называется *обучением*, или *настройкой*, *подгонкой* модели. В случае конечного *Y* говорят о *задаче классификации*, $Y = \mathbb{R}$ – *задаче восстановления регрессии* [1].

Основным требованием, предъявляемым к решению, является высокая обобщающая способность, т. е. обученная модель должна выдавать в среднем достаточно точные предсказания на новых (не входящих в обучающую выборку) прецедентах. Таким образом, оптимальное решение задачи индуктивного обучения должно удовлетворять условию:

$$f^* = \arg\min_{F \in K} M_{y,x} L(y, F(x)),$$

где $L(y, F(x))$ – неотрицательная функция потерь (штрафа), *K* – множество допустимых решений. Однако данный критерий неприменим в случае конечного набора известных данных и обычно заменяется на условие:

$$f = \arg\min_{F \in K} \sum_{i=1}^{N} L(y_i, F(x_i)),$$

где прецеденты $(x_i, y_i)$, $i = \overline{1,N}$, составляют обучающую выборку.

Один из общих подходов к решению задач обучения заключается в комбинировании моделей. Две основные конкурирующие идеи данного подхода – бэггинг (*bagging* от *Bootstrap Aggregating*) [2] и бустинг (*boosting*) [3]. Первая из них состоит в построении множества независимых (между собой) моделей с дальнейшим принятием решения путем голосования в случае задачи классификации и усреднения в случае регрессии. Данный подход реализован в алгоритме случайных деревьев (*random trees* или *random forest*). Основной сложностью применения этой идеи является обеспечение независимости построенных моделей. Бустинг, в противоположность бэггингу, обучает каждую следующую модель с использованием данных об ошибках предыдущих моделей.

В настоящей работе мы описываем алгоритм градиентного бустинга деревьев решений [4] и предлагаем программную реализацию этого метода. Насколько известно авторам, это первая открытая C/C++-реализация данного метода.

Результаты вычислительного эксперимента свидетельствуют о ее конкурентоспособности.

### Алгоритм градиентного бустинга деревьев решений

Основной задачей, которую требуется решить для обучения любой модели, является задача минимизации суммарного штрафа на прецедентах обучающей выборки:

$$\min_{F \in K} \mathfrak{L}(F), \quad (1)$$

где

$$\mathfrak{L}(F) = \sum_{i=1}^{N} L(y_i, F(x_i)).$$

Одним из методов ее приближенного решения служит жадная стратегия [1]. Так как суммарные потери $\mathfrak{L}(F)$ зависят не от самой функции $F(x)$, а лишь от ее значений в точках обучающей выборки, т. е. $\mathfrak{L}(F) = \mathfrak{L}(F(x_1), F(x_2),\ldots, F(x_N))$, на задачу (1) можно смотреть как на минимизацию функции $N$ переменных. Для этой задачи существует множество численных итерационных процедур. Все они ищут точку оптимума в виде суммы последовательных приближений $h_M = \sum_{m=0}^{M} F_m$, где $F_m \in \mathbb{R}^N$, а $F_0$ – начальная точка. Принципиальное различие между такими численными методами заключается в способе вычисления очередного слагаемого $F_m$. Так, метод наискорейшего градиентного спуска делает шаги в направлении антиградиента функции в текущей точке, т.е. $F_m = -\rho_m g_m$, где $g_m = \operatorname{grad}(\mathfrak{L}(F))_{F=h_{m-1}} =$

$$\left(\frac{\partial L(y_1, F(x_1))}{\partial F(x_1)}, \frac{\partial L(y_2, F(x_2))}{\partial F(x_2)}, \ldots, \frac{\partial L(y_N, F(x_N))}{\partial F(x_N)}\right)_{F=h_{m-1}}$$

а $\rho_m$ определяет длину шага и является решением одномерной задачи минимизации по направлению $g_m$:

$$\rho_m = \arg\min_{\rho} \mathfrak{L}(h_{m-1} - \rho g_m)$$

$$= \arg\min_{\rho} \sum_{i=1}^{N} L(y_i, h_{m-1}(x_i) - \rho g_{m,i}).$$

После осуществляется переход к следующей точке $h_m = h_{m-1} - \rho_m g_m$.

Метод градиентного спуска позволяет произвести минимизацию суммарного штрафа на обучающей выборке для любой дифференцируемой функции потерь, но нашей целью является вычисление функции $f = h_M$ в новых точках $x$, а вектор $g_m$ определен только на прецедентах $(x_i, y_i)$, $i = \overline{1, N}$. Данную проблему можно решить путем обучения базовой модели таким образом, чтобы она как можно точнее предсказывала компоненты антиградиента [4]. При этом возможно применение любой функции потерь $\Psi(y, h(x))$, весь вопрос в существовании соответствующего алгоритма обучения. Также алгоритм градиентного бустинга допускает использование любой функции регрессии в качестве базовой. Одним из наиболее популярных выборов являются деревья решений. Преимущества их использования заключаются в следующем: они позволяют производить обучение на исходных данных без их дополнительной предобработки, поддерживают наличие пропущенных значений, номинальных и количественных переменных, и, что немаловажно, существуют эффективные алгоритмы их обучения (CART [5], C4.5 [6]). Едва ли не единственным их недостатком является зачастую низкое качество обучения, но в бустинг-методах они хорошо зарекомендовали себя.

Дерево решений разбивает множество допустимых входных векторов $X$ на $J$ непересекающихся подмножеств $R_j$, $j = \overline{1, J}$, где $J$ – количество листьев в дереве. Каждой области $R_j$ приписана некоторая константа $\varphi_j$. Формально дерево можно представить в виде

$$T(x; \Theta) = \sum_{j=1}^{J} \varphi_j \cdot 1(x \in R_j),$$

где $\Theta$ – набор параметров, определяющих конкретное дерево решений: $\Theta = \{(R_j, \varphi_j), j = \overline{1, J}\}$, а $1(z)$ – функция, такая, что $1(z) = 1$, если $z$ истинно, и $1(z) = 0$, если $z$ ложно.

Как правило, $J$ является параметром, задаваемым до начала процесса обучения, а $\Theta$ подбирается в ходе настройки модели в два этапа: на первом производится разбиение множества $X$ на $R_j$, $j = \overline{1, J}$, а на втором вычисляются константы $\varphi_j$:

$$\hat{R} = (\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_J) = \arg\min_{\Theta} \sum_{i=1}^{N} \Psi(y_i, T(x_j; \Theta)),$$

$$\hat{\varphi}_j = \arg\min_{\varphi_j} \sum_{x_i \in \hat{R}_j} \Psi(y_i, \varphi_j), \quad j = \overline{1, J}.$$

Алгоритм 1 представляет собой частный случай метода градиентного бустинга при использовании деревьев решений в качестве базовых моделей с их обучением с применением квадратичного штрафа.

**Алгоритм 1.** *Градиентный бустинг деревьев решений для задачи восстановления регрессии* [3; 6].

1. Взять оптимальное константное решение за начальное приближение

$$h_0 = F_0(x) = \arg\min_{\rho} \sum_{i=1}^{N} L(y_i, \rho) .$$

2. Для всех $m = 1, \ldots, M$:

a) Вычислить компоненты вектора антиградиента

$$r_{i,m} = -\left[ \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F=h_{m-1}}, \quad i = \overline{1, N} .$$

b) Построить регрессионное дерево на выборке $\{(x_i, r_{i,m}), i = \overline{1, N}\}$

$$\hat{R}_m = (\hat{R}_{1,m}, \hat{R}_{2,m}, \ldots, \hat{R}_{J_m,m}) = \arg\min_{\Theta} \sum_{i=1}^{N} (r_{i,m} - T(x_j; \Theta))^2 .$$

c) Найти оптимальные константы для каждого листа дерева

$$\hat{\varphi}_{j,m} = \arg\min_{\varphi} \sum_{x_i \in \hat{R}_{j,m}} L(y_i, h_{m-1}(x_i) + \varphi), \quad j = \overline{1, J_m} .$$

d) Обновить модель

$$h_m(x) = h_{m-1}(x) + \nu \cdot \sum_{j=1}^{J_m} \hat{\varphi}_{j,m} \cdot 1(x \in \hat{R}_{j,m}) .$$

3. Конечная модель $f(x) = h_M(x)$.

Следует отметить, что для деревьев решений вычисление «длины шага» $\varphi_{j,m}$ выполняется независимо для каждого листа, в то время как градиентный спуск предполагает выбор лишь одной константы $\rho_m$ для масштабирования всего вектора антиградиента. Приведенный выше алгоритм имеет несколько параметров: количество бустинг-итераций $M$ и число листьев в каждом из обучаемых деревьев $J_m$, $m = \overline{1, M}$. При обучении одиночного дерева решений обычной стратегией является построение большого дерева (с большим числом листьев) с последующим удалением некоторых поддеревьев (pruning). Однако применение такого подхода в рамках бустинг-метода ведет к увеличению вычислительной сложности алгоритма. К тому же использование относительно больших деревьев решений ведет к переобучению и, следовательно, к увеличению обобщающей ошибки модели. В связи с этим было предложено [1] использовать деревья одинакового размера, т.е. $J_m \equiv J, \forall m = \overline{1, M}$, где количество листьев находится в пределах $4 \le J \le 8$. Данные ограничения были предложены из соображений учета взаимосвязанности переменных и являются лишь неплохим приближением, с которого можно начать подбор наилучших параметров для конкретной задачи. Что касается значений параметра $M$, то основная тенденция заключается в уменьшении тестовой ошибки с ростом $M$, однако выбор слишком больших значений ведет к появлению эффекта переобучения. В алгоритме 1 используется еще один параметр $\nu \in (0, 1]$ – коэффициент масштабирования (shrinkage). Использование $\nu<1$ позволяет снизить влияние отдельного дерева на результирующую модель, тем самым позволяя добиться более точных предсказаний. Параметры $M$ и $\nu$ тесно взаимосвязаны: уменьшение $\nu$ требует большего количества итераций алгоритма для достижения низкой ошибки. Следовательно, выбор $M$ и $\nu$ не должен быть независимым. Дополнительной модификацией рассматриваемого алгоритма может служить использование не всей обучающей выборки на каждой итерации, а лишь некоторой ее части. Алгоритм, реализующий подход формирования подвыборок (*subsampling*) для обучения базовых моделей, носит название *стохастического градиентного бустинга* [7]. Данная идея используется в бэггинг-алгоритмах, позволяя снизить разброс (*variance*). Таким образом, стохастический градиентный бустинг позволяет снизить вычислительные затраты на обучение и при этом может также приводить к уменьшению тестовой ошибки.

Алгоритм 1 представляет собой схему градиентного бустинга для решения задачи восстановления регрессии или бинарной классификации. Применение же функций потерь для многоклассовых задач требует, как отмечалось выше, построения не одной, а $K = |Y|$ аддитивных моделей. Соответствующая схема для функции кросс-энтропии приведена в алгоритме 2.

**Алгоритм 2.** *Градиентный бустинг деревьев решений для задачи классификации* [1].

1. Принять начальное приближение для каждой из $K$ аддитивных моделей равным нулю

$$h_{k,0} = F_{k,0}(x) = 0, \quad k = \overline{1, K} .$$

2. Для всех $m = 1, \ldots, M$:

a) Для всех $k = 1, \ldots, K$:

i) Вычислить компоненты вектора антиградиента

$$r_{i,k,m} = 1(y_{i,k} = k) - \frac{\exp h_{k,m-1}(x_i)}{\sum_{l=1}^{K} \exp h_{l,m-1}(x_i)}, \quad i = \overline{1, N} .$$

ii) Построить регрессионное дерево на выборке $\{(x_i, r_{i,k,m}), i = \overline{1, N}\}$

$$\hat{R}_{k,m} = (\hat{R}_{1,k,m}, \hat{R}_{2,k,m}, \ldots, \hat{R}_{J_m,k,m}) = \arg\min_{\Theta} \sum_{i=1}^{N} (r_{i,k,m} - T(x_j; \Theta))^2 .$$

*Таблица 1*

**Выражения для антиградиента различных функций потерь**

| Функция потерь $L(y_i, F(x_i))$ | $-\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}$ |
|---|---|
| Восстановление регрессии | |
| Квадратичная $\frac{1}{2}(y_i - F(x_i))^2$ | $y_i - F(x_i)$ |
| Абсолютная $\lvert y_i - F(x_i)\rvert$ | $\operatorname{sign}(y_i - F(x_i))$ |
| Хьюбер (Huber) $\frac{1}{2}(y_i - F(x_i))^2 1(\lvert y_i - F(x_i)\rvert \le \delta) + \delta\left(\lvert y_i - F(x_i)\rvert - \frac{\delta}{2}\right) 1(\lvert y_i - F(x_i)\rvert > \delta)$ | $y_i - F(x_i)$ при $\lvert y_i - F(x_i)\rvert \le \delta$<br>$\delta \cdot \operatorname{sign}(y_i - F(x_i))$ при $\lvert y_i - F(x_i)\rvert > \delta$ |
| Классификация | |
| Кросс-энтропия $-\sum_{k=1}^{K} 1(y_i = k) \ln p_k(x_i)$ | $1(y_i = k) - p_k(x_i)$ |

iii) Найти оптимальные константы для каждого листа дерева

$$\hat{\varphi}_{j,k,m} = \frac{K-1}{K} \cdot \frac{\sum_{x_i \in \hat{R}_{j,k,m}} r_{i,k,m}}{\sum_{x_i \in \hat{R}_{j,k,m}} \lvert r_{i,k,m}\rvert \cdot (1 - \lvert r_{i,k,m}\rvert)},$$

$$j = \overline{1, J_m}.$$

iv) Обновить модель

$$h_{k,m}(x) = h_{k,m-1}(x) + \nu \cdot \sum_{j=1}^{J_m} \hat{\varphi}_{j,k,m} \cdot 1(x \in \hat{R}_{j,k,m}).$$

3. Конечная модель

$$f(x) = \arg\max_k h_{k,M}(x).$$

Метод градиентного бустинга позволяет использовать любую дифференцируемую функцию штрафа и применим как для задачи восстановления регрессии, так и для классификации. В таблице 1 приведены компоненты вектора антиградиента для различных функций потерь.

## Программная реализация

Мы предлагаем программную реализацию алгоритма градиентного бустинга деревьев решений. Авторам известна лишь одна открытая реализация этого алгоритма [8]. Данная реализация выполнена в программной среде R и поэтому не обладает высокой производительностью. Программный код предлагаемой реализации написан на языке C++ с использованием открытой библиотеки компьютерного зрения OpenCV [9]: в качестве реализации деревьев решений используется класс CvDTree. Для обучения базовых моделей применяется алгоритм CART. Программный интерфейс реализации градиентного бустинга соответствует принятому в OpenCV, и, следовательно, данная реализация может быть интегрирована в эту библиотеку. Наша реализация позволяет решать как задачи восстановления регрессии, так и многоклассовой классификации, поддерживает все функции потерь, приведенные в таблице 1, допускает использование механизма формирования подвыборок на каждой итерации (стохастический градиентный бустинг). Программа написана в соответствии с приведенным в данной работе описанием, за исключением способа задания размера деревьев решений: вместо параметра *J* (количество листьев) используется ограничение на высоту *d*.

## Экспериментальные результаты

Основным способом оценки обобщающей способности алгоритма и его сравнения с другими подходами к решению той же задачи явля-

*Таблица 2*

**Тестовые наборы данных**

| Название | Общее количество прецедентов | Число переменных (количественные/ номинальные) | Количество классов |
|---|---|---|---|
| Восстановление регрессии | | | |
| auto-mpg | 398 | 7 (4/3) | — |
| Computer hardware | 209 | 8 (7/1) | — |
| Concrete slump | 103 | 9 (9/0) | — |
| Forestfires | 517 | 12 (10/2) | — |
| Boston housing | 506 | 13 (13/0) | — |
| imports-85 | 201 | 25 (14/11) | — |
| Servo | 167 | 4 (0/4) | — |
| Abalone | 4177 | 8 (7/1) | — |
| Классификация | | | |
| Agaricus lepiota | 8124 | 22 (0/22) | 2 |
| Liver disorders | 345 | 6 (6/0) | 2 |
| Car evaluation | 1728 | 6 (0/6) | 4 |

ется проведение экспериментов с реальными или искусственными данными. В данном разделе приведены некоторые экспериментальные результаты, показывающие достоинства и недостатки метода градиентного бустинга. Наряду с подходом, которому посвящена данная работа, были рассмотрены и конкурирующие алгоритмы: одиночные деревья решений, случайные деревья (случайные леса) [10], машина опорных векторов. Программной основой проведенных нами экспериментов является открытая библиотека компьютерного зрения OpenCV: все результаты, относящиеся к конкурирующим алгоритмам, были получены непосредственно с помощью ее компонентов: CvDTree, CvRTrees, CvERTrees и CvSVM.

Эксперименты проводились на наборах реальных данных, взятых с репозитория UCI [11]. Их краткие характеристики приведены в таблице 2.

Сравнение различных алгоритмов машинного обучения производилось по результатам 10-кратного перекрестного контроля, с помощью которого осуществлялся выбор наилучших значений параметров для каждой конкретной задачи и каждого алгоритма. Тестовая ошибка считалась при помощи нескольких критериев:

1) средняя абсолютная ошибка (*average-absolute-error*):

$$\mathrm{Err}_{\mathrm{abs}} = \frac{1}{10}\sum_{k=1}^{10}\frac{\sum_{i=1}^{T_k}\left|y_{j_{k,i}} - f(x_{j_{k,i}})\right|}{T_k},$$

где $T_k$ – объем $k$-й тестовой выборки, а $(x_{j_{k,i}}, y_{j_{k,i}})$ – $i$-й прецедент $k$-й тестовой выборки;

2) корень среднеквадратичной ошибки (*root-mean-squared error*):

$$\mathrm{Err}_{\mathrm{rms}} = \frac{1}{10}\sum_{k=1}^{10}\sqrt{\frac{\sum_{i=1}^{T_k}\left(y_{j_{k,i}} - f(x_{j_{k,i}})\right)^2}{T_k}};$$

3) для задачи классификации – подсчет частоты неправильной классификации прецедентов тестовой выборки:

$$\mathrm{Err}_{\mathrm{misclass}} = \frac{1}{10}\sum_{k=1}^{10}\frac{\sum_{i=1}^{T_k}1\left(y_{j_{k,i}} \neq f(x_{j_{k,i}})\right)}{T_k}.$$

Прецеденты с пропущенными значениями удалялись из обучающей и тестовой выборок при использовании CvSVM.

Наименьшие из полученных описанным способом ошибок приведены в таблицах 3 и 4. Из этих данных видно, что алгоритм градиентного бустинга, как правило, дает результат близкий к наилучшему для конкретной задачи, что подтверждает его универсальность и способность подстраиваться под специфику решаемой задачи. В то же время для некоторых из рассматриваемых задач существуют алгоритмы, дающие меньшую тестовую ошибку.

Рисунок 1 иллюстрирует сравнительную динамику изменения тестовой ошибки, полученной методами градиентного бустинга и случайных деревьев, с ростом количества итераций. Для обоих алгоритмов выбраны наилучшие параметры, с точки зрения результатов перекрестного контроля. Как можно видеть, для случайных деревьев ошибка достигает уровня ~1.5 уже после 100 итераций, на котором остается и при дальнейшем обучении. Бустинг, в свою очередь, обеспечивает постепенное снижение ошибки, которая достигает примерно того же уровня, что и для случайного леса, после 400 итераций. Оба метода комбинирования деревьев решений

*Таблица 3*

**Корни среднеквадратических ошибок (RMS) и средние абсолютные ошибки (ABS), полученные различными алгоритмами при 10-кратном перекрестном контроле**

| Название | Градиентный бустинг (GBT) | | Дерево решений (CvDTree) | | Случайные деревья (CvRTrees) | | Случайные деревья (CvERTrees) | | Машина опорных векторов (CvSVM) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | RMS | ABS | RMS | ABS | RMS | ABS | RMS | ABS | RMS | ABS |
| auto-mpg | 2.682 | 2 | 3.133 | 2.238 | *2.653* | *1.879* | 2.955 | 2.147 | 4.042 | 2.981 |
| Computer hardware | 23.55 | 12.62 | 30.13 | 15.62 | 26.02 | 11.62 | *19.12* | *9.631* | 50.51 | 37 |
| Concrete slump | 2.524 | 2.257 | 3.727 | 2.923 | 3.193 | 2.6 | 2.945 | 2.359 | *2.164* | *1.767* |
| Forestfires | 35.15 | 18.74 | 38.09 | 17.26 | 35.22 | 17.79 | *34* | 16.64 | 45.51 | *12.9* |
| Boston housing | *2.914* | *2.033* | 3.653 | 2.602 | 3.042 | 2.135 | 3.127 | 2.196 | 5.71 | 4.049 |
| imports-85 | 1827 | 1306 | 2317 | 1649 | *1821* | *1290* | 2146 | 1487 | 2583 | 1787 |
| Servo | *0.385* | *0.238* | 0.455 | 0.258 | 0.418 | 0.247 | 0.686 | 0.42 | 0.884 | 0.655 |
| Abalone | 2.144 | *1.47* | 2.281 | 1.604 | *2.115* | 1.492 | 2.124 | 1.498 | 2.644 | 2.091 |

*Таблица 4*

**Средние частоты неправильной классификации прецедентов, полученные различными алгоритмами при 10-кратном перекрестном контроле**

| Название | Градиентный бустинг (GBT) | Дерево решений (CvDTree) | Случайные деревья (CvRTrees) | Случайные деревья (CvERTrees) | Машина опорных векторов (CvSVM) |
|---|---|---|---|---|---|
| Agaricus lepiota | *0* | 0.000123 | *0* | *0* | *0* |
| Liver disorders | 0.251357 | 0.30543 | *0.227828* | 0.254299 | 0.278582 |
| Car evaluation | *0* | 0.0513824 | 0.0364987 | 0.0394574 | 0.0509819 |

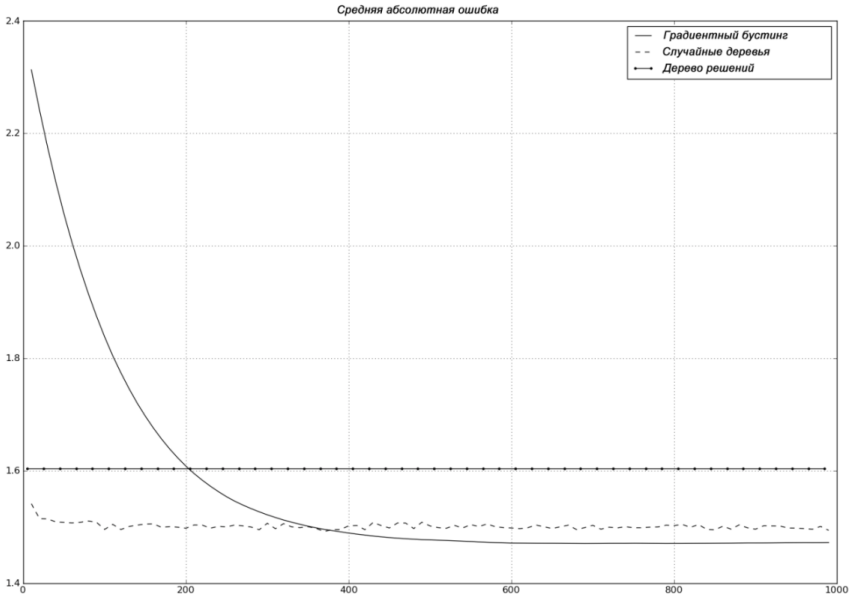


Рис. 1. Сравнение тестовой ошибки перекрестного контроля алгоритмов градиентного бустинга, случайных деревьев, и одиночного дерева решений на наборе данных Abalone. Для всех методов взяты наилучшие параметры, найденные с помощью 10-кратного перекрестного контроля

превосходят наилучшее из одиночных деревьев для данной задачи.

Для наборов данных с небольшим числом прецедентов более приемлемым является под-

ход многократного обучения и тестирования алгоритма, в то время как метод перекрестного контроля позволит осуществить подбор значений параметров. Для некоторых из перечисленных в таблице 2 наборов данных были проведены такие эксперименты. Данные случайным образом разбивались на обучающую и тестовую выборки в соотношении 9:1 для наборов Boston housing, Computer hardware и Servo и 8:2 для Abalone, после чего выполнялось обучение и предсказание. Процесс повторялся 100 раз для каждой задачи. Данный подход также позволяет проследить влияние значений некоторых параметров на обобщающую способность метода и служит методом для сравнения различных алгоритмов.

Средние значения ошибок, полученных в результате этих экспериментов, приведены в таблице 5. На рассмотренных нами наборах данных градиентный бустинг в большинстве случаев превосходит метод случайных деревьев или показывает сравнимый результат.

*Таблица 5*

**Сравнение средних ошибок алгоритмов градиентного бустинга и случайных деревьев**

| Набор данных | Тестовая ошибка алгоритма градиентного бустинга | | Тестовая ошибка алгоритма случайных деревьев | |
|---|---|---|---|---|
| | Средняя абсолютная ошибка | Средняя квадратичная ошибка | Средняя абсолютная ошибка | Средняя квадратичная ошибка |
| Boston housing | *1.995* | *8.234* | 2.13 | 9.689 |
| Computer hardware | *10.29* | *1096.1* | 13.996 | 2047.8 |
| Servo | *0.198* | *0.237* | 0.24 | 0.257 |
| Abalone | 1.517 | 4.732 | *1.514* | *4.653* |

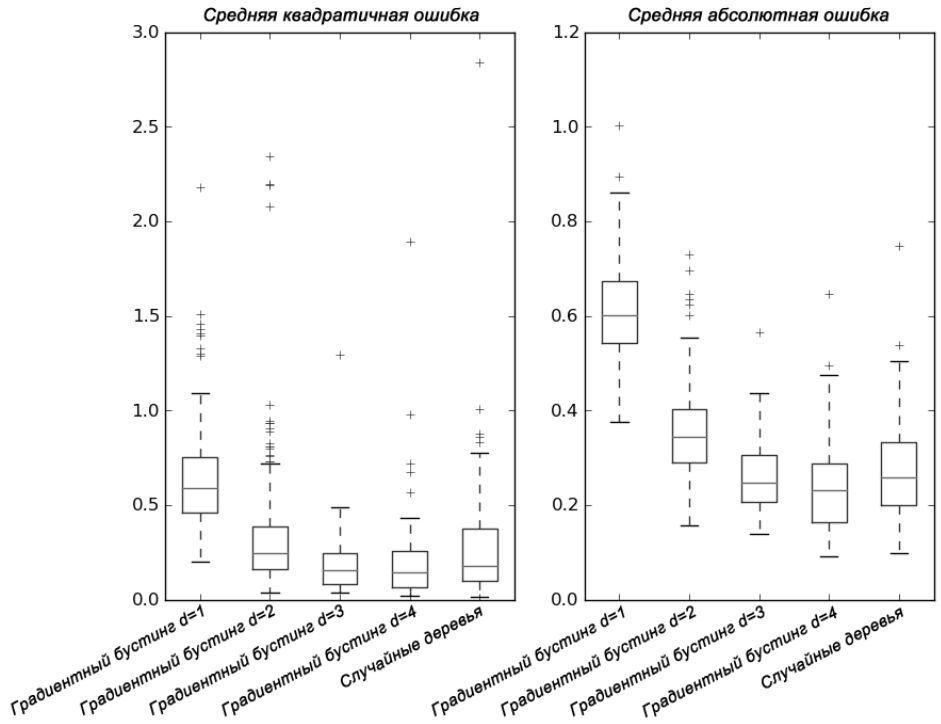


Рис. 2. Бокс-диаграммы для результатов тестирования алгоритма градиентного бустинга с использованием деревьев решений различной глубины и метода случайных деревьев на наборе данных Servo

Рассмотрим результаты некоторых экспериментов подробнее. На рис. 2 изображены бокс-диаграммы, которые показывают разброс результатов эксперимента на наборе данных Servo. Слева находятся диаграммы, соответствующие алгоритму градиентного бустинга с использованием деревьев решений различных размеров: глубина от 1 до 4. Самая правая бокс-диаграмма соответствует методу случайных деревьев. Таким образом можно наблюдать снижение тестовой ошибки бустинга при увеличении глубины используемых деревьев. Также эта диаграмма иллюстрирует небольшое превосходство алгоритма градиентного бустинга над случайными деревьями.

Авторы благодарят И.Б. Меерова за полезные обсуждения.

***Работа выполнена при поддержке федеральной целевой программы «Научные и научно-педагогические кадры инновационной России», госконтракт 02.740.11.5131.***

*Список литературы*

1. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. Springer, 2008.
2. Breiman L. Bagging predictors // Machine Learning. 1996. V. 26, № 2. P. 123–140.
3. Freund Y., Schapire R. Experiments with a New Boosting Algorithm // Machine Learning: Proceedings of the Thirteenth International Conference. 1996.
4. Friedman J.H. Greedy Function Approximation: a Gradient Boosting Machine. Technical Report. Dept. of Statistics, Stanford University, 1999.
5. Breiman L., Friedman J., Olshen R., Stone C. Classification and Regression Trees. Wadsworth, 1983.
6. Quinlan R. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.
7. Friedman J.H. Stochastic Gradient Boosting. Technical Report. Dept. of Statistics, Stanford University, 1999.
8. Ridgeway G. The state of boosting // Computing Science and Statistics. 1999. V. 31. P. 172–181.
9. OpenCV. URL: http://opencv.willowgarage.com. (дата обращения: 10.06.2010).
10. Breiman L. Random Forests // Mach. Learn. 2001. V. 45, № 1. P. 5–32.
11. UCI Machine Learning Repository. URL: http://archive.ics.uci.edu/ml (дата обращения: 10.06.2010).

## SOFTWARE IMPLEMENTATION OF THE GRADIENT TREE BOOSTING ALGORITHM

***P.N. Druzhkov, N.Yu. Zolotykh, A.N. Polovinkin***

Software implementation of the gradient tree boosting algorithm is described and its competitiveness is illustrated by the results of a computing experiment.

*Keywords*: machine learning, boosting, gradient boosting, decision trees.