УДК 004.852, 004.272.26

РЕАЛИЗАЦИЯ ПАРАЛЛЕЛЬНОГО АЛГОРИТМА ПРЕДСКАЗАНИЯ В МЕТОДЕ ГРАДИЕНТНОГО БУСТИНГА ДЕРЕВЬЕВ РЕШЕНИЙ

П.Н. Дружков, Н.Ю. Золотых, А.Н. Половинкин

PARALLEL IMPLEMENTATION OF PREDICTION ALGORITHM IN GRADIENT BOOSTING TREES METHOD

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

Описано несколько параллельных реализаций одного из алгоритмов обучения с учителем – градиентного бустинга деревьев решений (Gradient Boosting Trees) – с использованием библиотеки Intel Threading Building Blocks. Приводятся результаты экспериментального сравнения и анализ производительности различных подходов к распараллеливанию.

Ключевые слова: градиентный бустинг деревьев решений, Intel Threading Building Blocks.

Several variations of parallel implementations of one of the supervised learning algorithms, Gradient Boosting Trees (GBT), with the use of Intel Threading Building Blocks are described. Results of experimental comparison and performance analysis of different approaches to parallelization are discussed.

Keywords: gradient boosting trees, Intel Threading Building Blocks.

Введение

Машинное обучение является подразделом весьма обширной области науки, изучающей искусственный интеллект. Алгоритмы, относящиеся к данному направлению, используются при решении задач, для которых зачастую сложно или невозможно придумать явный алгоритм решения: предсказание погоды, прогнозирование экономических и социальных процессов, медицинская диагностика, детектирование объектов на фото или видео, распознавание текста, речи, создание антивирусных программ, алгоритмов фильтрации рекламы и спама и др. [9].

В настоящее время известно достаточно много алгоритмов обучения с учителем, предназначенных для решения задачи восстановления регрессии или классификации: машина опорных векторов [13], метод K ближайших соседей [9], нейронные сети [9], AdaBoost [9], деревья решений [2]. Используются различные ансамбли указанных выше методов: случайные деревья [1], полностью случайные деревья [8]. В работе рассматривается программная реализация одного из наиболее перспективных алгоритмов обучения с учителем — алгоритма градиентного бустинга деревьев решений [6, 7] (GBT — gradient boosting trees), которая является первой полнофункциональной C/C++ реализацией данного метода с открытым кодом. Результаты вычислительного эксперимента, проведенного с использованием широко распространенных наборов реальных данных, взятых из репозитория UCI [12], свидетельствуют о конкурентоспособности предлагаемой реализации по сравнению с реализациями

других алгоритмов. Разработанный код интегрирован в одну из наиболее известных свободно распространяемых библиотек компьютерного зрения OpenCV [11].

Многие из решаемых в настоящее время практических задач машинного обучения и компьютерного зрения требуют обработки значительного объема входных данных. В частности, каждый исследуемый объект может быть описан вектором признаков, содержащим сотни или даже тысячи переменных, а обучающая и тестовая выборки могут содержать десятки тысяч описаний объектов. Наглядным примером такой задачи может служить детектирование пешеходов [4], где в зависимости от выбираемых параметров необходимо классифицировать от 10000 до 185000 объектов на одно изображение. В связи с этим, наряду с качеством предсказания на одно из первых мест встает вопрос производительности используемого алгоритма. В работе рассматриваются аспекты параллельной реализации алгоритма обучения модели, а также предлагаются и анализируются различные подходы к распараллеливанию алгоритма предсказания на новых данных.

1. Градиентный бустинг деревьев решений

1.1. Постановка задачи

Одной из задач, изучаемой 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*, и 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*:

$$\{(x_i,\, y_i) \mid x_i \in X,\ y_i \in Y,\ i = 1, \ldots, n\}.$$

Другими словами, задача состоит в построении функции f из некоторого множества K, которая, получив на вход x, предсказала бы значение ответа y как можно точнее. В случае конечного Y, говорят о задаче *классификации*, если $Y = \mathbf{R}$ — задаче *восстановления регрессии* [9]. Процесс нахождения f называется *обучением* (*тренировкой*, *настройкой*) модели, процесс определения выхода по некоторому входу с помощью уже построенной модели — *предсказанием*.

1.2. Метод решения

Один из общих подходов решения задач обучения заключается в комбинировании моделей. Две основные конкурирующие идеи данного подхода — *бэггинг* (bagging от Bootstrap Aggregating) [3] и *бустинг* (boosting) [5]. 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). Бустинг, в противоположность бэггингу, обучает каждую следующую модель с использованием данных об ошибках предыдущих моделей.

Алгоритм градиентного бустинга деревьев решений является развитием бустинг-идеи. Он позволяет строить аддитивную функцию в виде суммы деревьев решений итерационно по аналогии с методом градиентного спуска. Данный подход позволяет расширить круг решаемых этим алгоритмом задач, а также зачастую получить выигрыш в точности предсказания.

Далее приводится краткое описание алгоритма обучения градиентного бустинга деревьев решений для задачи восстановления регрессии в случае использования квадратичной

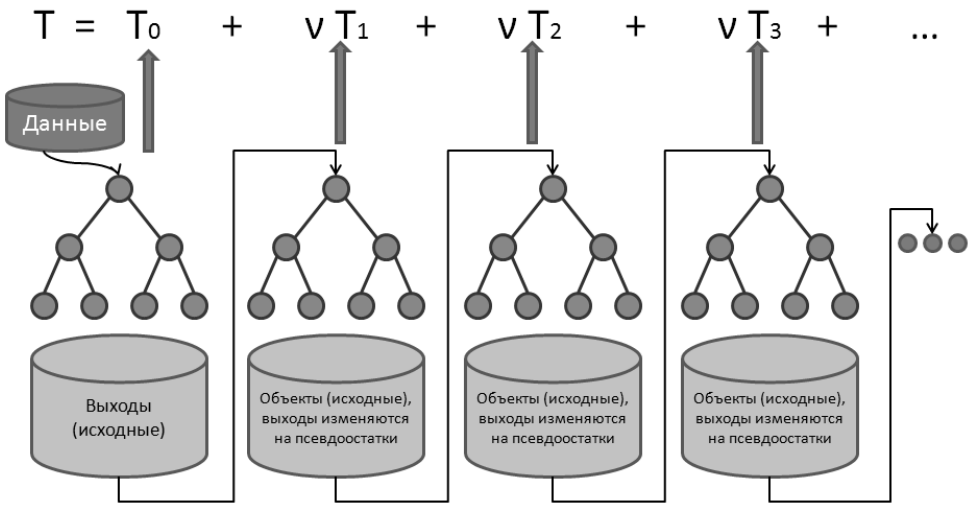

Рис. 1. Алгоритм обучения модели градиентного бустинга деревьев решений

функции потерь. Пусть обучающая выборка содержит n прецедентов, y_i — значение ответа для i-го прецедента, $T_j(x_i)$ — значение, предсказанное j-м деревом в ансамбле для i-го объекта, ν — коэффициент масшабирования. Тогда псевдоостатком для i-го объекта на k-м шаге алгоритма обучения называется значение

$$\widetilde{y}_i = y_i - T_0(x_i) - \nu \cdot \sum_{m=1}^{k-1} T_m(x_i),$$

равное разности между истинным значением ответа и значением, предсказанным ансамблем деревьев решений, построенным на $(k-1)$-м шаге алгоритма обучения. Пусть M — общее число деревьев в ансамбле. Тогда общая схема тренировки модели может быть сформулирована следующим образом:

1. Обучить дерево T_0 на исходном наборе данных (x_i, y_i), $i = 1, 2, \ldots, n$.
2. Для каждого $m = 1, 2, \ldots, M$
   - 2.1. для всех объектов в обучающей выборке вычислить псевдоостатки $\widetilde{y}$;
   - 2.2. добавить в ансамбль новое дерево, обученное на наборе данных $(x_i, \widetilde{y}_i)$, $i = 1, 2, \ldots, n$.

Детальное описание алгоритма обучения, а также подобности, связанные с тренировкой отдельных деревьев решений, и особенности реализации алгоритма для задач восстановления регресии с другими функциями потерь, а также классификации с двумя и более классами можно найти в [6].

Таким образом, на выходе алгоритма обучения мы получаем набор из M деревьев решений, и для осуществления предсказания, т. е. определения выхода y для нового объекта x, следует вычислить сумму

$$y = T_0(x) + \nu \cdot \sum_{m=1}^{M} T_m(x).$$

1.3. Реализация алгоритма градиентного бустинга деревьев решений

Авторами данной работы была выполнена программная реализация алгоритма градиентного бустинга деревьев решений, включающая как алгоритм тренировки модели, так и ее дальнейшее использование для предсказания. В данном разделе приведены некоторые экспериментальные результаты, показывающие достоинства и недостатки метода градиентного бустинга. Наряду с описываемым подходом были рассмотрены конкурирующие алгоритмы: одиночные деревья решений (алгоритм CART) [2], случайные деревья (случайные леса) [1, 8], машина опорных векторов [13]. Программной основой проведенных 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 из репозитория UCI, при этом мерой качества модели считалась средняя абсолютная ошибка 10-кратного перекрестного контроля.

Таблица 1

Результаты экспериментального сравнения алгоритмов обучения с учителем

Набор данных	*Градиентный бустинг* (GBT)	*Дерево решений* (CvDTree)	*Случайные деревья* (CvRTrees)	*Полностью случайные деревья* (CvERTrees)	*Машина опорных векторов* (CvSVM)
Auto-mpg	2	2.238	1.879	2.147	2.981
Computer hardware	12.62	15.62	11.62	9.631	37
Concrete slump	2.257	2.923	2.6	2.359	1.767
Forestfires	18.74	17.26	17.79	16.64	12.9
Boston housing	2.033	2.602	2.135	2.196	4.049
Imports-85	1306	1649	1290	1487	1787
Servo	0.238	0.258	0.247	0.42	0.655
Abalone	1.47	1.604	1.492	1.498	2.091

2. Параллельная реализация алгоритма

2.1. Алгоритм обучения

Обучение модели представляет собой достаточно трудоемкий процесс. Более того, для осуществления подбора наилучших параметров модели (количество деревьев в ансамбле, масштабный параметр, ограничение на размер деревьев) требуется неоднократное повторение процесса, что обуславливает необходимость оптимизации по скорости, в том числе за счет применения техник параллельных вычислений. Согласно результатам профилировки последовательной версии алгоритма обучения (использовался набор данных spambase из библиотеки UCI: размерность пространства признаков равна 56, число прецедентов в обучающей выборке — 2300), приведенным на рис. 2, основное время работы тратится на обучение одиночных деревьев решений.

К сожалению, бустинг-алгоритмы являются плохо распараллеливаемыми: подход, связанный с одновременным построением нескольких классификаторов в ансамбле, здесь невозможен в силу того, что тренировка каждого следующего дерева решений требует результатов предсказания всех предыдущих. Однако некоторые этапы построения одиночного дерева решений могут быть выполнены независимо друг от друга: например, нахождение разбиения во внутреннем узле дерева, которое осуществляется путем вычисления для каждой переменной уменьшения значения функции неоднородности [2] и выбора оптимального

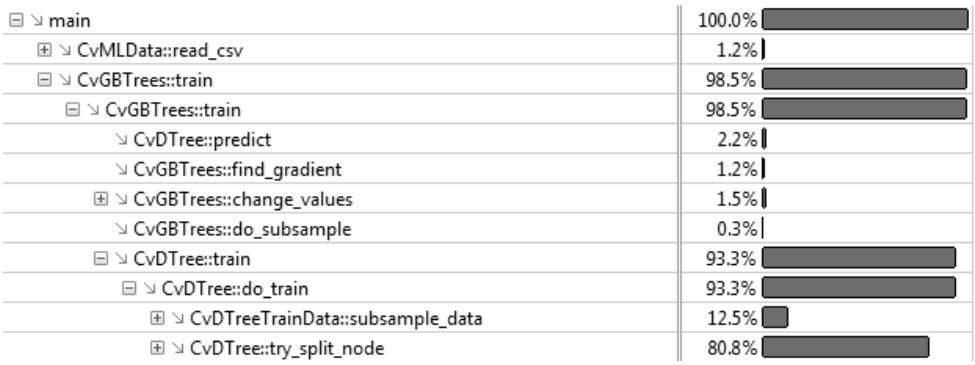

Рис. 2. Результаты профилировки алгоритма обучения модели градиентного бустинга деревьев решений

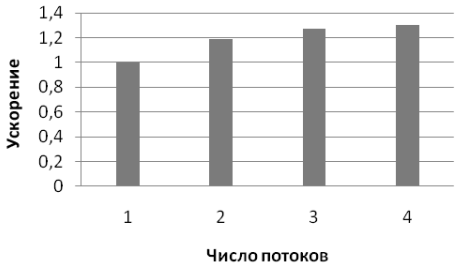

Рис. 3. Эффективность распараллеливания алгоритма обучения модели градиентного бустинга (на наборе данных spambase)

значения среди всех переменных. Именно на этом уровне выполнено распараллеливание обучения деревьев решений в библиотеке OpenCV.

Для изучения влияния параллельного подхода к отысканию оптимальной переменной каждого разбиения на скорость работы всего алгоритма обучения модели градиентного бустинга была проведена серия экспериментов.

Как видно из рис. 3, полученное ускорение значительно ниже линейного. Это связано с тем, что суммарное время работы алгоритма поиска оптимального разбиения составляет (по результатам профилировки) лишь около 40 % времени работы всего алгоритма обучения. Таким образом, максимальное ускорение, которое может быть получено, согласно закону Амдала равно 1.67.

2.2. Алгоритм предсказания

Несмотря на то, что предсказание в алгоритме градиентного бустинга деревьев решений не столь трудоемкий процесс по сравнению с построением модели, время работы этого алгоритма зачастую также является критичным. На это есть несколько причин. Обучение выполняется на «оффлайн» этапе, в то время как дальнейшее (и основное) применение модели заключается в осуществлении с ее помощью предсказаний на новых данных. Более того, на практике часто приходится выполнять не единичные предсказания, а целые серии: например, в некоторых алгоритмах, решающих задачу детектирования объектов на изображении, может потребоваться выполнение десятков тысяч предсказаний на одно изображение. Все это накладывает достаточно жесткие временные рамки на время вычисления предсказания — кроме того, в некоторых прикладных задачах требуется работа в режиме реального времени.

Рис. 4. Эффективность распараллеливания алгоритма предсказания по деревьям в ансамбле с использованием модели градиентного бустинга деревьев решений

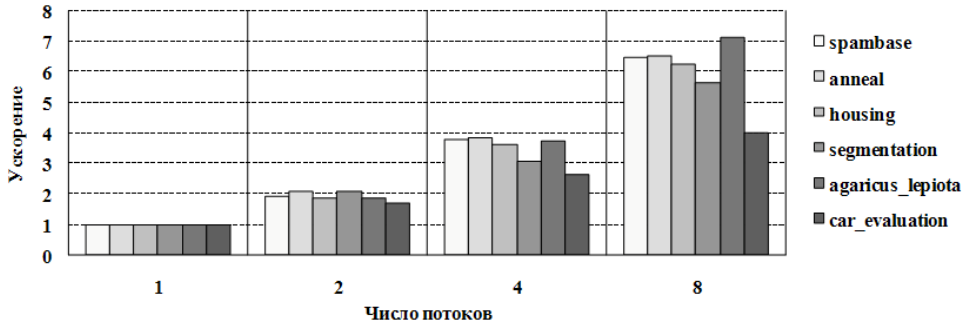

Рис. 5. Эффективность распараллеливания алгоритма предсказания по данным с использованием модели градиентного бустинга деревьев решений

В рамках данной статьи рассматривается два различных подхода к распараллеливанию алгоритма предсказания с использованием обученной модели градиентного бустинга деревьев решений. Напомним, что для получения выхода y по некоторому входу x необходимо вычислить сумму предсказаний всех деревьев из имеющегося ансамбля. В отличие от процесса обучения, здесь значение каждого слагаемого может быть получено независимо от остальных — отсюда возникает первая параллельная схема предсказания, в которой для одного объекта x значения предсказаний отдельных деревьев $T_m(x)$ в сумме $y = T_0(x) + \nu \cdot \sum_{m=1}^{M} T_m(x)$ вычисляются параллельно несколькими потоками. Другой рассматриваемый подход: распараллеливание по данным — основан на необходимости одновременного предсказания для большого количества новых объектов. В данном случае выполняется параллельное вычисление значений $y_i = T_0(x_i) + \nu \cdot \sum_{m=1}^{M} T_m(x_i)$ для нескольких объектов x_i. Программная реализация предложенных подходов к распараллеливанию выполнена с использованием технологии Intel Threading Building Blocks [10].

Как видно из рис. 4 – 5, оба рассматриваемых подхода дают существенное ускорение процесса предсказания. Однако метод распараллеливания по данным обладает лучшей масштабируемостью, так как число объектов, для которых требуется выполнить предсказание, как правило, значительно превосходит количество деревьев в модели градиентного бустинга.

3. Заключение

В статье рассмотрены аспекты параллельной реализации одного из наиболее перспективных алгоритмов обучения с учителем — градиентного бустинга деревьев решений. Про-

анализирована программная реализация обучения рассматриваемой модели, основанная на параллельной версии алгоритма тренировки отдельных деревьев решений, входящей в состав библиотеки OpenCV. Также в рамках данной работы были предложены и реализованы с использованием технологии Intel Threading Building Blocks две схемы распараллеливания алгоритма предсказания с помощью построенной модели.

Авторы благодарят И.Б. Мееров (ННГУ) и В.Л. Ерухимова (компания ITSeez) за ценные замечания и полезные обсуждения.

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

Статья рекомендована к публикации программным комитетом международной научной конференции «Параллельные вычислительные технологии 2011».

Литература

1. Breiman, L. Random Forests / L. Breiman // Machine Learning. – 2001. – V. 45, №1. – P. 5 – 32.
2. Classification and Regression Trees / L. Breiman, J. Friedman, R. Olshen, C. Stone. – Wadsworth, 1983.
3. Breiman, L. Bagging predictors / L. Breiman // Machine Learning. – 1996. – V. 26, №2. – P. 123 – 140.
4. Enzweiler, M. Monocular Pedestrian Detection: Survey and Experiments / M. Enzweiler, D.M. Gavrila // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2009. – V.31, №12. – P. 2179 – 2195.
5. Freund, Y. Experiments with a New Boosting Algorithm / Y. Freund, R. Schapire // Machine Learning: Proceedings of the Thirteenth International Conference. – San Francisco: Morgan Kauffman, 1996. – P. 148 – 156.
6. Friedman, J.H. Greedy Function Approximation: a Gradient Boosting Machine / J.H. Friedman. — Technical report. — Dept. of Statistics, Stanford University, 1999.
7. Friedman, J.H. Stochastic Gradient Boosting. Technical report. Dept. of Statistics, Stanford University, 1999.
8. Geurts, P. Extremely Randomized Trees / P. Geurts, D. Ernst, L. Wehenkel // Machine Learning. – 2006. – V. 36, №1. – P. 3 – 42.
9. Hastie, T. The Elements of Statistical Learning / T. Hastie, R. Tibshirani, J. Friedman. – Springer-Verlag, 2008.
10. Intel Threading Building Blocks. URL: http://www.threadingbuildingblocks.org (дата обращения: 07.12.2010).
11. OpenCV Wiki. URL: http://opencv.willowgarage.com/wiki (дата обращения: 07.12.2010).
12. UCI Machine Learning Repository. URL: http://archive.ics.uci.edu/ml (дата обращения: 07.12.2010).
13. Вапник, В.Н. Восстановление зависимостей по эмпирическим данным / В.Н. Вапник. – М.: Наука, 1979.

References

1. Breiman L. Random Forests. *Machine Learning*, 2001, v. 45, no. 1, pp. 5 – 32.
2. Breiman L., Friedman J., Olshen R., Stone C. *Classification and Regression Trees.* Wadsworth, 1983.

3. Breiman L. Bagging predictors *Machine Learning*, 1996, v. 26, no. 2, pp. 123 – 140.
4. Enzweiler M., Gavrila D.M. Monocular Pedestrian Detection: Survey and Experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009, v.31, no. 12, pp. 2179 – 2195.
5. Freund Y., Schapire R. Experiments with a New Boosting Algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*. San Francisco, Morgan Kauffman, 1996, pp. 148 – 156.
6. Friedman J.H. Greedy Function Approximation: a Gradient Boosting Machine. *Technical report*. Dept. of Statistics, Stanford University, 1999.
7. Friedman J.H. Stochastic Gradient Boosting. *Technical report*. Dept. of Statistics, Stanford University, 1999.
8. Geurts P., Ernst D., Wehenkel L. Extremely Randomized Trees. *Machine Learning*, 2006, v. 36, no.1, pp. 3 – 42.
9. Hastie T., Tibshirani R., Friedman J. *The Elements of Statistical Learning*. Springer-Verlag, 2008.
10. *Intel Threading Building Blocks*. Available at: http://www.threadingbuildingblocks.org (accessed 07.12.2010).
11. *OpenCV Wiki*. Available at: http://opencv.willowgarage.com/wiki (accessed 07.12.2010).
12. *UCI Machine Learning Repository*. Available at: http://archive.ics.uci.edu/ml (accessed 07.12.2010).
13. Vapnik V. *Estimation of dependences based on empirical data*. Springer, 1981.

Павел Николаевич Дружков, кафедра математической логики и высшей алгебры, Нижегородский государственный университет им. Н.И. Лобачевского (Россия, г. Нижний Новгород), druzhkov_paul@mail.ru.

Pavel Nikolaevich Druzhkov, Department of Mathematical Logic and Higher Algebra, N.I. Lobachevsky State University of Nizhni Novgorod (Russia, Nizhni Novgorod), druzhkov_paul@mail.ru.

Николай Юрьевич Золотых, кандидат физико-математических наук, доцент, кафедра математической логики и высшей алгебры, Нижегородский государственный университет им. Н.И. Лобачевского (Россия, г. Нижний Новгород), nikolai.zolotykh@gmail.com.

Nikolai Yur'evich Zolotykh, Candidate of Physico-mathematical Sciences, Department of Mathematical Logic and Higher Algebra, N.I. Lobachevsky State University of Nizhni Novgorod (Russia, Nizhni Novgorod), nikolai.zolotykh@gmail.com.

Алексей Николаевич Половинкин, кафедра математического обеспечения ЭВМ, Нижегородский государственный университет им. Н.И. Лобачевского (Россия, г. Нижний Новгород), alexey.polovinkin@gmail.com.

Aleksey Nikolayevich Polovinkin, Department of Software, N.I. Lobachevsky State University of Nizhni Novgorod (Russia, Nizhni Novgorod), alexey.polovinkin@gmail.com.

Поступила в редакцию 29 апреля 2011 г.