УДК 004.89

ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ОБУЧЕНИЯ ТЕКСТОВОГО КЛАССИФИКАТОРА ДЛЯ МНОГОПРОЦЕССОРНОЙ СИСТЕМЫ С ИЕРАРХИЧЕСКОЙ АРХИТЕКТУРОЙ

Т.А. Пескишева, Е.В. Котельников, О.А. Пестов

ФГБОУ ВПО «Вятский государственный гуманитарный университет», г. Киров

Рецензент д-р пед. наук, профессор С.М. Окулов

Ключевые слова и фразы: автоматическая текстовая классификация; метод опорных векторов; многопроцессорные системы; параллельные алгоритмы; системы с иерархической архитектурой.

Аннотация: Описана параллельная реализация алгоритма обучения текстового классификатора на основе метода опорных векторов. Предложены способы балансировки нагрузки между узлами многопроцессорной иерархической системы. Приведены результаты экспериментов на основе коллекции Reuters-21578, подтверждающие эффективность разработанного параллельного алгоритма.

Введение

Появление новых информационных технологий и развитие компьютерных сетей (как локальных, так и глобальных) привело к значительному росту количества информации, хранящейся и обрабатываемой в электронном виде. К настоящему времени различными хранилищами знаний (полнотекстовыми базами данных и электронными библиотеками) накоплены значительные информационные массивы.

Сложность обработки этих массивов увеличивается пропорционально их объему. Отсутствие возможности своевременно получать требуемую информацию по конкретной теме делает бесполезной бóльшую часть накопленных данных, представленных текстами в электронном виде.

Использование автоматической текстовой классификации позволяет сократить время на обработку электронных документов. На данный момент уже разработано достаточно большое количество систем и модулей автоматической рубрикации текстов [8], однако, их производительность существенно снижается в случае значительного роста объема обрабаты-

Пескишева Татьяна Анатольевна – ассистент кафедры «Прикладная математика и информатика»; Котельников Евгений Вячеславович – кандидат технических наук, доцент кафедры «Прикладная математика и информатика», e-mail: Kotelnikov.EV@gmail.com; Пестов Олег Александрович – ассистент кафедры «Прикладная математика и информатика», ФГБОУ ВПО «Вятский государственный гуманитарный университет», г. Киров.

ваемой информации, а также увеличения числа рубрик, по которым необходимо классифицировать документы.

Снижение производительности существующих систем связано со следующими особенностями задачи текстовой классификации. Во-первых, очень высока размерность пространства признаков (слов, словосочетаний) – десятки, иногда сотни тысяч. Во-вторых, все документы соотносятся с большим количеством классов (рубрик) – от нескольких десятков до сотен.

Повышение производительности программ автоматической текстовой классификации возможно за счет использования многопроцессорных вычислительных систем и комплексов. Современные многопроцессорные системы в большинстве случаев имеют иерархическую архитектуру, что позволяет выполнять распараллеливание алгоритмов не только на уровне вычислительных узлов, но и на уровне потоков.

Цель данной работы – предложить параллельный алгоритм обучения текстового классификатора для многопроцессорной системы с иерархической архитектурой и проанализировать эффективность способов распределения нагрузки между узлами вычислительного кластера.

Обучение классификатора

В задаче текстовой классификации каждый документ представляется в виде вектора признаков. В качестве признаков используются веса слов (или словосочетаний) документа. Вес дает числовую оценку значимости данного слова для определения тематики текста. Каждому вектору сопоставлено число, обозначающее класс, к которому он относится.

Описываемая в данной статье система многоклассовой автоматической классификации основана на одном из наиболее мощных методов машинного обучения – методе опорных векторов (Support Vector Machines – SVM), предложенном В. Н. Вапником [4].

Применение SVM сводится к двум основным этапам: обучению и классификации. В процессе обучения в n-мерном пространстве признаков строится гиперплоскость, разделяющая векторы разных классов. При этом SVM-классификатор выделяет так называемые опорные векторы – такие векторы, которые находятся ближе всего к разделяющей гиперплоскости. Только опорные векторы несут всю информацию о разделении классов, так что остальные векторы могут в дальнейшем не учитываться.

Обучение сводится к решению задачи квадратичной оптимизации с предельными ограничивающими условиями и одним ограничением линейного равенства. Методы решения задачи квадратичной оптимизации известны, но вычислительно сложны: при больших обучающих выборках обучение классификатора оказывается неприемлемо длительным.

Применительно к задаче текстовой классификации обучение происходит на основе документов, для которых уже известна соответствующая им рубрика или набор рубрик. После того, как на этапе обучения будут получены опорные векторы, на их основе может выполняться непосредственно классификация документов, о рубриках которых нет никакой предварительной информации.

Наиболее трудоемким является процесс обучения, поэтому важным представляется распараллеливание именно этого этапа работы классификатора.

Способы балансировки нагрузки

Для повышения эффективности работы многопроцессорной системы важно сбалансировать нагрузку на ее вычислительные узлы. В данной статье предлагаются четыре способа распределения нагрузки.

Все предлагаемые способы основаны на принципе «master-slave». Главный узел («master») передает подчиненным узлам («slave») задания, которые они должны выполнить. Подчиненные узлы независимо друг от друга выполняют свои задания, затем полученные результаты возвращают главному узлу.

Способы распределения нагрузки на узлы определяются двумя основными параметрами. Один из параметров зависит от того, в какой момент времени принимается решение о передаче обучающих векторов на узлы: заранее или непосредственно в процессе обучения. В первом случае применяется статическое распределение обучающих векторов, а во втором – динамическое.

Статическое распределение выполняется предварительно и не требует дополнительных ресурсов вычислительной системы на этапе обучения классификатора. Для распределения нагрузки используется «жадный» алгоритм. В общем случае работа «жадного» алгоритма заключается в принятии локально оптимальных решений на каждом этапе, допуская, что конечное решение также окажется оптимальным.

На практике при решении задачи сбалансированного распределения нагрузки «жадный» алгоритм почти всегда дает решение в достаточной степени близкое к оптимальному. Сначала главный узел сортирует классы по убыванию количества примеров. Каждому подчиненному узлу выделяется по одному классу в порядке невозрастания количества примеров. Очередной класс выделяется тому узлу, у которого общее количество всех выделенных ему примеров минимально (независимо от того, сколько классов уже выделено данному узлу).

Применение способа статического распределения приводит к недостаточно эффективной загрузке системы, так как некоторые узлы намного раньше завершают вычисления и простаивают.

Способ динамического распределения решает проблему неоптимального использования узлов системы за счет того, что узлы загружаются по мере их освобождения (динамически). Эффективность динамического распределения снижается, в свою очередь, из-за увеличения количества обменов информацией между главным и подчиненным узлами, а также времени ожидания узлов по сравнению со статическими методами.

Другой параметр способа балансировки нагрузки связан с видом информации, на основе которой выполняется распределение. В ситуации, когда есть предварительная информация о количестве опорных векторов в каждом классе, распределение нагрузки осуществляется на основании этих данных. Время обучения классификатора напрямую зависит от количества опорных векторов. Чем больше опорных векторов в классе, тем больше времени потребуется для построения классификатора. Однако на практике информация о количестве опорных векторов недоступна до завершения процесса обучения, поэтому применение способа динамического распределения нагрузки возможно только при неоднократном обучении

классификатора на одном и том же обучающем наборе, например, при поиске оптимальных параметров.

В результате может быть получено четыре метода балансировки нагрузки: статический на основе примеров, статический на основе опорных векторов, динамический на основе примеров и динамический на основе опорных векторов.

Однако реализация последнего из вышеперечисленных методов не представляет практического интереса. Объясняется это следующим. Имея возможность заранее оценить временные затраты на обучение классификатора по той или иной рубрике, оптимальнее распределить нагрузку по узлам до обучения классификатора, то есть статически. В случае динамического распределения на основе опорных векторов, полезная нагрузка на узлы не меняется, но при этом значительно увеличивается число обменов информацией между узлами и как результат – время простоя узлов.

Условия проведения экспериментов

Для тестирования и анализа предложенных методов распределения нагрузки в параллельном алгоритме обучения многоклассовой классификации было разработано программное приложение для многопроцессорной системы с иерархической архитектурой. Приложение создавалось в среде Microsoft Visual Studio 2010 на языках C# и C++ на основе принципов объектно-ориентированного программирования.

Для кластерной архитектуры параллельная реализация алгоритмов обучения проводилась с использованием библиотеки MPI.NET версии 1.0 [3]. Библиотека MPI.NET представляет собой свободно доступную реализацию интерфейса передачи сообщений MPI для среды Microsoft.NET.

Для запуска и выполнения потоков в среде с общей памятью применялась технология OpenMP [5].

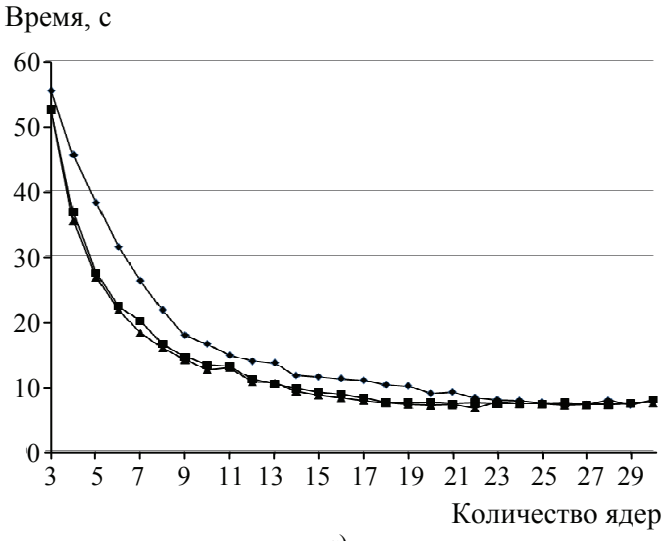
Расчеты проводились на вычислительном кластере Вятского государственного гуманитарного университета, состоящем из 30 вычислительных узлов. Каждый узел представляет собой персональный компьютер с процессором Intel Core 2 Duo 2 ГГц и 2 Гб оперативной памяти. Таким образом, максимальное количество вычислительных ядер кластера равно 60. Узлы связаны сетью Gigabit Ethernet. Кластер функционирует на базе операционной системы Microsoft Windows HPC Server 2008, на каждом узле установлена среда выполнения MPI.NET Runtime версии 1.0.

Для экспериментального исследования использовалась коллекция финансовых новостей агентства Reuters (Reuters-21578, Distribution 1.0) [4]. Обучающий набор документов был выделен в соответствии с общепринятым подходом, названным «ModApte split» [4]. При этом исключались документы, не помеченные ни одной рубрикой. Таким образом, обучающий набор в наших экспериментах представлен 7775 документами, каждый из которых относится к одной или нескольким из 115 рубрик.

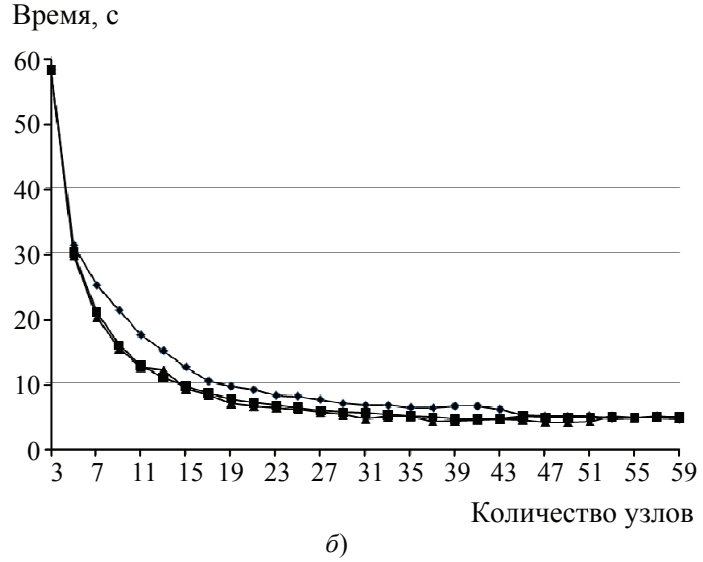
Для формирования векторов использовался морфологический анализатор Mystem 1.0 от компании Yandex [7], терминами считались все слова в нормальной форме, исключая стоп-слова. Для взвешивания терминов и получения числовых значений компонентов векторов применялся метод TF.IDF [6]. В результате получены 7775 обучающих векторов, размерность каждого из которых составляет 6103.

Результаты экспериментов

Тестировались три способа распределения нагрузки между узлами вычислительного кластера: статический на основе обучающих примеров, статический на основе опорных векторов и динамический на основе обучающих примеров. Для каждого способа варьировались следующие условия: использование технологии OpenMP (да/нет) и количество задействованных ядер кластера (от 3 до 30 без OpenMP и от 3 до 59 с OpenMP – на главном узле работает всегда одно ядро). Для каждого эксперимента с данным набором условий проводилось 5 запусков, затем результаты усреднялись.



а)



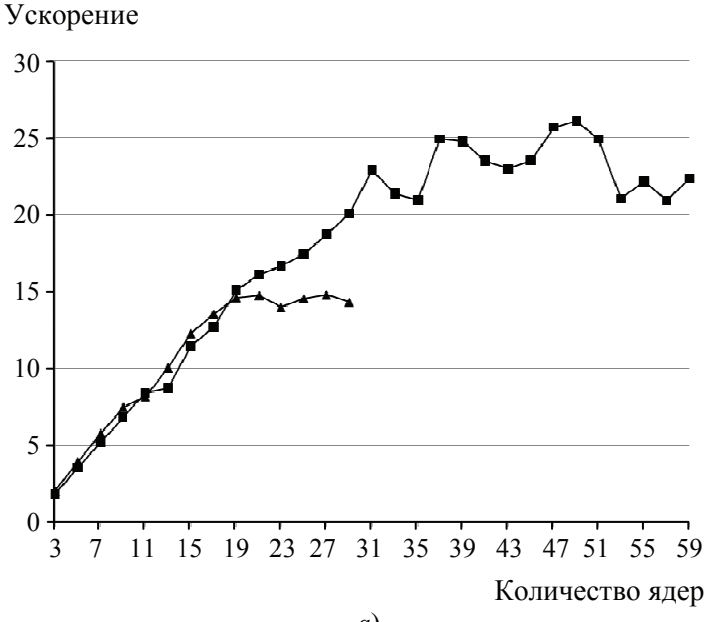
б)

Рис. 1. Результаты экспериментов без использования OpenMP (*а*) и с использованием OpenMP (*б*):
◆ – статический по примерам; ■ – статический по опорным векторам; ▲ – динамический по примерам

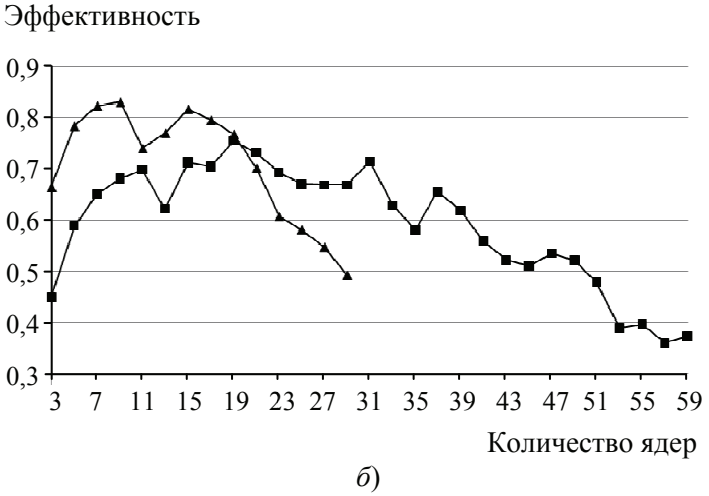
Для вычисления характеристик производительности параллельного алгоритма было измерено время выполнения последовательной версии решения данной задачи на одном узле кластера с применением технологии OpenMP и без нее. Полученные значения составляют 104,6 с (без OpenMP).

Результаты экспериментов приведены на рис. 1.

Значения ускорения и эффективности для способа динамического распределения нагрузки (как показавшего наилучшие результаты) приведены на рис. 2. Ускорение вычисляется как отношение времени решения задачи на одном вычислительном ядре ко времени решения на p вычислительных ядрах. Эффективность вычисляется как отношение ускорения к



а)



б)

Рис. 2. Зависимости ускорения (*а*) и эффективности (*б*) от количества ядер для способа динамического распределения:
—▲— – без OpenMP; —■— – с OpenMP

количеству задействованных ядер и отражает долю времени выполнения алгоритма, в течение которой вычислительные узлы были реально задействованы для решения задачи.

Заключение

По результатам экспериментов можно сделать вывод, что параллельный алгоритм существенно повышает эффективность решения задачи текстовой классификации. Когда априорная информация о количестве опорных векторов для каждой рубрики отсутствует, могут применяться способы статического и динамического распределения нагрузки на основе примеров, причем последний более предпочтителен. Если количество опорных векторов известно, тогда способ статического распределения нагрузки показывает результаты, сравнимые со способом динамического распределения.

Снижение эффективности при увеличении количества ядер объясняется достижением предельного ускорения для данного набора обучающих векторов (4,5 с против 104,6 при 59 ядрах). Поэтому в дальнейших исследованиях предполагается использовать более обширные текстовые коллекции, в частности RCV-1 [1].

Список литературы

1. RCV1: A New Benchmark Collection for Text Categorization Research / D.D. Lewis [at al.] // Journal of Machine Learning Research. – 2004. – No. 5. – P. 361–397.

2. Openmp.org. The OpenMP API Specification for Parallel Programming [Электронный ресурс]. – Режим доступа : http://openmp.org/. – Дата обращения : 01.04.2011. – Загл. с экрана.

3. Project MPI.NET // The Open Systems Lab, Indiana University [Электронный ресурс]. – Режим доступа : http://www.osl.iu.edu/research/mpi.net. – Дата обращения : 01.04.2011. – Загл. с экрана.

4. Reuters-21578, Distribution 1.0 [Электронный ресурс]. – Режим доступа : http://www.daviddlewis.com/resources/testcollections/reuters21578. – Дата обращения : 01.04.2011. – Загл. с экрана.

5. Salton, G. Term-Weighting Approaches in Automatic Text Tetrieval / G. Salton // Information Processing & Management. – Pergamon Press, 1988. – No. 5, Vol. 24. – P. 513–523.

6. Vapnik, V. Statistical Learning Theory / V. Vapnik . – Wiley, New York, 1998.

7. Морфологический анализатор Mystem от компании Yandex [Электронный ресурс]. – Режим доступа : http://company.yandex.ru/technology/mystem. – Дата обращения : 01.04.2011. – Загл. с экрана.

8. Пескишева, Т.А. Современные системы и модули автоматической рубрикации текстовых документов / Т.А. Пескишева ; Вят. гос. гуманитар. ун-т. – Киров, 2010. – 37 с. – Библиогр. : 30 назв. – Рус. – Деп. в ВИНИТИ 01.07.2010, № 410 – В 2010.

Text Classifier Training Parallel Algorithm for Multiprocessor System with Hierarchical Architecture

T.A. Peskisheva, E.V. Kotelnikov, O.A. Pestov

Vyatka State University of Humanities, Vyatka

Key words and phrases: automatic text categorization; multiprocessor systems; parallel algorithms; support vector machines; SVM; systems with hierarchical architecture.

Abstract: The paper describes parallel realization of text classifier training algorithm, using support vector machines. Load balancing methods between nodes of multiprocessor hierarchical system are suggested. The results of experiments based on collection Reuters-21578 proving the effectiveness of the introduced parallel algorithm are given.
