Оптимизация загрузки данных в формате libsvm при решении двухклассовой задачи SVM методом усреднения решающих правил в условиях большой обучающей совокупности

М.Ю. Курбаков[1], А.И. Макарова[1], В.В. Сулимова[1]

[1]Тульский государственный университет, Ленина 92, Тула, Россия, 300012

Аннотация. Метод опорных векторов (SVM) является одним из наиболее удобных и эффективных инструментов двухклассового распознавания. Однако существуют некоторые проблемы, препятствующие его применению для обучения в условиях больших объемов данных, в частности, проблема высокой вычислительной сложности процедуры обучения распознаванию и проблема хранения полного набора данных в оперативной памяти. В предыдущей работе нами был предложен метод усреднения решающих правил, направленный на решение первой проблемы, позволяющий быстро найти приближенное, но не сильно отличающееся от точного решение задачи SVM. В данной работе мы предлагаем решение второй проблемы - специализированную схему работы с данными, ориентированную на предложенный нами подход и оптимизирующую работу с памятью в условиях больших объемов данных. Предложенная схема основана на механизме отображения файлов в память и позволяет эффективно осуществлять загрузку произвольных подвыборок объектов из файла в традиционном формате libsvm. Экспериментальное исследование показывает преимущество данной схемы по сравнению с классическими способами работы с данными в том же формате.

1. Введение

Важной особенностью современных задач анализа данных является необходимость обучения в условиях больших обучающих совокупностей, размер которых может достигать десятков, сотен тысяч и даже миллионов объектов. В таких условиях многие хорошо зарекомендовавшие себя методы, такие, например, как метод опорных векторов (Support Vector Machines, SVM) [1], являющийся одним из наиболее удобных и эффективных инструментов обучения двухклассовому распознаванию для небольших объемов данных, оказываются неприменимыми из-за слишком высокой трудоемкости процедуры обучения и необходимости хранения полной обучающей совокупности в оперативной памяти.

В связи с этим появилось огромное количество исследований, направленных на повышение производительности метода SVM, в частности, различные инкрементные и декрементные методы [2-4], методы, адаптированные для параллельных и распределенных вычислений [5-8], а также различные подходы, позволяющие снизить требовательность алгоритмов к объему памяти [9,10] и т.д. Однако все имеют общий недостаток, унаследованный от методов работы с небольшими объемами данных - это итерационная природа алгоритмов, которая существенно

ограничивает возможности повышения производительности за счет привлечения параллельных и распределенных технологий.

В связи с этим в предыдущей работе [11] нами был предложен метод усреднения решающих правил - достаточно простой подход, позволяющий быстро найти приближенное (но не сильно отличающееся от точного) решение задачи SVM в условиях большого числа объектов, являющийся экономичным по памяти (что обеспечивает возможность его применения для обучения на одной вычислительной машине) и, в то же время, обладающий высокой степенью параллелизма, что дает возможность его эффективной реализации с применений технологий параллельных и распределенных вычислений.

С теоретической точки зрения данный подход не имеет ограничений по числу объектов. Более того, он имеет серьезное преимущество по сравнению с традиционными способами решения задачи SVM, поскольку не требует одновременного хранения в памяти всей обучающей совокупности, что само по себе может представлять проблему при больших объемах данных. Однако производительность метода во многом определяется эффективностью работы с данными, в связи с чем данный вопрос требует специального рассмотрения.

В данной работе предложено несколько стратегий загрузки данных в широко используемом формате libsvm и показано, что выбор оптимальной стратегии позволяет существенным образом повысить производительность работы метода.

2. Постановка задачи двухклассового распознавания

Задача двухклассового распознавания [1] является одной из наиболее распространенных задач анализа данных. Массовыми источниками таких задач являются такие важные области, как молекулярная биология, горнодобывающая и нефтяная промышленности, медицинские системы и системы видеонаблюдения, маркетинг и многие другие.

Задача двухклассового распознавания состоит из двух этапов - обучения и распознавания.

Исходными данными для обучения распознаванию является обучающая совокупность объектов $\Omega^* = \{\omega_j,\ j = 1, \dots, N\} \in \Omega$, каждый из которых представлен вектором его вещественных признаков $x_j = x(\omega_j)$ и снабжен меткой $y_j = y(\omega_j) \in \{+1; -1\}$, определяющей классовую принадлежность к одному из двух классов.

Требуется на основе анализа обучающей совокупности построить решающее правило распознавания - классифицирующая функция $\hat{y}(\omega): \Omega \to \{+1; -1\}$, которая для любого поступившего на ее вход объекта $\omega \in \Omega$ (в том числе не участвовавшего в обучении) определяла бы метку класса (рис. 1).

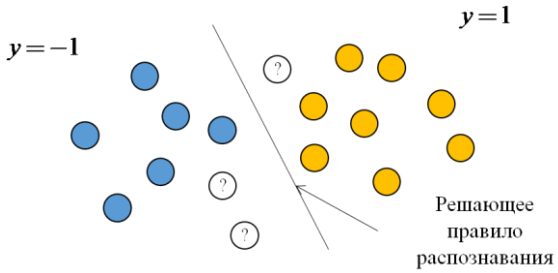

Рисунок 1. Иллюстрация задачи двухклассового распознавания.

На этапе распознавания осуществляется применение построенного решающего правила к новым объектам. Наиболее трудоемким с вычислительной точки зрения является этап обучения распознаванию, поэтому в данной работе основное внимание уделено повышению производительности именно этого этапа.

3. Метод усредненных решающих правил

Метод усредненных решающих правил, предложенный в [11], является быстрым приближенным методом нахождения решения двухклассовой задачи, возникающей в методе опорных векторов (SVM), согласно которому решение ищется в виде оптимальной линейной

разделяющей гиперплоскости, призванной как можно лучше разделить объекты двух классов [1].

Основная идея метода усредненных решающих правил заключается в формировании множества случайных небольших подвыборок обучающей совокупности, независимом обучении по методу SVM отдельно по каждой из подвыборок и последующем объединении частных решающих правил распознавания в одно общее решающее правило в виде линейной разделяющей гиперплоскости. В случае линейного признакового пространства объединение может быть осуществлено путем простого усреднения частных решающих правил [11].

При увеличении числа подвыборок усредненное решающее правило стабилизируется и перестает вести себя как случайная величина. Этот эффект хорошо виден в случае двухмерного признакового пространства (рис.2).

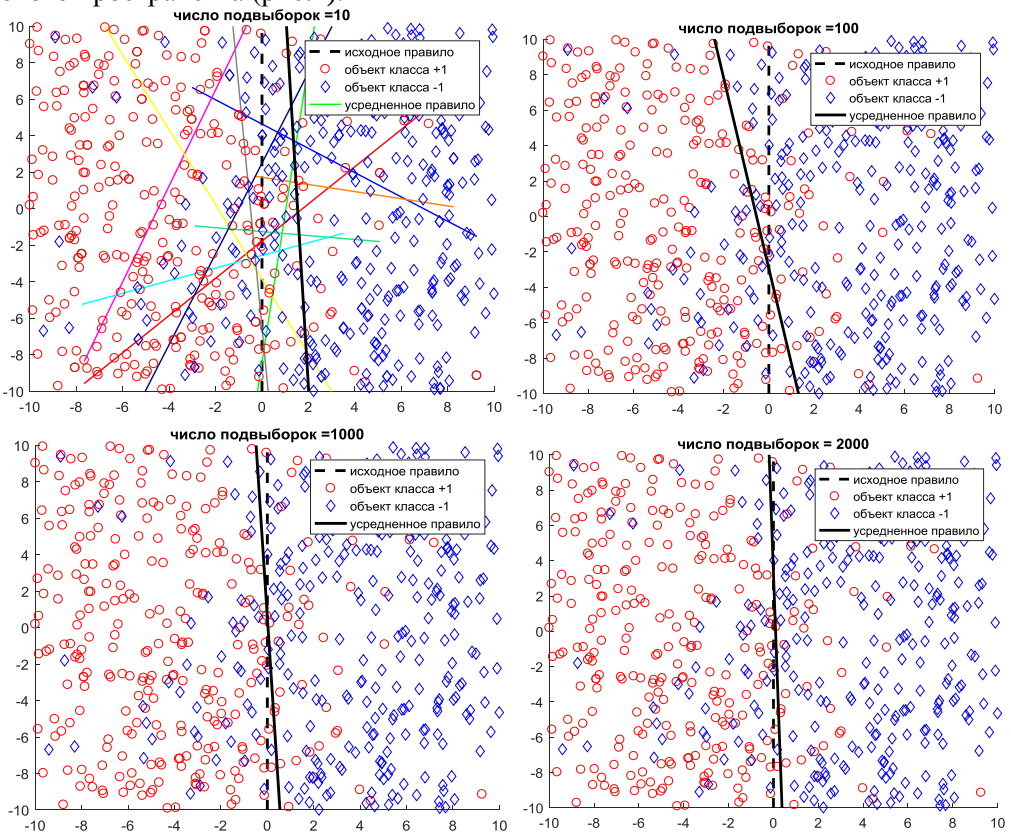

Рисунок 2. Изменение решающего правила при увеличении числа подвыбок. Пунктирной линией здесь показано решающее правило, построенное при обучении по всей обучающей совокупности, а сплошной – усредненное решающее правило.

Самый простой последовательный вариант алгоритма, реализующего данный метод, в качестве параметра имеет число подвыборок, размер одной подвыборки и обучающую совокупность. Такой алгоритм выполняет фиксированное количество итераций, на каждой из которых происходит обучение по отдельной подвыборке. Именно на такой вариант алгоритма мы ориентируемся в рамках данной работы, хотя возможны и другие варианты как последовательной, так и параллельной алгоритмической реализации [11].

Обучение по каждой отдельной подвыборке может осуществляться при помощи любой доступной реализации SVM. В данной работе мы используем с этой целью библиотеку libsvm [12,13].

Особенностью данного метода является то, что он не требует одновременного хранения в оперативной памяти всей обучающей совокупности - в данном случае оказывается достаточно иметь в памяти только текущую подвыборку, с которой идет работа на очередной итерации.

Однако следует отметить, что для достижения требуемой точности распознавания может потребоваться достаточно большое число подвыборок. Конкретное их количество зависит от решаемой задачи и размера подвыборки и может составлять, например, 100 или даже 1000 подвыборок. В связи с этим эффективность работы с данными оказывает существенное влияние на производительность метода в целом, требует специального рассмотрения и выбора оптимальной стратегии работы с данными.

4. Формат исходных данных libsvm для обучения распознаванию

В качестве формата исходных данных в этой работе применяется широко распространенный формат libsvm, используемый во многих открытых реализациях SVM, в том числе, в одноименной библиотеке libsvm, которую мы применяем для обучения по частным подвыборкам.

В файле в формате libsvm каждая отдельная строка соответствует одному объекту, а формат каждой строки имеет вид:

<метка класса> <номер признака>:<значение> ... <номер признака>:<значение>

На рисунке 1 представлен пример фрагмента файла в формате libsvm.

```
1  1:2596 3:3 4:258 6:510 9:148 10:6279 11:1 43:1
-1  1:2785 2:155 4:242 5:118 7:238 9:122 10:6211 11:1 44:1
1  1:2606 5:5 6:633 7:222 8:225 10:6256 11:1 43:1
1  1:2605 5:7 6:573 7:222 8:230 9:144 10:6228 11:1 43:1
-1  1:2742 2:134 4:150 5:69 7:248 9:92 10:6091 11:1 44:1
```

Рисунок 3. Пример фрагмента файла данных для распознавания в формате libsvm.

Особенность данного формата заключается в том, что в случае, если данные являются разреженными (содержат много нулевых значений признаков), он позволяет хранить информацию в сжатой форме, указывая значения только тех признаков, значения которых отличны от нуля. Соответственно, разреженные данные при хранении в таком формате занимают меньше места на диске. Однако следует отметить, что для не разреженных данных такой формат, наоборот, требует больше памяти по сравнению с традиционным способом хранения матриц в текстовых файлах.

5. Стратегии работы с данными в формате libsvm при обучении по методу усредненных решающих правил

5.1. Стратегия 1. Единовременная загрузка всех данных в оперативную память с последующим взятием подвыборок

При обучении по небольшим обучающим совокупностям при помощи традиционных алгоритмов такая стратегия работы с данными является наиболее оптимальной.

В случае же обучения в условиях большой обучающей совокупности все данные могут не поместиться в оперативную память совместно с задачей обучения. Еще сильнее проблема усугубляется при использовании параллельных реализаций, когда каждый процесс или поток загружает свою копию данных.

В связи с этим такая стратегия имеет существенные ограничения в применении.

5.2. Стратегия 2. Отдельная загрузка случайных подвыборок при помощи традиционного последовательного чтения

Метод усреднения решающих правил на каждой итерации требует нахождения в памяти только одной случайной подвыборки объектов. Загрузка подвыборок из файла по мере необходимости позволяет снять ограничение на объем обучающей совокупности.

Простейшим с организационной точки зрения способом загрузки случайной подвыборки является чтение всех объектов с заданными номерами из файла.

Однако следует обратить внимание, что при хранении данных в формате libsvm число ненулевых признаков у разных объектов и, соответственно, длина разных строк в файле в

общем случае отличается, в связи с чем позиция начала объекта с заданным номером не может быть вычислена. Поэтому в рамках данной работы рассматриваемая стратегия реализована путем последовательного прохождения по строкам файла, начиная с первой, в ходе которого выбираются все встречающиеся объекты с заданными номерами. При этом очевидно, что просмотр файла можно закончить при достижении последнего объекта подвыборки.
Открытие, чтение и закрытие файла в данном случае осуществляются при помощи традиционных функций работы с файлами языка С.

5.3. Стратегия 3. Отдельная загрузка случайных подвыборок при помощи традиционного чтения с предварительной разметкой
Для повышения эффективности предыдущей стратегии загрузки данных в данной работе предлагается осуществить предварительную разметку файла - создать и заполнить дополнительную структуру данных в памяти, хранящую для каждого объекта позицию его начала в файле.
Реализация такой стратегии требует однократного прохождения перед началом работы алгоритма по всему файлу, но ускоряет время доступа к одному объекту. Соответственно, данная стратегия оказывается тем более выгодной, чем больше размер и число подвыборок используется в алгоритме.
Следует отметить, что данная стратегия требует выделения дополнительного объема оперативной памяти, который зависит от числа объектов, но этот объем оказывается существенно меньше по сравнению с тем, который требуется для хранения всей обучающей совокупности.

5.4. Стратегия 4. Отдельная загрузка случайных подвыборок при помощи отображения файла в память с предварительной разметкой
Еще одним направлением для повышения производительности загрузки данных, предлагаемым в данной работе, является замена традиционных операций чтения данных из файла на диске на системные операции отображения файла в оперативную память процесса. В данной работе для этой цели использовались функции Windows API, такие, как CreateFile (открытие файла), CreateFileMapping (создание объекта ядра системы для отображения файла) и MapViewOfFile (создание окна просмотра отображения).
Основная идея применения данной стратегии заключается в том, что функция MpViewOfFile возвращает указатель на область в динамической памяти, в которую отображается содержимое файла и вся дальнейшая работа с данными производится как с обычными указателями, что удобнее и быстрее (особенно при организации произвольного доступа к данным, как в данном случае) по сравнению с применением традиционных функций для работы с файлом на диске.
Для дополнительного повышения производительности при работе с большим числом подвыборок в данном случае, так же, как и в предыдущей стратегии, описанной в разделе 5.3, мы выполняем предварительную разметку данных, сохраняя позиции начала объектов в отдельном массиве.

6. Экспериментальное исследование

Описанные в разделе 5 стратегии загрузки данных сравнивались на известных наборах данных из репозитория LibSVM [12,13], таких как ijcnn1, mnist-576, mnist-784 и kddcup122, характеристики которых приведены в таблице 1.
Исследуемые наборы данных имеют разное число объектов и признаков, а также, разную степень плотности данных (разную долю ненулевых признаков). Эти параметры оказывают влияние на скорость загрузки данных.
В таблице 2 приведено среднее время отдельных этапов работы с данными для каждой из предложенных стратегий 1-4, усредненное по 3 запускам. Время загрузки одной подвыборки дополнительно усреднялось по количеству подвыборок.

Таблица 1. Наборы реальных данных для тестирования реализаций SVM.

Название набора данных	Объектов на обучении	Объектов на контроле	Число признаков	Доля ненулевых признаков
ijcnn1	35000	91701	22	59,10%
mnist-576	60000	10000	576	66,66%
mnist-784	60000	10000	784	61,73%
kddcup122	4 898 430	145253	54	10,23%

Число подвыборок в [11] для всех наборов реальных данных было принято равным 100. Однако для того, чтобы показать, как будет меняться время загрузки данных при увеличении числа подвыборок в таблице 2 приведены результаты как для 100, так и для 500 подвыборок.
Эксперименты проводились на ЭВМ со следующей конфигурацией: процессор Intel Core i5-6200U 2.3 ГГц (Turbo Boost 2.8 ГГц), оперативная память 6 ГБ, операционная системы Windows 10 Home x64.
Лучшие результаты (отдельно для 100 и для 500 подвыборок) для каждого набора данных выделены жирным шрифтом.

Таблица 2. Среднее время загрузки данных для разных стратегий и наборов данных.

Этап работы с данными			Среднее время (с)			
			ijcnn1	mnist-576	mnist-784	kddcup122
Страт. 1	Чтение полной выборки в память		0,315	15,56	18,29	31,73
	Формирование одной подвыборки в памяти		1,5e-05	1,5e-05	1,5e-05	1,5e-05
	Общее время работы с данными	100 подвыборок	**0,3165**	15,562	18,2915	31,7315
		500 подвыборок	**0,3225**	15,568	18,2975	31,7375
Стр. 2	Формирование одной подвыборки в памяти		0,033	1,542	2,155	2,357
	Общее время работы с данными	100 подвыборок	3,3	154,2	215,5	235,7
		500 подвыборок	16,5	771	1077,5	1178,5
Страт. 3	Разметка файла (чтение)		0,10	2,8	4,1	5,1
	Формирование подвыборки (чтение)		0,0068	0,0798	0,1055	0,0060
	Общее время работы с данными	100 подвыборок	0,78	10,78	14,65	5,7
		500 подвыборок	3,5	42,7	56,85	8,1
Страт. 4	Разметка файла (отображение)		0,20	0,55	0,67	1,17
	Формирование подвыборки (отображение)		0,00223	0,02001	0,03452	0,00045
	Общее время работы с данными	100 подвыборок	0,423	**2,551**	**2,671**	**1,215**
		500 подвыборок	1,315	**10,551**	**17,93**	**1,395**

Из таблицы 2 видно, что на маленьком наборе данных ijcnn1 наилучшей оказывается стратегия 1, согласно которой вся выборка сразу загружается в память. Однако, как было отмечено ранее, такая стратегия имеет ограничения и не может быть применима для больших объемов данных из-за возможной нехватки памяти.
Наиболее выгодной стратегией при работе с большими объемами данных, как и ожидалось, оказалась стратегия 4. Правда, общее время работы данных зависит от числа подвыборок, но как видно из таблицы, не только для 100, но и даже для 500 подвыборок данная стратегия имеет преимущество перед остальными стратегиями.
Наибольший эффект от применения стратегии 4 достигается при работе с большими, сильно разреженными наборами. Так, для набора данных kddcup122 применение стратегии 4 позволило сократить время загрузки данных (относительно стратегии 1) примерно в 22 раза для 500 подвыборок и примерно в 26 раз для 100 подвыборок.

7. Заключение

В данное работе предложено несколько стратегий для оптимизации загрузки данных при обучении распознаванию образов по методу усредненных решающих правил в условиях больших обучающих совокупностей.

Результаты экспериментов показали, что, как и ожидалось, наиболее эффективной оказалась стратегия, основанная на отображении файла в память с предварительной разметкой позиций объектов. Данная стратегия обеспечивает повышение производительности на всех исследованных наборах данных, кроме самого маленького. Наибольший же эффект от применения данной стратегии достигается при работе с большими, сильно разреженными наборами данных.

8. Литература

[1] Vapnik, V. Statistical Learning Theory // John-Wiley & Sons Inc., 1998.
[2] Bottou, L. Stochastic Learning // Advanced Lectures on Machine Learning. Lecture Notes in Artificial Intelligence. – 2004. – Vol. 3176. – P. 146-168.
[3] Boser, B.E. A Training Algorithm for Optimal Margin Classifiers / B.E. Boser, I.M. Guyon, V. Vapnik // Fifth Annual Workshop on Computational Learning Theory, ACM, 1992.
[4] Platt, J. Sequential minimal optimization: A fast algorithm for training support vector machines // Technical Report MSR-TR-98-14. Microsoft Research, 1998.
[5] Distributed delayed stochastic optimization / A. Agarwal, J.C. Duchi // Advances in Neural Information Processing Systems, 2011. – P. 873-881.
[6] Communication-efficient distributed dual coordinate ascent / M. Jaggi, V. Smith, M. Takac, J. Terhorst, S. Krishnan, T. Hofmann, M.I. Jordan // Advances in Neural Information Processing Systems, 2014. – P. 3068-3076.
[7] Ma, C. Adding vs. averaging in distributed primal-dual optimization / C. Ma, V. Smith, M. Jaggi, M.I. Jordan, P. Richtarik, M. Takac // arXiv preprint arXiv:1502.03508, 2015.
[8] Yang, T. Trading computation for communication: distributed stochastic dual coordinate ascent // Advances in Neural Information Processing Systems, 2013. – P. 629-637.
[9] Chang, E.Y. Psvm: Parallelizing support vector machines on distributed computers / E.Y. Chang, K. Zhu, H. Wang, H. Bai, J. Li, Z. Qiu, H. Cui // NIPS. – 2007. – Vol. 20.
[10] Joachims, T. Training linear svms in linear time // ACM KDD, 2006. – P. 217-226.
[11] LIBSVM-A Library for Support Vector Machines [Электронный ресурс]. – Режим доступа: https://www.csie.ntu.edu.tw/~cjlin/libsvm/.
[13] Joachims, T. Making large-scale support vector machine learning practical // Advances in kernel methods: support vector learning. – Cambridge, MA, USA: MIT Press, 1999. – P. 169-184.

Благодарности

Работа выполнена при поддержке Российского Фонда Фундаментальных Исследований, грант 18–07–01087.

Data load optimization for solving SVM problem via averaging decision rules method for big training sets

M.U. Kurbakov[1], A.I. Makarova[1], V.V. Sulimova[1]

[1]Tula State University, Lenin Ave. 92, Tula, Russia, 300012

Abstract. The Support Vector Machines (SVM) is one of the most convenient and effective instruments of two-class recognition. But there are some problems of its application for training in big data sets. One of these problems is the high computational complexity and the other consists in the necessity to save the full data set in RAM. The first problem can be decided by our decision rule averaging method, which allows us to quickly find an SVM solution that is close to exact. In this paper a specialized data handling scheme is proposed, which allows to avoid a one-time download of the full training set into the RAM. The proposed approach is based on the system mechanism of mapping files into memory and allows us to efficiently load arbitrary subsamples of objects from a file in the libsvm format, providing a significantly higher speed of work on large training sets compared to traditional methods of working with data. The proposed approach can be applied jointly with any incremental training methods that require fast loading from a libsvm file of an arbitrary subsamples of objects.