Быстрое приближенное решение двухклассовой задачи SVM для больших обучающих совокупностей

А.И. Макарова[1], В.В. Сулимова[1]

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

Аннотация. Метод опорных векторов (SVM) является удобным и надежным инструментом решения двухклассовой задачи распознавания, однако в условиях больших обучающих совокупностей он оказывается чрезвычайно трудоемким. В данной работе предлагается простой подход нахождения решения задачи SVM, основанный на усреднении решающих правил, построенных по небольшим случайным, возможно пересекающимся подвыборкам исходной обучающей совокупности. Предлагаемый подход позволяет быстро найти приближенное (но не сильно отличающееся от точного) решение задачи SVM даже для больших обучающих совокупностей. При этом предлагаемый метод является экономичным по памяти (что обеспечивает возможность его применения для обучения на одной вычислительной машине) и, в то же время, обладает высокой степенью параллелизма, что дает возможность его эффективной реализации с применением технологий параллельных и распределенных вычислений.

1. Введение

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.

Решение задачи двухклассового распознавания состоит из двух этапов: обучение и распознавание. На этапе обучения на основе анализа некоторой доступной совокупности объектов $\Omega^* = \{\omega_j, j = 1,...,N\} \in \Omega$, снабженных метками $y_j = y(\omega_j) \in \{+1;-1\}$, определяющими их классовую принадлежность к одному из двух классов, строится 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) определяет метку класса. Этап распознавания заключается в применении построенного решающего правила к новым объектам для определения их классов.

Одним из наиболее удобных и точных методов решения задач обучения двухклассовому распознаванию является метод опорных векторов (Support Vector Machines, SVM), предложенный В.Н. Вапником [1].

Задача обучения по методу SVM является задачей квадратичного программирования. Она имеет единственное решение, которое при небольшой размерности признакового пространства и небольшом числе объектов может быть легко найдено классическими методами оптимизации. Однако важной особенностью современных задач анализа данных, обусловленной глобальной информатизацией, является необходимость обучения в условиях большого числа объектов. Обучающая совокупность может содержать десятки и сотни тысяч, а в ряде случае и миллионы

объектов. При этом процесс обучения оказывается весьма трудоемким - построение одного решающего правила может занимать десятки минут, часов или даже дней. В связи с этим в последнее время появилось множество работ, направленных на повышение производительности решения задачи SVM.

В частности, для обучения в условиях, когда объектов много, но они помещаются в памяти одного компьютера или имеет место потоковое поступление данных, были предложены методы, позволяющие сократить объем требуемой памяти и ускорить вычисления на одном компьютере, среди которых можно выделить две группы: 1) инкрементные и декрементные методы, в том числе метод стохастического градиентного спуска (SGD) [2] и 2) методы, основанные на декомпозиции: метод образования фрагментов (chunking) [3], метод последовательной оптимизации Sequential Minimal Optimization (SMO) [4] и другие. Методы декомпозиции положены в основу таких популярных библиотек для решения задачи SVM, как LIBSVM [5,6] и SVMLight [7,8].

Кроме того, целая серия работ, посвящена адаптации методов обучения к условиям распределенной обработки большого числа объектов: в частности, параллельные реализации стохастического градиентного спуска (SGD) с синхронным [9] и асинхронным [10–12] взаимодействием процессов; параллельные реализации покоординатного спуска при решении двойственной задачи обучения [11–13]; методы, осуществляющие аппроксимацию матрицы скалярных произведений [14], кэширование значений матрицы скалярных произведений и "сжатие вычислений" (shrinking) [7] для сокращения требуемого объема памяти и уменьшения сетевого трафика; методы, учитывающие разреженность признаков [15]. Также осуществлялись неоднократные попытки выразить процесс обучения в терминах парадигмы параллельной обработки больших данных MapReduce [16,17] и т.д.

Таким образом, в мировом сообществе накопилось большое количество подходов к решению задачи обучения по методу SVM. Однако, несмотря на массовость исследований в данной области, универсального средства для решения задачи SVM до сих пор не найдено. Каждый из подходов и его реализаций имеет свои достоинства и недостатки.

В связи с этим целью данной работы является исследование принципиальной возможности разработки метода быстрого приближенного (но не сильно отличающегося от точного) решения задачи SVM в условиях большого числа объектов, являющегося экономичным по памяти (что обеспечивало бы возможность его применения для обучения на одной вычислительной машине) и, в то же время, обладающего высокой степенью параллелизма, что дало бы возможность его эффективной реализации с применением технологий параллельных и распределенных вычислений.

2. Быстрая аппроксимация метода опорных векторов с высокой степенью параллелизма

2.1. Основная идея предлагаемого подхода

Основная идея предлагаемого подхода заключается в формировании множества случайных небольших подвыборок обучающей совокупности, независимом обучении отдельно по каждой из подвыборок и последующем объединении частных решающих правил распознавания в одно общее решающее правило.

В рамках данной работы рассматривается только обучение в линейном признаковом пространстве. Простейшим способом объединения отдельных решающих правил в данном случае является их простое усреднение. Такой подход иногда используется в литературе в сочетании с методом стохастического градиента (SGD) [18], однако особенности SGD не позволяют достичь желаемого эффекта при распараллеливании - либо результат получается очень неточным, либо скорость слабо зависит от числа используемых машин (например, разница при использовании 10 и 100 процессов оказывается практически не заметной [18].

В данной же работе в качестве базового метода обучения для каждой из подвыборок предлагается использовать любой подход, позволяющий найти решение задачи обучения с требуемой точностью, например, метод SMO, реализованный в библиотеке LibSVM.

Обоснованием для усреднения решающих правил могут служить следующие рассуждения.

Пусть $[X,Y],\ X=[\mathbf{x}_j,\ j=1,...,N],\ \mathbf{x}_j\in R^m,\ \ Y=[y_j,\ j=1,...,N],\ \ y_j\in\{-1;1\}$ – исходная обучающая совокупность, $[X,Y]^{(i)}\in[X,Y],i=1,...,k$ – набор случайных подвыборок из нее.

Результат обучения по каждой случайной подвыборке $[X,Y]^{(i)},i=1,...,k$ представляет собой оценки двух параметров – направляющего вектора оптимальной разделяющей гиперплоскости $\hat{\mathbf{a}}^{(i)}=\hat{\mathbf{a}}([X,Y]^{(i)})$ и смещения $\hat{\mathbf{b}}^{(i)}=\hat{\mathbf{b}}([X,Y]^{(i)})$. Очевидно, что эти оценки будут зависеть от того, какие наборы объектов попадут в соответствующие подвыборки.

При этом, поскольку формирование подвыборок осуществляется случайно, совокупность пар оценок $[\hat{\mathbf{a}}^{(i)},\hat{b}^{(i)}],i=1,...,k$ можно рассматривать как совокупность одинаково распределенных случайных величин с конечными математическим ожиданием m и дисперсией d.

Усредненная оценка параметров решающего правила $[\hat{\mathbf{a}},\hat{b}]$, где $\hat{\mathbf{a}}=\frac{1}{k}\sum_{i=1}^{k}\hat{\mathbf{a}}^{(i)}$ и $\hat{b}=\frac{1}{k}\sum_{i=1}^{k}\hat{b}^{(i)}$, также является случайной величиной с математическим ожиданием $M[\hat{\mathbf{a}},\hat{b}]=m,$ но ее дисперсия, согласно свойствам числовых характеристик случайных величин будет равна $D[\hat{\mathbf{a}},\hat{b}]=\frac{d}{k}$, т.е. с ростом числа подвыборок дисперсия оценок решающего правила имеет тенденцию уменьшаться.

Более того, согласно закону больших чисел, усредненная оценка параметров решающего правила сходится по вероятности к математическому ожиданию соответствующей случайной величины:

$$\lim_{k\to\infty}P\left\{\left|\frac{1}{k}\sum_{i=1}^{k}[\hat{\mathbf{a}}^{(i)},\hat{b}^{(i)}]-m\right|<\varepsilon\right\}=1,\ \ \forall\,\varepsilon>0. \qquad (1)$$

Таким образом, при увеличении числа подвыборок усредненная оценка должна стабилизироваться и переставать вести себя как случайная величина. Этот эффект хорошо виден в случае двухмерного признакового пространства (рисунок 1).

Правда, следует отметить, что значение математического ожидания m, к которому должно стремиться усредненное решающее правило, не обязано совпадать с точным решением задачи SVM по полной обучающей совокупности (в этом смысле получаемое решение является приближенным), однако есть основания предполагать, что в ряде случаев усреднение может позволить получить решение, обладающее большей обобщающей способностью по сравнению с традиционным подходом.

2.2. Последовательный алгоритм

Предлагаемый в данной работе подход допускает несколько вариантов алгоритмических реализаций. Самый простой вариант 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. Блок-схема соответствующего алгоритма представлена на рисунке 2.

Однако следует отметить, что предложенный подход обладает высокой степенью параллелизма, поскольку, фактически, все итерации данного алгоритма являются независимыми и могут быть выполнены параллельно. В результате предложенный подход может быть эффективно реализован с применением технологий параллельного и распределенного программирования для систем с общей и распределенной памятью.

3. Экспериментальное исследование

3.1. Условия проведения экспериментального исследования

Тестирование последовательных алгоритмов проводилось на ПК со следующими техническими характеристиками: Intel® Core™ i5-4210U (2.4GHz), 2 ядра, 6Gb RAM, а тестирование параллельных версий - на суперкомпьютерном комплексе МГУ имени М.В. Ломоносова [19].

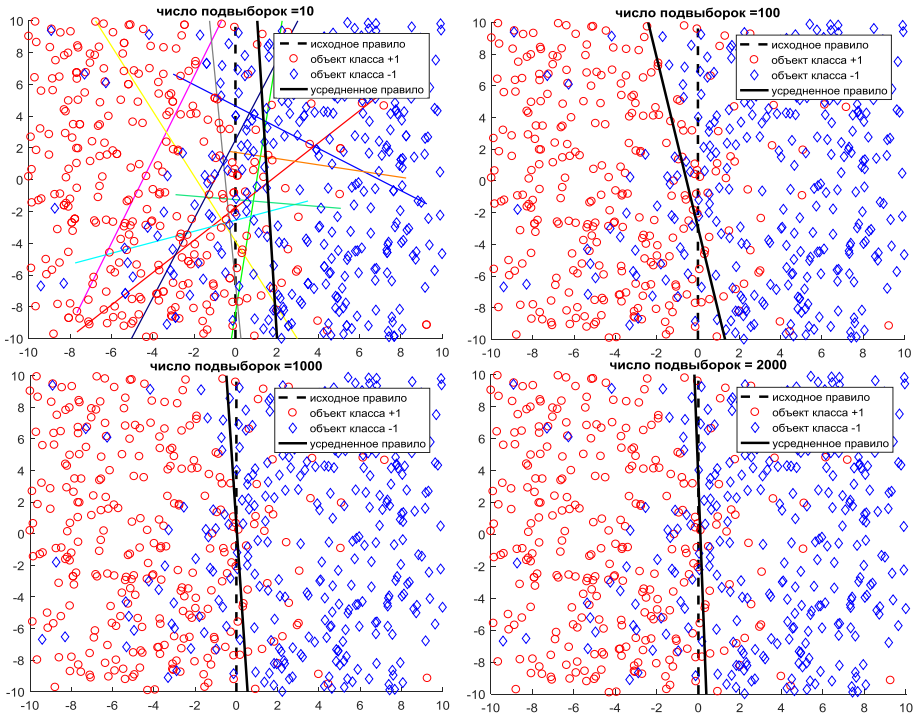

Рисунок 1. Изменение решающего правила при увеличении числа подвыборок. Пунктирной линией здесь показано решающее правило, построенное при обучении по всей обучающей совокупности, а сплошной – усредненное решающее правило.

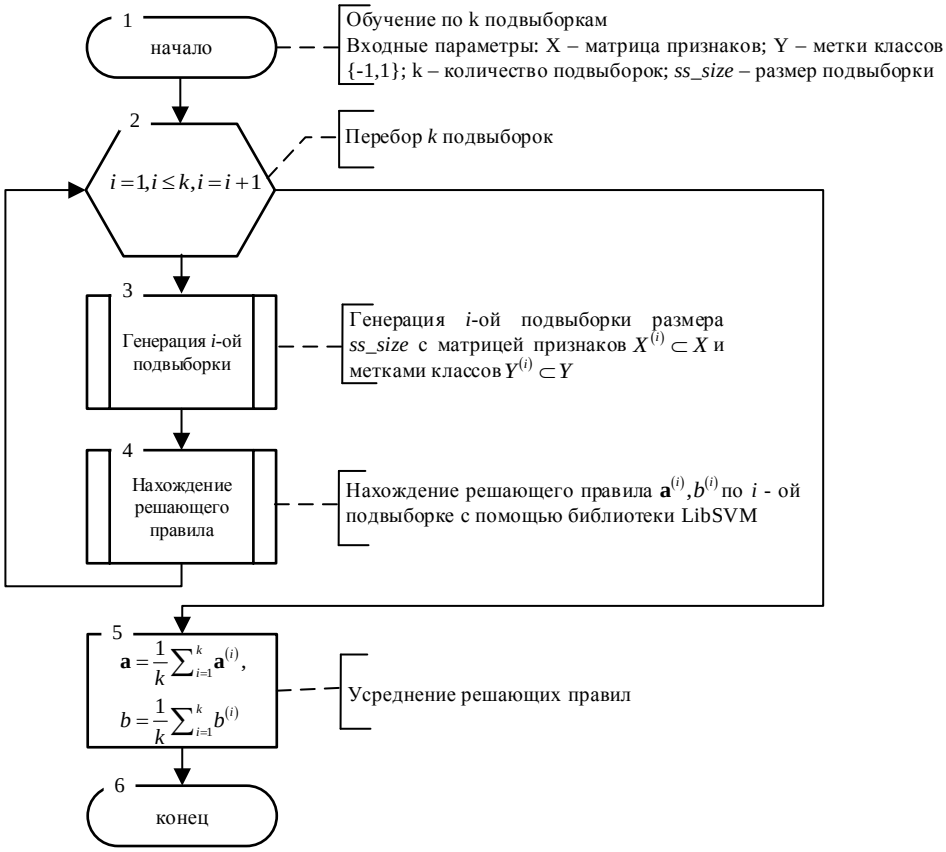

Рисунок 2. Блок-схема алгоритма быстрого приближенного решения задачи SVM для случая фиксированного числа подвыборок.

3.2. Исследование на модельных данных

3.2.1. Генерация данных

Генерация модельных данных в данной работе осуществляется согласно вероятностной модели метода опорных векторов [20], предполагающей, что объекты классов $y=1$ и $y=-1$ порождаются в соответствии с условными плотностями распределения вероятностей $\varphi(z \mid y=1;\mathbf{a},b;c)$ и $\varphi(z \mid y=-1;\mathbf{a},b;c)$, каждая из которых с одной стороны от гиперплоскости с параметрами $\mathbf{a},b$ имеет вид равномерного закона распределения, а с другой - экспоненциально убывает с ростом расстояния от гиперплоскости (рисунок 3). При этом параметр c определяет степень "перемешанности" объектов двух классов.

Пусть в признаковом пространстве размерности m задана некоторая гиперплоскость, описывающаяся уравнением $\mathbf{a}^T\mathbf{x}+b=0$.

Вообще говоря, гиперплоскость может быть любой, но для простоты будем считать, что она определяется следующим образом: $\mathbf{a}=[1,0,0,...,0]^T$, $b=0$.

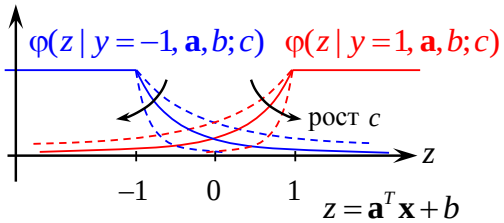

Рисунок 3. Распределение объектов класса.

Все объекты генерируются внутри гиперкуба со стороной α, такого, что заданная гиперплоскость делит его пополам. В таблице 1 представлены характеристики сгенерированных наборов данных.

Таблица 1. Описание сгенерированных выборок.

Набор данных	α	b	c	Число объектов	Число признаков
1	10	0	0,8	1000	2
2	10	0	0,8	1000	20
3	10	0	0,8	1000	200
4	10	0	0,8	10000	20
5	10	0	0,8	10000	200
6	10	0	0,8	100000	200

3.2.2. Результаты экспериментов на модельных данных

Результаты работы разработанного алгоритма (на MatLAB) сравнивались с результатами работы реализации метода стохастического градиента [21] для python (sklearn.linear_model. SGDClassifier), реализации библиотеки LIBLINEAR [22] для python (sklearn.svm.LinearSVC), а также библиотеки libSVM [5,6] для MatLAB.

На рисунке 4 представлены графики изменения качества усредненных решающих правил, полученных предложенным методом, в зависимости от времени работы алгоритма, которое увеличивается с увеличением числа используемых случайных подвыборок. Графики приведены для разных наборов модельных данных, описанных в таблице 1. Для сравнения предложенного подхода с существующими открытыми реализациями по скорости сходимости и качеству решения, на рисунке 4 приведены также графики для метода стохастического градиента (SGD), полученные при изменении количества эпох, влияющих на продолжительность работы алгоритма. Для остальных параметров метода SGD были взяты их значения по умолчанию. Библиотеки LibSVM и LIBLINEAR не имеют параметра, влияющего на скорость работы алгоритма. Результаты их работы отображены на графиках в виде прямых линий на уровне, соответствующем качеству построенного решающего правила.

Из графиков (рисунок 4) видно, что, как и ожидалось, при увеличении числа подвыборок усредненная оценка параметров решающего правила в предложенном подходе стабилизируется и перестает вести себя как случайная величина. Кроме того, можно отметить, что предложенный подход, как и метод стохастического градиента, являясь приближенным методом, в ряде случаев дает решение задачи SVM, характеризующееся большей точностью (accuracy) по сравнению с результатами, полученными при помощи библиотек LibSVM и LIBLINEAR, которые итерационно ищут точное решение задачи SVM.

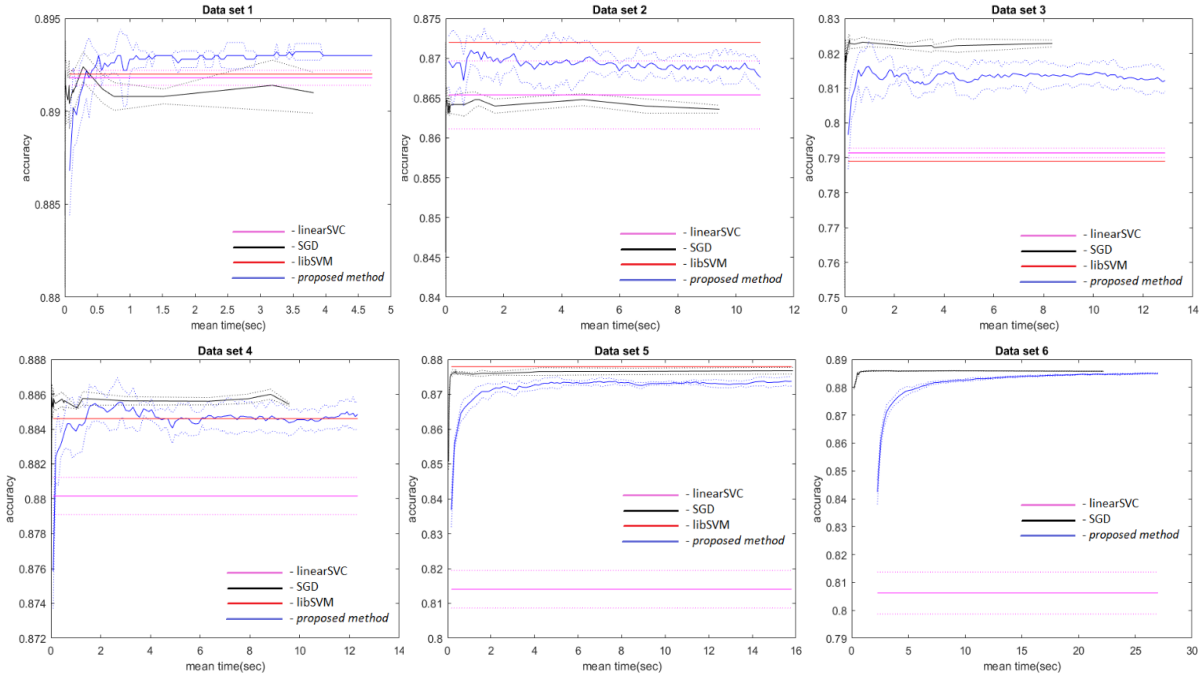

Рисунок 4. Графики зависимости точности (accuracy) построенного решающего правила от времени работы алгоритмов (в секундах) для разных наборов модельных данных.

3.2.3. Исследование на реальных данных

Предлагаемый подход экспериментально тестировался на известных наборах данных из репозитория LibSVM [6], таких как ijcnn1, mnist-576, mnist-784 и covtype, характеристики которых приведены в таблице 2.

Таблица 2. Наборы реальных данных для тестирования реализаций SVM.

Название набора данных	Объектов на обучении	Объектов на контроле	Число признаков
ijcnn1	35000	91701	22
mnist-576-rbf-8vr	60000	10000	576
mnist-784-poly-8vr	60000	10000	784
covtype-2vr	435759	145253	54

Приведенные в таблице 2 наборы данных являются разреженными и масштабированными, однако, было выявлено, что они не являются стандартизованными, т.е. математическое ожидание и дисперсия по многим признакам отличается от стандартных значений (0 и 1, соответственно). В таблице 3 приведены результаты сравнения точности и времени работы (в секундах) для ряда популярных реализаций SVM, находящихся в открытом доступе и предлагаемого подхода (приведенные результаты во всех экспериментах получены при усреднении решающих правил, построенных по 100 случайным подвыборкам из 300 объектов).

Таблица 3. Сравнение результатов тестирования открытых реализаций SVM с предлагаемым методом.

	SVMlight[а] (C)		libSVM[а] (C)		SVC[б] (LibSVM, python)		πSVM[а] (MPI libSVM, 1 процесс)		liblinear[а] (C)		linear-SVC[б] (Liblinear, python)		MPI - liblinear[а] (1 процесс)		SGD[б] (python)		предлагаемый метод[б]	
	время (с)	точ-ность	время (с)	точ-ность	время (с)	точ-ность	время (с)	точ-ность	время (с)	точ-ность	время (с)	точ-ность	время (с)	точ-ность	время (с)	точ-ность	время (с)	точ-ность
ijcnn1																		
обуч.	12,41	0,917	22,193	0,917	40,4	0,917	13,99	0,917	0,16	0,913	0,9151	0,9164	0,45	0,913	9,889	0,905	0,412	0,9147
распозн.	0,0267		21,146		25,72		24,67		0,34		0,02		0,09		0,006		0,004	
mnist_576																		
обуч.	51,26	0,994	512,736	0,994	499,8	0,994	95,41	0,994	2,17	0,989	4,5533	0,9935	4,73	0,992	0,511	0,903	5,74	0,987
распозн.	0,0067		30,506		44,08		35,12		1,03		0,0156		1,09		0,019		0,005	
mnist_784																		
обуч.	39,916	0,967	838,573	0,967	717,7	0,967	450,52	0,967	2,49	0,948	5,5304	0,9672	5,37	0,948	0,774	0,903	5,65	0,945
распозн.	0,03		64,767		50,13		68,45		1,27		0,0156		1,32		0,031		0,005	
covtype_2vr																		
обуч.	>40000	-	$>1{,}8{\cdot}10^5$	-	>3600	-	>40000	-	36,39	0,755	238,44	0.545	0,79	0,612	185	0,505	987	0,6347
распозн.	-		-		-		-		0,69		0,1718		0,8		0,172		0,013	
ijcnn1 (со стандартизацией)																		
обуч.	197,21	0,957	200,8	0,957	122,5	0,917	25,96	0,930	0,14	0,602	5,486	0,916	0,08	0,602	7,44	0,905	2,11	0,9232
распозн.	102,73		82,1		1,13		108,78		0,59		0,018		0,58		0,025		0,003	
mnist_576 (со стандартизацией)																		
обуч.	33072,2	0,903	31946,4	0,992	>3600	-	144,98	0,999	73,09	0,822	32,067	0,9905	36,95	0,822	117,5	0,988	5,31	0,9879
распозн.	1189,63		16,27		-		52,115		1,57		0,047		1,57		0,026		0,004	
mnist_784 (со стандартизацией)																		
обуч.	31083,3	0,903	19446,2	0,903	>3600	-	483,86	0,995	13,98	0,678	62,699	0,9458	13,26	0,679	83,83	0,947	6,256	0,9637
распозн.	1039,41		730,48		-		82,088		1,32		0,0312		1,3		0,016		0,006	
covtype_2vr (со стандартизацией)																		
обуч.	>40000	-	$>1{,}8{\cdot}10^5$	-	>3600	-	>40000	-	2,72	0,756	157,99	0,7561	2,47	0,754	129,6	0,693	11,61	0,7534
распозн.	-		-		-		-		4,24		0,0937		4,29		0,078		0,02	

[а]Суперкомпьютер НИИ ВЦ МГУ «Ломоносов»: 1 узел: Intel Xeon X5570 (2.93 GHz), 8 ядер, 8Gb RAM, узлов: 5104.
[б]ПК: Intel® Core™ i5-4210U (2.4GHz), 2 ядра, 6Gb RAM.

Помимо реализаций, участвовавших в исследовании на модельных данных, а именно, предложенного подхода (MatLAB), SGD [21] для python (sklearn.linear_model.SGDClassifier), библиотеки LibSVM для python(sklearn.svm.SVC), названной здесь для краткости SVC, и библиотеки LIBLINEAR [22] для python (sklearn.svm.LinearSVC), названной linearSVC, в данном сравнении использовались такие открытые реализации на языке C библиотек libSVM, LIBLINEAR и SVMLight [8], а также две параллельные реализации - πSVM [23] и MPIliblinear [24], написанные на языке C с применением технологии MPI и представляющие собой параллельные версии библиотек LibSVM и MPIliblinear, соответственно.

Таблица 4. Изменение времени обучения/распознавания для πSVM при увеличении числа процессов.

время (с)	Число процессов								точность
	1	2	4	8	16	32	64	128	
обучение	95,41	51,36	34,11	19,07	13,45	7,96	7,57	8,41	0,9935
распозн.	35,12	18,83	10,3	5,94	3,8	2,76	2,33	2,35	
обучение	13,99	9,28	7,79	5,87	6,24	10,65	23,89	39,93	0,9165
распозн.	24,67	12,75	6,61	3,55	2,07	1,38	1,06	1,03	
обучение	450,52	238,36	147,32	72,69	39,8	26,6	19,18	17,62	0,9672
распозн.	68,45	35,73	20,4	11,21	6,67	4,4	3,35	3,35	

В таблице 3 все приведенные результаты и для последовательных, и для параллельных версий получены при вычислении при помощи одного процесса, т.е. без использования возможностей распараллеливания. Зависимости изменения времени (в секундах) обучения и распознавания от числа процессов для параллельных реализаций πSVM и MPIliblinear приведены в таблицах 4 и 5, соответственно.
Для каждого набора данных в таблицах 3-5 приведены время обучения, время контроля и достигнутая точность.

Полученные результаты хорошо показывают наличие проблемы обучения в условиях большого числа объектов. Для ряда библиотек, даже для таких относительно небольших наборов данных, время обучения оказывается слишком большим.

Таблица 5. Изменение времени обучения/распознавания для MPIliblinear при увеличении числа процессов.

время (с)	Число процессов								точность
	1	2	4	8	16	32	64	128	
mnist-576									
обучение	4,73	2,37	1,23	0,6	0,31	0,18	0,09	0,08	0,9918
распозн.	1,09	1,06	1,07	1,14	1,14	1,12	1,14	1,12	
covtype									
обучение	0,79	0,41	0,2	0,1	0,05	0,05	0,05	0,05	0,6118
распозн.	0,8	0,77	0,78	0,78	0,78	0,77	0,78	0,78	
ijcnn1									
обучение	0,09	0,04	0,03	0,01	0,01	0,1	0,11	0,1	0,9139
распозн.	0,45	0,42	0,41	0,47	0,48	0,47	0,47	0,47	
mnist-784									
обучение	5,37	2,69	1,41	0,68	0,35	0,22	0,21	0,17	0,9476
распозн.	1,32	1,3	1,32	1,38	1,43	1,42	1,43	1,42	

Библиотека Liblinear, оптимизированная для обучения в разреженном признаковом пространстве, позволяет существенно ускорить решение задачи, однако эксперименты показывают, что в ряде случаев точность получаемого решения оказывается существенно ниже, чем у других библиотек.

Из таблицы 3 видно, что в ряде случаев точность распознавания позволяет повысить стандартизация. А для предложенного подхода, для набора данных covtype, в данном случае наблюдается еще и существенное повышение скорости решения. Такой эффект, по-видимому, связан с плохой сходимостью для не стандартизованных данных метода SMO, лежащего в основе библиотеки libSVM, которая использовалась для обучения по частным случайным подвыборкам при реализации предложенного подхода.

Очевидно, что использование параллельных реализаций позволяет повысить скорость решения, однако, даже при использовании 128 процессов суперкомпьютерного комплекса "Ломоносов", библиотека πSVM оказывается существенно менее производительной по сравнению с предлагаемым подходом. Такой эффект обусловлен в первую очередь нелинейной вычислительной сложностью метода libSVM и его параллельной версии πSVM относительно числа объектов, из-за которой решение большого числа задач с небольшим числом объектов (как в предлагаемом подходе) требует меньше времени, чем решение одной большой задачи даже без применения технологий параллельного программирования в первом случае.

Более того, следует обратить внимание, что увеличение числа процессов при использовании параллельных реализаций оказывается целесообразным лишь до некоторого предела, после чего получаемое ускорение начинает снижаться. Это обусловлено тем, что метод, лежащий в основе πSVM являтся итерационным и имеет зависимости по данным, требующие синхронизации и обмена информацией между процессами после каждой итерации, что существенно ограничивает возможный эффект от применения технологий параллельного программирования. Предложенный подход имеет существенно более высокую степень параллелизма по данным, что открывает широкие возможности для дальнейшего повышения производительности.

Наиболее высокую производительность в рамках данного исследования демонстрирует реализация MPIliblinear, однако она наследует особенности исходной библиотеки liblinear, которая в ряде случаев может существенно отставать по точности решения от других библиотек, особенно на стандартизованных данных.

В целом, проведенное исследование показывает, что предложенный подход позволяет достаточно быстро получить результат, не сильно отличающийся от точного, а в некоторых случаях, как и для модельных данных, точность получается более высокой, чем у других библиотек. Кроме того, в ряде случаев, точность предложенного подхода может быть повышена за счет увеличения размера случайных подвыборок.

4. Заключение

В данной работе предложен простой подход для обучения по методу SVM в условиях большой обучающей совокупности, основанный на усреднении решающих правил, построенных по случайным подвыборкам исходного обучающего множества объектов. Данный подход позволяет быстро найти приближенное (но не сильно отличающееся от точного) решение задачи SVM в условиях большого числа объектов, является экономичным по памяти (что обеспечивает возможность его применения для обучения на одной вычислительной машине) и, в то же время, обладает высокой степенью параллелизма, что дает возможность его эффективной реализации с применений технологий параллельных вычислений.

5. Литература

[1] Vapnik, V. Statistical Learning Theory. – John-Wiley & Sons Inc., 1998.
[2] Bottou, L. Stochastic Learning // Advanced Lectures on Machine Learning. – 2004. – 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] Chang, C.-C. LIBSVM: a library for support vector machines / C.-C. Chang, C.-J. Lin // Software available, 2001. [Electronic resource]. – Access mode: http://www.csie.ntu.edu.tw/ cjlin/libsvm.
[6] LIBSVM-A Library for Support Vector Machines [Электронный ресурс]. Режим доступа: https://www.csie.ntu.edu.tw/~cjlin/libsvm/
[7] 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.
[8] svmLight Support Vector Machine [Электронный ресурс]. – Режим доступа: http://svmlight. joachims.org.
[9] Dekel, O. Optimal distributed online prediction using mini-batches / O. Dekel, R. Gilad-Bachrach, O. Shamir, L. Xiao // J. Mach. Learn. Res. – 2012. – Vol. 13(1). – P. 165-202.
[10] Agarwal, A. Distributed delayed stochastic optimization / A. Agarwal, J.C. Duchi // Advances in Neural Information Processing Systems. – 2011. – P. 873-881.
[11] Jaggi, M. 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.
[12] 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.
[13] Yang, T. Trading computation for communication: distributed stochastic dual coordinate ascent // Advances in Neural Information Processing Systems, 2013. – P. 629-637.
[14] 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.
[15] Joachims, T. Training linear svms in linear time // ACM KDD, 2006. – P. 217-226.
[16] Chu, C.-T. Map reduce for machine learning on multicore / C.-T. Chu, S.K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A.Y. Ng, K. Olukotun // NIPS, 2006.
[17] Zhao, H.X. Parallel support vector machines on multi-core and multiprocessor systems / H.X. Zhao, F. Magoules // 11th International Conference on Arti_cial Intelligence and Applications (AIA), 2011. DOI: 10.2316/P.2011.717-056.

[18] Agarwal, A. A reliable effective terascale linear learning system / A. Agarwal, O. Chapelle, M. Dudık, J. Langford // Journal of Machine Learning Research. – 2014. – Vol. 15(1). – P. 1111-1133.
[19] Воеводин, В.В. Практика суперкомпьютера "Ломоносов" / В.В. Воеводин, С.А. Жуматий, С.И. Соболев, А.С. Антонов, П.А. Брызгалов, Д.А. Никитенко, К.С. Стефанов, В.В. Воеводин // Открытые системы. – 2012. – № 7. – С. 36-39.
[20] Татарчук, А.И. Байесовские методы опорных векторов для обучения распознаванию образов с управляемой селективностью отбора признаков // Диссертация к.ф.-м.н. – Вычислительный центр РАН, 2014.
[21] Stochastic Gradient Descent SVM classifier [Электронный ресурс]. – Режим доступа: https:// github.com/joa-ofaro/SVMSGD.
[22] Mpliblinear Support Vector Machine [Электронный ресурс]. – Режим доступа: https:// www.csie.ntu.edu.tw/~ cjlin/liblinear/.
[23] PiSvMSoftware [Электронный ресурс]. – Режим доступа: http://pisvm.source–forge.net.
[24] Mpiliblinear Support Vector Machine [Электронный ресурс]. – Режим доступа: https:// www.csie.ntu.edu.tw/~cjlin/libsvmtools/distributed–liblinear/

Благодарности
Работа выполнена с использованием оборудования Центра коллективного пользования сверхвысокопроизводительными вычислительными ресурсами МГУ имени М.В. Ломоносова.
Работа выполнена при поддержке Российского Фонда Фундаментальных Исследований, грант 18–07–01087.

Fast approximate two-class SVM learning for large training sets

A.I. Makarova[1], V.V. Sulimova[1]

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

Abstract. Support Vector Machines (SVM) is one of the most convenient and reliable tools for solving the two-class recognition problem, but it is very laborious in the case of large training sets. In this paper we propose a simple approach to finding a solution of the SVM problem, based on averaging the decision rules, which are constructed from small random, possibly intersecting subsamples of the initial training set. The proposed approach allows to quickly find an approximate (but not very different from the exact) solution of the SVM problem, even for large training sets. In addition, the proposed approach is memory-efficient and so provides the possibility of training on a single computer, and, at the same time, has a high degree of parallelism and can be effectively implemented using parallel and distributed computing technologies.