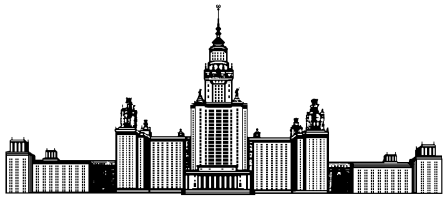

Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Щекалев Алексей Андреевич

Параллельная реализация стохастического логического корректора при помощи технологии CUDA

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
д.ф-м.н.
Дюкова Елена Всеволодовна

Москва, 2018

Содержание

1 Введение

В различных областях человеческой деятельности возникают задачи, в которых требуется найти решение на основе анализа большого объема накопленных знаний. К ним относятся задачи классификации, распознавания и прогнозирования, возникающие в различных плохо формализованных областях знаний таких, как медицинская диагностика, обработка социологической информации, техническое и геологическое прогнозирование, анализ банковской деятельности и т.д. Для решения перечисленных задач успешно применяются методы распознавания образов, в частности методы, основанные на обучении по прецедентам.

Постановка задачи обучения по прецедентам заключается в следующем. Исследуются некоторое множество объектов M, про которое известно, что оно может быть разбито на непересекающиеся подмножества (классы) $K_1, \ldots, K_l,\ l \geq 2$. Под обучающей информацией понимается совокупность примеров описаний изучаемых объектов, полученных на основе наблюдения ряда характеристик этих объектов. Для каждого примера задан класс, которому он принадлежит. Требуется уметь классифицировать объекты, не вошедшие в обучающую выборку, т.е. по признаковому описанию каждого объекта определять, какому классу он принадлежит.

При решении прикладных задач классификации хорошо себя зарекомендовали методы, основанные на логическом анализе признаковых описаний объектов. В случае, когда данные представлены в целочисленном виде, используется понятие элементарного классификатора (эл.кл.). *Эл.кл.* — это элементарная конъюнкция, определенная на признаковых описаниях объектов. Если на описании некоторого объекта S элементарная конъюнкция обращается в единицу, то говорят, что объект S *содержит данный эл.кл.*

В классических логических процедурах распознавания ставится задача построения множества корректных элементарных классификаторов. Эл.кл. называется *корректным* для класса K, если не существует пары обучающих объектов, в которой один объект принадлежит K, а второй не принадлежит K, и такой, что оба объекта содержат данный эл.кл. В процессе обучения для каждого класса K строится некоторое семейство корректных эл.кл. На этапе распознавания каждый из найденных эл.кл. участвует в процедуре голосования и формирования оценок принадлежности

рассматриваемого объекта к классам. В таких моделях корректность распознающего алгоритма обеспечивается корректностью каждого из найденных эл.кл.

Однако, существуют прикладные задачи классификации, на которых классические логические алгоритмы показывают плохие результаты. Например, такая ситуация возникает, когда признаки принимают слишком большое число значений. Тогда не удается найти достаточное число информативных корректных эл.кл. Один из способов решения этой проблемы — использование логических корректоров.

Логический корректор — современная модель логических процедур распознавания, при конструировании которых используются идеи алгебраического подхода. При этом в качестве базовых алгоритмов используется произвольный элементарный классификатор. Итоговый алгоритм корректен за счет использования корректных наборов элементарных классификаторов.

В [1] показано, что задача построения корректных наборов эл.кл. для класса K сводится к задаче поиска покрытий булевой матрицы L_K, специальным образом построенной по обучающей выборке. Число столбцов в матрице L_K равно числу используемых эл.кл. В [3], [4] рассмотрены вопросы практического использования различных моделей логического корректора. При этом строились модели, в которых каждый набор обладает хорошей распознающей способностью. Для решения этой задачи использовался генетический алгоритм.

В [3] проведено исследование двух моделей: модели, основанной на построении корректных наборов по мощности близких к минимальным и модели, основанной на построении коллектива тупиковых корректных наборов, каждый из которых обладает хорошей распознающей способностью. В качестве особей генетического алгоритма выступали корректные наборы эл.кл., порождаемые элементарными конъюнкциями ранга 1. Дополнительно исследован случай использования только монотонных корректных наборов из эл.кл., когда в качестве корректирующей выступает монотонная булева функция. Наилучшие результаты получены при использовании модели логического корректора MON, основанной на построении коллектива монотонных корректных наборов из эл.кл.

Использование эл.кл. ранга 1 — довольно жесткое условие. Снятие этого ограничения приводит к дополнительным временным затратам. В [4] предложено искать

корректные наборы эл.кл. не среди всего множества наборов элементарных классификаторов, а среди некоторого подмножества, называемого локальным базисом. В [5] построена модель стохастического логического корректора MONS на базе эл.кл. произвольного ранга. Данные модели показали хорошие результаты на тестовых задачах.

При больших объемах обучающей выборки и высокой значности признаков даже генетический алгоритм работает достаточно медленно. В настоящей работе существенного сокращения временных затрат удалось достичь за счет применения параллельных вычислений на основе программно-аппаратной архитектуры унифицированных вычислений CUDA (Compute Unified Device Architecture). Указанная технология позволяет выполнить операции, не требующие длительного времени, на центральном процессоре компьютера (CPU), а все сложные операции на графических процессорах. На каждом графическом процессоре GPU может быть задействовано большое число вычислительных ядер. По сравнению с традиционными подходами к организации вычислений посредством возможностей графических API у CUDA отмечаются определённые преимущества.

В настоящей работе построена модель логического корректора CLC (CUDA logical corrector), базирующиеся на эффективном распараллеливании поиска корректных наборов элементарных классификаторов с хорошей распознающей способностью. В модели CLC снято ограничение на ранг элементарного классификатора и осуществляется поиск минимальных по длине корректных наборов.

Для дополнительного сокращения временных затрат реализована стохастическая модель логического корректора, позволяющая находить достаточно представительную случайную выборку искомых корректных наборов эл.кл. Проведено тестирование разработанной модели логического корректора на реальных задачах. Полученные результаты свидетельствуют об эффективности предложенных подходов к обработке больших данных методами алгебро-логического анализа.

2 Логический корректор

2.1 Определения и обозначения

Определение 1. T — множество обучающих объектов. $T \cap K$ — обучающие объекты из класса K. $T \cap \overline{K}$ — множество обучающих объектов не принадлежащих классу K.

Определение 2. Пусть $H = \{x_{j_1}, \ldots, x_{j_r}\}$ — набор различных целочисленных признаков, $\sigma = \{\sigma_1, \ldots, \sigma_r\}$ — набор допустимых значений этих признаков.
Элементарный классификатор (эл.кл.)— конъюнкция вида $x_{j_1}^{\sigma_1}, \ldots, x_{j_r}^{\sigma_r}$, обозначаемая далее $B_{(H,\sigma)}$

Определение 3. Объект S *содержит эл.кл.* $B_{(H,\sigma)}$, если $B_{(H,\sigma)}(S) = 1$

Определение 4. Эл.кл. считается *корректным*, если не существует двух объектов из разных классов, содержащих данный эл.кл.:

$$\nexists\ S' \in T \cap K,\ S'' \in T \cap \overline{K} : B_{(H,\sigma)}(S') = B_{(H,\sigma)}(S'') = 1$$

Определение 5. Множество $U = \left\{B_{(H_1,\sigma_1)}, \ldots, B_{(H_q,\sigma_q)}\right\}$ — *набор элементарных классификаторов.*

Определение 6. Набор эл.кл. U называется *корректным для класса* K, $K \in \{K_1, \ldots, K_l\}$, если для $\forall(S', S'')$ — пара обучающих объектов, $S' \in K$, $S'' \notin K$ существует монотонная функция алгебры логики $F_{U,K}$:

$$F_{U,K}(U(S')) \neq F_{U,K}(U(S''))$$

Определение 7. Корректный для класса K набор из эл.кл. U называется *классифицирующим набором для класса* K

Определение 8. Классифицирующий набор U для класса K называется *монотонным*, если существует монотонная функция алгебры логики $F_{U,K}$:

$$F_{U,K}(U(S)) = \begin{cases} 1, & \text{если } S \in K \cap T, \\ 0, & \text{если } S \notin K \cap T \end{cases}$$

Определение 9. Классифицирующий набор U для класса K называется *антимонотонным*, если существует монотонная функция алгебры логики $F_{U,K}$:

$$F_{U,K}\left(U\left(S\right)\right)=\begin{cases}0, & \text{если } S\in K\cap T,\\ 1, & \text{если } S\notin K\cap T\end{cases}$$

Обозначение 1. $C_K^{\mathcal{A}}$ — множество эл.кл. для класса K, построенное распознающим алгоритмом $\mathcal{A}$.

2.2 Детерминированная модель логического корректора

Построение распознающего алгоритма $\mathcal{A}$ основано на конструировании для каждого класса $K,\ K\in\{K_1,\ldots,K_l\}$, множества корректных наборов эл.кл. $W_{\mathcal{A}}\left(K\right)$ и проведении процедуры голосования по каждому набору эл.кл. из $\bigcup\limits_{i=1} W_{\mathcal{A}}\left(K_i\right)$

В зависимости от значения функции $B_{(H,\sigma)}\left(S\right),\ (H,\sigma)\in C_K$ распознаваемый объект S либо получает голос за принадлежность классу K, либо не получает этого голоса. Оценка $\Gamma_K\left(S\right)$ принадлежности объекта S к классу K вычисляется на основе суммирования голосов, полученных при голосовании по всем эл.кл. из наборов $U,\ U\in W_{\mathcal{A}}\left(K\right)$. Анализ оценок $\Gamma_{K_1}\left(S\right),\ldots,\Gamma_{K_l}\left(S\right)$ позволяет классифицировать объект S. Таким образом, общая схема работы логического корректора состоит из двух этапов: обучение и распознавание.

На этапе обучения для каждого класса $K,\ K\in\{K_1,\ldots,K_l\}$ распознающий алгоритм $\mathcal{A}$ конструирует множество из корректных наборов элементарных классификаторов $W_{\mathcal{A}}\left(K\right)$

На этапе распознавания для объекта S вычисляется оценка принадлежности этого объекта к каждому из классов K по следующей формуле:

$$\Gamma_K\left(S\right)=\frac{1}{\left|W_{\mathcal{A}}\left(K\right)\right|}\sum_{U\in W_{\mathcal{A}}(K)}F_{U,K}\left(U\left(S\right)\right),$$

$$\text{где } F_{U,K}\left(S\right)=\frac{1}{\left|T\cap K\right|}\sum_{S'\in T\cap K}\delta_U\left(S,S'\right)$$

А функция голосования классифицирующего набора U по обучающему объекту S' для S имеет вид:

$$\delta_U\left(S,S'\right)=\prod_{B_{(H,\sigma)}\in U}\left[B_{(H,\sigma)}\left(S\right)\geq B_{(H,\sigma)}\left(S'\right)\right]$$

Далее алгоритм относит неизвестный объект S к классу с наибольшей оценкой принадлежности. Если таких классов несколько, то алгоритм отказывается от распознавания.

2.3 Стохастическая модель логического корректора CLC

В детерминированном алгоритме $\mathcal{A}$, поиск семейства корректных наборов эл.кл. с хорошей распознающей способностью осуществляется среди множества всевозможных эл.кл. $C_K^{\mathcal{A}}$. Но мощность данного множества очень велика, т.к. каждый объект класса K формирует 2^n эл.кл. (возможно с повторениями). Поэтому $\left|C_K^{\mathcal{A}}\right| \sim 2^n \cdot |K|$ и для задач с большим числом признаков, множество эл.кл. для класса K становится слишком большим для хранения в памяти компьютера. Вместо $C_K^{\mathcal{A}}$ рассматривается подмножество $\widetilde{C_K^{\mathcal{A}}} \subseteq C_K^{\mathcal{A}}$, которое представляет собой случайную выборку из $C_K^{\mathcal{A}}$.

Алгоритм формирования $\widetilde{C_K^{\mathcal{A}}}$:

1. Выбирается случайный объект S из класса K и генерируется случайное число i от 1 до n.
2. В качестве H выступают i случайных признака, $H = \{x_{j_1}, \ldots, x_{j_i}\}$.
3. В качестве σ выступают значения признаков объекта S в столбцах H, $\sigma = \{a_1, \ldots, a_i\}$
4. Если эл.кл. $B_{(H,\sigma)}$ не содержится в $\widetilde{C_K^{\mathcal{A}}}$, то $\widetilde{C_K^{\mathcal{A}}} = \widetilde{C_K^{\mathcal{A}}} \cup B_{(H,\sigma))}$, иначе переход к шагу 5.
5. Если $\left|\widetilde{C_K^{\mathcal{A}}}\right| < N_K$, (где N_K — мощность множества эл.кл. для класса K) то переход к шагу 1, иначе выход.

В остальном принцип работы стохастического логического корректора не отличается от детерминированного алгоритма.

2.4 Поиск покрытий матрицы сравнения L_K

После формирования множества элементарных классификаторов $\widetilde{C_K^{\mathcal{A}}} =$
$= \left\{B_{(H_1,\sigma_1)}, \ldots, B_{(H_q,\sigma_q)}\right\}$ осуществляется поиск классифицирующих наборов $U \subseteq C_K$.

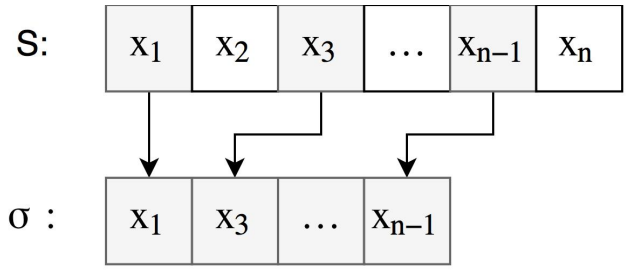

Рис. 1: Формирование эл.кл. (H, σ)

Эта задача сводится к нахождению покрытий булевой матрицы L_K, которая формируется по обучающей выборке следующим образом.

Паре объектов S' и S'' ставится в соответствие строка $D(S', S'') = (d_1, \ldots, d_{N_K})$, в которой:

$$d_i = \begin{cases} 1, & \text{если } B_{(H_i,\sigma_i)}(S') = 1 \text{ и } B_{(H_i,\sigma_i)}(S'') = 0, \\ 0, & \text{иначе} \end{cases}$$

Для каждого класса K формируется булева матрица сравнения L_K из всех строк вида $D(S', S'')$ таких что S', S'' — обучающие объекты и $S' \in K,\ S'' \in \overline{K}$. Поиск семейства тупиковых корректных наборов эл.кл. для класса K эквивалентен задаче нахождения неприводимых покрытий матрицы L_K.

Задача о покрытии относится к классу NP - полных. Известные алгоритмы поиска точного решения имеют экспоненциальную вычислительную сложность и малопригодны на практике. Поэтому для прикладных задач больших размерностей ищутся приближенные решения. Одним из таких решений является генетический алгоритм.

3 Генетический алгоритм

3.1 Общие сведения

Работа генетических алгоритмов напоминает развитие биологических организмов. На каждом шаге генетического алгоритма обновляется множество приближенных решений рассматриваемой задачи, называемое *популяцией*, элементы популяции называются *особями*. Каждая особь описывается числовым вектором, элементы которого называются *хромосомами*. Качество найденного приближенного решения харак-

теризуется *функцией приспособленности*. Существует несколько стратегий развития популяции:

1. *Классическая стратегия* — заключается в полном обновлении популяции на каждом шаге генетического алгоритма. При этом самые приспособленные особи не всегда переходят в следующее поколение.

2. *Элитарная стратегия* — заключается в сохранении наилучших особей данной популяции на последующих итерациях.

3. *Частичная замена популяции* — только часть популяции переходит в следующее поколение без каких-либо изменений.

В дипломной работе рассмотрен третий вариант развития популяции, в котором на каждом шаге генетического алгоритма будет происходить замена одной особи популяции на более приспособленную новую. Таким образом, обновление популяции происходит так, что наименее приспособленные особи постепенно ее покидают.

Для предотвращения преждевременной сходимости генетического алгоритма (попадание в локальный минимум) из-за сходства большей части особей, необходимо, чтобы все особи популяции были различны. Новые особи получаются путем применения операторов скрещивания и мутации, которые являются аналогами соответствующих процессов в биологии. Будем рассматривать генетический алгоритм, действующий по следующей схеме:

1. Формирование начальной популяции заданного объема N. Вычисление функции приспособленности особей популяции.

2. Выбор родительских особей.

3. Получение потомка с помощью оператора скрещивания.

4. Мутация генов потомка.

5. Вычисление функции приспособленности потомка.

6. Замещение менее приспособленной особи популяции.

7. Проверка условия окончания работы генетического алгоритма. Если условие не выполняется, то переход к шагу 2.

3.2 Представление особей в генетическом алгоритме

Для выбора способа представления особей в популяции следует определить множество объектов, среди которых будет осуществляться поиск оптимального решения задачи. Такое множество называется *множеством допустимых решений*. Для решения задачи о покрытии в качестве множества допустимых решений в данном случае используется множество неприводимых покрытий матрицы L.

Далее задается функция кодирования $g(H)$ преобразующая набор столбцов H матрицы L в набор хромосом. Возможные способы представления особей.

- **Бинарное представление**.

  Функция кодирования $g(H)$ преобразует набор столбцов H в бинарный вектор $g = (g1, \ldots, g_n)$, где i-я компонента вектора равна единице, если столбец с номером i принадлежит H и нулю в противном случае.

- **Целочисленное представление**. Функция кодирования $g(H)$ преобразует набор столбцов H в целочисленный вектор $g = (g_1, \ldots, g_m)$, где i-я компонента равна номеру столбца, который покрывает строку с номером i.

Особь в популяции будет описываться как пара (H, g), где H — описываемое покрытие матрицы L, $g = g(H)$. Далее будет рассмотрено бинарное представление особей.

3.3 Формирование начальной популяции

Пусть N — заданный размер популяции P. Процедура формирования начальной популяции состоит из нескольких шагов.

1. Формируется набор хромосом $g = (g_1, \ldots, g_n)$, каждая компонента которого случайно выбирается из множества $\{0, 1\}$. Если полученный набор столбцов не является покрытием матрицы L, то набор дополняется столбцами до покрытия.

2. По вектору значений g строится набор столбцов H матрицы L.

3. Для каждого столбца j из H в порядке убывания весов проверяется, является ли набор столбцов $H \setminus \{j\}$ покрытием. Если является, то столбец j удаляется из H.

4. Если в P нет особи (H, g), то она добавляется в P.

5. Если число сгенерированных особей меньше N, то происходит переход к первому шагу. В противном случае формирование начальной популяции завершается.

3.4 Функция приспособленности

Функцию приспособленности будем вычислять по следующей формуле:

$$f_i = w_i - \min_{j \in \{1,\dots,N\}} w_j + 1, \tag{1}$$

где w_i — вес i-ой особи в популяции P, $i \in \{1, \dots, N\}$. В качестве веса отдельной особи берется длинна покрытия, т.е. $w_i = \sum_{i=1}^{m} g_i$.

3.5 Выбор родительских особей

Существует несколько подходов к выбору родительской пары. Наиболее распространенными операторами выбора родителей являются следующие:

1. *Панмиксия* — все особи равновероятно могут войти в «брачную пару».

2. *Инбридинг* — первый родитель выбирается равновероятно и случайным образом. Вторым становится особь популяции, которая наиболее схожа с первым по заданной метрике.

3. *Аутбридинг* — также использует понятие схожести особей. Однако теперь второй родитель выбирается максимально непохожим на первый.

4. *Селективное скрещивание* — при образовании «брачной пары» родителям могут стать только те особи, значения функции приспособленности которых не меньше некоторой пороговой величины.

Существует большое число более сложных правил выбора, однако будет рассмотрен первый способ, в котором все особи могут равновероятно попасть в пару.

3.6 Оператор скрещивания

Для формирования новых особей в генетических алгоритмах применяются операторы скрещивания, которые на базе хромосом родителей формируют особь потомок. Рассмотрим возможные варианты:

1. *Одноточечный кроссовер* — формируется «брачная пара», случайным образом выбирается точка в наборе хромосом (точка разрыва), в которой оба родительских набора делятся на две части и обмениваются ими. Таким образом получается две новые особи-потомки.

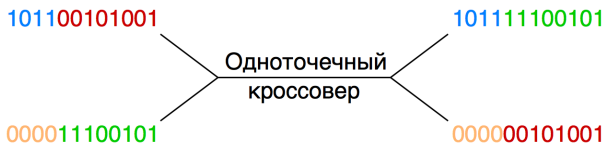

Рис. 2: Одноточечный кроссовер

2. *Многоточечный кроссовер* — выбирается несколько точек разрыва.

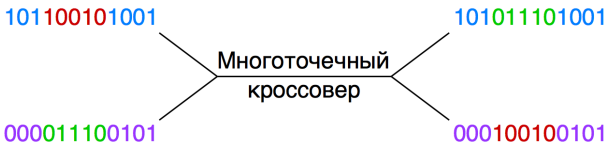

Рис. 3: Многоточечный кроссовер

3. *Однородный кроссовер* — единственный потомок воспроизводится таким образом, что значение каждой хромосомы копируется в набор потомка равновероятно из набора одного из родителей.

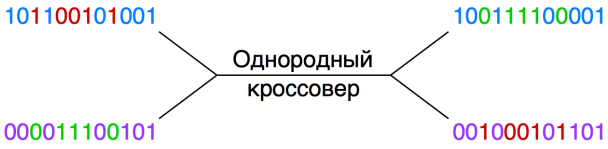

Рис. 4: Однородный кроссовер

3.7 Оператор мутации

При использовании в популяции только оператора скрещивания, как правило, наблюдается сходимость решений к некоторому локальному минимуму. Такая ситу-

ация возникает в случае, когда в популяции находятся достаточно большое количество особей с примерно одинаковыми описаниями. Оператор скрещивания для таких особей в качестве потомка будет выдавать схожий с ним объект, то есть объект, находящийся рядом с некоторым локальным минимумом. Для обеспечения возможности перехода к тем решениям, к которым невозможно перейти из начальной популяции только с помощью скрещивания, применяется оператор мутации.

Действие оператора мутации заключается в изменении некоторого набора хромосом особи. Размер этого набора может быть константой, выбираться случайно или в зависимости от каких-то параметров. Так же можно увеличивать число мутируемых хромосом с ростом числа шагов алгоритма.

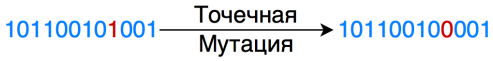

Рис. 5: Точечная мутация

3.8 Восстановление допустимости решения

Применение операторов скрещивания и мутации над особями популяции может нарушить допустимость решения, т.е. полученный набор столбцов не будет являться неприводимым покрытием или покрытием вообще, как в случае бинарного представления. Для восстановления допустимости решения предлагается следующий алгоритм.

Пусть (H, g) — особь популяции, не удовлетворяющая условию допустимости, M_H — номера строк матрицы L, которые не покрыты ни одним столбцом из H. M_j — номера строк матрицы L, которые покрываются j-ым столбцом. Процедура восстановления допустимости решения состоит из двух этапов:

1. Множество столбцов H дополняется до покрытия матрицы L последовательным добавлением столбцов, покрывающих строки из M_H и максимизирующих функционал $|M_H \cap M_j|$, т.е. выбирается столбец, который покрывает как можно больше строк.

2. Из покрытия H формируется неприводимое покрытие матрицы L: в порядке убывания весов для всех столбцов j из H, если $H \setminus \{j\}$ является покрытием, то $H = H \setminus \{j\}$.

4 Реализация генетического алгоритма в модели логического корректора CLC с использованием технологии CUDA

При больших объемах обучающей выборки и высокой значности признаков даже генетический алгоритм работает достаточно медленно. Существенного сокращения временных затрат в работе генетического алгоритма удалось достичь за счет применения параллельных вычислений на основе программно-аппаратной архитектуры унифицированных вычислений CUDA. В данном разделе описана схема работы параллельной версии генетического алгоритма и приведены вычислительные эксперименты, показавшие эффективность предложенного подхода.

4.1 Схема работы параллельного генетического алгоритма

Результаты запусков последовательной версии генетического алгоритма показали, что примерно 90% времени уходит на проверку: является ли набор столбцов покрытием матрицы сравнения L_K. На каждой итерации формируется новая особь популяции, а она состоит из неприводимых покрытий матрицы L_K. После операций скрещивания и мутации новый набор столбцов не обязательно будет неприводимым покрытием матрицы сравнения. Поэтому необходимо проверять, является ли особь неприводимым покрытием после скрещивания и мутации, дополнения до покрытия и сокращения до неприводимого покрытия.

Матрица сравнения L_K содержит $\left|K\right| \times \left|\overline{K}\right|$ строк. Если задача содержит большое число объектов в каждом классе, то проверка набора столбцов на покрытие займет основную часть времени счета программы. Поэтому в алгоритме указанная проверка набора столбцов реализована на графическом процессоре компьютера. Для этого использовалась следующая процедура.

Каждое ядро графического процессора осуществляет проверку отдельной строки матрицы L_K. Поскольку все строки проверяются независимо, то ядра видеокарты могут работать параллельно. Таким образом, за счет большого числа вычислительных ядер удается достичь многократного выигрыша в производительности.

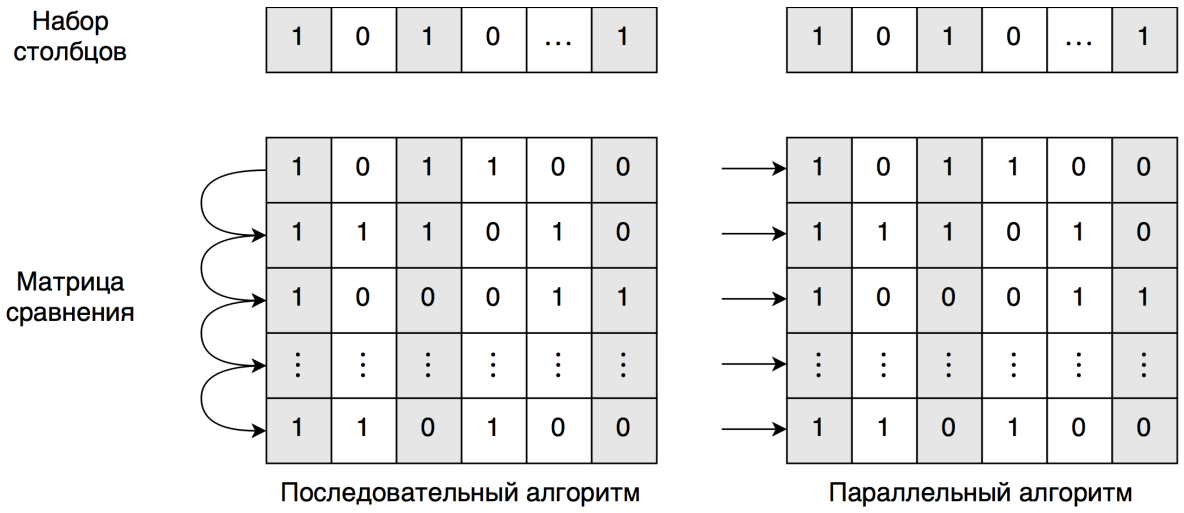

Рис. 6: Последовательный и параллельный алгоритмы

4.2 Экспериментальные результаты

Для сравнения последовательной и параллельной версии процедуры проверки набора столбцов на покрываемость, был поставлен эксперимент. Каждый алгоритм запускался на случайных булевых матрицах различной размерности. Рассматривались матрицы L_K размером от 100×100 до 10000×10000 и было произведено сравнение времени работы параллельного и последовательного алгоритмов. Для это вычислялась величина $K = t_{cpu}/t_{gpu}$, где t_{cpu} — время работы последовательного алгоритма, а t_{gpu} — время работы параллельного алгоритма.

На графике, изображенном на рисунке 7, показана зависимость коэффициента ускорения K от числа столбцов в матрице L_K. Приведены три кривые для матриц с числом строк равным $100, 1200$ и 3400. Видно, что с увеличением числа столбцов в матрице L_K, стремительно растет эффективность параллельного алгоритма. На матрицах размером 3400×10000 проверка набора столбцов на покрываемость осуществляется в 100 раз быстрей чем у последовательного алгоритма. Однако, на

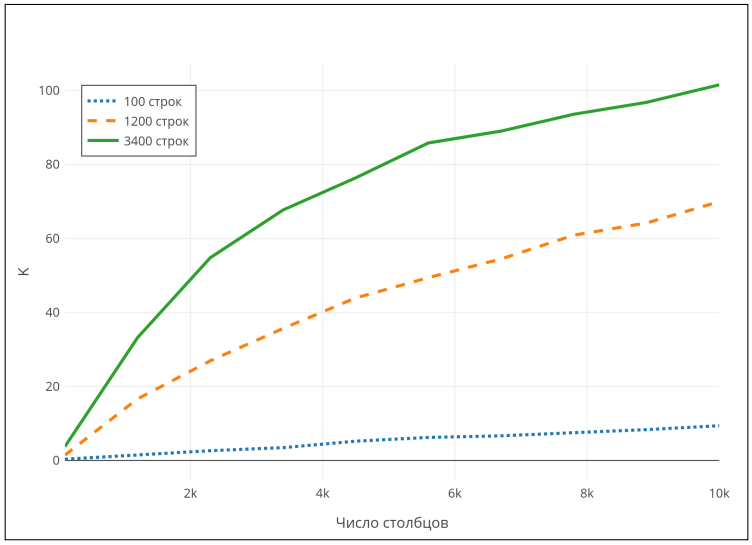

Рис. 7: Сравнение последовательной и параллельной версии алгоритмов

матрицах меньшего размера параллельный алгоритм проигрывает в скорости работы, что связано с дополнительными временными затратами, которые возникают при обмене данными между центральным и графическим процессорами.

При формировании нового потомка, необходимо многократно передавать бинарный вектор в память видеокарты, что существенно замедляет общее время работы. Улучшения производительности удается достичь только тогда, когда сложность решаемой задачи во много раз превосходит сложность передачи данных. Поэтому параллельный алгоритм не достаточно быстро работает на задачах с небольшими матрицами L_K.

5 Тестирование логического корректора CLC

Тестирование проводилось по методу скользящего контроля на 23 прикладных задачах, собранных в репозитории UCI. Характеристики всех задач. представлены в таблице 1. Для каждой задачи указаны число объектов m, число признаков n, число классов k, распределение объектов по классам и доля уникальных значений признаков. Для оценки качества распознавания использован следующий функционал:

$$R = \frac{1}{l}\sum_{i=1}^{l}\frac{TP_i}{|K_i|},$$

где TP_i — число правильно распознанных объектов из класса K_i. Функционал имеет смысл средней по всем классам точности распознавания.

Название	m	n	k	Распределение объектов по классам	Доля уникальных значений
botwinklbl	196	9	2	23/173	2.95%
botwinSt	196	17	2	23/173	2.49%
crest	100	2	2	50/50	30.00%
dorovskih	33	12	2	16/17	25.76%
ech_l	60	8	2	40/20	10.42%
ech_r	71	8	2	48/23	7.39%
echu	131	9	2	89/42	5.35%
eco_l	144	7	4	76/33/24/11	2.28%
Hea_r	136	13	5	78/24/14/15/5	8.37%
Hep_l	72	19	2	15/57	5.26%
Hep_r	83	19	2	17/66	4.06%
manelis1	145	35	2	38/107	1.91%
manelis2	107	35	2	35/72	2.38%
manelis3	73	35	2	38/35	5.83%
manelis4	110	35	2	38/72	2.29%
matchak2	132	24	2	30/102	5.05%
melanoma	80	33	3	29/30/21	35.87%
patomorfoz	77	7	2	47/30	7.42%
Pnevmo_r	57	41	4	17/18/12/10	11.60%
SARComa	80	18	2	40/40	18.19%
stupenexper	61	18	2	39/22	16.67%
surv	77	8	2	52/25	16.72%
TLMEL	48	33	3	17/20/11	37.88%
TUML	114	5	3	60/15/39	2.63%

Таблица 1: Характеристики используемых задач

В таблице 2 приведены результаты эксперимента описанного выше. В каждой строке указано число $R \cdot 100\%$. Последний столбец таблицы содержит результат работы логического корректора CLC. Реализованной алгоритм логического корректора показал хорошие результаты на прикладных задачах, во многих случая опережая остальные алгоритмы по качеству классификации. Алгоритм близок по качеству

Task	MON	MONS	TstAlg	Repr Sets	DecTree	Random Forest	CLC
botwinklbl	69,00%	73,93%	62,20%	74,50%	50,00%	61,89%	74,27%
botwinSt	63,56%	61,90%	62,05%	64,64%	50,00%	54,21%	61,13%
dorovskih	87,87%	90,99%	87,85%	97,06%	0,00%	73,16%	85,06%
ech_l	72,50%	86,25%	60,00%	85,00%	50,00%	68,75%	74,85%
ech_r	81,84%	82,97%	71,95%	81,75%	45,85%	62,18%	68,80%
echu	84,37%	87,73%	77,40%	89,49%	78,55%	81,15%	89,45%
eco_l	93,94%	94,51%	84,73%	94,27%	96,35%	93,47%	94,14%
Hea_r	36,19%	41,81%	57,00%	49,14%	24,44%	29,16%	34,64%
Hep_l	67,93%	68,77%	71,60%	70,70%	46,50%	62,28%	64,56%
Hep_r	79,32%	79,99%	74,75%	84,54%	44,70%	72,68%	80,27%
manelis1	80,98%	77,12%	67,00%	87,90%	72,35%	75,68%	84,43%
manelis2	71,17%	74,52%	72,25%	75,91%	50,00%	70,89%	74,12%
manelis3	80,93%	79,47%	70,50%	92,97%	72,35%	87,48%	87,83%
manelis4	81,79%	78,66%	80,90%	87,79%	79,55%	77,34%	82,75%
matchak2	65,71%	69,22%	66,25%	72,55%	50,00%	63,43%	66,10%
melanoma	89,45%	90,48%	70,77%	90,52%	57,20%	59,78%	90,26%
patomorfoz	91,21%	91,96%	88,95%	88,48%	94,50%	92,87%	90,68%
Pnevmo_r	76,26%	78,70%	69,83%	85,00%	17,40%	61,18%	78,87%
SARComa	95,00%	95,00%	80,00%	96,25%	0,00%	85,00%	90,34%
sigapur	93,33%	94,39%	93,30%	49,71%	92,25%	94,39%	92,53%
stupenexper	89,62%	93,18%	80,50%	90,91%	80,50%	81,53%	91,04%
surv	75,12%	78,15%	63,75%	80,15%	50,00%	57,27%	64,71%
TLMEL	88,05%	84,71%	64,43%	95,30%	30,87%	48,98%	61,38%
TUML	63,85%	76,84%	77,70%	78,50%	61,67%	74,36%	73,54%

Таблица 2: Результаты тестрования

прогнозирования к логическим корректорам MON и MONS. При этом, благодаря использованию вычислений на графическом процессоре, удалось ускорить этап обучения.

6 Заключение

В выпускной квалификационной работе получены следующие результаты:

1. Построена новая модель логического корректора CLC , основанная на стохастическом формировании множества корректных минимальных наборов эл.кл.
2. Модель логического корректора CLC реализована с помощью технологии CUDA. Распараллелена работа генетического алгоритма.
3. Проведены эксперименты, показавшие вычислительную эффективность логического корректора CLC.
4. Показана эффективность логического корректора CLC при решении прикладных задач классификации.

Список литературы

[1] *Дюкова Е.В., Журавлев Ю.И., Рудаков К.В.* Об алгебро-логическом синтезе корректных процедур распознавания на базе элементарных алгоритмов // Ж. вычисл. матем. и матем. физ., 2000. Т.40 №8. — С. 1264 - 1278

[2] *Журавлев Ю.И.* Об алгебраическом подходе к решению задач распознавания или классификации // Проблемы кибернетики. №33, М.: Наука, 1978. — С.5-66

[3] *Dyukova E.V., Zhuravlev Yu.I., Sotnezov M.R.* Construction of an Ensemble of Logical Correctors on the Basis of Elementary Classifiers // Pattern Recognition and Image Analysis, 2011, Vol. 21, №4, pp. 599 - 605

[4] *Dyukova E.V., Prokofjev P.A.* Models of Recognition Procedure with Logical Correctors // Pattern Recognition and Image Analysis, 2013, Vol. 23, №2, pp. 235 - 244

[5] *Дюкова Е.В., Любимцева М.М., Прокофьев П.А.* Об алгебро-логической коррекции в задачах распознавания по прецедентам // Машинное обучение и анализ данных, 2013. Т.41 №6 — С.705-713

[6] *Djukova E.V, Lyubimtseva M.M., Prokofjev P.A.* Logical correctors in recognition problems // 11th International Conference «Pattern Recognition and Image Analysis: New Information Technologies (PRIA-11-2013)», — Samara, September 23-28, 2013. Conference Proceedings, Vol. I. Samara: IPSI RAS.P.82-83.

[7] *Дюкова Е.В., Любимцева М.М., Прокофьев П.А.* Логические корректоры в задачах классификации по прецедентам // Тезисы докладов Всероссийской конференции «Математические методы распознавания образов (ММРО-16)», Казань, 6-12 октября 2013. — М.:Торус Пресс. С.7.

[8] *Дюкова Е.В.* Асимптотические оптимальные тестовые алгоритмы в задачах распознавания // Проблемы кибернетики. Вып. 39. М.: Наука, 1982.—С.165-199.

[9] *Воронцов К.В.* Лекции по логическим алгоритмам классификации — 2010. http://www.machinelearning.ru