УДК 519.688;004.93'12

ПРИМЕНЕНИЕ ТЕХНОЛОГИИ CUDA ДЛЯ УСКОРЕНИЯ ВЫЧИСЛЕНИЙ В НЕЙРОННЫХ СЕТЯХ

В.В. Парубец, О.Г. Берестнева, Д.В. Девятых

Томский политехнический университет
E-mail: VParubets.mail@gmail.com

Рассматривается применение искусственных нейронных сетей в задачах медицинской диагностики. Описывается возможность ускорения вычислений в сетях с технологией NVIDIA CUDA для реализации параллельных вычислений в обучении и распознавания образов сетью. Показано увеличение производительности сети.

Ключевые слова:
Искусственная нейронная сеть, NVIDIA CUDA.

Key words:
Artificial Neural Network, NVIDIA CUDA.

Введение

В современной медицине всё больше возрастает роль компьютерной диагностики. Особенно ярко это выявляется в случаях, когда необходимо произвести классификацию большого количества разнородных данных, чтобы выявить те или иные общие признаки для, например, разделения больных на группы. Искусственные нейронные сети изначально хорошо себя зарекомендовали в таких областях применения.

Существенную роль при анализе медико-биологической информации играет её особенности: описательный характер, использование формализмов, подверженность крайней вариабельности. Данные, даже описанные с помощью чисел, в большинстве случаев не могут быть хорошо упорядочены и классифицируемы, так как изменяются в зависимости от клинических традиций различных школ, геосоциальных особенностей регионов и даже отдельных учреждений, а также от времени.

Все задачи, решаемые человеком, с позиций нейроинформационных технологий можно условно классифицировать на две группы [1]:

1. Задачи, имеющие известный и определенный набор условий, на основании которого необходимо получить четкий, точный, недвусмысленный ответ по известному и определенному алгоритму.
2. Задачи, в которых не представляется возможным учесть все реально имеющиеся условия, от которых зависит ответ, а можно лишь выделить приблизительный набор наиболее важных условий. Так как часть условий при этом не учитывается, ответ носит неточный, приблизительный характер, а алгоритм нахождения ответа не может быть выписан точно.

Для решения задач первой группы с большим успехом можно использовать традиционные компьютерные программы. Как бы ни был сложен алгоритм, ограниченность набора условий (входных параметров) дает возможность составления алгоритма решения и написания конкретной программы, решающей данную задачу.

При решении задач второй группы применение нейротехнологии оправдывает себя по всем параметрам, при выполнении, однако, двух условий: *во-первых*, наличия универсального типа архитектуры и единого универсального алгоритма обучения (отсутствие необходимости в их разработке для каждого типа задач), *во-вторых*, наличия примеров (предыстории, фиксированного опыта), на основании которых производится обучение нейронных сетей.

Практически вся медицинская и биологическая наука состоит именно из задач, относящихся ко второй группе, и в большинстве этих задач достаточно легко набрать необходимое количество примеров для выполнения второго условия [2]. Это задачи диагностики, дифференциальной диагностика, прогнозирования, выбора стратегии и тактики лечения и др. Медицинские задачи практически всегда имеют несколько способов решения и «нечеткий» характер ответа, совпадающий со способом выдачи результата нейронными сетями.

Нейронные сети часто используются для классификации в области компьютерной диагностики. Создание оптимальной нейронной сети включает:

1) выбор активационной функции нейронов скрытого слоя;
2) выбор топологии сети;
3) выбор метода обучения;
4) обучение сети.

На сегодняшний день представлен широкий спектр нейронных сетей: многослойный перцептрон, самоорганизующиеся карты Кохонена, рекуррентные нейронные сети, свёрточные нейронные сети. Все они имеют свои отличительные особенности в топологии, функциях активации и методах обучения.

Известным ограничением развития нейросетевых алгоритмов следует признать высокие вычислительные затраты на реализацию таких методов [3]. К традиционным способам решения данной проблемы относят организацию параллельных и распределенных вычислений на специализированном аппаратном обеспечении.

Описание технологии CUDA

Особенностью оборудования, поддерживающего технологию *CUDA* (Compute Unified Device Architecture – унифицированная архитектура вычислительного устройства), является возможность обеспечивать на порядок большую (по сравнению с кластерами) пропускную способность при работе с памятью [4].

В графических ускорителях *NVIDIA*, начиная с восьмой серии, реализована архитектура параллельных вычислений *CUDA*, которая предоставляет специализированный программный интерфейс для не графических вычислений.

Логически графический процессор с поддержкой *CUDA* можно рассматривать как набор многоядерных процессоров. Основными вычислительными блоками таких видеочипов являются мультипроцессоры, которые состоят из восьми ядер, нескольких тысяч 32-битных регистров, 16 Кбайт общей памяти, текстурного и константного кэшей.

До официального появления технологии *CUDA* проводились эксперименты по использованию графических карт настольных систем для реализации потоковых вычислений. Так, с помощью графических программно-аппаратных интерфейсов и представления данных в качестве массивов текстур, удалось добиться троекратного увеличения производительности в экспериментах. Стоит отметить также и одно из ключевых достоинств технологии *CUDA* – отсутствие необходимости в разработке программ следовать графическим «метафорам» – типам данных и принципам построения вычислений, характерным исключительно для обработки вершин и пикселей при построении кадра.

Реализация нейронной сети с сочетанием *CUDA* и *OpenMP* и параллелизма центральных процессоров на уровне языка программирования и программно-аппаратного интерфейса создает еще один уровень прироста производительности.

Литература, касающееся не только *CUDA*, но и предыдущие аппаратные реализации генетических алгоритмов и нейронных сетей демонстрируют ускорение между 2× и 15× [4–6].

Основным источником информации по применению данной технологии может послужить портал разработчика компании *NVIDIA*, где можно найти как наиболее свежую версию программных библиотек, так и основную информацию о непосредственном внедрении графических процессоров для общих расчетов в своём проекте.

Несмотря на большую информативность портала компании *NVIDIA*, непосредственные оценки производительности и некоторые тонкости применения графических процессоров для задач численного интегрирования, опирались на работы [4–6].

Таким образом, применение описанных выше алгоритмов в связке друг с другом имеет право на существование и возможность развития.

Математическая постановка задачи

Поставим задачу диагностики пациента как задачу распознавания образов [7].

Пусть X – множество описаний объектов, Y – множество номеров (или наименований) классов. Существует неизвестная целевая зависимость значения которой известны только на объектах конечной обучающей выборки. Требуется построить алгоритм способный классифицировать $x \in X$.

Математически исходная информация о лабораторных показателях представляется в виде матрицы:

$$X_{N\times P} = \left\| x_{ij} \right\|,$$

где i=1,2,..., n – номер элемента; j=1,2,..., p – номер показателя; x_{ij} – значение j-го показателя для i-го элемента; n – размер выборки; p – число показателей.

Также есть вектор значений индивидуальных интегральных показателей здоровья для каждого ребенка:

$$Y_{N\times 1} = \{y_i\}.$$

Запишем функцию вида:

$$F(X \cdot W) = Y\_out,$$

где $W_{p\text{x}1}$ – вектор весовых коэффициентов синапсов; $Y\_out_{n\text{x}1}$ – вектор выходных значений.

Пусть имеется нейронная сеть, выполняющая преобразование F: $X \to Y$ матрицы X из признакового пространства входов X в матрицу – столбец Y выходного пространства Y. Нейронная сеть находится в состоянии W из пространства состояний W. Пусть далее имеется обучающая выборка (X^α, Y^α), α=1...p. Рассмотрим полную ошибку E, получаемую сетью в состоянии W.

$$E = E(W) = \sum_\alpha \left\| F(X^\alpha; W) - Y^\alpha \right\| =$$

$$= \sum_\alpha \sum_i [F_i(X^\alpha; W) - Y^\alpha]^2.$$

Необходимо найти такую совокупность адаптивных весов нейронной сети W*, которая доставляет минимум некоторому квадратичному функционалу: $\min E(W) \to 0$.

Задача кластеризации тогда будет представлена следующим образом. Пусть m – размерность входного пространства, а входной вектор выбирается из этого пространства случайно и обозначается так:

$$x = [x_1, x_2, \ldots, x_m]^T.$$

Вектор синаптических весов каждого из нейронов имеет ту же размерность, что и входное пространство. Обозначим синаптический вес нейрона следующим способом:

$$w_j = [w_{j1}, w_{j2}, \ldots, w_{jm}]^T,\ j = 1, 2, \ldots, l,$$

где l – общее количество нейронов сети. Для того чтобы подобрать наилучший вектор w_j, соответ-

ствующий входному вектору x, нужно сравнить скалярные произведения $w_j^T x$ для $j = 1,2,..., l$ и выбрать наибольшее значение, таким образом, мы определяем местоположение, которое должно стать центром топологической окрестности возбужденного нейрона.

Критерий соответствия основанный на максимизации скалярного произведения математически эквивалентен минимизации Евклидова расстояния между векторами x и w. Если использовать индекс $i(x)$ для идентификации нейрона, то сущность кластеризации проявится в данном выражении:

$$i(x) = \arg\min_j \left\| x - w_j \right\|, \; j = 1, 2, ..., l.$$

Необходимо реализовать процесс обучения искусственной нейронной сети, распараллелив для применения на графических процессорах.

Выбор структуры нейронной сети обуславливается спецификой и сложностью решаемой задачи [7]. Для решения некоторых типов задач разработаны оптимальные конфигурации. В большинстве случаев выбор структуры нейронной сети определяется на основе объединения опыта и интуиции разработчика. Однако существуют основополагающие принципы, которыми следует руководствоваться при разработке новой конфигурации:

- возможности сети возрастают с увеличением числа ячеек сети, плотности связей между ними и числом выделенных слоев;
- введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о динамической устойчивости сети;

Вопрос о необходимых и достаточных свойствах сети для решения того или иного рода задач представляет собой целое направление нейрокомпьютерной науки. Так как проблема синтеза нейронной сети сильно зависит от решаемой задачи, дать общие подробные рекомендации затруднительно. Очевидно, что процесс функционирования нейронной сети, то есть сущность действий, которые она способна выполнять, зависит от величин синаптических связей, поэтому, задавшись определенной структурой нейронной сети, отвечающей какой-либо задаче, разработчик сети должен найти оптимальные значения всех переменных весовых коэффициентов.

Из теории нейронных сетей известно [7], что количество нейронов во входном слое предпочтительно брать равным количеству входных сигналов. В выходном же слое также предпочтительно брать количество нейронов соответственно равное количеству выходных сигналов. Для нейронной сети, имеющей один скрытый слой, количество синаптических весов этого слоя можно приближенно оценить по формуле:

$$\frac{n_y n_p}{1 + \log_2 n_p} \le n_w \le n_y \left(\frac{n_p}{n_y} + 1 \right) (n_x + n_y + 1) + n_y,$$

где n_w – необходимое число синаптических весов создаваемой искусственной нейронной сетью; n_x и n_y – размерность входного и выходного сигналов; n_p – число элементов обучающей выборки.

В свою очередь, получив n_w, можно найти количество нейронов во внутреннем (скрытом) слое:

$$n = \frac{n_w}{n_x + n_y},$$

где n_w – приблизительное количество нейронов в скрытом внутреннем слое.

В исследованиях [8] показана зависимость точности выполнения операции распознавания образов в медицинской диагностике от числа нейронов в скрытом слое. Выявлено, что сравнительно малое количество нейронов позволяет достичь большей точности.

В нашем случае число пациентов в каждой выборке равно 87, количество входных сигналов и, соответственно количество нейронов во входном слое, равно 16, а количество нейронов в выходном слое – 2. Получим, что приблизительное количество нейронов в скрытом слое лежит в пределах от 1 до 91.

В качестве активационной функции для нейронов скрытого слоя была выбрана сигмоидальная функция [7]:

$$F(x) = \frac{1}{1 + e^{-\beta x}},$$

а для нейронов выходного слоя – линейная. Здесь β – коэффициент, контролирующий «пологость» активационной функции.

Прямое распространение сигнала

Прямое распространение сигнала в нейросети описывается следующими формулами:

$$S_j^1 = \sum_{i=1}^{m} W_{ij}^1 x_i + W_0^1;$$

$$u_j = F(S_j^1), \;\; j = 1, ..., n;$$

$$S_j^2 = \sum_{i=1}^{n} W_{ij}^2 u_i + W_0^2;$$

$$y_l = S_l^2 \, l = 1, ..., n.$$

Здесь x_i – внешний сигнал, поступающий на i-й нейрон входного слоя; u_j – сигнал, вырабатываемый j-м нейроном скрытого слоя; y_l – сигнал, вырабатываемый l-м выходным нейроном. Верхние индексы у весовых коэффициентов обозначают номер слоя. Отметим, что матрицы весов можно объединить с соответствующими матрицами смещений (W_0^1, W_0^2). В дальнейшем под матрицей весов будем понимать именно объединенную матрицу.

Алгоритм обучения на графическом процессоре

Для того, чтобы в полной мере использовать возможности параллельной системы, которой является видеокарта, алгоритм настройки нейросети должен быть соответствующим образом преобразован. Для уменьшения числа обменов с глобальной памятью видеокарты и увеличения отношения

объема вычислений к объему загружаемых из нее данных по возможности все массивы данных должны быть записанными в двумерные массивы и обрабатываться на двумерной решетке. Для этого, например, всю обучающую выборку, т. е. совокупность пар $(\vec{X}, \vec{D})$, поместим в две матрицы с размерами $P\times M$ и $P\times K$ соответственно. С этой же целью избавляемся от матриц, хранящих ошибки обучения в текущей эпохе, заменив их одним массивом для хранения весов $W^2_{N\times K}$, рассчитанных на предыдущей операции.

Естественно, что если число обучающих примеров очень велико, то эти матрицы придется разбивать на блоки. В этом случае пакетная обработка обучающей выборки позволяет снизить частоту обменов между центральным и графическим процессорами, которая приводит к существенному замедлению всего алгоритма. В результате этого при прямом распространении сигнала операция умножения матрицы весов на вектор, получаемый с входа нейронов, заменятся на операцию перемножения двух матриц:

$$Y_{P\times K} = F(X_{P\times M} W^1_{M\times N})W^2_{N\times K}.$$

Наконец, изменив порядок вычислений, выделим ядра потоковой обработки, выполняющиеся на видеокарте, по возможности преобразуя схему вычислений так, чтобы во время обработки данных оперировать блоками нужного размера и обеспечить загруженность процессорных элементов, способную скрыть задержку при доступе к глобально памяти видеокарты. Это также позволит ускорить обработку за счет такой особенности параллелизма видеокарты, как векторность вычислений на графической карте.

Таким образом, процесс обучения сети сводится к следующему алгоритму:

1) получение сигнала, поступающего на скрытый слой,

$$U_{P\times N} = F(X_{P\times M} W^1_{M\times N});$$

2) вычисление выходного сигнала сети

$$Y_{P\times K} = U_{P\times N} W^2_{N\times K};$$

3) итеративное изменение весовых коэффициентов – элементов матрицы $W^2_{N\times K}$

$$W^{2*}_{jk} = W^2_{jk} - \alpha \sum_{p=1}^{P} (Y_{pk} - D_{pk}) U_{pj};$$

4) получение вспомогательной матрицы $U'_{P\times N}$

$$U'_{pj} = \sum_{k=1}^{K} (Y_{pk} - D_{pk}) W^2_{jk};$$

5) изменение весовых коэффициентов $W^1_{M\times N}$ на новом шаге обучения

$$W^{1*}_{ij} = W^1_{ij} - \beta\alpha \sum_{p=1}^{P} [U_{pj}(1 - U_{pj}) X_{pi} U'_{pj}].$$

Основным приемом перевода последовательного алгоритма в параллельный для реализации в среде CUDA является замена циклов параллельно выполняющимися командами вида SIMD (Single Instruction, Multiple Data – множественный набор данных, обрабатываемых одной инструкцией). При этом ускоряется выполнение операций умножения векторов и матриц. Вначале нужно разбить задачу на вычислительные блоки и потоки [10], после чего, если необходимо, внести изменения в способ их взаимодействия и записать алгоритм в среде CUDA. Помимо этих преобразований при реализации алгоритмов, рассмотренных в настоящей работе были использованы специальные приемы работы с различными видами видеопамяти [10]:

- использование общей памяти видеокарты для хранения часто используемых значений;
- планирование порядка исполнения вычислительных потоков, чтобы избежать невыровненного и непоследовательного (некогерентного) доступа к глобальной памяти видеокарты;
- использование текстур для осуществления невыровненного доступа к глобальной памяти;
- изменение порядка обработки данных во избежание конфликтов доступа к отдельным участкам памяти.

Реализация алгоритма

В качестве входных данных была использована база лабораторных показателей: общего анализа крови, общего анализа урины, анализ каленинкреиновой системы 87 детей обоего пола в возрасте от 1 до 36 мес. с патологией выделительной системы. Выборка содержала 31 представителя контрольной группы, которые были здоровы, у остальных был диагностирован пиелонефрит различных степеней тяжести. Каждый пациент описывается 16 входными показателями.

Смоделированы нейронные сети со следующими параметрами:

- топология сети – многослойный перцептрон;
- количество нейронов во входном/скрытом/выходном слое – 16/64/2;
- активационная функция – сигмоидальная;
- метод обучения – обратное распространение ошибки;
- ошибка, при которой пример считается распознанным верно, 0,1.

В первом случае, операции над матрицами производились на центральном процессоре. Во втором случае, помимо ресурсов центрального процессора применялся CUDA-совместимый графический процессор.

Сравнение производительности нейронных сетей производилось на рабочей станции с процессором *Intel Pentium E5200*, графической картой *NVIDIA GeForce GTX550TI* и объёмом оперативной памяти 4 ГБ. Время выполнения обучения/тестирования алгоритмом, реализованным для центрального процессора – 712/26 с, с применением CUDA – 88/4 с.

Выводы

1. Рассмотрены приемы повышения производительности искусственных нейронных сетей в задачах медицинской диагностики при реализации параллельных алгоритмов с применением CUDA-устройств.
2. Показана возможность ускорения вычисления в среде CUDA при обучении/тестировании сети в 8/6,5 раз по сравнению с алгоритмом, реализованным на центральном процессоре.

СПИСОК ЛИТЕРАТУРЫ

1. Галушкин А.И. О методике решения задач в нейросетевом логическом базисе // Нейрокомпьютер. – 2006. – № 2. – С. 49–70.
2. Мызников А.В., Россиев Д.А., Лохман В.Ф. Нейросетевая экспертная система для оптимизации лечения облитерирующего тромбангиита и прогнозирования его непосредственных исходов // Ангиология и сосудистая хирургия. – 1995. – № 2. – С. 100–103.
3. Горбань А.Н., Россиев Д.А. Нейронные сети на персональном компьютере. – Новосибирск: Наука, 1996. – 276 с.
4. Harris M. Mapping Computational Concepts to GPUs // GPU Gems 2. – 2006. – № 2. – P. 493–508.
5. Hall J.D., Carr N.A., Hart J.C. Cache and Bandwidth Aware Matrix Multiplication on the GPU. Technical Report. – UIUCDCS-R-2003–2328. 2012. URL: http://graphics.cs.uiuc.edu/~jch/papers/UIUCDCS-R-2003–2328.pdf (дата обращения: 17.01.2012).
6. Horn D. Stream Reduction Operations for GPGPU Applications // GPU Gems 2. – Addison Wesley, 2006. – P. 573–589.
7. Хайкин С. Нейронные сети полный курс. – М.: ООО «И.Д. Вильямс», 2006. – 1104 с.
8. Rossiev D.A., Savchenko A.A., Borisov A.G., Kochenov D.A. The employment of neural-network classifier for diagnostics of different phases of immunodeficiency // Modelling, Measurement & Control. – 1994. – V. 42. – № 2. – P. 55–63.
9. Горбань А.Н., Дунин-Барковский В.Л., Кардин А.Н. Нейроинформатика / Отв. ред. Е.А. Новиков. – Новосибирск: Наука, 1998. – 296 с.
10. NVIDIA CUDA Programming Guide Version 2.3.1 // NVIDIA – World Leader in Visual Computing Technologies. 2011. URL: http://www.nvidia.com/object/cuda_develop.html (дата обращения: 17.01.2012).

Поступила 24.01.2012 г.

УДК 004.652.3

ПРИМЕНЕНИЕ МЕТОДОВ ВИЗУАЛИЗАЦИИ ПРИ ИССЛЕДОВАНИИ СТРУКТУРЫ ЭКСПЕРИМЕНТАЛЬНЫХ МНОГОМЕРНЫХ ДАННЫХ

В.А. Воловоденко, О.Г. Берестнева, Е.В. Немеров*, И.А. Осадчая

Томский политехнический университет
*Сибирский государственный медицинский университет, г. Томск
E-mail: ogb@tpu.ru

Рассмотрены методы структурного анализа многомерных данных, представлены различные подходы к визуализации результатов экспериментальных исследований. Приведены примеры решения прикладных задач с использованием NovoSparkVisualizer.

Ключевые слова:
Кластерный анализ, методы визуализации, когнитивная графика, спектральные представления.

Key words:
Cluster analysis, imaging, cognitive graphics, spectral representation.

В настоящее время накоплен обширный арсенал средств анализа многомерных данных. Наиболее полное изложение применяемых здесь подходов, сопровождающееся подробными ссылками на ключевые работы, содержится в [1]. В [2] приведена классификация основных методов анализа структуры многомерных данных:

1) визуализация данных: линейные методы снижения размерности, нелинейные отображения, многомерное шкалирование, заполняющие пространство кривые;
2) автоматическое группирование: факторный и кластерный анализ объектов и признаков, иерархическое группирование, определение «точек сгущения».

В основу приведенной классификации положен признак, отображающий степень участия экспериментатора в выделении особенностей взаимоотношений между исследуемыми объектами и признаками. Применение методов визуализации данных нацелено на поиск наиболее выразительных изображений совокупности исследуемых объектов для последующего максимального задействования потенциала зрительного анализатора экспериментатора.

Компьютерная обработка данных предполагает некоторое математическое преобразование данных с помощью определенных программных средств. Для этого необходимо иметь представление как о математических методах обработки данных, так и о соответствующих программных средствах [2].