УДК 681.3

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ РЕШЕНИЯ ЗАДАЧ ПРОГНОЗИРОВАНИЯ И КЛАССИФИКАЦИИ НА КЛАСТЕРЕ НА ОСНОВЕ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

С.В. Минухин, С.В. Знахур

Харьковский национальный экономический университет
61001, Харьков, проспект Ленина, 9-а. Тел.: (057)702 1831,
e-mail: ms_vl@mail.ru, sergznakhur@ukr.net

Рассмотрен подход к параллельному решению задач классификации и прогнозирования для аппарата искусственных нейронных сетей в условиях большой размерности исходных и результирующих данных. Получены временные характеристики решения задачи обучения нейронных сетей с различной архитектурой, количеством нейронов и алгоритмов обучения. Приведены условия их эффективности при распараллеливании на вычислительном кластере.

An approach to the parallel solution of problems of classification and prediction for the apparatus of artificial neural networks in a large dimension of input and output data. Obtain the temporal characteristics of solving the problem of training neural networks with different architectures, the number of neurons and learning algorithms. We give conditions for their effectiveness in parallelizing on a computational cluster.

Введение

В настоящих условиях развития глобальных вычислительных кластерных систем актуальной проблемой является формализация задач в таком виде, которая бы позволила использовать вычислительные кластеры и многопроцессорные системы для параллельного решения поставленных задач. Одним из вариантов параллельной реализации определенных классов математических задач является использование искусственных нейронных сетей (ИНС). Задачи аппроксимации, кластеризации, классификации, прогнозирования могут быть успешно реализованы с помощью моделей, основанных на принципе суперпозиций элементов (многослойных персептронов). Существуют определенные ограничения использования ИНС для параллельного программирования, которые связаны с особенностями построения сети и передачи данных с одного слоя (позиции) на другой. В данной статье рассмотрены подходы к оптимальному использованию возможностей вычислительного кластера, содержащего множество вычислительных узлов, для реализации как параллельного обучения нейронной сети, так и для решения сервисных задач для поиска оптимальной структуры и настроек ИНС.

Цель работы – провести обзор и анализ подходов к распараллеливанию решения задач классификации, аппроксимации и прогнозирования на основе использования аппарата ИНС для вычислительных кластеров с определенным количеством узлов. В результате опытного тестирования необходимо определить условия для уменьшения времени поиска оптимальной структуры ИНС, обучения ее весов при ограничении на допустимую точность (ошибку) решения (обучения). Данная проблема возникает в случае прогнозирования большого количества взаимосвязанных между собой данных, например, для прогнозирования курсов акций на фондовой бирже. Уменьшение времени обучения сети является важным в условиях часто меняющихся значений курса акций (обновление котировок акций осуществляется в пределах малых временных интервалов – нескольких минут).

Основная постановка задачи При создании параллельных программ предлагается, в первую очередь, распараллеливать не обучение нейронной сети, а некоторые сервисные процедуры, такие как методы оптимизации предобработки данных, схемы отбора примеров в обучающую выборку, выбор оптимальной архитектуры сети, отбор информативных независимых переменных и т.д. Обучение нейронной сети может быть распараллелено только в рамках парадигмы растущих сетей или при использовании отдельных методов обучения. В работе предлагаются направления создания параллельных программ искусственных нейронных сетей. В качестве основной нейроархитектуры взяты нейронные сети, обучаемые алгоритмами обратного распространения ошибки и решающие задачи обучения с учителем [1, 4].

Актуальность параллельной реализации нейронной сети основана на следующих факторах:

1. Пользователь обычно использует программу - нейроимитатор для обычных персональных компьютеров, чем специализированное аппаратное обеспечение для нейровычислений, которое часто требует программирования на более низком уровне, чем программирование для современных персональных компьютеров, и имеет пиковую производительность, как правило, на достаточно узком круге задач или при определенных свойствах обрабатываемых данных.
2. Развитие кластерной организации компьютеров, GRID-вычислений и упрощение практики программирования под кластерные платформы позволяет быстро и дешево организовывать высокопроизводительные и одновременно универсальные вычислительные мощности.

3. Скорость работы (в последовательном, а не параллельном режиме) нейрона и, соответственно, нейропрограммы, возрастает практически до максимума при представлении наборов сигналов и переменных нейронные сети как векторов и матриц и использовании расширений системы команд процессоров Intel. Дальнейшее увеличение общей скорости работы может быть достигнуто только реализацией параллельных алгоритмов.

Простота достижения предела скорости нейрона как базового способа повышения эффективности и ничтожность выигрыша при распараллеливании функций сумматора, активации, расчета градиента, весов для параллельного использования множества вычислительных узлов делают необходимым распараллеливание преимущественно сервисных алгоритмов. Так, например, исследования показали, что выигрыш времени параллельной обработки вышеперечисленных функций сопоставима со временем передачи данных и результатов между вычислительными узлами [5].

В данной работе предлагается несколько групп сервисных задач и алгоритмов, показана возможность их параллельной реализации для решения задач классификации и прогнозирования.

Задача выбора параметров обучения нейронные сети. Коррекция адаптивных внутренних параметров нейронные сети может идти как после просмотра каждого очередного примера обучающей выборки, так и путем накопления градиентов всех примеров (тестов). Второй вариант может быть легко распараллелен, но теоретически и практически проигрывает по достигаемой скорости обучения сети online-обучению [1]. Суммарный градиент, особенно при его дополнении эффективными методами оптимизации типа сопряженных градиентов, часто приводит к переобучению сети [1]. Поэтому для предотвращения побочных эффектов лучше использовать online-обучение, которое принципиально не распараллеливается. Поэтому предлагается распараллеливать, например, схему начального определения оптимального шага коррекции весов, дающего высокую скорость обучения наравне с достижением высоких обобщающих способностей сети. Online-обучение мало чувствительно к величине шага [1], поэтому реально выигрыш от распараллеливания процесса обучения нейронной сети можно получить только в случае большого размера обучающих выборок (десятки и сотни тысяч примеров). Тогда параллельный расчет суммарного градиента может привести к сокращению общего времени обучения или при адаптации шага на каждой итерации (эпохе) обучения, когда можно параллельно выполнить пробы нескольких значений величин шагов обучения.

Предобработка данных, выбор примеров в обучающую выборку. Скорость обучения существенно зависит от схемы предобработки данных для нейронной сети [1, 2, 4]. Вычисление константы Липшица обучающей выборки как индикатора оптимальности предобработки может быть распараллелено. Есть и другие возможности и стратегии обучения на подмножестве обучающей выборки, например, селекция после каждой итерации обучения очередного малого числа обучающих примеров. Также предлагается не фиксировать размер обучающей выборки (или соотношение размеров обучающей и тестовой выборок), а начинать с некоторой малой обучающей выборки и после обучения на ней нескольких нейронных сетей переносить в нее из тестовой выборки примеры, после чего обучать новые сети на уже увеличенной выборке и исследовать систематическую ошибку. Используемые в этих случаях этапы тестирования всей массы примеров или обучения нескольких нейронных сетей могут быть распараллелены.

Методы поиска и исправления «нетипичных» примеров (выбросов) в данных. Любая робастная схема имеет гипотезы о природе нетипичности, обоснованность которых для конкретной выборки не всегда удается подтвердить. На этапах предобработки данных, отбора подмножеств примеров, поиска нетипичных наблюдений, при разведочном анализе данных повышение скорости работы алгоритмов (через их распараллеливание) позволит проводить более быстрый, глубокий и всесторонний анализ свойств данных. Рассмотрение графиков точностей решения обучающих и тестовых выборок вдоль оси размера обучающей выборки или оси размера нейронные сети позволяет определить минимально необходимый размер обучающей выборки, с которого заканчивает проявляться чувствительность качества решения к размеру выборки, или максимальный размер сети, с ростом которого начинает проявляться эффект "переобучения" ("запоминания" свойств обучающей выборки с одновременным снижением качества обобщения).

Задача обучения нескольких нейронных сетей для последующего принятия решения о выборе тех или иных настроек является также примером параллельной реализации. Распараллеливание обучения единичной сети возможно в схемах подбора размера сети одновременно с ее обучением путем старта с минимального размера и добавления нейронов в сеть. Этот подход позволяет параллельно реализовать варианты помещения нейронов в разные места сети (в один из имеющихся слоев или в новый слой), либо варианты нейронов с разными нелинейными функциями, затем выбирать и реально встраивать в сеть только один нейрон с наилучшим критерием качества. Может быть осуществлено параллельное обучение набора сетей для последующего объединения в кластер.

Гибридные алгоритмы используют два и более метода (не обязательно нейросетевых). Например, для построения иерархического многошагового решения: первый метод строит разделение областей компетенции, а второй – решающую модель для области. Классическим примером может быть построение кусочно-линейного регрессора (с каждым кластером связано свое уравнение регрессии). Для прогноза скалярного временного ряда может быть предложен способ идентификации скрытой марковской цепи, для каждого из состояний которой строится нейросетевая прогнозная модель. Очевидно, что обучение набора моделей-экспертов, каждая из которых работает в своей области данных, легко распараллеливается. Следует отметить, что шаг кластеризации данных, т.е. идентификации этих областей, требует повышения его робастности и возможности выделения кластеров, здесь также можно специально привлекать эффективно распараллеливаемые методы, ставить и решать задачи, трудозатратные при однопроцессорных расчетах.

Отбор информативных независимых переменных. Метод отбора и группировки информативных независимых переменных использует обучение и редукцию структур достаточно большого числа нейросетей. В настоящее время используется схема идентификации информативных признаков и получения координат, разделяющих классы поверхностей путем исследования чувствительности значений выходных сигналов нейронные сети в различных точках пространства независимых переменных в ходе мелкошагового сканирования области значений в пространстве достаточно большой размерности (все входные сигналы сети, неинформативные, информативные, поскольку информативность определяется по результатам этой процедуры), т. е. является прямой переборной процедурой. Достаточно интересной для параллельного программирования является парадигма multitask learning: если в предметной области есть несколько задач прогнозирования или классификации, опирающихся на общий набор независимых признаков, то одновременное решение всех задач одним аппаратом (нейронной сетью с несколькими выходными сигналами) может повысить обобщающие способности, поскольку такая нейронная сеть строит и использует "промежуточные рассуждения", пригодные для решения всех задач, т.е. потенциально более адекватные причинно-следственным закономерностям предметной области и менее подверженные шумам в измерениях значений отдельных независимых признаков. Можно искусственно создавать ситуацию одновременного обучения нескольким задачам путем перевода найденных неинформативных независимых признаков в число зависимых.

В работе дана попытка обобщить группы задач, использующих нейромоделирование, и методов, внедрение которых в практику повысит эффективность использования нейронных сетей, но которые при этом требуют значительных вычислительных затрат и поэтому требуют параллельных реализаций. Некоторые случаи (например, задачи комбинаторного перебора примеров выборки) наиболее эффективно могут быть решены при запросе программой числа параллельных процессоров и оптимальном, возможно, параллельно-последовательном разделении фрагментов задачи между процессорами с целью поместить все данные, обрабатываемые процессором, в его кэш и получить нелинейное ускорение за счет исключения повторных обращений к обычной памяти компьютера. Это еще один способ повышения скорости работы программ, который можно применять при практическом программировании.

При реализации ИНС важными параметрами являются время вычислений и точность. Под временем вычислений понимается время, требуемое всей ИНС для формирования выходного результата. Точность можно рассматривать в двух аспектах: как точность получаемого результата (на выходе ИНС) и как точность промежуточных вычислений (выход отдельного нейрона).

С учетом этого можно сформулировать основные требования к программным средствам, предназначенным для реализации ИНС в следующем виде:

минимизировать время вычислений сети;

уменьшить объем и сложность вычислений;

обеспечить требуемую точность вычислений.

Последнее из приведенных требований связано, в первую очередь, со способом реализации элемента, реализующего функции отдельного нейрона. Два первых условия – с выбором числа таких элементов. Очевидно, они являются противоречивыми.

Для поиска компромиссного числа (C) элементов, реализующих функции отдельного нейрона, неохо-димо учесть следующие положения.

Вычисления, соответствующие отдельному нейрону, имеют конечную длительность (τ). При использовании только одного вычислителя (вычислительного узла) $(C=1)$, последовательно решающего задачи каждого из P нейронов сети, время вычислений сети многократно увеличится по сравнению с интервалом τ, составив величину $t=\tau\cdot P$.

Для снижения времени вычислений сети, проводимых всей сетью, естественно увеличить число элементов, реализующих функции отдельного нейрона и работающих параллельно во времени.

На практике используются различные структуры ИНС. Часто применяются многослойные сети. Послойная организация сети оказывает влияние на пути ее практической реализации. Пусть P нейронов сети разбиты на $N(N\le P)$ слоев. Как правило, число нейронов в разных слоях разное. Пусть наибольшее число нейронов в отдельном слое составляет $M_N(M_N<P)$.

Послойная структура ИНС определяет необходимость проведения вычисления сначала в первом слое, затем – во втором, и т. д. Другими словами, в отдельный момент времени работают вычислители (нейроны) только одного слоя. Естественно при этом должно быть обеспечено поступление соответствующих входных данных (выходы нейронов предыдущего слоя) и фиксация результатов вычислений.

Отсюда следует, что для многослойной сети все вычислительные узлы не могут работать одновременно. Даже если число вычислительных узлов равно числу нейронов, то есть каждый вычислитель работает в качестве только одного нейрона.

Другой вывод – для многослойной ИНС целесообразно использовать число элементов, реализующих функции отдельного нейрона и работающих параллельно во времени, не меньшим, чем число нейронов в самом насыщенном слое $(C \geq M_N)$. В этом случае время вычислений сети составит $T = \tau \cdot M_N$.

Следует иметь в виду также, что при практической реализации ИНС с использованием больше, чем одного вычислительного узла $(C > 1)$, возникает проблема обмена данными между отдельными вычислительными узлами. Обеспечение эффективного обмена данными между вычислительными узлами может привести к проблемам сетевой реализации параллельного вычисления в кластерной структуре. Поэтому может оказаться целесообразным нахождение таких алгоритмических, процедурных и сервисных блоков обучения и эксплуатации сети, которые сохраняют паритет между временем вычислений сети и простоту ее кластерной реализации, уменьшая число вычислительных узлов $(C \to 1)$.

Из приведенных рассуждений следует важный практический вывод. Для актуальной многослойной ИНС увеличение числа элементов, реализующих функции отдельного нейрона, свыше наибольшего числа нейронов в отдельном слое (M_N) не приведет к сокращению времени вычислений сети. Кроме того, не всегда есть смысл использовать даже такое (M_N) количество элементов. Так, исходя из удобства реализации связей между нейронами, может оказаться целесообразным использовать уменьшенное по сравнению с количеством нейронов в самом насыщенном слое число вычислителей $(C < M_N)$, а в пределе – только один вычислитель $(C = 1)$.

Как упоминалосьранее, по отношению к ИНС понятие «точность» имеет двойной смысл.

Рассматривая отдельный элемент (вычислительный узел), реализующий функции отдельного нейрона, следует говорить о точности вычислений, производимых им. Актуальность данного рассмотрения определяется ограниченностью разрядности такого вычислительного узла, а также возможной их гетерогенностью.

Еще более остро встает эта проблема, если для эмуляции ИНС, работающей с действительными числами, используется вычислительный узел, использующий целочисленную арифметику. В этом случае используется масштабирование обрабатываемых величин, обеспечивающее переход от действительных чисел к целым (и обратно).

Однако опыт показывает, что при рациональном выборе параметров ИНС даже использование целочисленной арифметики позволяет получать точность вычислений, обеспечивающую успешное решение прикладной задачи

Второй смысл понятия «точность» в данном контексте связан с корректностью результата, получаемого на выходе ИНС. Пусть вычисления, соответствующие отдельному нейрону, проводятся с идеальной точностью. Но даже в этом случае результат, получаемый на выходе ИНС, может быть далеким от идеала. Рассматриваемая точность выходного результата зависит от того, насколько корректно заданы параметры ИНС. Под параметрами тут понимаются структура сети, а для сетей, использующих предварительное обучение, еще и значения весовых коэффициентов и смещений.

Указанные параметры не связаны непосредственно с аппаратной реализацией сети, а определяются на этапах конструирования сети (структура сети) и подготовки ее к работе – обучения (значения весовых коэффициентов и смещений). Целесообразно остановиться на последнем этапе подробнее.

Наибольший интерес при практической реализации обычно представляют ИНС, решающие задачи

кластеризации;

классификации (при использовании предварительного обучения);

прогнозирования.

Например, по отношению к сетям, решающих задачи классификации, цикл функционирования можно разбить на два этапа:

обучение – подготовка сети к работе;

работа – однократное формирование результата классификации предъявленных входных данных.

После того как ИНС спроектирована – выбрана ее структура, а применительно к многослойной сети – это число нейронов в каждом слое и связи между слоями – должно быть проведено ее обучение. Для проведения обучения требуется обеспечить:

формирование обучающей выборки;

представительность и непротиворечивость обучающей выборки;

сходимость процесса обучения.

Для формирования обучающей выборки необходимо отобрать обучающие примеры и преобразовать соответствующую информацию к виду, используемому на входе ИНС. Так, например, применительно к системе распознавания объектов на изображениях это соответствует выбору некоторого набора изображений (или участков на изображениях) и преобразованию участков изображений, соответствующих выбранным объектам в некоторое множество чисел, число которых соответствует числу входов ИНС. Такой набор значений входа сети удобно назвать наблюдением. Совокупность наблюдений, каждое из которых сформировано по отдельному объекту, класс которого требуется распознавать, образует обучающую выборку. Практика показала, что формирование обучающей выборки наиболее целесообразно проводить «учителю» с использованием специализированного программного обеспечения.

Для успешного обучения ИНС полезно, чтобы различные классы, которые ИНС должна распознавать, были представлены в обучающей выборке примерно поровну. Лучше всего это обеспечивать на этапе формирования обучающей выборки. В противном случае можно прореживать выборки наиболее широко представленных классов или, что дублировать выборки, соответствующие классам, имеющим малое представительство. При таком прореживании (размножении) полезно использовать понятие «качества записи» в выборке.

Для этого в ходе формирования выборки каждому используемому объекту следует приписать некоторую оценку. Например, оценку того, насколько уверенно его можно отнести к данному классу. Другой вариант формирования оценки – учет того, насколько часто такой объект может встречаться на практике. При прореживании (размножении) какая оценка может служить формальным критерием отбрасывания (дублирования) конкретной записи.

Сформированная выборка должна быть представительной и непротиворечивой. Представительность еще можно контролировать в процессе формирования. Однако для проверки непротиворечивости наиболее удобно использовать кластерный анализ.

Процесс обучения ИНС с использованием обучающей выборке является недетерминированным. Это означает, что в зависимости от свойств используемой выборки и, конечно, используемого алгоритма обучения он может или идти чрезмерно долго, или не обеспечивать требуемой точности решения. Реализовать обучение возможно либо на одном ПК, или на вычислительном кластере (вычислительных узлах) либо на некотором многопроцессорном устройстве. В последних двух вариантах, как говорилось выше, могут возникнуть проблемы с обменом данными между процессорами.

Произвольность формирования обучающей выборки, в зависимости от строгости ее проверки на противоречивость и объема, может привести к чрезмерному росту времени обучения или к тому, что его результаты будут неудовлетворительными. Типичный подход к этой ситуации, упоминаемый в литературе, заключается в модификации процесса обучения, сводящейся, как правило, к уложению алгоритма. Однако, при создании прикладных систем можно рекомендовать и иной подход. Он заключается в упрощении задачи, возлагаемой на ИНС, за счет дополнительной нагрузки на нейросетевые алгоритмы обработки информации. Степень такого упрощения задачи ИНС выбирается таким образом, чтобы обеспечить эффективность обучения сети.

Рассмотрим особенности решения задачи прогнозирования (анализа временных рядов) на основе использования ИНС, реализованных на нескольких вычислительных узлах. В качестве примера задачи прогнозирования рассмотрим задачу определения значений котировок акций на момент времени t+1. Требуется определить значения котировок основных акций фондовой биржи с учетом предположения, что их поведение на рынке обусловлено взаимным влиянием, т. е. существует рефлексия поведения котировки акции в зависимости от поведения других котировок акций во времени.

Следует сказать, что задача прогнозирования котировок акций на фондовом рынке является достаточно сложной задачей, предполагающей, кроме учета временных рядов самих котировок, учета множества факторов, таких как PFTS-индекс, курс доллара, объем предложения акций, объем последних закупок и т.д. Поэтому в данной работе используются некоторые упрощения прогнозной модели. На настоящий момент времени Украинская Фондовая Биржа (УФБ) является неразвитой в том смысле, что количество зарегистрированных акций, которые имеют постоянные котировки, не превышает 602, а сами торги имеют низкие объемы сделок (например, по сравнению с РФБ). Однако публичные данные (месячные) котировок акций позволят получить упрощенную прогнозную модель поведения цен (котировок) акций на УФБ для работы трейдера для дальнейшей ее модификации и апробации на данных международных фондовых бирж. Следует отметить, что, например, данные с РФБ обновляются в течение нескольких секунд (от 3–6 с.), и, следовательно существует незначительный лимит времени для решения задачи прогнозирования для международных фондовых бирж. Также необходимо отметить, что поведения акций на рынке связаны между собой как на уровне коинтеграции временных рядов, описывающих их изменение, так и вследствие влияния общих внешних факторов. Существует значительная вычислительная сложность прогнозирования поведения одновременно всех акций фондового рынка, тем более в условиях необходимости получения результата в течение нескольких секунд. Поэтому предлагается осуществить прогнозирование на основе архитектуры многослойных персептронов (MLP) с выбранным окном и горизонтом прогноза. Выбор обусловлен возможностью логического распараллеливания сложных вычислительных операций (расчет сумматора, функции активации, градиента, весов) для каждого нейрона слоя, а также сервисных задач, связанных с настройкой и обучением сети. Была предложена архитектура MLP (602–1200–602), включающая 602 входа и 602 выхода, которая предполагает использовать при окне равным 1 (т. е. прогноз только на основе предыдущего наблюдения) 1802 нейрона (1200 промежуточных + 602 выходных). Эксперимент показывает, что для обучающей выборки в 128 наблюдений и количестве эпох 300 расчет осуществляется в пределах 6 мин на ПК с процессором с частотой 2 ГГц и оперативной памятью 2 Гб.

Визуальный пример сети MLP (602-1200-602) показан на рис. 1.

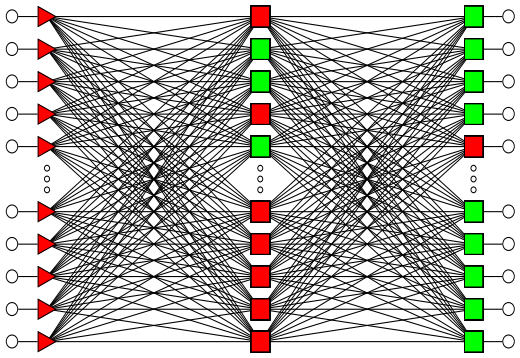

Рис. 1. Пример сети MLP (602-1200-602)

Тестирование работы многослойного персептрона в разных режимах параллельного обучения и предпроцессорной обработки данных показали следующие результаты.

С помощью пакета Statistica 7.1 [3] были отобраны восемь базовых архитектур, которые имеют на выходе 602 нейрона. В качестве функции активации использовалась экспоненциальная функция. В примере показана вычислительная сложность обучения сетей разных архитектур при условии, что они запускались одновременно на 18 вычислительных узлах.

В качестве вычислительных узлов были использованы ПК, объединенные в локальную сеть, на которых одновременно запускались на обучение сети с разными архитектурами. В качестве критерия остановки было выбрано условие прохождения 100 эпох. На рис. 2 показан результат обучения сетей, где ось ординат представляет собой время (секунды), а ось абсцисс – количество нейронов в архитектуре. Следует отметить, что при росте количества нейронов время обучения экспоненциально возрастает. На одном компьютере обучение восьми архитектур ИНС заняло 28 минут, при распараллеливании обучения каждой архитектуры на соответствующем узле максимальное время обучения составило 4,8 мин. Соответственно, использование нескольких вычислительных узлов действительно существенно позволит ускорить подбор оптимальной архитектуры сети для решения поставленных задач.

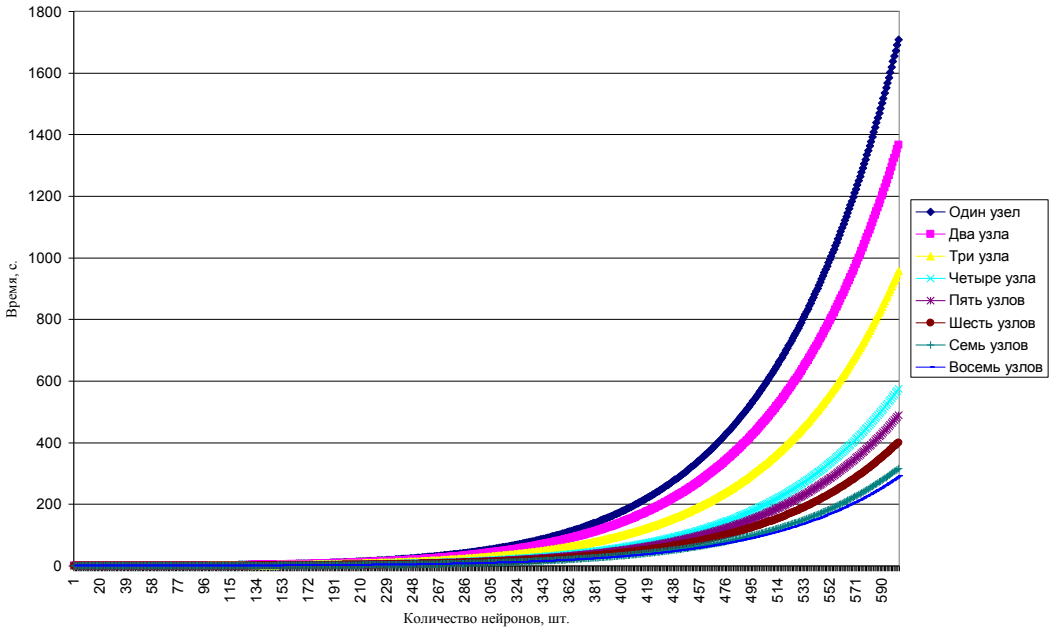

Рис. 2. Тестирование работы сети для разного количества вычислительных узлов и нейронов ИНС

Анализ производительности разных алгоритмов при обучении ИНС разных архитектур (с разным количеством нейронов) показан на рис. 3, где алгоритм BackPropogation менее эффективен, чем алгоритм Conjugate Gradient при увеличении количества нейронов в архитектуре многослойного персептрона. Для данной задачи выигрыш во времени расчета достигает до 5 минут.

При использовании подхода к распараллеливанию обучения архитектуры MLP (602–1200–602) на двух вычислительных узлах (с помощью операционной системы MOSIX) время выполнения задачи уменьшается на 27 % , однако с увеличением количества нейронов и эпох обучения существует тенденция к увеличению временной сложности. Следовательно, при большом количестве нейронов (больше 1000) и количестве эпох больше 300 кластерная операционная система, работающая с множеством узлов процессоров и общей памятью, становится неэффективной, потому что передача параметров расчета и промежуточных результатов занимает много времени и требует хранения больших массивов в памяти вычислительных узлов [5–7].

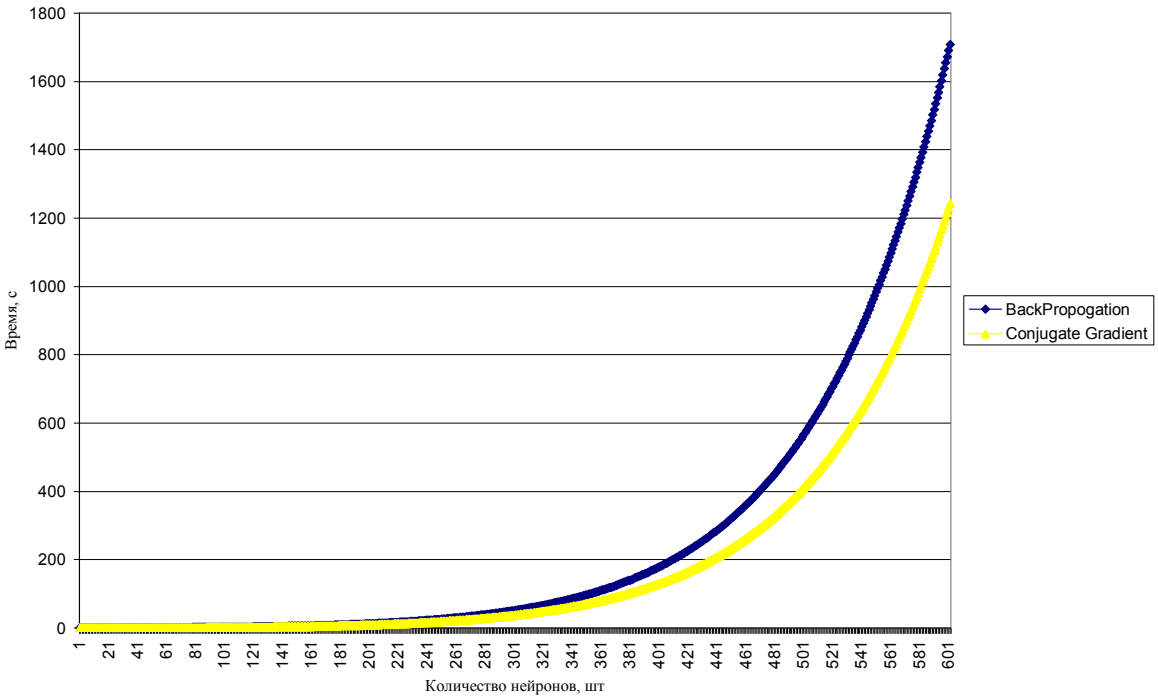

Рис. 3. Сравнительный анализ эффективности алгоритмов обучения ИНС

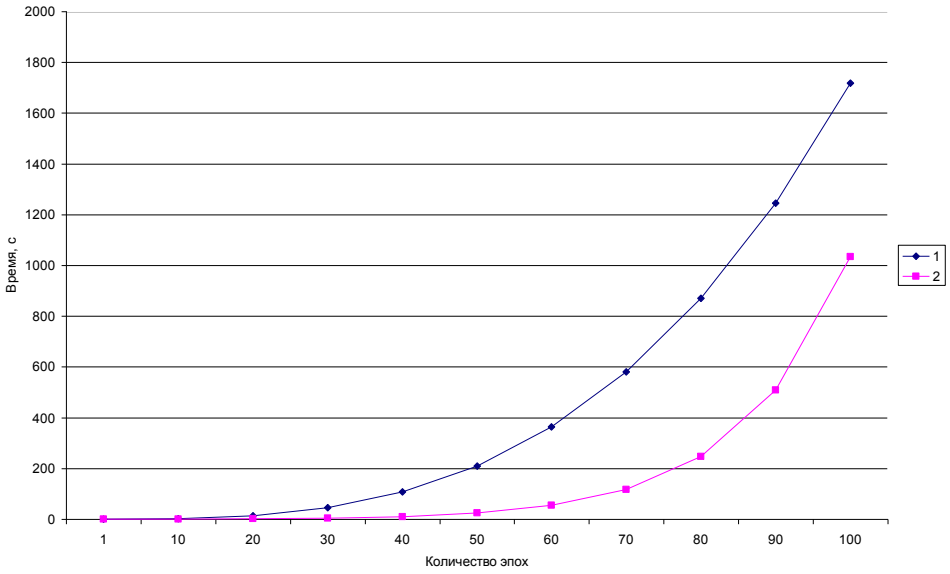

Рис. 4. Сравнительный анализ временной сложности обучения по эпохам для ИНС MLP (602–1200–60) для одного узла и двух узлов в кластерной ОС MOSIX

Модель ИНС для анализа и тестирования подходов к обучению на основе алгоритма BackPropogation и сервисному распараллеливанию реализована в C# в виде отдельных функций, которые могут быть вызваны и запущены на разных вычислительных узлах. Фрагмент программы, реализующей обучение нейронной сети, приведен ниже.

```
using System;
using System.Collections.Generic;

namespace Neuro
{
   public class Axon
   {
      private Double potential = .0;
      private List<Dendrite> postSynapticDendrites = null;
      public Axon(Double potential)
```

```
        {
            this.postSynapticDendrites = new List<Dendrite>();
            this.potential = potential;
        }
        public Double Potential
        {
            get { return this.potential; }
            set { this.potential = value; }
        }
        public void ConnectDendrite(Dendrite dendrite)
        {
            this.postSynapticDendrites.Add(dendrite);
        }
    }
    public class Dendrite
    {
        private Double synapticWeight = .0;
        private Double potential = .0;
        public Dendrite(Double weight, Double potential)
        {
            this.synapticWeight = weight;
            this.potential = potential;
        }
        public Double SynapticWeight
        {
            get { return this.synapticWeight; }
            set { this.synapticWeight = value; }
        }
        public Double Potential
        {
            get { return this.potential; }
            set { this.potential = value; }
        }
    }
    public class Neuron
    {
        private Axon axon = null;
        private List<Dendrite> dendrites = null;
        public Neuron(Int32 numDendrites)
        {
            this.axon = new Axon(new Random().NextDouble());
            this.dendrites = new List<Dendrite>();
            for (Int32 i = 0; i < numDendrites; i++)
            {
                dendrites.Add(new Dendrite(new Random().NextDouble(), new Random().NextDouble()));
            }
        }
        public void CalculateAxonPotential()
        {
            Double weightedSum = .0;
            foreach (Dendrite dendrite in this.dendrites)
            {
                weightedSum += dendrite.Potential * dendrite.SynapticWeight;
            }
            this.axon.Potential = Math.Exp(-0.2 * weightedSum);
        }
        public Dendrite GetDendrite(Int32 index)
        {
            return this.dendrites[index];
        }
        public Axon GetAxon()
        {
            return this.axon;
        }
    }
}
```

Реализация алгоритма обучения для 200 выходных нейронов

```
using System;
using Neuro;
using System.Collections.Generic;
```

```
using System.Windows.Forms;
namespace NeuroModel
{
   public partial class MainForm : Form
   {
      public MainForm()
      {
         this.InitializeComponent();
      }
      private void buttonOk_Click(object sender, EventArgs e)
      {
         this.Close();
      }
      private void buttonCreateNetwork_Click(object sender, EventArgs e)
      {
         Double[] X = new Double[50];
         Double[] Y = new Double[50];
         Neuron[] neurons = new Neuron[200];
         Double[] gradient = new Double[200];
         //инициализация входного выходного вектора
         for(int i = 0; i < 50; i++)
         {
            X[i] = i * 10;
            Y[i] = (i * 10) - 100;
         }
         try
         {
            //инициализация сети
            for (Int32 i = 0; i < 50; i++)
            {
               neurons[i] = new Neuron(1);
               neurons[i].GetDendrite(0).Potential = X[i];
               neurons[i].CalculateAxonPotential();
            }
            for (Int32 i = 50; i < 150; i++)
            {
               neurons[i] = new Neuron(50);
               for (int j = 0, d = 0; j < 50; j++, d++)
               {
                  neurons[i].GetDendrite(d).Potential = neurons[j].GetAxon().Potential;
               }
               neurons[i].CalculateAxonPotential();
            }
            for (Int32 i = 150; i < 200; i++)
            {
               neurons[i] = new Neuron(100);
               for (int j = 50, d = 0; j < 150; j++, d++)
               {
                  neurons[i].GetDendrite(d).Potential = neurons[j].GetAxon().Potential;
               }
               neurons[i].CalculateAxonPotential();
            }
            //обучение сети
            for (Int32 iteration = 0; iteration < 100; iteration++)
            {
               for (Int32 i = 150, y = 0; i < 200; i++, y++)
               {
                  gradient[i] = neurons[i].GetAxon().Potential * (1 - neurons[i].GetAxon().Potential) * (Y[y] - neu-
rons[i].GetAxon().Potential);
               }
               for (Int32 i = 50, d = 0; i < 150; i++, d++)
               {
                  Double sigma = .0;
                  for (Int32 j = 150; j < 200; j++)
                  {
                     sigma += neurons[j].GetDendrite(d).SynapticWeight * gradient[j];
                  }
                  gradient[i] = neurons[i].GetAxon().Potential * (1 - neurons[i].GetAxon().Potential) * sigma;
               }
               //перестройка весовых коэффициентов
```

```
                for (Int32 i = 150; i < 200; i++)
                {
                    for (Int32 j = 50, d = 0; j < 150; j++, d++)
                    {
                        neurons[i].GetDendrite(d).SynapticWeight += 0.1 * gradient[i] * neurons[j].GetAxon().Potential;
                    }
                }
                for (Int32 i = 50; i < 150; i++)
                {
                    for (Int32 j = 0, d = 0; j < 50; j++, d++)
                    {
                        neurons[i].GetDendrite(d).SynapticWeight += 0.1 * gradient[i] * neurons[j].GetAxon().Potential;
                    }
                }
                //прямое прохождение сигнала
                for (Int32 i = 50; i < 150; i++)
                {
                    for (int j = 0, d = 0; j < 50; j++, d++)
                    {
                        neurons[i].GetDendrite(d).Potential = neurons[j].GetAxon().Potential;
                    }
                    neurons[i].CalculateAxonPotential();
                }
                for (Int32 i = 150; i < 200; i++)
                {
                    for (int j = 50, d = 0; j < 150; j++, d++)
                    {
                        neurons[i].GetDendrite(d).Potential = neurons[j].GetAxon().Potential;
                    }
                    neurons[i].CalculateAxonPotential();
                }
            }
            this.dataGridView.Columns.Add("Input", "Входные значения");
            this.dataGridView.Columns.Add("Output", "Выходные значения");
            this.dataGridView.Columns.Add("Axon", "Значения аксонов");
            for (Int32 i = 0; i < 50; i++)
            {
                this.dataGridView.Rows.Add(X[i], Y[i], neurons[i + 150].GetAxon().Potential);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
  }
}
```

Заключение. В результате проведенного исследования были предложен подход к распараллеливанию решения статистических задач классификации и прогнозирования для вычислительного кластера на основе распараллеливания сервисных функций обучения, настройки и выбора архитектуры ИНС. Было показано, что для кластерных структур распараллеливание расчетов по слоям и нейронам (в рамках последовательного алгоритма BackPropogation) не дает ощутимого выигрыша, поскольку возрастает время передачи промежуточных результатов и ожидания расчетов от сопряженных вычислительных узлов. Вместе с тем, для архитектур ИНС, которые используют небольшое количество узлов и эпох обучения, выигрыш составит от 25 до 30 % для двух узлов. В перспективе представляется целесообразным исследовать эффективность распараллеливания на многопроцессорном вычислительном кластере при условии параллельной реализации блоков алгоритмов BackPropogation и Conjugate Gradient.

1. *Хайкин С.* Нейронные сети: полный курс. – М.: Вильямс, 2008. – 1104 с.
2. *Барсегян А.А., Куприянов М.С,. Степаненко В.В, Холод И.И.*. Методы и модели анализа данных: OLAP и Data Mining. – БХВ-Петербург, 2004.
3. *Боровиков В. Statistica.* Искусство анализа данных на комьютере. – 2003. – 688 с.
4. *Круглов В.В., Борисов В.В.* Искусственные нейронные сети. Теория и практика.– М.: Горячая линия –Телеком, 2002. – 382 с.
5. http://parallel.ru/vvv/lec7.html
6. http://www.ccas.ru/paral/prog/comm_msg/intro.html
7. http://osp.admin.tomsk.ru/os/index.htm

PARALLEL REALIZATION OF THE SOLUTION OF PROBLEMS OF PREDICTION AND CLASSIFICATION FOR CLUSTER BASED ON ARTIFICIAL NEURAL NETWORKS

S.V. Minukhin,S.V. Znakhur