УДК 004.855.5

ОБУЧЕНИЕ НЕЙРОННЫХ СЕТЕЙ МЕТОДОМ ЛЕВЕНБЕРГА-МАРКВАРДТА В УСЛОВИЯХ БОЛЬШОГО КОЛИЧЕСТВА ДАННЫХ

С. С. Пархоменко, Т. М. Леденёва

Воронежский государственный университет

Поступила в редакцию 15.05.2014 г.

Аннотация. Для достижения высокой точности обучения нейронной сети часто применяют алгоритм Левенберга-Марквардта. Однако алгоритм требует сложные вычисления, занимающей много времени. В данной статье представлено подробное описание метода и предложены способы его оптимизации и распараллеливания с целью увеличения производительности.

Ключевые слова: нейронная сеть прямого распространения, обучение нейронных сетей, метод Левенберга-Марквардта

Annotation. The Levenberg-Marquardt method achieves a high accuracy in a neural network training. However, it requires a complex time-consuming calculation. This article provides a detailed description of the method and suggests approaches to its optimization and parallelization for the performance improvement.

Keywords: backpropagation neural network, training neural networks, Levenberg-Marquardt method.

1. ПОСТАНОВКА ЗАДАЧИ

Нейронная сеть прямого распространения (НСПР) – нейросетевая архитектура, состоящая из следующих друг за другом слоёв нейронов; в качестве входов каждый последующий слой использует выходы предыдущего слоя (обычно называемого *скрытым слоем*) за исключением *входного слоя*, непосредственно принимающего входные сигналы [1]. Основная функция скрытых слоёв – выделение статистики высокого порядка.

Обучение нейронных сетей прямого распространения – это процесс определения значений весов сети на основе примеров, образующих *обучающее множество* для сети с n входами и m выходами, состоит из N вход-выходных значений – *обучающих примеров*. Структура обучающего множества имеет вид:

$$\left\langle \tilde{X}, \tilde{Y} \right\rangle = \left\{ \left(\tilde{x}_i, \tilde{y}_i \right) \right\}_{i=\overline{1,N}}, \qquad (1)$$

где $\tilde{x}_i \in \mathbb{R}^n$ – входной вектор из i-го примера, $\tilde{y}_i \in \mathbb{R}^m$ – вектор ожидаемых значений (указаний учителя).

Степень близости вектора-ответа сети y_i на i-м примере и соответствующего вектора указаний учителя $\tilde{y}_i$ при текущем векторе весов нейронной сети $w \in \mathbb{R}^W$, где W – количество весовых коэффициентов НСПР, характеризуется мгновенным функционалом качества [2]:

$$Q_i = Q\left(\varepsilon_i\left(w\right)\right) = \varepsilon_i^T\left(w\right) \cdot V \cdot \varepsilon_i\left(w\right), \qquad (2)$$

где $\varepsilon_i\left(w\right) = y_i\left(w\right) - \tilde{y}_i \in \mathbb{R}^m$ – вектор отклонений выходов сети от указаний учителя, $V \in \mathbb{R}^{m \times m}$ – положительно определённая матрица, задающая взвешенную норму вектора $\varepsilon_i\left(w\right)$. Обычно V – единичная матрица, что сводит функционал к евклидовой норме вектора отклонений

$$\begin{aligned} Q\left(\varepsilon_i\left(w\right)\right) &= \varepsilon_i^T\left(w\right)\varepsilon_i\left(w\right) = \\ &= \left(y_i\left(w\right) - \tilde{y}_i\right)^T \left(y_i\left(w\right) - \tilde{y}_i\right) = \\ &= \sum_{j=1}^{m} \left(y_i\left(w\right) - \tilde{y}_i\right)^2. \end{aligned} \qquad (3)$$

Степень соответствия сети данным из обучающего множества задаётся интегральным функционалом качества обучения [2]

© Пархоменко С. С., Леденёва Т. М., 2014

$$E(w)=\sum_{i=1}^{N}Q_i(w)=\sum_{i=1}^{N}\sum_{j=1}^{m}\left(y_i(w)-\tilde{y}_i\right)^2. \quad (4)$$

Для случая с одним выходом $(m=1)$ и с учётом его обозначения как $F(x,w)$ функционал принимает следующий вид:

$$E(w)=\sum_{i=1}^{N}Q_i(w)=\sum_{i=1}^{N}\left(y_i(w)-\tilde{y}_i\right)^2= \\ =\sum_{i=1}^{N}\left(F(x_i,w)-\tilde{y}_i\right)^2. \quad (5)$$

Цель обучения НСПР – определение такого вектора весов w^*, чтобы функционал (5) принимал минимальное значение, что превращает процесс обучения сети в решение задачи безусловной оптимизации

$$w^*=\arg\min_{w\in\mathbb{R}^W}E(w). \quad (6)$$

Многовыходная нейронная сеть с m выходами может быть заменена совокупностью m одновыходных сетей, что позволяет без ограничения общности рассмотреть методы обучения лишь для случая сетей с $m=1$.

Для решения (6) существует множество методов. Одним из популярных методов можно выделить *метод обратного распространения ошибки*. Несмотря на широкое применение, его главные недостатки – медленная сходимость и негативное влияние локальных минимумов поверхности $E(w)$ [3]. Существуют методы, которые не обладают этими недостатками, среди которых известен метод Левенберга-Марквардта.

Цель статьи заключается в оценке производительности метода Левенберга-Марквардта и разработке подходов к сокращению времени обучения НСПР, при условии $N\gg M$.

2. МЕТОД ЛЕВЕНБЕРГА-МАРКВАРДТА

Для устранения указанных недостатков обычно используется информация высокого порядка об $E(w)$. В рамках квадратичной аппроксимации ошибки в окрестности точки w имеет вид

$$E(w+\Delta w)\approx E(w)+\nabla E(w)\Delta w+ \\ +\frac{1}{2}\Delta w^T\nabla^2E(w)\Delta w. \quad (7)$$

На основе квадратичной аппроксимации разработаны широко известные методы Гаусса-Ньютона и Левенберга-Марквардта (ЛМ-метод), которые сводят задачу (6) для (7) к уравнению

$$\nabla E(w)+\nabla^2E(w)\Delta(w)=0. \quad (8)$$

При этом

$$\nabla E(w)=\frac{\partial\varepsilon(w)}{\partial w}\cdot\varepsilon(w)= \\ =\frac{\partial\left(y(w)-\tilde{y}\right)}{\partial w}\cdot\varepsilon(w)= \\ =\frac{\partial F(x,w)}{\partial w}\cdot\varepsilon(w), \quad (9)$$

$$\nabla^2E(w)=\left(\frac{\partial\varepsilon(w)}{\partial w}\right)^T\cdot\frac{\partial\varepsilon(w)}{\partial w}+ \\ +\sum_{i=1}^{N}\varepsilon(w)\nabla^2\varepsilon(w). \quad (10)$$

Ключевое различие между ними – подход к вычислению матрицы Гессе $\nabla^2E(w)$. Если представить (10) в виде

$$H=J^TJ+S, \quad (11)$$

где S – информация о вторых производных, то для метода Гаусса-Ньютона $S=0$, в то время как в ЛМ-методе S аппроксимируется эвристическими правилами.

Исходя из (8–11), метод Левенберга-Марквардта заключается в решении уравнения относительно $\Delta(w)$

$$\left(\underbrace{J^TJ}_{H^*}+\underbrace{\lambda I}_{\tilde{S}}\right)\Delta(w)=-J^T\varepsilon(w) \quad (12)$$

или в другой интерпретации

$$\left(J^TJ+\lambda I\right)\delta=J^T\tilde{\varepsilon}(w), \\ \tilde{\varepsilon}(w)=-\varepsilon(w)=\tilde{y}-y(w), \quad (13)$$

где λ – коэффициент затухания Левенберга, δ – вектор, состоящий из величин приращения весов.

Найденный вектор δ позволяет изменить вектор весов w. Элементы вектора w обычно упорядочиваются сначала по слою, затем по нейронам, и, наконец, по весу каждого нейрона и его смещению.

Параметр λ задаётся изначально и определяет поведение алгоритма, делая его более

похожим на градиентный метод или метод Гаусса-Ньютона. В самом начале обучения, когда функция $F(x,w)$ подобрана грубо, удобно использовать метод наискорейшего спуска, поэтому λ выбирается относительно большим. По мере уточнения коэффициентов w более эффективным становится метод Гаусса-Ньютона (при этом λ становится малой величиной; при $\lambda = 0$ метод вырождается в метод Гаусса-Ньютона). Так ЛМ-метод реализует адаптивную модель перехода между классами методов с явной аппроксимацией S и без неё.

В классическом методе Гаусса-Ньютона требуется невырожденность матрицы H. Для гарантированного обращения H К. Левенберг предложил подправлять элементы главной диагонали, путём добавления к ней матрицы λI (I – единичная матрица).

После того, как при заданном λ вектор δ будет вычислен, необходимо принять решение о принятии модификации или её отклонения. Для этого необходимо рассчитать $E(w+\delta)$ и сравнить полученное значение с $E(w)$. Если $E(w+\delta) \le E$, то необходимо уменьшить λ и изменить веса $w+\delta$, иначе λ увеличивается и метод применяется заново для нового λ.

Для настройки величины λ часто используется вспомогательная величина v, (обычно $v = 10$). Если λ необходимо увеличить, то λ умножается на v, иначе – делится. Умножение повторяется до тех пор, пока $E(w+\delta) > E(w)$. Как только выполняется неравенство $E(w+\delta) \le E(w)$, считается, что *один обучающий цикл* (эпоха) нейросети завершился.

В результате процедура, реализующая обучающий цикл НСПР, имеет вид [4]:

1) Построить матрицу Якоби J;

2) Рассчитать градиент ошибки $g = J^T \tilde{\varepsilon}(w)$;

3) Рассчитать приближенную матрицу Гессе с помощью матрицы Якоби $H^* = J^T J$;

4) Решить уравнение $(H^* + \lambda I)\delta = g$ относительно неизвестного вектора δ;

5) Вычислить $E(w+\delta)$;

6) Если $E(w+\delta) > E(w)$, то $\lambda := v\lambda$ и перейти на шаг 4, иначе $\lambda := \frac{\lambda}{v}$, $E(w) := E(w+\delta)$ и закончить цикл обучения.

Одним из дополнительных критериев останова в ЛМ-методе является то, что λ становится слишком большим.

К недостаткам ЛМ-метода относят:

1) высокую чувствительность к начальным значениям весовых коэффициентов;

2) за счёт использования данных из всей выборки возможно переобучение шума [4];

3) высокая вычислительная сложность, под которой подразумевается большая длительность вычислений при большом количестве обучающих примеров N и/или весовых коэффициентов W.

Для решения проблем ЛМ-метода существуют следующие подходы:

1) *Модификации*. Для увеличения сходимости к оптимальному решению предложены модификации различной сложности (примеры: настройка параметра $\lambda = 0.01\tilde{\varepsilon}(w)^T \tilde{\varepsilon}(w)$ [5], использование геодезического ускорения [6], включение в метод регуляризации Байеса [7] и др.);

2) *Оптимизация вычислительного процесса*, направленная на сокращение времени вычислений одной эпохи;

3) *Распараллеливание* – процесс, который непосредственно не влияет на вычислительную сложность, но позволяет более эффективно использовать ресурсы компьютера.

Остановимся на двух последних подходах.

Вычисление J обычно занимает значительную часть времени относительно остальных шагов цикла обучения, особенно при $N \gg W$, а значит оптимизация позволит существенно сократить время, затрачиваемое на обучение нейронной сети ЛМ-методом. В [8] указано на возможность снижения вычислительной сложности за счёт снижения точности вычисления матрицы J и/или отказа от универсальности.

В первом случае точный расчёт J заменяется приближенным с помощью метода Бройдена. При наличии на шаге n матрицы J_n получим J_{n+1} с помощью следующей формулы [9]:

$$J_{n+1} = J_n + \frac{\left(F\left(x, w_{n+1}\right) - F\left(x, w_n\right) - J_n h\right)}{h^T h} h^T, \quad (14)$$
$$h = w_{n+1} - w_n.$$

С теоретической точки зрения, было бы разумно использовать этот подход на каждом шаге ЛМ-метода, однако, на практике с течением времени аппроксимация становится грубее, что, в свою очередь, влияет на вектор-градиент $J^T \tilde{\varepsilon}\left(w\right)$ и делает необходимым повторный пересчёт матрицы Якоби более точными методами. Эту операцию можно проводить после каждого неудачного изменения вектора δ, который привёл к увеличению ошибки, или раз в две эпохи [6].

Если математическая модель НСПР фиксированная, то допускается отказ от универсального, но затратного по времени, поиска J через разностные методы вычисления производных [10]. Если рассмотреть сеть с n входами, m нейронами в скрытом слое и одним выходом, то получим

$$F\left(x, w\right) =$$
$$= act_1\left(\left[\sum_{i=1}^{m} \tilde{w}_i \cdot act_2\left(\sum_{j=1}^{n} w_{ij} x_j + \sigma_i\right)\right] + \tilde{\sigma}\right), \quad (15)$$

$$act_1\left(x\right) = act_2\left(x\right) = \frac{1}{1 + e^{-\alpha x}}. \quad (16)$$

где x_j – принимаемая j-ым нейроном величина, w_{ij} – весовой коэффициент, связывающий j-ый входной нейрон с i-ым нейроном скрытого слоя, σ_i – коэффициент смещения для i-го нейрона скрытого слоя, $\tilde{w}_i$ – весовой коэффициент, связывающий i-ый нейрон скрытого слоя в выходным нейроном, $\tilde{\sigma}$ – коэффициент смещения для выходного нейрона, $act_1\left(x\right)$, $act_2\left(x\right)$ – активационные функции для выходного нейрона и нейронов скрытого слоя.

Для (15) с учётом (16) аналитически выведем формулы для поиска элементов строки J:

$$s_i = \sum_{j=1}^{n} w_{ij} x_j + \sigma_i, \; S = \left(\sum_{i=1}^{m} \frac{\tilde{w}_i}{e^{-\alpha s_i} + 1}\right) + \tilde{\sigma},$$
$$\left[Es_i\right] = e^{\alpha s_i}, \quad \left[ES\right] = e^{\alpha S},$$
$$\left[EC'\right] = \frac{\alpha\left[ES\right]}{\left(\left[ES\right]+1\right)^2}, \quad \left[EC''\right] = \alpha\left[EC'\right], \quad (17)$$
$$\left[ECS_i\right] = \frac{\tilde{w}_i\left[EC''\right]\left[Es_i\right]}{\left(\left[Es_i\right]+1\right)^2},$$
$$\frac{\partial F\left(x, w\right)}{\partial w_{ij}} = x_j\left[ECS_i\right], \quad \frac{\partial F\left(x, w\right)}{\partial \sigma_i} = \left[ECS_i\right],$$
$$\frac{\partial F\left(x, w\right)}{\partial \tilde{w}_i} = \frac{\left[EC'\right]\left[Es_i\right]}{\left[Es_i\right]+1}, \quad \frac{\partial F\left(x, w\right)}{\partial \tilde{\sigma}} = \left[EC'\right]. \quad (18)$$
$$\left(1 \le i \le m, \; 1 \le j \le n\right).$$

Необходимо отметить, что при выводе (17) учитывалась возможность эффективного использования предварительных вычислений.

Аналитический подход к вычислению матрицы Якоби не универсален, так как для каждой новой модели нейронной сети (при изменении количества скрытых слоёв и/или замене активационных функций) потребуется заново выводить формулы. Однако он требует значительно меньше вычислений, чем, например, расчёт через центральные разностные производные, при этом не теряя в точности.

3. РАСПАРАЛЛЕЛИВАНИЕ

В рассматриваемой задаче некоторые вычисления сводятся к выполнению однотипной обработки большого набора данных. В первую очередь это касается матричных вычислений. В этом случае можно говорить, что существует параллелизм по данным.

Распределение вычисления строк матрицы J между потоками позволяет организовать её параллельное заполнение, поскольку элементы рассчитываются независимо друг от друга. Таким образом, реализуется один из вариантов *ленточной схемы* [11].

Пусть τ – максимально допустимое одновременно выполняемых потоков в вычислительной среде (такая ситуация характерна, например, при использовании технологии OpenMP). Будем считать, что $\tau \ll N$. Очевидно, что в среднем каждый поток обрабатывает примерно $\frac{N}{\tau}$ строк матрицы J. Однако примем для удобства, что одна строка J – минимальная обрабатываемая единица, которую необходимо задействовать в макси-

мально возможном количестве операций внутри параллельного блока. Обозначим за Ξ_t множество номеров строк J, обработанных потоком t.

Согласно (12), $H^* = J^T J$. Для экономии памяти можно отказаться от раздельного хранения матриц J^T и J, поскольку для любых $n \in [1, N]$ и $m \in [1, W]$ имеет место $J_{nm} = J^T_{mn}$, что даёт возможность получать элементы J^T из J перестановкой индексов местами.

Для вычисления элемента матрицы Гессе

$$H^*_{pq} = \sum_{i=1}^{N} J^T_{qi} J_{ip} = \sum_{i=1}^{N} J_{iq} J_{ip}, \quad (1 \le p \le W, \ 1 \le q \le W), \tag{19}$$

необходимо, чтобы все элементы p-го и q-го столбцов матрицы J были известны. В самом простом случае достаточно дождаться вычисления всей матрицы J. Однако это породит дополнительный барьер – точку синхронизации, которую должны достичь все процессы перед тем, как продолжится выполнение любого из них. Избежать этого позволит разложение матрицы H^* в виде суммы

$$H^* = \sum_{t=1}^{\tau} \left({}^{[t]}H \right) = {}^{[1]}H + {}^{[2]}H + {}^{[3]}H + \cdots + {}^{[\tau]}H, \tag{20}$$

$${}^{[t]}H = \left\{ {}^{[t]}H_{pq} \right\}_{W \times W},$$

где ${}^{[t]}H$ – матрица накопительной суммы потока t, ${}^{[t]}H_{pq}$ – элемент этой матрицы.

Для каждой рассчитанной строки матрицы J, можно произвести перемножение всех элементов этой строки между собой, получив матрицу-слагаемое ${}^{[t]}H^+_k$:

$${}^{[t]}H^+_k \Big|_{ij} = J^T_{ik} J_{kj} = J_{ki} J_{kj}, \quad (k \in \Xi_t, \ 1 \le i \le W, \ 1 \le j \le W), \tag{21}$$

где ${}^{[t]}H^+_k \Big|_{ij}$ – элемент матрицы ${}^{[t]}H^+_k$.

Суммируя матрицы-слагаемые отдельного потока получим матрицу накопительной суммы:

$${}^{[t]}H = \sum_{k \in \Xi_t} {}^{[t]}H^+_k, \tag{22}$$

которые будучи просуммированные с накопительными матрицами из других потоков за пределами параллельной области дадут результирующую матрицу Гессе H^*.

Таким образом, распределив вычисление скалярного матричного произведения по ${}^{[t]}H$ удалось совместить расчёт строки J и матрицы H внутри одного параллельного блока.

Расчёт $g = J^T \tilde{\varepsilon}(w)$ также требует все элементы p-го столбца матрицы J:

$$g_p = \sum_{i=1}^{N} J^T_{pi} \tilde{\varepsilon}(w) = \sum_{i=1}^{N} J_{ip} \tilde{\varepsilon}_i(w), \quad (1 \le p \le W). \tag{23}$$

Снова разложив вектор g на накопительные вектора ${}^{[t]}g$ для каждого потока, а их в свою очередь на вектора-слагаемые ${}^{[t]}g^+_k$ для всех строк J, обработанных внутри одного потока, получим аналогичный способ вычислений, использованный для расчёта H^*

$${}^{[t]}g^+_k \Big|_i = J^T_{ik} \tilde{\varepsilon}_k(w) = J_{ki} \tilde{\varepsilon}_k(w),$$

$${}^{[t]}g = \sum_{k \in \Xi_t} {}^{[t]}g^+_k, \quad g = \sum_{t=1}^{\tau} {}^{[t]}g, \quad (k \in \Xi_t, \ 1 \le i \le W). \tag{24}$$

Заметим, что после обработки строки J и всех сопутствующих вычислений (19–24), она больше не представляет интереса, следовательно, нет необходимости её хранить в оперативной памяти, как и целиком всю матрицу J. Достаточно построчно в нескольких потоках перебирать примеры $\langle \tilde{X}, \tilde{Y} \rangle$ и получать массив частных производных по весовым коэффициентам для проведения дальнейших расчётов. Это даёт ощутимую экономию памяти.

4. ПРАКТИЧЕСКИЙ ЭКСПЕРИМЕНТ

Для проведения экспериментов была разработана программа на языке C с применением технологии параллельного программирования OpenMP. В программе реализованы следующие однопоточные и параллельные варианты:

1) классического ЛМ-метода с расчётом J с помощью формулы центральной разностной производной;

2) ЛМ-метода с расчётом J методом Бройдена раз в две эпохи;

3) классического ЛМ-метода с расчётом J с помощью аналитически выведенных формул.

Для испытаний использован ПК под управлением ОС GNU/Linux с процессором AMD FX-8320, обладающий 8-ю ядрами, работающими на частоте 3.3ГГц.

Целью вычислительного эксперимента является оценка времени работы каждого метода с учётом предложенных способов расчёта матрицы Якоби.

Выберем для экспериментов несколько искусственных функций, применяемых для апробации методов оптимизации (рис. 2):

• функция Бирда [12]:

$$f(x,y)=\sin(x)\cdot e^{(1-\cos(y))^2}+\cos(y)\cdot e^{(1-\sin(x))^2}+$$
$$+(x-y)^2,\quad x,y\in[-7,7];$$

• функция Букина [12]:

$$f(x,y)=100\sqrt{\left|y-0.01x^2\right|}+0.01\left|x+10\right|,$$
$$x\in[-15,-5],\ y\in[-3,3];$$

• шестигорбовая обратная верблюжья функция на промежутке [13]:

$$f(x,y)=\left(4-2.1x^2+\frac{x^4}{3}\right)x^2+xy+$$
$$+\left(-4+4y^2\right)y^2,\ x\in[-3,3],\ y\in[-2,2];$$

• функция МакКормика [12]:

$$f(x,y)=\sin(x+y)+(x-y)^2-$$
$$-1.5x+2.5y+1,\quad x,y\in[-3,4];$$

• волновая функция [13]:

$$f(x,y)=-\frac{1+\cos\left(12\sqrt{x^2+y^2}\right)}{\frac{1}{2}\left(x^2+y^2\right)+2},$$
$$x,y\in[-2,2];$$

• долина Розенброка [13]:

$$f(x,y)=100\left(y-x^2\right)^2+(1-x)^2,$$
$$x,y\in[-2,2].$$

Изначально, эти функции предназначались для поиска точек экстремума на их поверхности. В случае с нейронными сетями, цель обучения заключается в максимально точном запоминании и восстановлении поверхности. Для оценки качества обучения используем мгновенный функционал $E(w)$.

Для обучения используем нейронную сеть с двумя входами, 15-ю или 20-ю (в зависимости от эксперимента) нейронами и одним выходом. Обучающую выборку построим на основе предложенной для каждой из функции области определения, спроецировав сетки 200×200 и 400×400, тем самым получив $\left\langle \tilde{X},\tilde{Y}\right\rangle_1=\left\{(x_i,y_i),z_i\right\}_{i=\overline{1,40000}}$ и $\left\langle \tilde{X},\tilde{Y}\right\rangle_2=\left\{(x_i,y_i),z_i\right\}_{i=\overline{1,160000}}$. Максимальное количество итераций ограничим 400.

Результаты вычислений представлены в табл. 1.

Анализ результатов вычислительного эксперимента позволяет сделать следующие выводы:

1) при незначительном увеличении количества нейронов в скрытом слое значительно увеличивается время вычислений;

2) время вычислений увеличивается пропорционально объёму выборки;

3) при фиксированном ограничении на количество шагов и увеличении нейронов в скрытом слое отмечается рост $E(w)$. При увеличении количества настраиваемых параметров (в данном случае это элементы вектора w) требуется больше шагов для их корректировки;

4) с помощью метода Бройдена удалось примерно в 1.82–1.96 раза (в зависимости от эксперимента) снизить время вычислений относительно варианта с использованием центральной разностной производной, однако видно, что $E(w)$ увеличилась;

5) с помощью прямых вычислений удалось примерно в 11.5–14 раз (в зависимости от эксперимента) снизить время вычислений относительно варианта с использованием центральной разностной производной. При малом размере обучающей выборки прямые вычисления чаще показывают минимальное $E(w)$ из всех трёх подходов;

6) параллельные варианты методов в среднем работают в примерно в 6.5–7 раз быстрее в зависимости от эксперимента;

7) $E(w)$ почти не изменяется при изменении количества потоков косвенно даёт по-

Таблица 1

Результаты замеров

Функция	Центр. разност.		Бройден		Прямое выч.	
	t, сек.	$E(w)$	t, сек.	$E(w)$	t, сек.	$E(w)$
1 поток, 15 нейронов в срытом слое, $N = 40000$						
Бирда	1213.84	40.169850	648.92	42.017745	98.89	39.267781
Букина	1217.04	5.737951	649.99	8.693253	99.08	5.738459
Волновая	1218.87	534.663815	648.45	535.959216	99.73	527.876536
МакКормика	1219.46	0.059028	649.10	0.063145	98.75	0.059026
Розенброка	1220.01	0.238130	647.77	0.285570	99.45	0.209310
Шестигорб.	1224.55	1.677874	649.26	1.593291	98.88	1.672548
8 поток, 15 нейронов в срытом слое, $N = 40000$						
Бирда	179.49	40.169850	94.19	42.017745	14.68	39.267781
Букина	174.03	5.737951	94.19	8.693253	14.72	5.738459
Волновая	175.32	534.663455	95.03	535.959214	14.82	527.876536
МакКормика	173.96	0.059028	93.74	0.063145	14.89	0.059026
Розенброка	173.86	0.238130	93.98	0.285570	14.72	0.209310
Шестигорб.	173.82	1.677874	95.32	1.593291	14.80	1.672548
1 поток, 20 нейронов в срытом слое, $N = 40000$						
Бирда	2070.48	40.094523	1088.57	49.598327	145.30	37.575327
Букина	2073.59	4.885433	1089.97	8.342167	145.67	4.726689
Волновая	2060.71	504.997225	1089.63	534.877796	146.12	524.204555
МакКормика	2065.83	0.063091	1089.16	0.051283	145.84	0.062646
Розенброка	2068.28	0.168774	1085.86	0.218997	145.73	0.168326
Шестигорб.	2067.05	1.544811	1086.49	1.968521	145.08	1.017207
8 поток, 20 нейронов в срытом слое, $N = 40000$						
Бирда	310.39	40.094513	163.13	49.598262	21.80	37.575330
Букина	305.15	4.885433	158.82	8.342167	21.71	4.726690
Волновая	310.23	508.322556	158.66	534.877796	21.80	524.204558
МакКормика	310.55	0.063091	158.84	0.051283	21.78	0.062646
Розенброка	311.42	0.168774	158.59	0.218997	21.70	0.168326
Шестигорб.	310.27	1.544811	158.79	1.968521	21.73	1.017207
1 поток, 20 нейронов в срытом слое, $N = 160000$						
Бирда	8286.56	152.256003	4352.31	169.806005	606.17	154.611018
Букина	8278.27	22.328333	4343.18	26.046442	579.60	22.536436
Волновая	8271.45	1895.123088	4352.49	2099.547681	581.21	1936.021446
МакКормика	8272.26	0.198018	4339.09	0.132353	583.32	0.456437
Розенброка	8262.71	0.462430	4357.74	0.799574	584.53	0.802374
Шестигорб.	8262.71	4.282578	4350.97	6.536533	584.36	5.834115
8 поток, 20 нейронов в срытом слое, $N = 160000$						
Бирда	1192.92	152.255916	630.11	169.806000	86.24	154.611018
Букина	1205.72	22.328333	630.05	26.046441	86.42	22.536436
Волновая	1220.88	1895.122097	627.98	2099.551337	87.76	1936.021463
МакКормика	1224.04	0.198017	629.52	0.132353	86.79	0.456437
Розенброка	1220.76	0.462430	628.46	0.799573	89.49	0.802374
Шестигорб.	1215.28	4.282578	629.29	6.536533	86.41	5.834115

нять, что параллельная реализация реализована без ошибок;

8) максимально достигнутое ускорение составляет примерно 82–96 раз в зависимости от эксперимента.

Благодаря параллельному программирования и применение дополнительных математических методов удалось ускорить работу метода Левенберга-Марквардта более чем в $10 \cdot \tau$ раз. В зависимости от решаемых задач, пользу от ускорения можно выразить как в сокращении затраченного времени на вычисления, так и в увеличении потенциально допустимого объёма обрабатываемых данных.

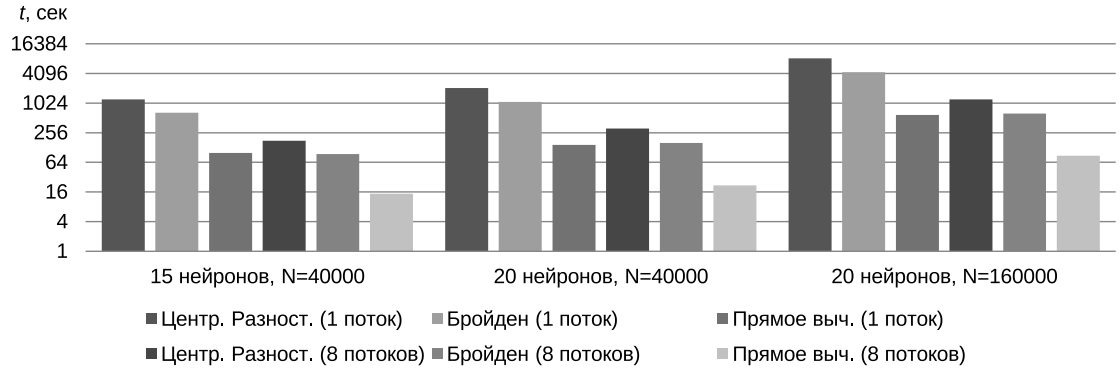

Рис. 1. Среднее время выполнения экспериментов

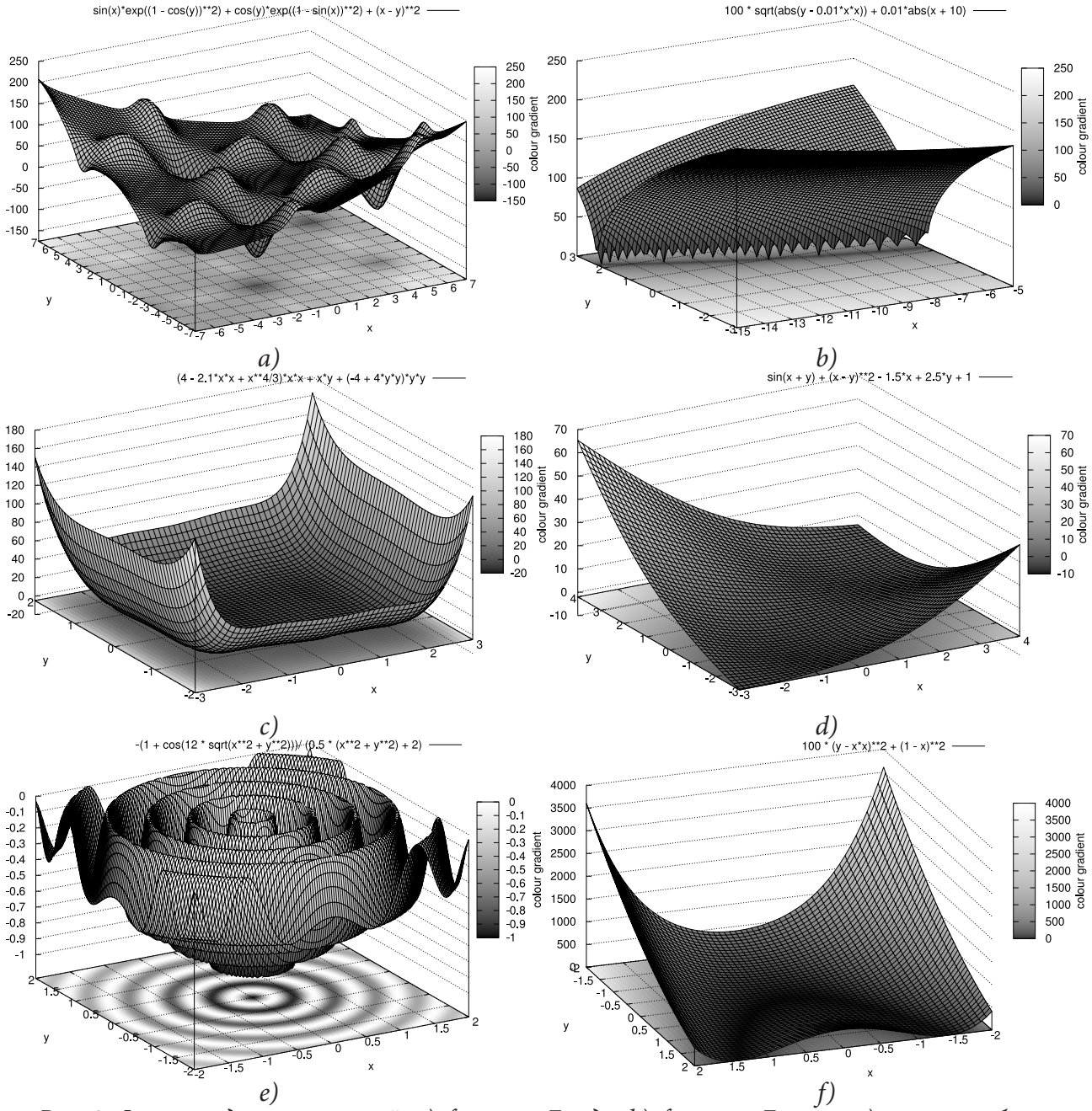

Рис. 2. Функции для испытаний: a) функция Бирда, b) функция Букина, c) шестигорбовая обратная верблюжья функция (на графике самих горбов не видно из-за масштабирования), d) функция МакКормика, e) волновая функция, e) долина Розенброка

СПИСОК ЛИТЕРАТУРЫ

1. *Кохонен Т.* Толковый словарь «нейронных» терминов // Самоорганизующиеся карты. – М. : БИНОМ. Лаборатория знаний, 2013. – С. 532.

2. *Сараев П.В.* Суперпозиционное линейно-нелинейное нейроструктурное моделирование : Дисс... кандидата наук / Павел Викторович Сараев ; Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Липецкий государственный технический университет». – 2012.

3. *Хайкин С.* Преимущества и ограничения обучения методом обратного распространения // Нейронные сети. – М. : Вильямс, 2006. – С. 304–314.

4. *Sousa C.* Neural network learning by the levenberg-marquardt algorithm with Bayesian regularization (part 1). – 2009. – URL: http://crsouza.blogspot.com/2009/11/neural-network-learning-by-levenberg_18.html.

5. *Suratgar A.A., Tavakoli M.B., Hoseinabadi A.* Modified levenberg-marquardt method for neural networks training.

6. *Transtrum M.K., Sethna J.P.* Improvements to the levenberg-marquardt algorithm for nonlinear least-squares minimization // Journal of Computational Physics. – 2012.

7. *Sousa C.* Neural network learning by the levenberg-marquardt algorithm with Bayesian regularization (part 2). – 2009. – URL: http://crsouza.blogspot.com/2009/11/neural-network-learning-by-levenberg.html.

8. *Пархоменко С.С.* О сокращении времени обработки большого количества данных нейронными сетями методом Левенберга-Марквардта // Международный научно-исследовательский журнал / Под ред. А.В. Миллер. – 20. – ООО «Импекс», 2014. – Январь. – С. 80–83.

9. *Madsen K., Nielsen H.B., Tingleff O.* A Secant Version of the L-M Method // Methods for Non-Linear Least Squares Problems. – 2 edition. – Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby : Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2004. – P. 40–45.

10. *Амосов А.А., Дубинский Ю.А., Копченова Н.В.* Простейшие формулы численного дифференцирования // Вычислительные методы для инженеров. – М. : Высшая школа, 1994. – С. 364–369.

11. *Гергель В.П., Лабутина А.А.* Разделение вычислений на независимые части // Учебно-образовательный комплекс по методам параллельного программирования. – Нижний Новгород, 2007. – С. 10–12. – Учебно-методические материалы по программе повышения квалификации «Информационные технологии и компьютерное моделирование в прикладной математике». URL: http://www.unn.ru/pages/e-library/aids/2007/7.pdf

12. Some new test functions for global optimization and performance of repulsive particle swarm method : MPRA Paper / University Library of Munich, Germany ; Executor: Sudhanshu Kumar Mishra : 2006. – URL: http://EconPapers.repec.org/RePEc:pra:mprapa:2718.

13. *Molga M., Smutnicki C.* Test functions for optimization needs. – 2005.

Пархоменко Станислав Сергеевич – аспирант кафедры вычислительной математики и прикладных информационных технологий Воронежского государственного университета. E-mail: ibm8086@yandex.ru

Леденёва Татьяна Михайловна – д.т.н., проф. заведующий кафедрой вычислительной математики и прикладных информационных технологий Воронежского государственного университета. Тел.: +7(473) 228-11-60 (доб. 1470).
E-mail: ledeneva-tm@yandex.ru

ParkhomenkoStanislavSergeevich – postgraduate student of the Department of Computational Mathematics and Applied Information Technology, Voronezh State University.
E-mail: ibm8086@yandex.ru

Ledeneva Tatyna Michaylovna – Doctor of Technic Sciences, Professor, Head of the Department of Computational Mathematics and Applied Information Technology, Voronezh State University. Phone: +7(473) 228-11-60 (1470).
E-mail: ledeneva-tm@yandex.ru.