УДК 519.71

НЕЙРОСЕТЕВАЯ АППРОКСИМАЦИЯ МНОГОМЕРНЫХ ФУНКЦИЙ ПРИ НАЛИЧИИ ПОМЕХ ИЗМЕРЕНИЙ

О. Г. Руденко[1], А. В. Островерхий[2]

[1] ХНУРЭ, г. Харьков, Украина, rudenko@kture.kharkov.ua

[2] ХНУРЭ, г. Харьков, Украина, aostrich@gmail.com

В статье приводится сравнительный анализ восстановления зашумленных многомерных функций нейронными сетями CMAC различной архитектуры и РБС. Рассматривается проблема выбора базисных функций сети CMAC и параметров РБС. Показано, что для решения данной задачи весьма эффективными являются иерархическая и линейная архитектуры CMAC при выборе в качестве базисной параболической функции, а применение РБС обеспечивает заданную точность восстановления, требуя меньшего объема памяти, но значительно больших вычислительных затрат.

АППРОКСИМАЦИЯ, БАЗИСНЫЕ ФУНКЦИИ, ВОССТАНОВЛЕНИЕ, НЕЙРОННАЯ СЕТЬ CMAC, МНОГОМЕРНАЯ ФУНКЦИЯ, РАДИАЛЬНО-БАЗИСНАЯ СЕТЬ.

Введение

Решение широкого круга задач науки, техники, экономики, например, идентификация, фильтрация, сглаживание, прогнозирование и т. д. связано с аппроксимацией некоторых нелинейных функций вида

$$y(k)=f[x(k)]+\xi(k), \qquad (1)$$

где $f[\bullet]$ — непрерывная нелинейная функция; $\xi(k)$ — помеха измерения; $k=1, 2, \ldots$ — дискретное время.

Отсутствие информации о виде функции $f[\bullet]$ зачастую делает традиционные методы аппроксимации неэффективными, а в ряде случаев — неприменимыми. Альтернативой традиционным методам являются нейросетевые технологии.

Являясь универсальными аппроксиматорами, некоторые типы искусственных нейронных сетей (ИНС) позволяют восстановить с любой заданной точностью любую сколь угодно сложную непрерывную нелинейную функцию. Наибольшее распространение при решении данной задачи получили многослойный персептрон (МП) [1], радиально-базисные сети (РБС) [2] и сеть CMAC (Cerebellar Model Articulation Controller) [3]. Все эти сети используют представление нелинейного оператора некоторой системой базисных функций, реализуемой нейронами, сводя задачу аппроксимации к обучению сети, т.е. настройке параметров нейронов на основе предъявления обучающих пар. Такими обучающими парами служат значения аргументов $x(k)$ и функции $y(k)$.

Увеличение размерности решаемой задачи существенно усложняет задачу аппроксимации, а наличие в измерениях помех не только приводит к увеличению времени обучения ИНС, но и выдвигает определенные требования к используемому алгоритму обучения.

Обучение МП, содержащего несколько (чаще всего не больше двух) скрытых слоев, осуществляется обычно с помощью алгоритма обратного распространения ошибки (ОРО), реализация которого связана с существенными вычислительными трудностями. Поэтому более предпочтительными представляются РБС, содержащие лишь один скрытый слой нейронов, для обучения которых применяют рекуррентный метод наименьших квадратов или фильтр Калмана, и сеть CMAC, не имеющая скрытых слоев и использующая простой, но обеспечивающий высокую скорость обучения алгоритм настройки параметров.

Целью данной работы является сравнительный анализ сетей CMAC и РБС в задаче аппроксимации нелинейных функций большой размерности при наличии помех измерений.

1. Сеть CMAC

Сеть CMAC в общем случае осуществляет следующие преобразования:

$$S: X \Rightarrow A, \qquad (2)$$

$$P: A \Rightarrow y, \qquad (3)$$

где X — N-мерное пространство непрерывных входных сигналов; A — n-мерное пространство ассоциаций; y — вектор выходных сигналов.

Преобразование (2) соответствует кодированию информации

$$a=S(x), \qquad (4)$$

а (3) — вычислению выходного сигнала.

Входной слой сети состоит из нейронов, имеющих, как правило, одинаковые базисные (активационные) функции (БФ).

Традиционная сеть CMAC использует БФ прямоугольной формы, однако в некоторых задачах такой выбор является неэффективным, а в ряде слу-

чаев и неприемлемым. Если в сети используются нейроны с БФ, отличными от прямоугольных, то преобразование, осуществляемое сетью CMAC, принимает вид:

$$\hat{y} = a^T \Phi(x) w , \qquad (5)$$

где $\Phi(x)$ — диагональная матрица с элементами $\Phi_i(x) = \prod_{j=1}^{N} \phi_{ij}(x_j)$, $\phi_{ij}(x_j)$ — значение выбранной базисной функции в точке x_j.

Одним из наиболее простых и наиболее эффективных является выбор в качестве БФ тригонометрических функций, например, косинусоидальной [4]:

$$\Phi(x) = \begin{cases} \cos\left(\pi \dfrac{x-m}{\lambda}\right); \text{ при } x \in \left[m - \dfrac{\lambda}{2}; m + \dfrac{\lambda}{2}\right] \\ 0; \text{ в противном случае} \end{cases} \qquad (6)$$

и параболической [5]:

$$\Phi(x) = \begin{cases} 1 - \left(2 \cdot \dfrac{x-m}{\lambda}\right)^2; \text{ при } x \in \left[m - \dfrac{\lambda}{2}; m + \dfrac{\lambda}{2}\right] \\ 0; \text{ в противном случае,} \end{cases} \qquad (7)$$

которая не только имеет форму, близкую к тригонометрическим, но и требует значительно меньших вычислительных затрат при ее реализации. В (6), (7) приняты обозначения: m — центр гиперкуба, λ — длина гиперкуба.

Обучение сети CMAC, как практически и всех других ИНС, заключается в настройке вектора ее весовых параметров w размерности $n \times 1$.

При выборе БФ, отличных от прямоугольных, алгоритм обучения имеет вид:

$$w(k+1) = w(k) + \gamma(k) \times \left(\frac{y(k) - a^T(k)\Phi(x)w(k)}{\left\| \Phi(x)a(k) \right\|^2} \Phi(x)a(k) \right), \qquad (8)$$

где $\gamma(k)$ — некоторый в общем случае переменный параметр.

Свойства алгоритма (8) в значительной степени зависят от выбора $\gamma(k)$. Несложно показать, что оптимальное значение этого параметра, обеспечивающее максимальную скорость обучения при отсутствии помех ξ, будет равно единице. Для обеспечения же сходимости алгоритма (8) при наличии помех измерений параметр $\gamma(k)$ должен удовлетворять условиям Дворецкого.

Следует, однако, отметить, что с ростом размерности пространства входных переменных N объем памяти, требуемый для хранения информации о весах сети CMAC, растет экспоненциально. Кроме того, ограниченные размеры памяти сужают сферу применения данной ИНС в реальных приложениях, а увеличение размерности N приводит к возрастанию сложности кодирования информации.

2. Модифицированные архитектуры CMAC

В качестве эффективного способа уменьшения объема требуемой памяти в сети CMAC при работе с многомерными объектами в [8] предлагается построение иерархической структуры CMAC (Hierarchical CMAC — HCMAC), состоящей из нескольких более простых модулей, например, двумерных CMAC. На рис. 1 приведена топология HCMAC, учитывающая, что каждая CMAC содержит два входа, и выходной сигнал CMAC первого слоя является входным сигналом для CMAC второго слоя и т. д. Архитектура HCMAC может быть, соответственно, расширена с использованием данной топологии бинарного дерева.

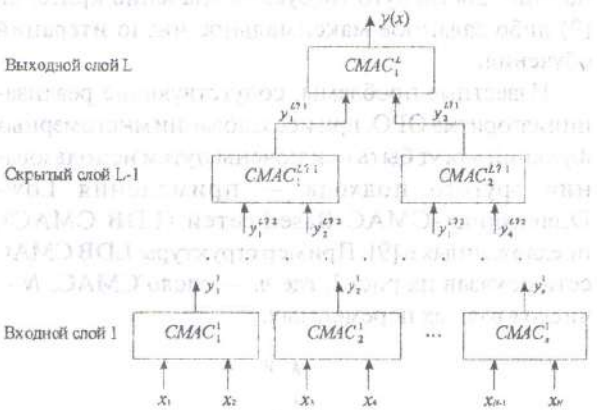

Рис. 1. Топологическая структура нейронной сети HCMAC

На рис. 1 использованы следующие обозначения: x_i ($i = 1, 2, ..., N$) — i-ый вход нейронной сети HCMAC; y_j^l ($j = 1, 2, ..., n$; $l = 1, 2, ..., L$) — выход j-ого CMAC слоя l; $y(x)$ — выходной сигнал HCMAC для входного сигнала x.

При выборе дифференцируемых БФ [4–7] для обучения данной сети (настройки её параметров) может быть применён алгоритм ОРО. Если в качестве минимизируемого выбран квадратичный функционал ошибки

$$E = \frac{1}{2}\left(\hat{y}(x) - y(x)\right)^2 , \qquad (9)$$

где $y(x)$, $\hat{y}(x)$ — требуемый и реальный выходные сигналы HCMAC для входного сигнала x, соответственно, то обобщенная процедура обучения сети может быть представлена так.

1) *Настройка CMAC выходного слоя* (практически ничем не отличается от настройки обычной двухвходовой CMAC, например, по алгоритму (8), за исключением того, что входными для данного слоя являются выходные сигналы предыдущего слоя).

2) *Настройка сетей CMAC скрытых слоев* по правилу:

$$w_h^l(k+1) = w_h^l(k) + \gamma \cdot \left(\hat{y}_{\lfloor h/2 \rfloor}^{l+1}(y^l) - y_{\lfloor h/2 \rfloor}^{l+1}(y^l)\right) \times \times \frac{\partial y_{\lfloor h/2 \rfloor}^{l+1}(x)}{\partial y_h^l} \cdot \Phi(y^l) a(k); \quad (10)$$

$$\frac{\partial y_{\lfloor h/2 \rfloor}^{l+1}(x)}{\partial y_h^l} = \sum_{i=1}^{\rho} a_i(y^l) \cdot w_{hi}^l \cdot \left[\prod_{j=2\lfloor h/2 \rfloor}^{2\lfloor h/2 \rfloor+1} \Phi_i(y_j^l)\right] \cdot \frac{\partial \Phi_i(y_h^l)}{\partial y_h^l}, \quad (11)$$

где $h = 1, 2, ..., n$; w_{hi}^l — значение веса в i-ой ячейке памяти CMAC_h слоя l; $]\bullet[$ — означает округление в сторону ближайшего большего целого числа.

Данная процедура повторяется до тех пор, пока не будет достигнуто требуемое значение критерия (9) либо заданное максимальное число итераций обучения.

Известные проблемы, сопутствующие реализации алгоритма ОРО, при исследовании многомерных функций могут быть исключены путем использования другого подхода — применения Low-Dimensional-CMAC-Based сетей (LDB CMAC), предложенных в [9]. Пример структуры LDB CMAC сети показан на рис. 2, где m — число CMAC, N — число входных переменных.

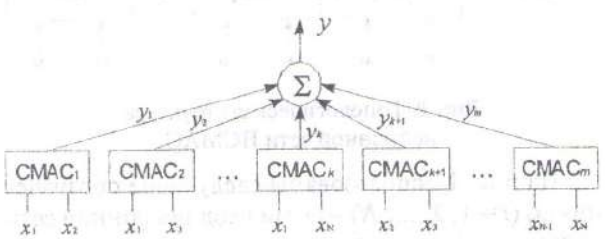

Рис. 2. Структура сети LDB CMAC

Данную структуру образуют множество малоразмерных (базовых) сетей CMAC (например, двумерных), на которые подаются все возможные парные комбинации входных сигналов. Взвешенные весами z_i ($i = 1, 2, ..., m$) выходы этих CMAC формируют общий выход сети

$$y(x) = \sum_{i=1}^{m} y_i(x) z_i, \quad (12)$$

где $y_i(x)$ — выходное значение i-ой сети CMAC; z_i — вес соответствующего выхода $y_i(x)$.

Данная архитектура позволяет создавать также и неполные структуры, учитывающие не все возможные комбинации пар входных переменных, а лишь часть их.

Процедура обучения сети LDB CMAC состоит в следующем:

1) *Настройка весов* z_i, например, по алгоритму

$$z(k+1) = z(k) + \frac{y(x) - \hat{y}(x)}{\|Y(k)\|^2} Y(k), \quad (13)$$

где $Y(k) = (y_1(k), y_2(k), ..., y_m(k))$ — вектор выходов базовых сетей CMAC.

2) *Настройка параметров базовых сетей* CMAC_i по правилу (8).

Следует отметить, что данная структура позволяет применять градиентный метод обучения при выборе БФ любой формы, включая прямоугольную.

3. Радиально-базисная сеть

Аппроксимация нелинейной функции (1) радиально-базисными функциями

$$\Phi_i(x) = \exp\left\{-\frac{\|x - \mu_i\|^2}{\sigma_i^2}\right\}, \quad (14)$$

где μ_i, y_i — центры и радиусы базисных функций, соответственно; $\|\bullet\|$ — евклидова норма; $\Phi_0(x) = 1$; приводит к нейросетевой модели

$$\hat{y}(k) = \sum_{i=0}^{N} c_i \Phi_i(x), \quad (15)$$

где c_i — весовые коэффициенты.

Изменение структуры сети осуществляется ее постепенным усложнением с добавлением новых нейронов, проводимым каждый раз, когда при появлении очередного входного сигнала возникает ошибка аппроксимации $e = y - \hat{y}$, превышающая допустимую. В этом случае, если в l-й момент времени сеть содержала N нейронов, а появление сигнала $x(l)$ привело к появлению ошибки на i-м выходе $e_i(l) > e_{i\,\text{доп}} = \alpha_i$, в сеть вводится новый, $(N+1)$-й, нейрон, а центр базисной функции, её вес и радиус принимаются равными, соответственно, $\mu_{N+1} = x(l)$, $c_{N+1} = e(l)$, $\sigma_{N+1} = \|x(l) - \mu_m(l)\|$, где $\mu_m(l)$ — центр базисной функции для l-го входа сигнала. Таким образом, условием введения нового нейрона является выполнение неравенств

$$e_i(l) > \alpha_i, \quad (16)$$

$$\|x(l) - \mu_m(l)\| > \beta, \quad (17)$$

где α_i и β — априорно устанавливаемые предельно допустимые значения ошибки реакции сети и отклонения обобщенного сигнала $x(l)$ от ближайшего к данному входу центра.

Обучение сети состоит в определении её параметров μ_i, σ_i и c_i и сводится к минимизации обычно квадратичного функционала (9).

В настоящее время существует множество методов настройки параметров сети, среди которых достаточно широко используется рекуррентный алгоритм метода наименьших квадратов (РМНК) с экспоненциальным взвешиванием информации, согласно которому вектор оценок настраиваемых параметров

$$w(k) = (c_0^T(k), c_1^T(k), \mu_1^T(k), \sigma_1(k), \ldots, c_N^T(k), \mu_N^T(k), \sigma_N(k)) \quad (18)$$

корректируется следующим образом:

$$w(k) = w(k-1) + K(k)e(k);$$

$$K(k) = P(k-1)\nabla_w \hat{y}(k) \times \left[\lambda I + \nabla_w^T \hat{y}(k) P(k-1) \nabla_w \hat{y}(k)\right]^{-1}; \quad (19)$$

$$P(k) = \lambda^{-1}\left(P(k-1) + K(k)\nabla_w^T \hat{y}(k) P(k-1)\right),$$

где

$$\nabla_w \hat{y}(k) = [I, \Phi_1(x(k)), \Phi_1(x(k))\frac{2c_1}{\sigma_1^2}(x(k)-\mu_1)^T,$$

$$\Phi_1(x(k))\frac{2c_1}{\sigma_1^3}\|x(k)-\mu_1\|^2, \ldots, \Phi_N(x(k)),$$

$$\Phi_N(x(k))\frac{2c_N}{\sigma_N^2}(x(k)-\mu_N)^T,$$

$$\Phi_N(x(k))\frac{2c_N}{\sigma_N^3}\|x(k)-\mu_N\|^2]^T;$$

I — единичная матрица; $\lambda \in [0, 1]$.

Несмотря на огромное число работ, в которых используется алгоритм (19), общих рекомендаций по выбору оптимального значения коэффициента λ в настоящее время, к сожалению, не существует, поэтому при решении практических задач чаще всего ограничиваются выбором $\lambda = 0{,}995 \div 0{,}999$.

4. Экспериментальные исследования

Сравнительный анализ аппроксимирующих свойств модифицированных CMAC и РБС производился на ПК с процессором Intel Pentium 4 3.2 GHz в среде MatLab 7.0 под ОС Linux 2.6.20.

Для оценки эффективности работы сетей использовались:

1. Время вычислений в секундах — T;
2. Средняя квадратичная ошибка — MSE.

$$MSE = \frac{\sqrt{\sum_{i=1}^{M}(y_i(x) - \hat{y}_i(x))^2}}{M},$$

где M — количество экспериментов.

Рассматривалась задача восстановления зашумленной нелинейной четырехмерной функции $F(x_1, x_2, x_3, x_4)$:

$$F(x_1, x_2, x_3, x_4) = x_1 + \sin(\pi x_1)\cdot\cos(\pi x_2)\times \times \sin(\pi x_3)\cdot(\sin(\pi x_4)^2 - 1). \quad (20)$$

Сечение данной функции при $x_2 = x_3 = 0{,}25$, т. е. $F(x_1, 0{,}25, 0{,}25, x_4)$, показано на рис. 3.

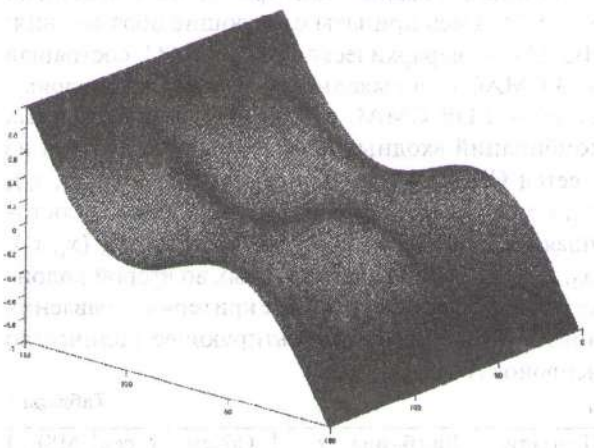

Рис. 3. Вид исходной функции $F(x_1, 0{,}25, 0{,}25, x_4)$

На данную функцию был наложен равномерно распределенный шум с амплитудой 0,1, в результате чего она приобрела вид, приведенный на рис. 4.

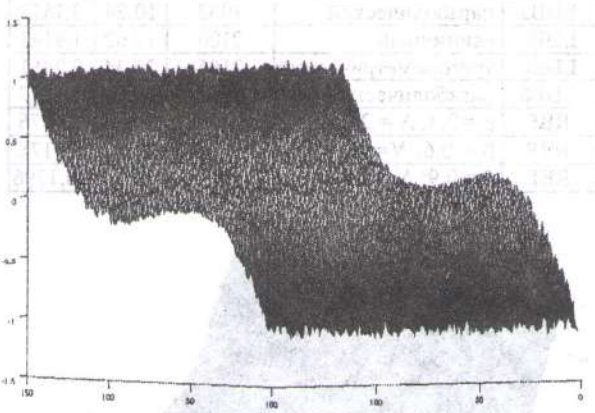

Рис. 4. Вид зашумленной функции $F(x_1, 0{,}25, 0{,}25, x_4)$

Обучение сетей осуществлялось на выборке из 20000 случайных точек, равномерно распределенных в интервале [–1; 1]. Целью эксперимента было исследование влияния различных характеристик сетей на их восстанавливающие свойства. В качестве моделируемой использовалась окрестность зашумленной функции $F(x_1, 0{,}25, 0{,}25, x_4)$ $F(x_1, 0 \ldots 0{,}5, 0 \ldots 0{,}5, x_4)$.

Все исследуемые базовые сети CMAC использовали $R = 50$ (уровней квантования), распределен-

ных по $\rho = 10$ ступеням квантования. Базовая двумерная сеть CMAC с такими характеристиками требует 350 ячеек памяти. Для восстановления данной функции обычной четырехмерной CMAC потребовалось бы 12119 ячеек памяти.

При исследовании РБС базисные функции выбирались гауссовскими (14), параметры сети настраивались по алгоритму (19) с $\lambda = 0{,}99$. Критерии введения нового нейрона имели вид (16), (17) с $\alpha_i = 0{,}01$ и различными значениями $\beta = 0{,}3;\ 0{,}6;\ 0{,}9$.

Результаты вычислений приведены в табл. 1 и на рис. 5–9. Здесь приняты следующие обозначения: HCMAC — иерархическая сеть CMAC, состоящая из 3 CMAC (1 в выходном и 2 в скрытом слоях); LDB6 — LDB CMAC с полным набором парных комбинаций входных переменных, состоящая из 6 сетей CMAC с входами (x_1, x_2), (x_1, x_3), (x_1, x_4), (x_2, x_3), (x_2, x_4), (x_3, x_4); LDB3 — LDB CMAC, состоящая из 3 сетей CMAC с входами (x_1, x_2), (x_1, x_4), (x_3, x_4); RBF — РБС, для которых во второй колонке указано заданное значение критерия добавления нового нейрона β и результирующее количество нейронов N.

Таблица 1

Тип сети	Вид базисных функций	Объем памяти, ячеек	T, сек	MSE, $*10^{-3}$
HCMAC	тригонометрический	1050	9,88	2,4034
HCMAC	параболический	1050	9,42	2,5923
LDB3	единичный	1053	10,12	4,8339
LDB3	тригонометрический	1053	10,43	3,9295
LDB3	параболический	1053	10,24	3,1513
LDB6	единичный	2106	19,62	1,9143
LDB6	тригонометрический	2106	20,34	2,2461
LDB6	параболический	2106	20,11	2,1517
RBF	$\beta = 0{,}3;\ N = 221$	1326	368,48	2,4376
RBF	$\beta = 0{,}6;\ N = 46$	276	85,65	7,0176
RBF	$\beta = 0{,}9;\ N = 16$	96	45,94	10,1796

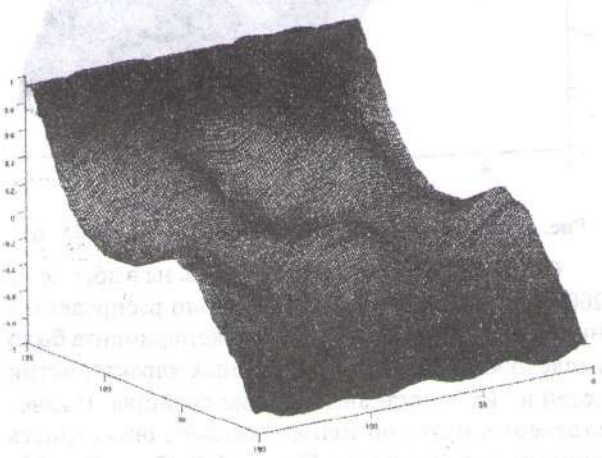

Рис. 5. Восстановление HCMAC с тригонометрическими БФ

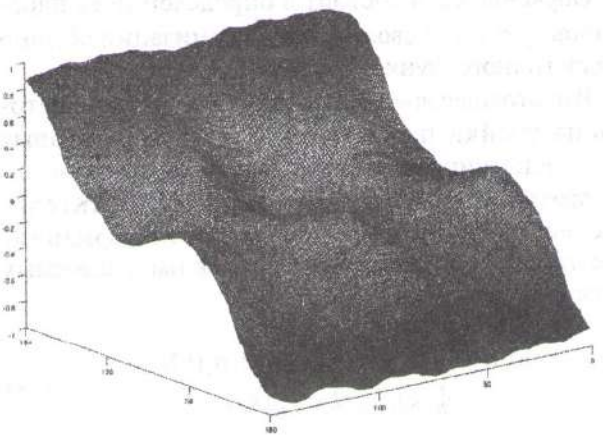

Рис. 6. Восстановление LDB3 CMAC с параболическими БФ

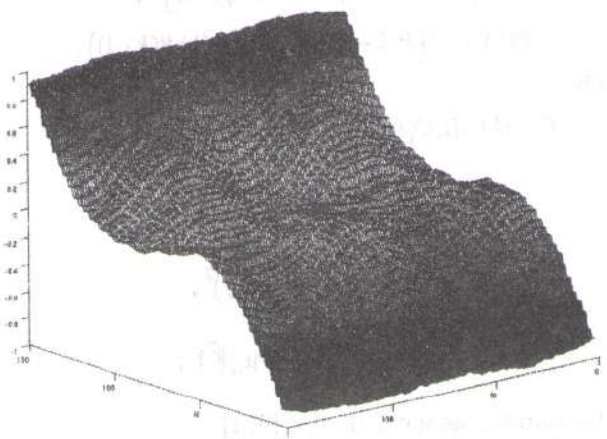

Рис. 7. Восстановление LDB6 CMAC с единичными БФ

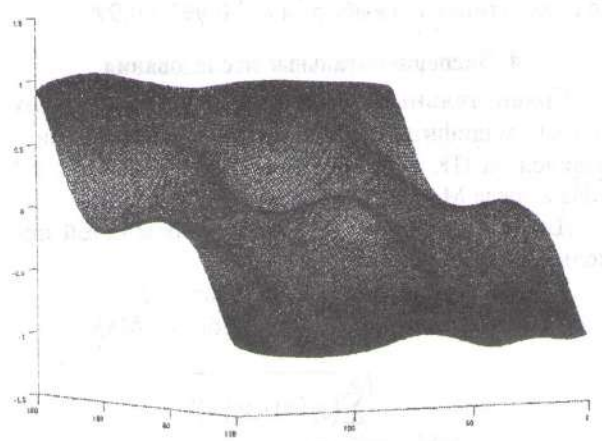

Рис. 8. Восстановление РБС с $\beta = 0{,}3$ ($N = 221$)

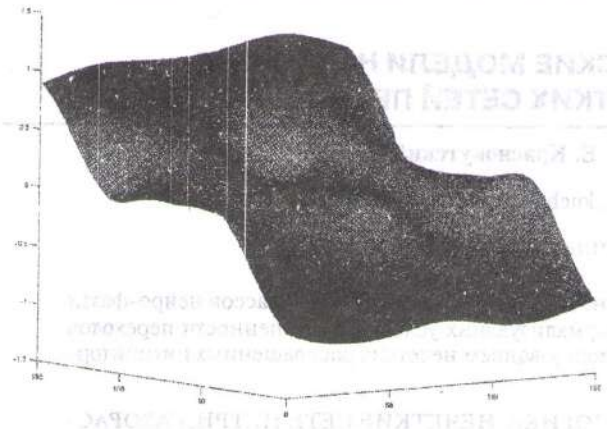

Рис. 9. Восстановление РБС с $\beta = 0{,}6$ ($N = 46$)

Выводы

Результаты исследований свидетельствуют о том, что восстановление многомерных функций может быть достаточно эффективно осуществлено путем применения сетей CMAC модифицированной архитектуры (иерархической HCMAC или линейной LDB CMAC, образованной сетями малой размерности) либо РБС.

В случае применения HCMAC используемые БФ обязательно должны быть дифференцируемыми, в то время как LDB CMAC позволяет применять произвольные БФ. Среди дифференцируемых наиболее простой и в то же время весьма эффективной является параболическая функция. Однако во всех случаях удается существенно сократить объем требуемой памяти сети CMAC и добиться приемлемого качества восстановления многомерной нелинейной функции. Для рассмотренного примера объем требуемой CMAC памяти удалось сократить примерно в 5–11 раз.

Применение РБС обеспечивает заданную точность восстановления, требуя меньшего объема памяти, но значительно больших вычислительных затрат. Повышение точности восстановления, достигаемое путем введения новых нейронов, сопровождается резким возрастанием времени обучения.

Список литературы: 1. *Хайкин С.* Нейронные сети: полный курс, 2-е издание. — М.: Изд. дом «Вильямс», 2006. — 1104 с. 2. *Руденко О. Г., Бодянский Е. В.* Искусственные нейронные сети. — Харьков: Компания «СМИТ», 2005. — 408 с. 3. *Albus J. S.* A new approach to manipulator control: the cerebellar model articulation controller (CMAC) // ASME Trans., J. Dynamic Systems, Measurement and Control. — 1975. — 97. — №3. — P. 220–227. 4. *Руденко О. Г., Бессонов А. А.* О выборе базисных функций в нейронной сети CMAC // Проблемы управления и информатики. — 2004. — № 2. — С. 143–155. 5. *Руденко О. Г., Островерхий А. В., Островерхая Н. Н.* Аппроксимация многомерных функций с помощью нейронной сети CMAC // Бионика интеллекта: Научн.-техн. журнал. — 2006. — № 2 (65). — С. 8–13. 6. *Chiang Ch.-T., Lin Ch.-Sh.* CMAC with General Basis Functions // Neural Networks. — 1996. — 9. — №7. — P. 1199–1211. 7. *Lane S.H., Handelman D.A., Gelfand J.J.* Theory and development of higher-order CMAC neural networks // IEEE Control Systems. — 1992. — 12. — №2. — P. 23–30. 8. *Lee H.-M., Chen Ch.-M., Lu Yu.-F.* A Self-organizing HCMAC Neural Network Classifier // IEEE Transactions on Neural Networks. — 2003. — 14. — № 1. — P. 15–26. 9. *Lin Ch.-Sh., Li Ch.-K.* A Low-Dimensional-CMAC-Based Neural Network // IEEE Int. Conf. on Neural Networks. — 1996. — P. 1297–1302.

Поступила в редколлегию 28.02.07