УДК 519.71

А.А. Бессонов, А.В. Островерхий, А.А. Шамраев, Н.Н. Островерхая

Харьковский национальный университет радиоэлектроники, Харьков

О НЕЙРОСЕТЕВОМ ПОДХОДЕ К ВОССТАНОВЛЕНИЮ МНОГОМЕРНЫХ ФУНКЦИЙ ПРИ НАЛИЧИИ ПОМЕХ ИЗМЕРЕНИЙ

В статье приводится сравнительный анализ восстановления зашумленных многомерных функций нейронными сетями CMAC, РБС и МП различных архитектур. Рассматривается проблема выбора параметров этих сетей, а также базисных функций сети CMAC. Показано, что наименьшего времени обучения позволяют добиться сети CMAC, а применение РБС и МП обеспечивает заданную точность восстановления, требуя меньшего объема памяти, но значительно больших вычислительных затрат.

аппроксимация, базисные функции, восстановление, нейронная сеть CMAC, многомерная функция, многослойный персептрон, радиально-базисная сеть, фильтрация

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 требует довольно существенных вычислительных затрат, поэтому в данной статье исследуется применение МП, содержащих только один скрытый слой. Кроме того, весьма предпочтительными представляются РБС, которые также содержат лишь один скрытый слой нейронов, для обучения которых применяют рекуррентный метод наименьших квадратов или фильтр Кальмана, и сети CMAC, не имеющие скрытых слоев и использующие простой, но обеспечивающий высокую скорость обучения, алгоритм настройки параметров.

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

1. Сеть CMAC

Сеть CMAC в общем случае осуществляет следующие преобразования:

$$S : X \Rightarrow A ; \quad (2)$$

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

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

Преобразование (2) соответствует кодированию информации

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

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

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

где $\Phi(x)$ – диагональная матрица с элементами $\Phi_i(x) = \prod_{j=1}^{N} \varphi_{ij}(x_j)$, а $\varphi_{ij}(x_j)$ – 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]

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

и параболической [5]

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

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 × 1.

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

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

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.

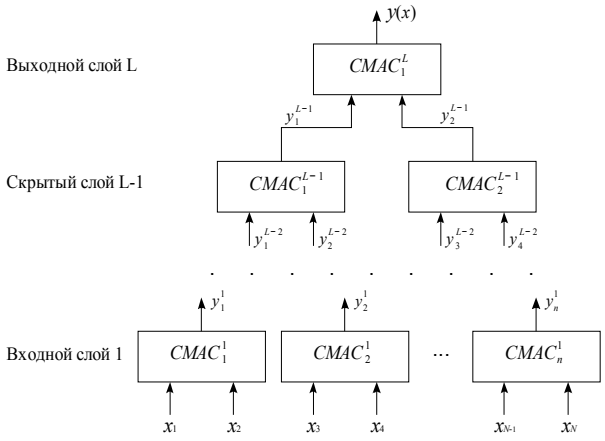

Рис. 1. Топологическая структура нейронной сети HCMAC

На рис. 1 использованы следующие обозначения: x_i (i = 1,2,...,N) – i-й вход нейронной сети HCMAC; y_j^l (j = 1, 2, ... , n; l = 1, 2, ... , L) – выход j-го 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 НСМАС 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

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

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

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

Данная процедура повторяется, 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.

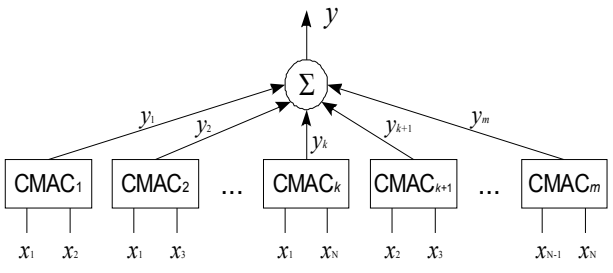

Рис. 2. Структура сети LDB CMAC

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

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

где $y_i(x)$ – выходное значение i-й 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

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

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

2) настройка параметров базовых сетей CMAC 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[10-11]

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

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

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

где c_i – весовые коэффициенты; N – число нейронов. Структурная схема РБС показана на рис. 3.

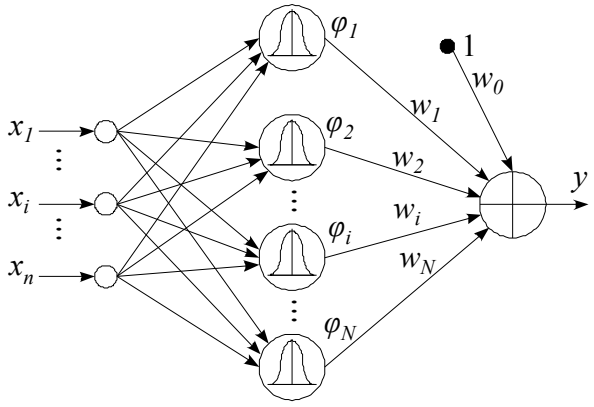

Рис. 3. Структура РБС

Изменение структуры сети осуществляется ее постепенным усложнением с добавлением новых нейронов [12], 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 $e_i(l) > e_{i\ доп} = \alpha_i$, 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

$$e_i(l) > \alpha_1 ; \tag{16}$$

$$\left\|x(l) - \mu_m(l)\right\| > \beta , \tag{17}$$

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

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

корректируется следующим образом:

$$w(k) = w(k-1) + K(k)e(k) ; \tag{19}$$

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

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

где

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

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

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

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

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. Нормализованная радиально-базисная сеть

Нормализованная радиально-базисная сеть [2] (НРБС) – это РБС, где используются нормализованные базисные функции (НБФ), вычисляемые следующим образом:

$$\overline{f_i}(x) = f_i(x) \Big/ \sum_{j=1}^{N} f_j(x) , \tag{20}$$

где N – количество нейронов в сети.

Свойство данной сети, состоящее в том, что для любой точки входного сигнала сумма всех базисных функций равна единице, делает их привлекательными для многих приложений. Как и РБФ, НРБФ являются универсальными аппроксиматорами.

Сеть имеет архитектуру, аналогичную архитектуре радиальной базисной сети общего вида, и отличается от нее структурой второго слоя, в котором вычисляется нормированные значения первого слоя.

Следует отметить, что нормализация (20) существенно влияет на вид базисной функции, изменяя тем самым свойства сети.

На рис. 4 показаны ненормализованные, а на рис. 5 – нормализованные гауссовские функции с параметрами $c_1 = -3$, $c_2 = 0$, $c_3 = 1$, $\sigma_1^2 = \sigma_2^2 = \sigma_3^2 = 1$.

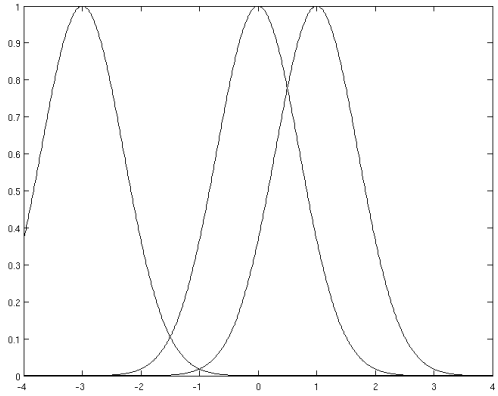

Рис. 4. Ненормализованные гауссовские функции

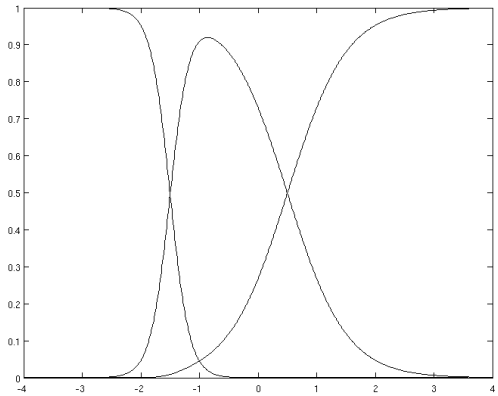

Рис. 5. Нормализованные гауссовские функции

Как видно из приведенных рисунков, нормализация существенно изменяет форму, причем на эту форму оказывает влияние как ширина исходных БФ

(при уменьшении σ_i^2 форма нормализованных функций становится более прямоугольной), так и их взаимное расположение, характеризуемое параметрами c_i.

Как следует из (20), каждая нормализованная БФ является функцией всех исходных БФ. Поэтому изменение вида какой-либо исходной БФ или введение дополнительной БФ (что характерно для начального этапа функционирования РБС, когда определяется ее структура и происходит обучение) приводит к изменению всех нормализованных БФ. Это является весьма неудобным, если предполагается обучение и функционирование сети в режиме on-line.

Кроме того, дополнительные неудобства применения нормализованных БФ обусловлены тем, что при неравномерном распределении центров исходных БФ или при различной их ширине, максимумы НБФ будут опущены по отношению к центрам, а сами НБФ могут стать не монотонно убывающими. Последнее приводит к неправильной реакции нейрона на входные сигналы: так, на сигналы, находящиеся дальше от центра БФ нейрона, реакция его будет более сильной, чем на сигналы, расположенные ближе к центру.

При использовании НРБС в задачах большой размерности необходимо также учитывать, что с ростом размерности задачи увеличивается число используемых НБФ, приводящее не только к указанным выше последствиям, но и к резкому уменьшению значений максимумов НБФ.

Достоинством сети можно считать определенность структуры: сеть фактически вмещает в себя все обучающие данные. С другой стороны, такая структура нейросети и является ее самым большим недостатком: при большом объеме обучающих данных скорость работы сети падает, иногда очень существенно, по причине заметного увеличения сложности архитектуры.

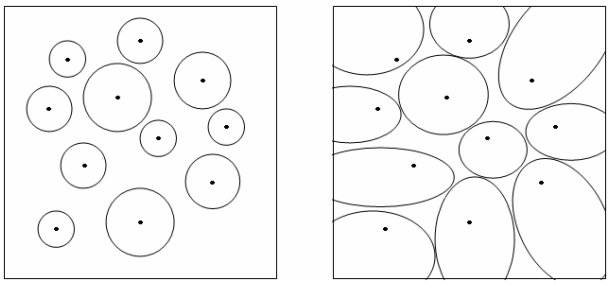

Рис. 6. Покрытие двухмерного пространства РБС и НРБС

Следует отметить, что НРБС являются достаточно удобными и весьма эффективными, если центры БФ равномерно распределены в пространстве входных сигналов, а ширина БФ примерно одинакова. В этом случае в процессе обучения сети ни центры, ни ширина БФ не изменяются, а корректируются лишь ее весовые коэффициенты w. Применение НБФ, в отличие от ненормализованных БФ, обеспечивает в равной степени покрытие всех точек входного пространства, делая сеть менее чувствительной к неудачному выбору центров. Рис. 6 иллюстрирует разные возможности покрытия двумерного пространства входных сигналов сетями РБС и НРБС при одних и тех же заданиях центров БФ.

5. Многослойный персептрон

Хорошие аппроксимирующие свойства МП позволяют использовать эти сети в задачах идентификации нелинейных объектов.

МП состоят из модулей (узлов, нейронов), размещаемых в двух или более слоях (уровнях). Стандартом архитектуры МП являются сети прямого распространения (рис. 7), обучение которых сводится к настройке весовых коэффициентов, характеризующих силу связи между нейронами.

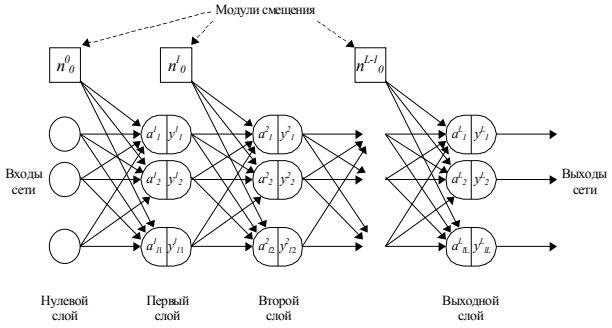

Рис. 7. Архитектура МП

Обучение нейронной сети включает несколько шагов:

- выбор начальной конфигурации и установка начальных значений весов и смещений;
- проведение ряда экспериментов с различными конфигурациями сети и выбор той, которая дает минимальное значение функционала ошибки или, при достижении заданной точности решения, требует меньших вычислительных затрат;
- если качество обучения недостаточно, следует увеличить число нейронов слоя или количество слоев;
- если наблюдается явление переобучения, следует уменьшить число нейронов в слое или удалить один или несколько слоев.

Традиционным методом обучения МП является алгоритм ОРО (backpropagation). Однако во многих применениях он оказался медленно сходящимся и ненадежным.

Для повышения скорости сходимости и надежности алгоритмов обучения МП, в последнее время стали применяться методы, основанные на методах оптимизации второго порядка. К ним относятся методы сопряженного градиента, квазиньютоновские методы, метод Левенберга-Марквардта [13]. Эти методы используют информацию второго порядка, содержащуюся во вторых частных производных

целевой функции по независимым переменным.

В данной работе использовался метод Левенберга-Марквардта. Этот метод был специально разработан для обучения искусственных нейронных сетей, для которых функционал качества обучения определяется как сумма квадратов ошибок [13, 14].

Типичные данные, используемые для обучения искусственных нейронных сетей, представляют собой набор K точек (a_k, b_k) (k = 1, … , K), и значений функции Ф(a, w). Для решения задачи обучения необходимо настроить вектор w, включающий N свободных параметров w_j (j = 1, … , N) так, чтобы функция Ф(a, w) лучше всего аппроксимировала экспериментальные данные.

Согласно методу Левенберга-Марквардта, целевая функция определяется выражением

$$F(w_k) = \frac{1}{2} r(w_k)^T r(w_k), \qquad (21)$$

где $r(w_k)$ (или r_k) является остаточным вектором с элементами

$$r_k = \Phi(a_k, w) - b_k, \quad k = 1,...,K. \qquad (22)$$

В терминах обучения многослойного персептрона (МП), элементы остаточного вектора определяются как

$$r_{i,k} = y^L_{i,k} - t_{i,k}, \quad i = 1,...,N^L, k = 1,...,K, \qquad (23)$$

где $t_{i,k}$ – i-й элемент целевой функции t_k; $y^L_{i,k}$ – выход нейрона n^L_i выходного слоя для образца k; N^L – число выходных узлов.

В этом случае градиент и матрица Гессе для функции (21) имеют особый вид относительно остаточного вектора r и матрицы Якоби J размерностью K × N с элементами $J_{ij} = \frac{\partial r_k}{\partial w_j}$.

В терминах r и J на k-й итерации градиент определяется как

$$g_k = J_k^T r_k, \qquad (24)$$

а матрица Гессе как

$$G_k = J_k^T J_k + s_k;$$
$$s_k = \sum_{i=1}^{M} r_{i,k} \cdot \nabla^2 r_{i,k}, \qquad (25)$$

где $\nabla^2 r_{i,k}$ – вторая производная i-го элемента вектора разности r.

К сожалению уравнения (24) и (25) нельзя взять за основу нелинейного алгоритма, так как слагаемое s_k обычно недоступно. В методе Левенберга-Марквардта s_k игнорируется, так как вблизи решения первое слагаемое в (25) доминирует.

Итерационная формула, на которой основан алгоритм Левенберга-Марквардта, имеет вид

$$w_{k+1} = w_k - [J_k^T J_k + \mu_k I]^{-1} J_k^T r_k, \qquad (26)$$

где I – единичная матрица; μ_k – радиус доверительного интервала.

Таким образом, для обучения МП по методу Левенберга-Марквардта необходима оценка матрицы Якоби J_k на каждой итерации k. В терминах архитектуры МП и набора обучения, матрица Якоби имеет $K \cdot N^L \times W$ элементов вида $\partial r_{b,k} / \partial w^{lm}_{ij}$, которые могут быть рассчитаны согласно выражению

$$\frac{\partial r_{b,k}}{\partial w^{lm}_{ij}} = \delta^l_{i,b,k} \, y^m_{j,k}, \qquad (27)$$

где $\delta^l_{i,b,k}$ рассчитывается при обратном проходе согласно формуле

$$\delta^L_{i,b,k} = f'(a^L_{i,k}), \ i = b;$$
$$\delta^L_{i,b,k} = 0, \ i \neq b; \qquad (28)$$
$$\delta^m_{j,b,k} = f'(a^m_{j,k}) \sum_{n^l_i \in T^m_j} w^{lm}_{ij} \delta^l_{i,b,k}, \ m < L.$$

6. Экспериментальные исследования

Сравнительный анализ аппроксимирующих свойств нейронных сетей CMAC, РБС и МП различных архитектур 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:

$$MSE = \frac{\sqrt{\sum_{i=1}^{M} (y_i(x) - \ _i(x))^2}}{M}, \qquad (29)$$

где M – количество экспериментов.

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

$$F(x_1, x_2, x_3, x_4) = x_1 + \sin(\pi x_1) \cdot \cos(\pi x_2) \times \times \sin(\pi x_3) \cdot (\sin(\pi x_4)^2 - 1). \qquad (30)$$

Сечение данной функции при $x_2 = x_3 = 0.25$ показано на рис. 8.

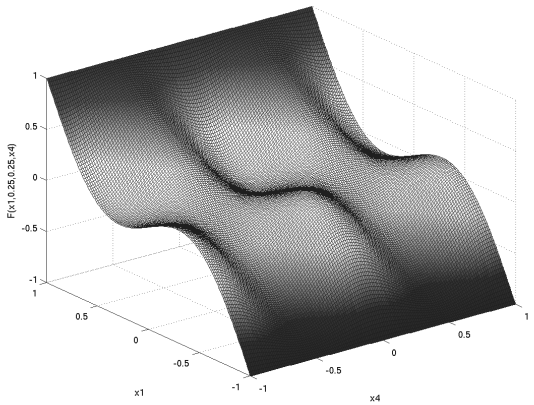

Рис. 8. Вид исходной функции $F(x_1; 0{,}25; 0{,}25; x_4)$

На данную функцию был наложен равномерно распределенный шум с амплитудой 0,1, в результате чего она приобрела вид, приведенный на рис. 9.

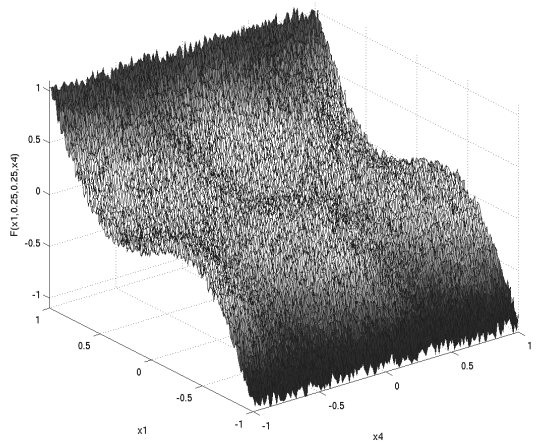

Рис. 9. Вид зашумленной функции F(x1; 0,25; 0,25; x4)

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 $F(x_1; 0{,}25; 0{,}25; x_4)$ $F(x_1; 0...0{,}5; 0...0{,}5; x_4)$.

Все исследуемые базовые сети CMAC использовали R = 50 (уровней квантования), распределенных по ρ = 10 ступеням квантования. Базовая двумерная сеть CMAC с такими характеристиками требует 350 ячеек памяти. Для восстановления данной функции обычной четырехмерной CMAC потребовалось бы 12 119 ячеек памяти.

При исследовании РБС базисные функции выбирались гауссовскими (14), параметры сети настраивались по алгоритму (19) с λ = 0,99. Критерии введения нового нейрона имели вид (16), (17) с α_i = = 0,01 и различными значениями β = 0,3; 0,6; 0,9.

Также рассматривались три архитектуры МП: 10-1; 20-1 и 30-1, в первом слое в качестве функции активации использовался гиперболический тангенс, а в выходном – линейная функция. Начальная инициализация весов и смещений осуществлялась согласно правилу, при котором каждый вес равномерно распределен в диапазоне [–r, r], где r задается формулой

$$r\left(w_{ij}^{lm}\right) = \frac{1}{N^m + 1}. \qquad (31)$$

Обучение сети осуществлялось методом Левенберга-Марквардта. Процесс обучения завершался по достижению заданного уровня точности ε = 0,0001 (в процессе обучения ни разу достигнут не был) либо по прошествии максимально заданного числа эпох обучения Epochs.

Результаты вычислений приведены в табл. 1 и на рис. 10 – 15.

Таблица 1

Результаты вычислений

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	β = 0,3; N = 221	1327	368,48	2,4376
RBF	β = 0,6; N = 46	277	85,65	7,0176
RBF	β = 0,9; N = 16	97	45,94	10,1796
NRBF	β = 0,3; N = 231	1387	843,85	1,7775
NRBF	β = 0,6; N = 40	241	166,26	2,5711
NRBF	β = 0,9; N = 16	97	62,78	4,1257
MP	N = 10, Epochs = 20	61	20,32	8,4653
MP	N = 10, Epochs = 50	61	45,74	4,1960
MP	N = 10, Epochs = 100	61	91,72	3,6858
MP	N = 20, Epochs = 20	121	42,22	1,3515
MP	N = 20, Epochs = 50	121	98,80	1,3838
MP	N = 20, Epochs = 100	121	197,06	1,2366
MP	N = 30, Epochs = 20	181	70,22	1,2888
MP	N = 30, Epochs = 50	181	153,44	0,7835
MP	N = 30, Epochs = 100	181	306,66	0,1844

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 базовых сетей CMAC; LDB3 – LDB CMAC, состоящая из 3 сетей CMAC с входами (x_1, x_2), (x_1, x_4), (x_3, x_4); RBF и NRBF – РБС и НРБС соответственно, для которых во второй колонке указано заданное значение критерия добавления нового нейрона β и результирующие количество нейронов N; MP – многослойный персептрон, для которого во второй колонке указывается число нейронов скрытого слоя N и число эпох обучения Epochs.

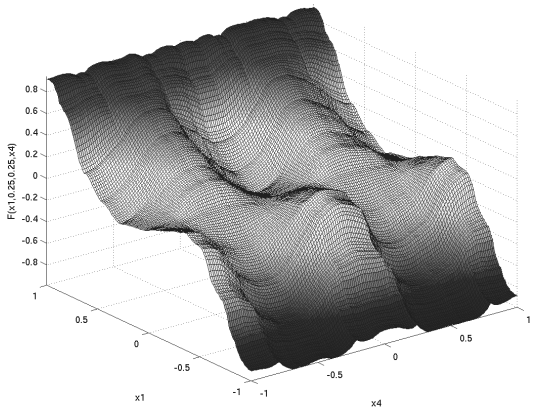

Рис. 10. Восстановление HCMAC с тригонометрическими БФ

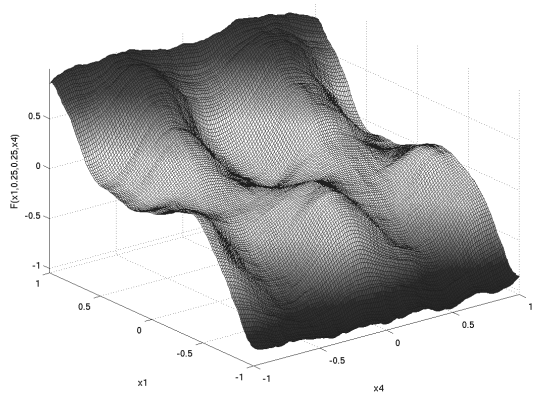

Рис. 11. Восстановление LDB3 CMAC с параболическими БФ

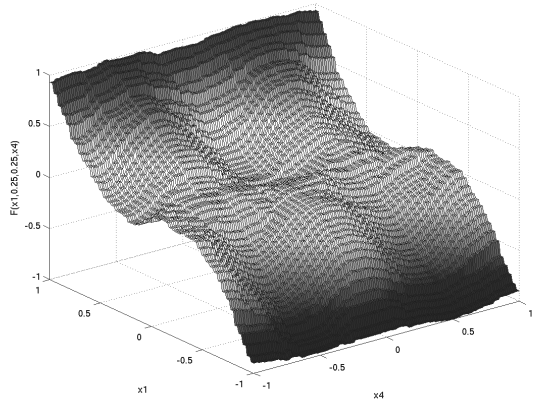

Рис. 12. Восстановление LDB6 CMAC с единичными БФ

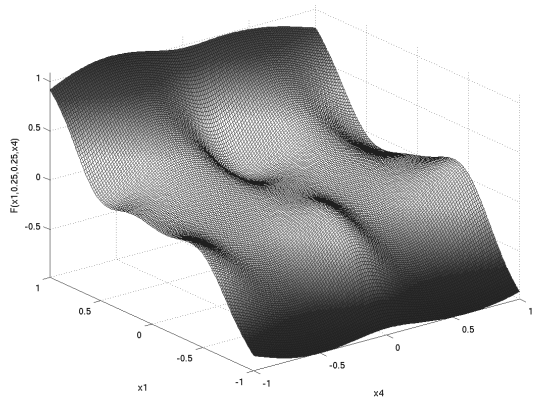

Рис. 13. Восстановление РБС с $\beta = 0.3$ (N = 221)

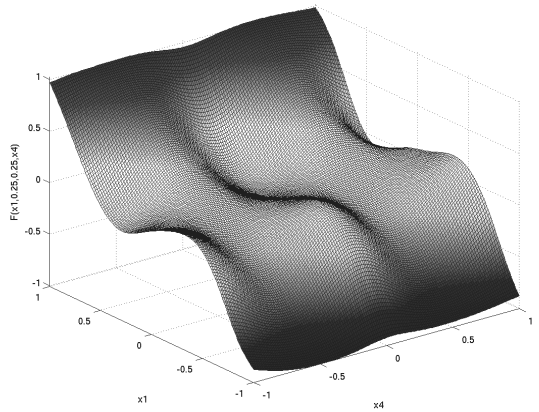

Рис. 14. Восстановление НРБС с $\beta = 0.6$ (N = 40)

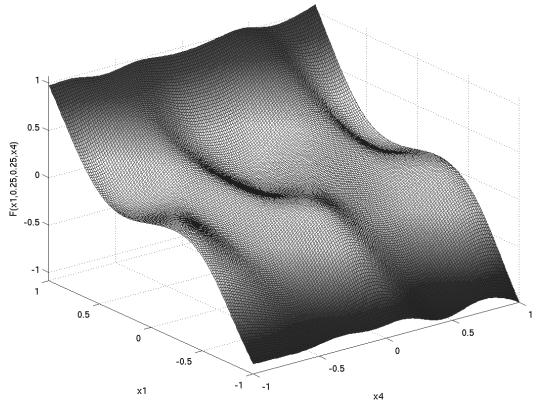

Рис. 15. Восстановление МП с числом нейронов скрытого слоя N = 20, Epochs = 20

Выводы

Результаты исследований свидетельствуют о том, что восстановление зашумленных многомерных функций может быть достаточно эффективно осуществлено путем применения рассмотренных нейронных сетей CMAC, РБС и МП различных архитектур.

Сети CMAC различных архитектур требуют минимума вычислительных затрат и, соответственно, обеспечивают наименьшее время обучения сети. Применение CMAC модифицированных архитектур (линейной LDB CMAC и иерархической HCMAC) во всех случаях позволяет существенно сократить объем требуемой памяти при сохранении приемлемого качества восстановления многомерной нелинейной функции. Для рассмотренного примера объем требуемой CMAC памяти удалось сократить примерно в 6 – 11 раз.

В случае применения HCMAC используемые БФ обязательно должны быть дифференцируемыми, в то время как LDB CMAC позволяет применять произвольные БФ, в том числе и простейшие единичные, что позволяет за счёт небольшого снижения точности аппроксимации ещё более снизить вычислительные затраты. Среди дифференцируемых наиболее простой и в то же время весьма эффективной является параболическая функция.

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.

Применение НРБС позволяет добиться ещё более высокой точности аппроксимации, но за счёт ещё большей вычислительной сложности: время обучения сети по сравнению с ненормализованной РБС увеличивается примерно в два раза.

Наиболее гибким вариантом является использование МП. Отказ от применения традиционных методов обучения в пользу более сложных алгоритмов (метод Левенберга-Марквардта) обеспечивает очень хорошую сходимость и высокую точность аппроксимации даже при несложных конфигураци-

ях сети. Вычислительные затраты, как и в случае применения РБС, довольно высоки, но объем требуемой памяти и среднее значение ошибки аппроксимации при этом существенно ниже.

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. – Х.: 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. – V. 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. – V. 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. – V. 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.

10. Powell M.J.D. Radial Basis Functions for Multivariable Interpolation: A review // Proc. of IMA Conf. on Algorithms for the Approximation of Functions and Data, Shrivenham, UK. – 1985. – P. 143-167.

11. Robert J. Schilling, James J. Carroll, Jr, and Ahmad F. Al-Ajlouni, Approximation of Nonlinear Systems with Radial Basis Function Neural Networks, IEEE Trans. Neural Networks. – 2001. – Vol. 12. – P. 17.

12. Karayiannis N.B. and Mi G.W. Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques // IEEE Trans. – Neural Networks. – 1997. – Vol. 8. – P. 1492-1506.

13. Shepherd A.J. Second-Order Methods for Neural Networks. – New York: Springer, 1997. – 146 p.

14. Zhou G., Si J. Advanced neural-network training algorithm with reduced complexity based on Jacobean deficiency // IEEE Trans. – Neural Networks. – 1998. – V. 9, № 3. – P. 448-453.

Поступила в редколлегию 15.10.2007

Рецензент: д-р техн. наук, проф. О.Г. Руденко, Харьковский национальный университет радиоэлектроники, Харьков.