УДК 519.713

АЛГОРИТМ ОБУЧЕНИЯ ФОРМАЛЬНОГО НЕЙРОНА С ПОЛИНОМИАЛЬНОЙ АКТИВАЦИОННОЙ ФУНКЦИЕЙ

БОДЯНСКИЙ Е.В., ПОЛУШКИНА Н.А.

Предлагается простейшая активационная функция нейрона МакКаллоха-Питтса в виде кубического полинома, удовлетворяющая всем необходимым требованиям, и соответствующий ей алгоритм настройки синаптических весов и параметра крутизны. В основе алгоритма лежит процедура Левенберга-Марквардта, обладающая сглаживающими свойствами и отличающаяся простотой реализации.

1. Введение

В настоящее время искусственные нейронные сети широко применяются для решения разнообразных задач, связанных с интеллектуальным анализом данных в условиях априорной и текущей неопределённости. Этот факт связан, прежде всего, с универсальными аппроксимирующими свойствами нейросетей, доказанными в работах Цыбенко и Хорника [1-3]. Эти авторы показали, что конечная линейная комбинация некоторых одномерных функций особого вида может однозначно аппроксимировать любую непрерывную функцию n действительных переменных на заданном гиперкубе. Эти одномерные конструкции, получившие название сигмоидальных активационных функций, являются непрерывными, монотонными, возрастающими, ограниченными кривыми, выбор которых в общем случае не ограничен какими-либо формальными соображениями.

2. Постановка задачи

Одним из наиболее распространённых элементов нейронных сетей является формальный нейрон МакКаллоха-Питтса, для которого была использована активационная функция

$$0<\psi_1(\gamma_j u_j)=(1+e^{-\gamma_j u_j})^{-1}<1, \qquad (1)$$

определяемая на множестве всех действительных чисел и принимающая только положительные значения. На рис. 1 приведена схема нейрона с активационной функцией (1).

Более удобной оказалась биполярная функция гиперболического тангенса

$$-1<\psi_2(\gamma_j u_j)=\tanh(\gamma_j u_j)=\frac{1-e^{-2\gamma_j u_j}}{1+e^{-2\gamma_j u_j}}<1, \qquad (2)$$

связанная с (1) соотношением

$$\psi_1(\gamma_j u_j)=\frac{1}{2}(\tanh\frac{\gamma_j u_j}{2}+1).$$

Также в качестве активационных достаточно часто используются функции вида

$$\psi_3(\gamma_j u_j)=\frac{\gamma_j u_j}{\sqrt{1+\gamma_j^2 u_j^2}}, \qquad (3)$$

$$\psi_4(\gamma_j u_j)=\sin(\frac{\pi}{2}\gamma_j u_j), \qquad (4)$$

$$\psi_5(\gamma_j u_j)=\frac{2}{\pi}\operatorname{arctg}(\gamma_j u_j) \qquad (5)$$

и им подобные, определённые на квадрате $-1<u_j<1$, $-1<\psi_j(\gamma_j u_j)<1$. Перечисленные выше функции обладают всеми свойствами сигмоидальных активационных функций на заданном единичном квадрате.

Все эти функции имеют подобный друг другу вид и отвечают всем требованиям, сформулированным в [1-3]. Естественно, что выражениями (1)-(5) возможный выбор не ограничивается.

Процесс обучения нейронной сети сводится к настройке вектора синаптических весов

$$w_j=(w_{j0},w_{j1},w_{j2},\ldots,w_{jn})^T$$

путём минимизации целевой функции, в качестве которой достаточно часто используется выражение $E_j(k)=\frac{1}{2}e_j^2(k)$, где $e_j(k)=d_j(k)-y_j(k)$ - ошибка обучения, $d_j(k)$ – внешний обучающий сигнал. Использование процедуры градиентного спуска приводит к так называемому дельта-правилу обучения, имеющему вид [4]:

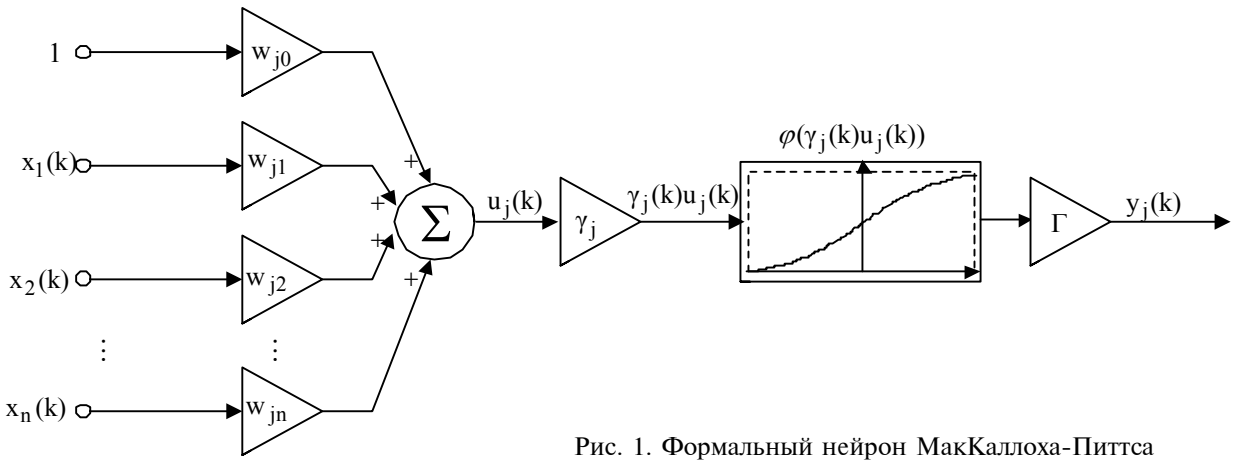

Рис. 1. Формальный нейрон МакКаллоха-Питтса

$$w_j(k+1)=w_j(k)-\eta_w(k)\nabla_{w_j}E_j(k)=$$
$$=w_j(k)-\eta_w(k)\frac{\partial E_j(k)}{\partial e_j(k)}\cdot\frac{\partial e_j(k)}{\partial u_j(k)}\nabla_{w_j}u_j(k)=$$
$$=w_j(k)-\eta_w(k)e_j(k)\nabla_{w_j}e_j(k)=$$
$$=w_j(k)+\eta_w(k)e_j(k)\frac{\partial\psi(\gamma_j(k)u_j(k))}{\partial u_j(k)}x(k)=$$
$$=w_j(k)+\eta_w(k)\delta_j(k)x(k),$$

где $\delta_j(k)=e_j(k)\frac{\partial\psi(\gamma_j(k)u_j(k))}{\partial u_j(k)}$ – локальная ошибка; $\eta_w(k)$ – скалярный параметр, определяющий скорость сходимости процесса настройки синаптических весов.

Хотя в практике обучения искусственных нейронных сетей параметр γ_j обычно полагается постоянным, целесообразно введение настраиваемого параметра крутизны, обеспечивающего повышение скорости обучения и улучшение аппроксимирующих свойств сети. Алгоритм обучения этого параметра был предложен в [5] и также представляет собой градиентную процедуру

$$\gamma_j(k+1)=\gamma_j(k)-\eta_\gamma(k)\frac{\partial E_j(k)}{\partial\gamma_j(k)}=$$
$$=\gamma_j(k)+\eta_\gamma(k)\frac{\partial\psi(\gamma_j(k)u_j(k))}{\partial\gamma_j(k)}.$$

Несложно заметить, что конкретный вид алгоритма обучения определяется используемой активационной функцией, а его "скоростные" характеристики – выбором параметров шага $\eta_w(k)$ и $\eta_\gamma(k)$. Именно выбору достаточно простой и универсальной активационной функции и оптимизации параметров шага посвящена настоящая работа.

3. Полиномиальные активационные функции

В [6] было показано, что активационные функции (2)-(5) могут быть представлены в виде степенного ряда

$$\psi(\gamma_j u_j)=\alpha_0\gamma_j u_j+\alpha_1\gamma_j^3u_j^3+\alpha_2\gamma_j^5u_j^5+\ldots=$$
$$=\sum_{l=0}^{L}\alpha_l(\gamma_j u_j)^{2l+1}, \qquad (6)$$

где коэффициенты разложения α_l определяются соотношениями $\alpha_{l+1}=\beta\alpha_l$, $-1<\beta<0$, $\alpha_0=1$, т.е. фактически представляют собой устойчивый колебательный процесс авторегрессии первого порядка.

Для обучения нейрона с активационной функцией (6) в [7] был предложен алгоритм вида

$$\begin{cases} w_j(k+1)=w_j(k)+\eta_w(k)e_j(k)\times \\ \times\left(\sum_{l=0}^{L}(2l+1)(\gamma_j(k)u_j(k))^{2l}\alpha_l\gamma_j(k)\right)x(k), \\ \gamma_j(k+1)=\gamma_j(k)+\eta_\gamma(k)e_j(k)\times \\ \times\left(\sum_{l=0}^{L}(2l+1)(\gamma_j(k)u_j(k))^{2l}\alpha_l u_j(k)\right) \end{cases} \qquad (7)$$

с неопределёнными параметрами шага обучения.

Простейший вариант (7), удовлетворяющий всем необходимым требованиям, возникает при L=1, т. е. кубический полином

$$\psi(\gamma_j u_j)=\gamma_j u_j-\frac{1}{3}\gamma_j^3u_j^3, \qquad (8)$$

приведенный на рис. 2, также является активационной функцией.

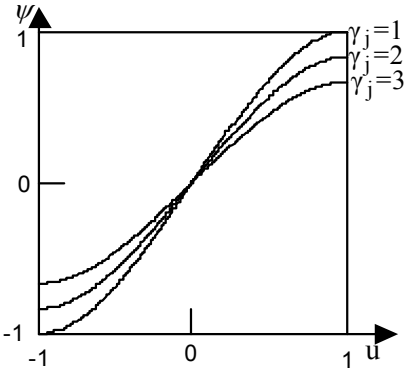

Рис. 2. Кубическая активационная функция

Записав производные (8) по настраиваемым параметрам

$$\frac{\partial(\gamma_j(k)u_j(k)-\frac{1}{3}\gamma_j^3(k)u_j^3(k))}{\partial u_j(k)}=$$
$$=\gamma_j(k)(1-\gamma_j^2(k)u_j^2(k))=\gamma_j(k)\varphi(\gamma_j(k)u_j(k)),$$
$$\frac{\partial(\gamma_j(k)u_j(k)-\frac{1}{3}\gamma_j^3(k)u_j^3(k))}{\partial\gamma_j(k)}=$$
$$=u_j(k)(1-\gamma_j^2(k)u_j^2(k))=u_j(k)\varphi(\gamma_j(k)u_j(k)),$$

приходим к простому алгоритму обучения:

$$\begin{cases} w_j(k+1)=w_j(k)+\eta_w(k)e_j(k)\times \\ \times\varphi(\gamma_j(k)u_j(k))x(k), \\ \gamma_j(k+1)=\gamma_j(k)+\eta_\gamma(k)e_j(k)\times \\ \times\varphi(\gamma_j(k)u_j(k)), \end{cases} \qquad (9)$$

где $\varphi(\gamma_j(k)u_j(k))$ в данном случае есть не что иное, как квадратичная радиально-базисная функция [8] с параметром ширины γ_j^{-1}, приведенная на рис. 3.

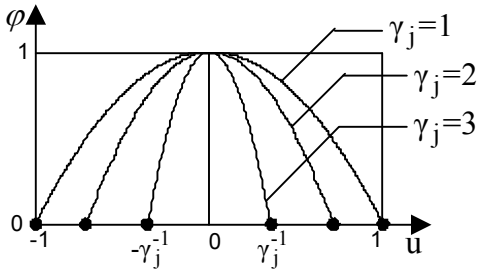

Рис. 3. Квадратичная радиально-базисная функция

На рис.4 показана также схема обучения формального нейрона с помощью алгоритма (9).

Выходной сигнал нейрона, приведенного на рис.4, описывается выражением $y_j(k)=\Gamma_j\psi(\gamma_j(k)w_j^T(k)x(k))$,

где $\Gamma_j<\frac{1}{\gamma_j-\frac{1}{3}\gamma_j^3}$ – коэффициент, определяющий экстремальные значения выходного сигнала $y_j(k)$ и реализуемый дополнительным усилительным блоком, z^{-1} – элемент чистого запаздывания.

4. Алгоритм обучения

Введём составные векторы расширенных входов и настраиваемых параметров:

$$w_j^a(k)=w_j(k)\oplus\gamma_j(k)=(w_j^T(k)\vdots\gamma_j(k))^T,$$

$$x^a(k)=\varphi(\gamma_j(k)u_j(k))(\gamma_j(k)x(k)\oplus u_j(k))=$$

$$=\varphi(\gamma_j(k)u_j(k))(\gamma_j(k)x^T(k)\vdots u_j(k))^T$$

и запишем общий градиентный алгоритм обучения на основе процедуры (9):

$$w_j^a(k+1)=w_j^a(k)+\eta_a(k)e_j(k)x^a(k). \qquad (10)$$

Повышение скорости сходимости процесса настройки всех параметров нейрона может быть обеспечено использованием вместо градиентного спуска (10) алгоритма Левенберга-Марквардта [9], принимающего в данном случае форму

$$w_j^a(k+1)=w_j^a(k)+(x^a(k)x^{aT}(k)+rI)^{-1}e_j(k)x^a(k), \qquad (11)$$

или при малых значения параметра регуляризации r

$$w_j^a(k+1)=w_j^a(k)+(x^a(k)x^{aT}(k))^+e_j(k)x^a(k),$$

где $I-(n+2)\times(n+2)$ – единичная матрица, $(\bullet)^+$ – псевдообращение по Муру-Пенроузу.

Применив к (11) формулу Шермана-Моррисона, после несложных преобразований можно переписать алгоритм обучения в гораздо более простой форме:

$$w_j^a(k+1)=w_j^a(k)+\frac{e_j(k)x^a(k)}{r+\left\|x^a(k)\right\|^2}=$$

$$=w_j^a(k)+r^{-1}(k)e_j(k)x^a(k). \qquad (12)$$

Заметим, что в отличие от (11), процедура (12) не использует операции обращения матриц и является по сути одной из модификаций популярного в теории обучения нейронных сетей алгоритма Уидроу-Хоффа [10]. Рассчитывая параметр шага в (12) согласно рекуррентному соотношению

$$r(k)=rr(k-1)+\left\|x^a(k)\right\|^2, \quad 0\le r\le 1,$$

приходим к расширенной версии процедуры стохастической аппроксимации [11] , получившей широкое распространение в задачах адаптивной идентификации динамических систем. Варьируя параметром r, можно обеспечить процессу обучения как сглаживающие (фильтрующие), так и следящие свойства.

5. Заключение

Предложен простой и эффективный алгоритм обучения формального нейрона с полиномиальной активационной функцией, предназначенный для использования в многослойных нейронных сетях и позволяющий существенно упростить численную реализацию процедуры обратного распространения ошибок.

Литература: 1. *Cybenko G.* Approximation by superposition of sigmoidal function //Math.Contr.Sign.Syst. 1989. №2.

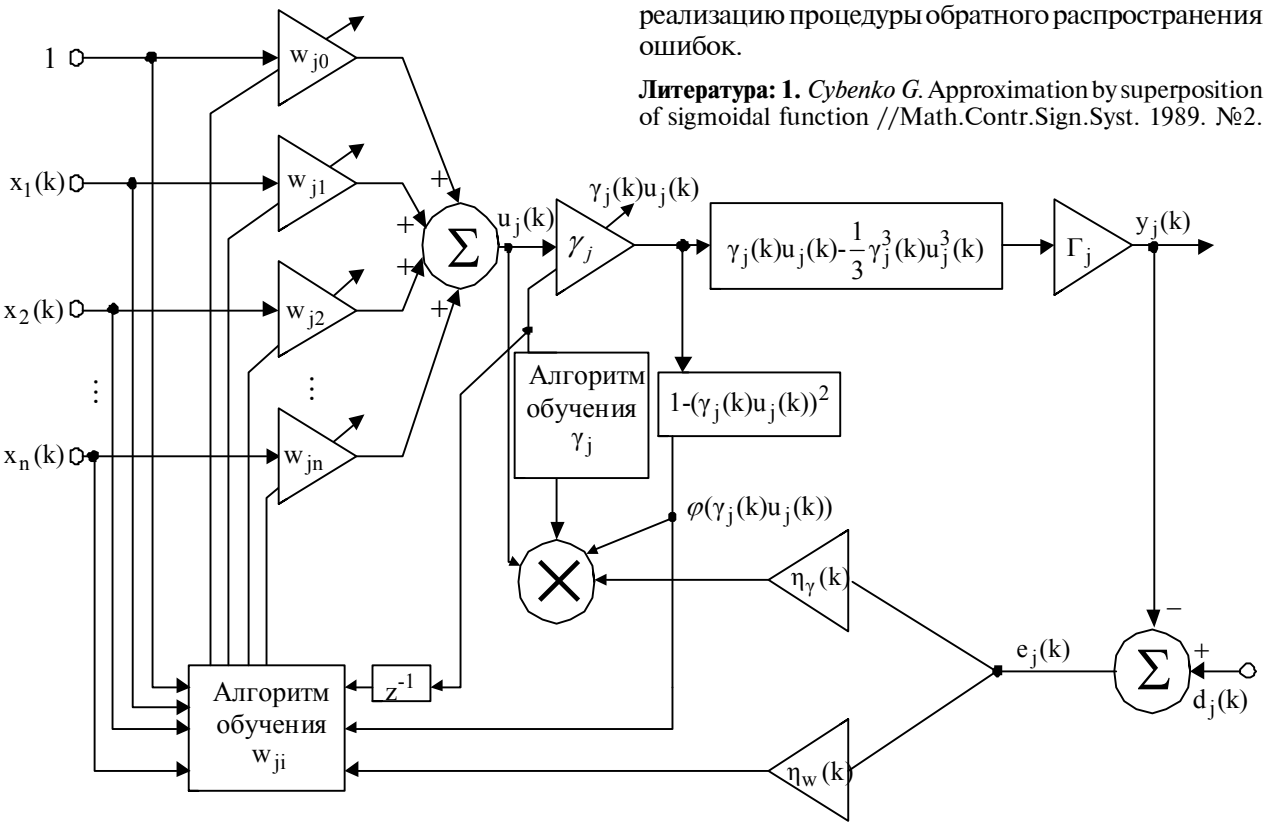

Рис. 4. Схема обучения формального нейрона с полиномиальной функцией активации

P.303-304. **2.** *Hornik K., Stinchcombe M., White H.* Multilayer feedforward networks are universal approximators // Neural Networks.1989.№2.P.359-366. 3**.** *Hornik K.* Approximation capabilities of multiplayer feedforward networks // Neural Networks.1991.4.P.251-257. **4.** *Golden R.M.* Mathematical Methods for Neural Networks Analysis and Design.-Cambridge, Massachusetts: The MIT Press, 1996. 420p. **5.** *Kruschke J.K., Movellan S.R.* Benefits of gain: Speeded learning and minimal layers backpropagation networks// IEEE Trans. on Syst., Man and Cybernetics.1991.21. P.273-280. **6.** *Бодянский Е.В., Кулишова Н.Е., Руденко О.Г.* Обобщённый алгоритм обучения формального нейрона // Кибернетика и системный анализ. 2002. №6. С.176-182. **7.** *Бодянский Е.В., Кулишова Н.Е., Руденко О.Г.* Об одной модели формального нейрона // Доповіді НАН України. 2001. №4. С.69-73. **8.** *Bodyanskiy Ye., Chaplanov O., Kolodyazhniy V., Otto P.* Adaptive quadratic radial basis function networks for time series forecasting // Proc. East West Fuzzy Coll.-Zittau/Goerlitz: HS, 2002. P.164-172. **9.** *Shepherd A.J.* Second-Order Methods for Neural Networks.-London: Springer-Verlag, 1997. 145p. **10.** *Ham F.M., Kostanic I.* Principles of Neurocomputing for Science and Engineering. N.Y.:Mc Graw-Hill, Inc., 2001. 642p. **11.** *Goodwin G.C., Ramadge P.J., Caines P.E.* A globally convergent adaptive predictor // Automatica.-1981. №1. P.135-140.

Поступила в редколлегию 23.07.2003

Рецензент: д-р техн. наук, проф. Любчик Л.М.

Бодянский Евгений Владимирович, д-р техн. наук, профессор кафедры искусственного интеллекта, научный руководитель ПНИЛ АСУ ХНУРЭ, член IEEE, WSES. Научные интересы: нейро-фаззи-системы. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-890.

E-mail:Bodyanskiy@ieee.org, bodya@kture.kharkov.ua

Полушкина Наталия Александровна, ст. лаборант каф. филосифии ХНУРЭ. Научные интересы: искусственные нейронные сети, нейросетевое прогнозирование. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-465. E-mail: Natalie_p@mail.ru

УДК 519.5

ОБ ОДНОМ ПОДХОДЕ К ПОСТРОЕНИЮ МАТЕМАТИЧЕСКОЙ МОДЕЛИ ПРОГРАММНОГО АГЕНТА

ФИЛАТОВ В.А.

Рассматривается подход к построению модели программного агента на основе фреймовой структуры. Обосновывается формальный подход, используемый в теории образов, к задаче проектирования логической структуры взаимосвязанных программных агентов. Предложенные модели могут использоваться для разработки мультиагентных систем администрирования информационных ресурсов распределенных вычислительных систем.

Введение

Исследования в области интеллектуальных агентов и мультиагентных систем представляют собой одну из наиболее интересных областей развития современных информационных технологий. По оценкам специалистов в области искусственного интеллекта агентно-ориентированный подход может сыграть важнейшую роль в становлении информационного общества будущего. Уже сегодня он широко применяется в таких областях как управление информационными потоками (workflow management), диагностика и управление в сетевых структурах (network management), информационный поиск (information retrieval), электронная коммерция (e-commerce), обучение (education) и ряде других.

Существует большое количество определений понятия "программный агент" в зависимости от взгляда на распределенную обработку знаний. С точки зрения распределенных вычислений, агент – это самостоятельный процесс, имеющий определенное состояние и способный взаимодействовать с другими агентами посредством передачи сообщений. В этом смысле его можно рассматривать как естественное развитие парадигмы объектно-ориентированного параллельного программирования. По существу программные агенты выполняют задачи, делегируемые им пользователями. В зависимости от поставленной задачи, агенты могут размещаться на Web-серверах, настольных персональных компьютерах или в локальных сетях.

Другая точка зрения ассоциируется с областью распределенного искусственного интеллекта, где термин "агент" имеет более узкое и специфическое значение, чем упомянутое выше. В соответствии с ним, "агент" – это компьютерная система, которая в дополнение к перечисленным выше свойствам реализована с использованием концепций, свойственных человеку. Например, с такими понятиями, как знания, убеждения, намерения, обязательства [1].

Цель исследования

Целью проводимых исследований является разработка математической модели программного агента, которая позволит реализовать многоуровневую, иерархическую систему автономного администрирования распределенных информационных ресурсов вычислительной системы.

Рассмотрим один из вариантов построения модели управления информационными ресурсами на основе мультиагентной системы. Главными составляющими такой системы являются:

– агент – приложение, функционирующее на выделенном ему компьютере и используемое для сбора, анализа, защиты информации;

– система управления агентами (менеджер агентов);
– программа-администратор мультиагентного пространства, осуществляющая контроль работы системы, накопления и анализа информации, полученной от агентов в процессе их функционирования;

– конструктор агентов – программный модуль, обеспечивающий логическое и физическое конструирование агентов. Он отвечает за построение, генерацию и запуск агентов на удаленные компьютеры;

– библиотека функций – набор модулей, из которых строится программный агент. Каждая конкретная логическая функция на этапе проектировании должна иметь реализацию в виде программного модуля;