СБОРНИК **НАУЧНЫХ ТРУДОВ НГТУ. – 2016. – № 4 (86).** – 104–113

УДК 519.688

НЕЙРОННАЯ СЕТЬ ДЛЯ СОРТИРОВКИ МАССИВА ЧИСЕЛ*

К.А. ЧЕРДАНЦЕВ[1], А.В. КЛАДЬКО[2]

[1] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, студент факультета автоматики и вычислительной техники. E-mail: medmene@yandex.ru*
[2] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, студент факультета автоматики и вычислительной техники. E-mail: tonkladko@ya.ru*

Сети Петри и нейронные сети в последнее время набирают популярность и стали применяться в различных сферах деятельности, таких как управление в автоматизированных системах, анализ текстовых данных, анализ графических данных. Одна из важных тем программирования – сортировка массивов данных. В нашем случае данные – это массивы чисел в диапазоне от нуля до ста. Нейронная сеть, обученная сортировать массивы чисел в интервале 0…100, может сортировать и большие числа. Нейронные сети бывают однослойные и многослойные. Последние позволяют «видеть» более сложные закономерности у входных параметров. В работе использованы два варианта нейронных сетей для полной оценки эффективности нейронных сетей в обработке массивов данных. Нейронная сеть написана на языке C++. Модель класса Neuron, нейронная сеть и многослойная сеть разработаны без использования готового кода.

Ключевые слова: нейронные сети Петри, алгоритмы, эксперимент, сортировка, многослойная нейронная сеть, однослойная нейронная сеть, числовые массивы, оценка алгоритмов, программирование

DOI: 10.17212/2307-6879-2016-4-104-113

ВВЕДЕНИЕ

При решении задач управления, анализа текстовых и графических данных в последние годы стали использовать как теорию сетей Петри [1–8] (разработка встраиваемого программного обеспечения, на пример для центров дистанционного контроля и управления, при разработке АСУ ТП, управление манипуляторами и т. д.)

* Статья получена 12 октября 2016 г.

Задача сортировки элементов массивов данных является классической, это одна из важных задач программирования. В классическом виде в ее реализации используют точные алгоритмы, самые известные из которых пузырьковая сортировка, сортировка вставками, сортировка выбором. Проведение эксперимента для сортировки массива данных позволит определить эффективность нейронных сетей в области анализа и обработки массивов данных.

1. ПОСТАНОВКА ЗАДАЧИ

Чтобы реализовать сортировку массива чисел с использованием нейронной сети, нужно реализовать класс neuron и все его методы, направленные на обработку информации, реализовать класс «нейронная сеть» и методы группировки нейронов, подачи информации, получение результатов их работы и их обучение.

Нейронная сеть и соответственно каждый нейрон будут иметь четыре входа, то есть четыре числа из массива, и, следовательно, надо реализовать функцию для получения всех комбинаций этих четырех чисел.

2. РЕАЛИЗАЦИЯ

Однослойная нейронная сеть

В реализации нейронной сети использованы следующие формулы:

активационная функция $f = \frac{1}{1+e^{-s}}$, где s – функция ядра нейрона;

сумматор (функция ядра) $s = \sum_{i=1}^{n} x_i w_i$, где x_i – i-й вход, w_i – i-й синапс, n – количество нейронов;

функция корректировки весов $w_i = w_i + C(o_j - y)x_i |w_i|$, где w_i – i-й синапс, C – коэффициент обучения, o_j – ответ j-го нейрона, x_i – i-й вход.

Class Neuron

В классе содержатся следующие поля, отвечающие за основные характеристики нейрона:

int number – номер нейрона

int countIn – количество входов у нейрона

vector<double> weights – веса синапсов

Функции работы с нейроном:

– void ReWeights() – функция, нужная для обучения, перераспределяет веса

– double activFunc – функция активации нейрона, приводит ответ нейрона к виду 0 или 1

– double summate – функция ядра (ячейка нейрона), суммирует произведения входов и весов синапсов

– double answer – функция ответа нейрона.

Описание функций класса Neuron:

– функция ReWeights аналогична конструктору нейрона, она с помощью стандартной функции rand присваивает новые значения весам

– функция, реализующая формулу активационной функции

– функция, реализующая формулу сумматора

– опрос ответа нейрона с помощью пунктов b & c.

Class NeuralNetwork

В классе содержатся следующие поля:

int inputs – входы нейронов

vector<Neuron> neuron – непосредственно массив нейронов

int countNeurons – количество нейронов

double coef – коэффициент обучения

double *nrn_rez – массив ответов нейронов

Эти поля отвечают за основные характеристики ИНС.

Также есть функции:

– NeuralNetwork – конструктор, который создает первоначальную, необученную нейронную сеть

– void Study(int *in, int rez) – функция обучения, получающая вектор входных значений и ожидаемый результат

– int answ – функция ответа сети, в отличие от обучения она не корректирует веса, а только дает ответ сети

– void SaveWeights & void LoadWeights() – функция сохранения весов с наименьшей ошибкой в файл

– void ReWeight() – функция перераспределения весов на уровне сети.

Принципы работы функций класса NeuralNetwork:

– конструктор поочередно инициализирует все нейроны и распределяет им веса;

– функция обучения сначала получает ответы нейронов на данный входной вектор значений, а затем меняет веса нейронов по методу наискорейшего спуска или же, если ответ верный, ничего не меняет;

– функция ответа получает ответы нейронов и формирует ответ сети, не меняя при этом веса;

– функции сохранения и чтения весов нужны для как можно более точного обучения ИНС, которое длится долгое время;

– функция перераспределения весов нужна в том случае, если количество ошибок стало не уменьшаться, а увеличиваться.

Многослойная нейронная сеть

Класс Neuron не поменялся, а вместо класса NeuralNetwork был реализован схожий класс NeuralNetworkManyLayers, в котором:

int inputs – входы нейронов

vector<vector<Neuron>> neuron – непосредственно массив нейронов

int countNeurons – количество нейронов

double coef – коэффициент обучения

double **nrn_rez – массив ответов нейронов

int layers – количество слоев

Конструктор NeuralNetwork изменен на NeuralNetworkManyLayers(int inp_, int neur_, int lay_). Он теперь учитывает количество слоев и также случайным образом распределяет веса нейронам. Остальные методы класса NeuralNetwork не изменились. Добавился метод RezultLay, который возвращает значение ответа нейрона предыдущего слоя для подачи на вход нейрона текущего слоя.

3. ОБУЧЕНИЕ НЕЙРОННОЙ СЕТИ ДЛЯ ДОСТИЖЕНИЯ МИНИМАЛЬНОЙ ОШИБКИ

Чтобы обучить нейронную сеть сортировать массив из четырех элементов, была создана вспомогательная функция, возвращающая все комбинации четырех чисел, чтобы далее каждую комбинацию подать на нейронную сеть и проверить, является она сортированной или нет.

Основой обучения нейронной сети был бесконечный цикл (while(true)) в котором:

- генерировался входной вектор значений (случайным образом) и генерировался ответ к нему;
- на каждом втором шаге цикла вектор сортировался, чтобы обучать ИНС на сортированных и несортированных последовательностях;
- сортированная или несортированная последовательность подавалась на ИНС вместе с ожидаемым ответом;

- последним шагом являлась проверка на количество ошибок. Задавался цикл на 1000 экспериментов и переменная «счетчик», которая по итогу цикла показывала количество ошибок;
- для отслеживания минимальной ошибки за все эксперименты была введена дополнительная переменная mink;
- при достижении определенного порога ошибок веса нейронов сохранялись в файл, а также выводились в консоль, чтобы отслеживать обучение сети.

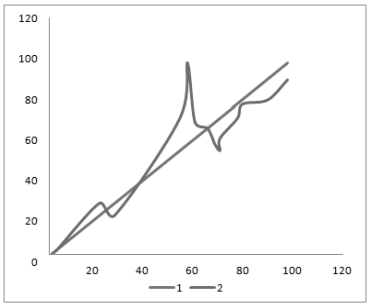
4. АЛГОРИТМ СОРТИРОВКИ БОЛЬШИХ МАССИВОВ

Для сортировки массива размерностью больше четырех элементов использовался следующий алгоритм:

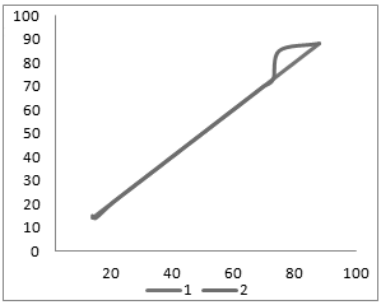
- считываем массив чисел из файла;
- находим число chunk, равное длине массива, деленной на четыре;
- формируем второй массив (выходной), который будем редактировать;
- заводим переменную смещения (k) для постепенного перемещения по всему массиву;
- при условии, что $k = 0$, сортируем массив блоками по четыре элемента до конца, затем увеличиваем переменную k;
- при условии, что $k! = 0$, сортируем массив блоками с k-го элемента до предпоследнего блока, затем увеличиваем переменную k;
- как только $k = 4$, присваиваем ей значение 0;
- проверяем отсортированность массива циклом (если текущий элемент меньше следующего, то увеличиваем переменную слежения);
- если количество отсортированных элементов не меняется более десяти раз, то выводим на экран результат сортировки.

5. РЕЗУЛЬТАТЫ

На рисунке приведены результаты эксперимента.




```
Input size(max. 1000):
16
Start array:
76 68 51 10 38 59 46 51 74 58 40 43 78 99 7 36
Sorted array:
10 7 36 38 40 46 43 51 51 58 68 59 74 76 78 99
Для продолжения нажмите любую клавишу . . .
```




```
Input size(max. 1000):
16
Start array:
85 34 40 69 50 54 21 15 14 32 73 35 88 70 32 53
Sorted array:
15 14 21 32 32 34 35 40 50 53 54 69 70 73 85 88
Для продолжения нажмите любую клавишу . . .
```

Отклонение массива отсортированного ИНС от идеального

В результате первого этапа данная ИНС имеет ошибку 0,04 (40 ошибок из 1000).

ЗАКЛЮЧЕНИЕ

По результатам этой работы видно, что ИНС не очень хорошо подходит для сортировки массива чисел. Это было заранее известно, так как при сортировке обычно используют четкие, известные и явно реализуемые алгоритмы, которые дают точный результат. В случае с ИНС мы получаем массив, примерно сортированный по возрастанию, с редкими значительными отклонениями от полностью отсортированного массива. Точность, достигнутая для данной ИНС, равна 4 % (40 ошибок из 1000 экспериментов) для однослойной ИНС за 24 часа непоследовательного обучения, для многослойной нейронной сети точность достигла 15 % (150 ошибок из 1000) за два часа последовательного обучения.

БЛАГОДАРНОСТИ

Авторы выражают свою искреннюю благодарность профессору кафедры автоматики А.А. Воеводе за помощь при выполнении работ, а также полезное обсуждение полученных результатов.

СПИСОК ЛИТЕРАТУРЫ

1. *Воевода А.А., Романников Д.О., Зимаев И.В.* Применение UML диаграмм и сетей Петри при разработке встраиваемого программного обеспечения // Научный вестник НГТУ. – 2009. – № 4. – С. 169–174.

2. *Воевода А.А., Романников Д.О.* Редуцирование пространства состояний сетей Петри для объектов одного класса // Научный вестник НГТУ. – 2011. – № 4. – С. 136–139.

3. *Коротиков С.В., Воевода А.А.* Применение сетей Петри в разработке программного обеспечения центров дистанционного управления и контроля // Научный вестник НГТУ. – 2007. – № 4. – С. 15–32.

4. *Воевода А.А., Романников Д.О.* О компактном представлении языков раскрашенных сетей Петри // Сборник научных трудов НГТУ. – 2008. – № 3 (53). – С. 105–108.

5. *Воевода А.А., Марков А.В., Романников Д.О.* Разработка программного обеспечения: проектирование с использованием UML диаграмм и сетей Петри на примере АСУ ТП водонапорной станции // Труды СПИИРАН. – 2014. – Вып. 3 (34). – С. 218–231.

6. *Воевода А.А., Марков А.В.* Рекурсия в сетях Петри // Сборник научных трудов НГТУ. – 2012. – № 3 (69). – С. 115–122.

7. *Марков А.В., Воевода А.А.* Развитие системы «перемещение манипулятора в пространстве с препятствиями» при помощи рекурсивных функций // Автоматика и программная инженерия. – 2013. – № 2 (4). – С. 35–41.

8. *Capannini G., Silvestri F., Baraglia R.* Sorting on GPUs for large scale datasets: a thorough comparison // Information Processing and Management. – 2011. – Vol. 48 (5). – P. 903–917.

9. *Галушкин А.И.* Нейронные сети. Основы теории. – М.: Горячая Линия-Телеком, 2010. – 496 с.

10 *Каллан Р.* Основные концепции нейронных сетей. – М.: Вильямс, 2001. – 291 с.

11. *Барский А.Б.* Нейронные сети: распознавание, управление, принятие решений. – М.: Финансы и статистика, 2004. – 176 с.

12. *Хайкин С.* Нейронные сети. Полный курс. – М.: Вильямс, 2006. – 1104 с.

13. *Медведев В.С., Потемкин В.Г.* Нейронные сети. – М.: Диалог-МИФИ, 2002. – 496 с.

14. *Яхъяева Г.Э.* Нечеткие множества и нейронные сети. – М.: Бином, 2006. – 374 с.

15. *Комашинский В.И., Смирнов Д.А.* Нейронные сети и их применение в системах управления связи. – М.: Горячая Линия-Телеком, 2003. – 94 с.

Черданцев Константин Артурович, студент факультета автоматики и вычислительной техники Новосибирского государственного технического университета. Имеет одну публикацию. E-mail: medmene@yandex.ru.

Кладько Антон Владимирович, студент факультета автоматики и вычислительной техники Новосибирского государственного технического университета. E-mail: tonkladko@ya.ru.

Sorting neural network*

K.A. Cherdantsev[1], A.V. Klad'ko[2]

[1] *Novosibirsk State Technical University, 20 Karl Marks Avenue, Novosibirsk, 630073, Russian Federation, doctor of Technical Sciences, professor of the automation department. E-mail: ucit@ucit.ru*

[2] *Novosibirsk State Technical University, 20 Karl Marks Avenue, Novosibirsk, 630073, Russian Federation, candidate of Technical Sciences, associate professor of the automation department. E-mail: dmitry.romannikov@gmail.com*

The applicationof microcontrollers is becoming increasingly widespread in various areas of information processing. Actual questions on the implementation of various algorithms, the development of software, is encountered in an increasing number of papers. Some algorithms are fairly simple, but require careful software development. But there are more complex algorithms, that should simulate the functioning, for example, of neural networks and Petri nets. In this paper, we show the sequence of designing a device that, at the first sight, is not complex but which requires careful study. The study outlines the progress of the project to create an LED matrix for displaying any information.

Keywords: Neural networks, algorithms, experiment, sorting, multilayer neural network, single layer neural network, numeric arrays, evaluation algorithms, programming

DOI: 10.17212/2307-6879-2016-4-104-113

REFERENCES

1. Voevoda A.A., Romannikov D.O., Zimaev I.V. Primenenie UML diagramm i setei Petri pri razrabotke vstraivaemogo programmnogo obespecheniya [An approach to the using UML and Petri nets for embedded software designing]. *Nauchnyi vestnik Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta – Science bulletin of the Novosibirsk state technical university*, 2009, no. 4, pp. 169–174.

2. Voevoda A.A., Romannikov D.O. Redutsirovanie prostranstva sostoyanii setei Petri dlya ob"ektov odnogo klassa [Reducing the state space of Petri nets for objects of one class]. *Nauchnyi vestnik Novosibirskogo gosudarstvennogo tekhnich-*

* *Received 12 October 2016.*

eskogo universiteta – Science bulletin of the Novosibirsk state technical university, 2011, no. 4, pp. 136–139.

3. Korotikov S.V., Voevoda A.A. Primenenie setei Petri v razrabotke pro-grammnogo obespecheniya tsentrov distantsionnogo upravleniya i kontrolya [Using Petri nets in software development of remote monitoring and control center]. *Nauchnyi vestnik Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta – Science bulletin of the Novosibirsk state technical university*, 2007, no. 4, pp. 15–32.

4. Voevoda A.A., Romannikov D.O. O kompaktnom predstavlenii yazykov raskrashennykh setei Petri [On the compact representation of the languages of colored Petri nets]. *Sbornik nauchnykh trudov Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta – Transaction of scientific papers of the Novosibirsk state technical university*, 2008, no. 3 (53), pp. 105–108.

5. Voevoda A.A., Markov A.V., Romannikov D.O. Razrabotka programmnogo obespecheniya: proektirovanie s ispol'zovaniem UML diagramm i setei Petri na primere ASU TP vodonapornoi stantsii [Software development: software design using UML diagrams and Petri nets for example automated process control system of pumping station]. *Trudy SPIIRAN – SPIIRAS proceedings*, 2014, iss. 3 (34), pp. 218–231.

6. Voevoda A.A., Markov A.V. Rekursiya v setyakh Petri [The concepts recursion in Petri nets]. *Sbornik nauchnykh trudov Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta – Transaction of scientific papers of the Novosibirsk state technical university*, 2012, no. 3 (69), pp. 115–122.

7. Markov A.V., Voevoda A.A. Razvitie sistemy "peremeshchenie manipulyatora v prostranstve s prepyatstviyami" pri pomoshchi rekursivnykh funktsii [Development of the system "moving the manipulator in the obstacle space" with the help of recursive functions]. *Avtomatika i programmnaya inzheneriya – Automatics & Software Enginery*, 2013, no. 2 (4), pp. 35–41.

8. Capannini G., Silvestri F., Baraglia R. Sorting on GPUs for large scale datasets: a thorough comparison. *Information Processing and Management*, 2011, vol. 48 (5), pp. 903–917.

9. Galushkin A.I. *Neironnye seti. Osnovy teorii* [Neural network. Base theory]. Moscow, Hotline-Telecom Publ., 2010. 496 p.

10. Callan R. *The essence of neural networks*. London, Prentice Hall Europe, 1999 (Russ. ed.: Kallan R. *Osnovnye kontseptsii neironnykh setei*. Moscow, Vil'yams Publ., 2001. 291 p.).

11. Barskii A.B. *Neironnye seti: raspoznavanie, upravlenie, prinyatie reshenii* [Neural network: recognition, control, accept solutions]. Moscow, Finansy i statistika Publ., 2004. 176 p.

12. Haykin S. *Neural networks*. Upple Saddle River, Prentice Hall, 1999 (Russ. ed.: Khaikin S. *Neironnye seti. Polnyi kurs*. Moscow, Vil'yams Publ., 2006. 1104 p.).

13. Medvedev V.S., Potemkin V.G. *Neironnye seti* [Neural network]. Moscow, Dialog-MIFI Publ., 2002. 496 p.

14. Yakh"yaeva G.E. *Nechetkie mnozhestva i neironnye seti* [Fuzzy sets and neural networks]. Moscow, Binom Publ., 2006. 374 p.

15. Komashinskii V.I., Smirnov D.A. *Neironnye seti i ikh primenenie v sistemakh upravleniya svyazi* [Neural networks. Using in communications system]. Moscow, Hotline-Telecom Publ., 2003. 94 p.