СПИСОК ЛИТЕРАТУРЫ

1. Колесенков А. Н. Современные подходы к обработке данных при построении геоинформационных систем экологического мониторинга // Изв. Тульского гос. ун-та. Техн. науки. 2016. № 9. С. 103–111.

2. The space syntax toolkit: Integrating depthmapX and exploratory spatial analysis workflows in QGIS / J. Gil, T. Varoudis, K. Karimi, A. Penn // SSS 2015–10th Intern. Space Syntax Symp. Space Syntax Laboratory, The Bartlett School of Architecture. London, UK: University College London, 2015. Vol. 10. P. 1–19.

3. Доан Д. Х., Крошилина С. В., Жулева С. Ю. Использование нечеткой кластеризации в анализе статистической нечеткой медицинской информации для формирования наборов вариантов течения болезни в системах поддержки принятия медицинских решений // Тр. Междунар. науч.-техн. конф. Воронеж: ФГБОУ ВО «Воронежский гос. техн. ун-т», 2017. Т. 1. С. 268.

4. Сайт RapidMiner: Data Science Platform. URL: https://rapidminer.com/ (дата обращения 01.06.2018).

Ya. A. Bekeneva, S. I. Lebedev, I. I. Kholod, E. S. Novikova
Saint Petersburg Electrotechnical University «LETI»

EVENT CLASSIFICATION DEPENDING ON THE SET OF INDEPENDENT ATTRIBUTES

Events at the production facility that can be recorded by monitoring devices could be not related to each other and carried out by different types of entities. In this case, the attribute sets may not coincide completely, except for the attributes that are common to all the fixed events. In this regard, large data sets describing a variety of different processes often consist of records that differ significantly in structure and composition of attributes. When aggregate analysis of such data by the methods of intellectual analysis, there may be a problem with the choice of a method suitable for such a set of data. In addition, the authors suggest that different analysis methods can be used for different data groups. In this paper, we propose a method for splitting the data sets into groups, depending on the composition of the attributes.

Data grouping, data attributes, data from heterogeneous sources, decision tree, data mining

УДК 006.72

А. А. Воевода, Д. О. Романников
Новосибирский государственный технический университет

Формирование структуры нейронной сети посредством декомпозиции исходной задачи на примере задачи управления роботом-манипулятором

Предлагается способ формирования структуры нейронной сети, основанный на декомпозиции исходной задачи, результатом которой является набор состояний, в которых может находиться система, и признаков смены состояний. Предлагается построить конечный автомат, в котором переходы и сам конечный автомат представлены составными частями нейронной сети, а каждому состоянию соответствует отдельная нейронная сеть, с помощью которой выполняется управление роботом-манипулятором в соответствующем состоянии. Построение нейронной сети состоит из трех этапов: 1) реализация части сети для определения признаков переходов между состояниями; 2) реализация части сети для определения состояния системы; 3) реализация нейронной сети для каждого состояния при формировании выходных сигналов и объединение выходных сигналов от разных состояний. Итоговая нейронная сеть позволяет получить большую наблюдаемость и возможность «отладки», а также в значительной мере упростить процесс обучения за счет использования более простых типов нейронных сетей и решаемых задач.

Нейронные сети, структура нейронной сети, обучение с подкреплением, автоматизация, конечный автомат, управление роботом-манипулятором

В настоящее время нейронные сети применяются для решения таких задач, как распознавание речи и изображений [1], генерация заголовков изображений [2] и др. В [3] предлагается метод обучения нейронных сетей с подкреплением (reinforcement learning), который на основе взаимо-

действия со средой позволяет минимизировать целевую функцию для произвольной системы.

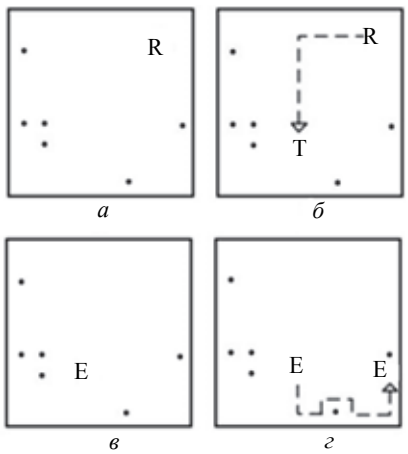
Метод обучения с подкреплением применялся при создании системы для игры в го, системы управления охлаждением в дата-центрах, системы управления игроком в игре Doom, где предложено использование дополнительных задач с целью ускорения обучения [4]. В данной статье в отличие от вышеуказанных подходов предлагается решение задачи управления роботом-манипулятором, основанное на декомпозиции исходной задачи на составные части и представлении их в виде состояний, соединенных переходами. Каждому состоянию соответствует работа одной из составных частей нейронной сети, а с помощью переходов осуществляется их смена.

Несмотря на существенные успехи применения метода обучения с подкреплением, данный метод не лишен недостатков, к которым можно отнести общие недостатки нейронных сетей: неопределенность при выборе структуры нейронной сети, начальных условий, метода обучения, проблемы переобучения (overfitting) и недообучения (underfitting) и др. С другой стороны, существует необходимость наличия тестовой среды, а также сложность описания функции вознаграждения (reward function), в частности, проблемы «забывания» устаревших данных, получения противоречивых данных для обучения и сложность выявления причин, которые привели к текущему состоянию [5].

Проблему медленного обучения нейронных сетей при использовании метода обучения с подкреплением решали многие авторы. Так в [4] предлагается метод обучения, основанный на использовании предварительно обученной нейронной сети при помощи «демонстрации» «правильных» действий (Deep Q-learning from Demonstrtions); в [6] предлагается подход к ускорению обучения за счет более частого обучения на приоритетных данных, где приоритет определяется на основании ошибки временной разницы (temporal difference) при переходе между состояниями в марковской цепи.

С другой стороны, в настоящий момент накоплен существенный опыт проектирования и реализации программных систем, сходных с нейронными сетями (в части функционального подхода). Стоит отметить, что среди вышеперечисленных методов и подходов к ускорению обучения выделяется метод обучения с предварительной демонстрацией, использующий знания о конкретной задаче. Такой подход можно перенести и на другие аспекты нейронных сетей. В частности, в статье предлагается объединение идеи о программном представлении нейронной сети и использовании знаний о специфике конкретной задачи при помощи построения структуры нейронной сети на основании декомпозиции задачи на состояния, в которых может находиться система, а также на переходы между этими состояниями. Таким образом, исходная задача рассматривается как конечный автомат, в котором каждому состоянию и переходу соответствует своя нейронная сеть. Предлагаемый способ позволяет не только ускорить процесс обучения нейронной сети, но и получить частичную наблюдаемость нейронной сети, а именно возможность отладки каждого состояния или перехода нейронной системы.

Постановка задачи. Предлагаемый способ построения нейронной сети рассматривается на основе примера реализации автоматической системы управления роботом-манипулятором, которая способна выполнять последовательность таких действий, как перемещение к указанной цели, перемещение по заданной траектории, возврат к исходной позиции. Входными данными для разрабатываемой системы являются координатное поле (представляется в виде сетки с размерами 15 × 15), на котором отмечены робот-манипулятор (символ R), препятствия (отмечены точками), заданные координаты (символ T), достигший заданных координат робот-манипулятор (символ E). Визуализированные примеры координатных полей приведены на рис. 1. Пример, соответствующий начальному положению робота, изображен на рис. 1, *а*; пример появления заданных координат приведен на рис. 1, *б*. Штриховой линией отмечена одна из возможных траекторий перемещения робота к заданным координатам. Пример достижения роботом заданных координат изображен на рис. 1, *в*. Последний пример (рис. 1, *г*) показывает часть траектории (отмеченной штрихами) перемещения роботом-манипулятором по периметру поля после захвата цели в указанных координатах T.



а *б* *в* *г*

Рис. 1

Последовательность выполняемых действий следующая: изначально робот находится в состоянии покоя и не должен совершать никаких действий; при появлении координат робот должен перемещаться к ним, а по их достижении двигаться по периметру координатной сети обходя препятствия; после завершения обхода робот-манипулятор должен вернуться на исходную позицию. Для управления должны выполняться следующие действия: перемещения влево/вправо/вверх/вниз. Отсутствие перемещения определяется отсутствием команд на перемещение в любую из сторон.

Построение структуры нейронной сети. В [7] рассматривается синтез структуры нейронной сети для логико-арифметических задач на основании предварительной разработки сети Петри, которая выступает языком описания алгоритма. Описание алгоритма для решения поставленной задачи управления роботом-манипулятором выполняется при помощи построения конечного автомата (рис. 2), отражающего те состояния, в которых робот-манипулятор находится при совершении заданной последовательности действий. Каждое из этих состояний можно разбивать на более детальные до тех пор, пока реализация и процесс обучения нейронной сети, соответствующие состоянию, не станут достаточно простыми задачами.

Согласно [4] любой конечный автомат можно представить в виде рекуррентной нейронной сети. Тогда конечный автомат на рис. 2 можно преобразовать в нейронную сеть, структура которой изображена на рис. 3.

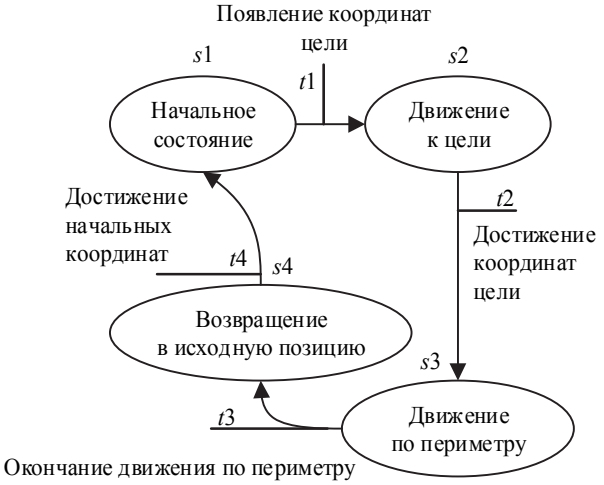


Рис. 2

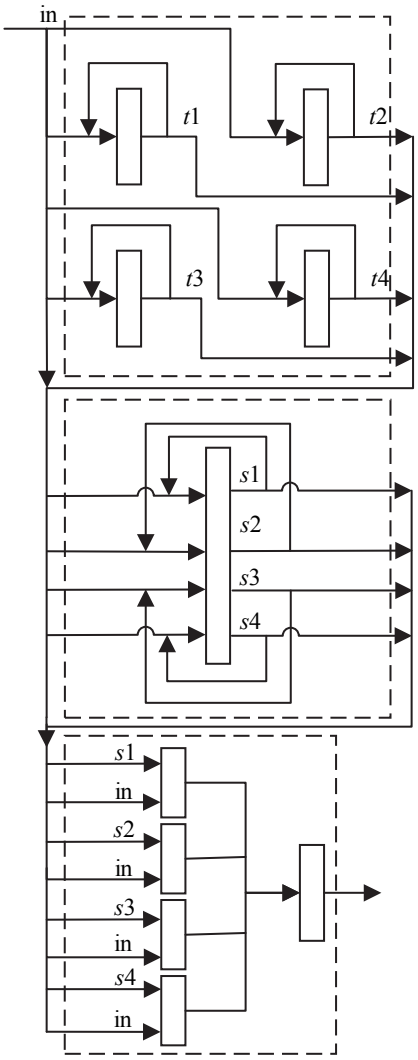


Рис. 3

На вход in нейронной сети (рис. 3) поступает двумерный массив координатной сети, из которого формируются выходные сигналы переходов $t1$–$t4$ при помощи рекуррентных нейронных сетей. На каждую нейронную сеть для формирования сигналов перехода, кроме данных координат, также поступает выходной сигнал перехода для формирования сигнала на выходах $t1$–$t4$. Все сигналы переходов поступают на часть нейронной сети, в которой определяются состояния $s1$–$s4$. Эта часть сети также является рекуррентной, так как выходной сигнал зависит как от значений переходов, так и от текущих значений состояний. В каждый момент времени на выходе подсети может быть только одно состояние $s1$–$s4$. После определения состояний значения их сигналов $s1$–$s4$ поступают вместе со значениями координатной сети in на входы подсетей, в которых определяются управляющие сигналы в зависимо-

сти от состояния. Последней частью нейронной сети является объединение выходных сигналов управления от всех подсетей. Другой возможный вариант – формирование управляющих сигналов в подсетях состояний независимо от текущего состояния с последующим объединением с учетом значений состояний.

Подход к формированию нейронной сети после выделения каждого из состояний и переходов с целью формирования нейронной сети для смены состояний может быть алгоритмизирован. Алгоритм построения нейронной сети основан на следующих требованиях: 1) на выходе состояния сигнал единицы должен присутствовать все время, в течение которого состояние активно; 2) состояние меняется тогда, когда возникает переход из активного состояния; 3) при нулевых значениях всех переходов и состояний активным считается первое состояние. Исходя из этих требований можно сформулировать алгоритм, приведенный в псевдокоде:

```
nns1 = {} // хеш-таблица состояния к массиву
          // слоев нейронной сети входного слоя
nns2 = {} // хеш-таблица состояния к массиву
          // слоев нейронной сети выходного слоя

for каждого состояния s ∈ S:
   nns1.put(s, [])
   nns2.put(s, [])

nns1.get(s1).add(слой для нулевого состояния)

for каждого состояния s ∈ S:
   for каждого перехода t ∈ T:
      if t исходит из s:
         nns1.get(s)
             .add(слой для перехода t и состояния s)
         nns1.get(s)
             .add(слой для состояния s и отсутствия
переходов)

for каждого состояния s ∈ S:
   outs = []
   for каждого входного слоя l ∈ nns1.get(s):
      outs.add(выход нейронной сети слоя l)
   nns2.get(s)
       .add(слой для объединения выходов outs)
```

Работа алгоритма начинается с того, что две хеш-таблицы (nns1, nns2), в которых будут храниться слои нейронной сети, инициализируются пустыми массивами всех состояний. В nns1 добавляется слой для определения начального состояния – при нулевых значениях всех состояний и переходов. Далее для каждого исходящего из итерируемого состояния s перехода добавляется нейронный слой для определения состояния s и слой для формирования сигнала единицы на выходе нейронной сети при состоянии s и отсутствии остальных переходов. После формирования входных слоев нейронной сети для объединения результатов для одинаковых состояний необходимо их сгруппировать. Для этого для каждого состояния s выходы нейронных слоев, хранящихся в nns1[s], объединяются по условию «или» и являются выходными нейронами сети для определения состояний.

Пример применения описанного алгоритма приведен на рис. 4. На вход нейронной сети подаются признаки переходов $t1$–$t4$ и состояний $s1$–$s4$. Состояние $s1$ формируется объединением результатов случая начального состояния (нулевые сигналы $t1$–$t4$ и $s1$–$s4$), срабатывания перехода $t4$ при активном состоянии $s4$ и при отсутствии сигналов переходов $t1$–$t4$ (на рис. 4 обозначается горизонтальной чертой над переходами $t1$–$t4$) и активном состоянии $s1$. Признаки срабатывания других состояний формируются аналогично, за исключением учета начального состояния. Обучение данной сети может быть выполнено двумя способами: традиционным обучением с учителем, при котором с помощью размеченного набора данных одним из методов обучения определяются значения параметров обучения сети, или обсуждаемым в [7] способом, при котором значения коэффициентов определяются вручную.

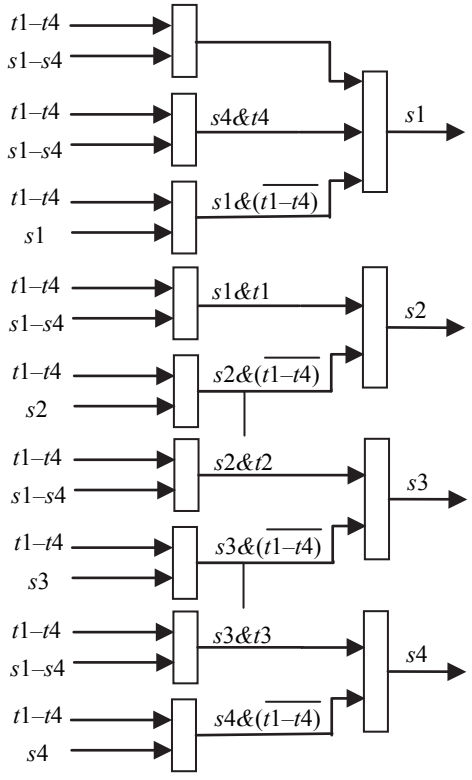


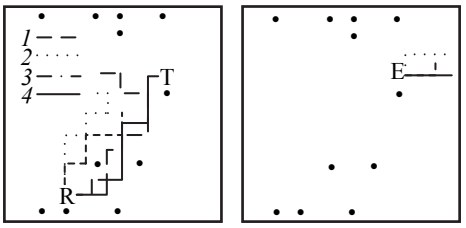
Рис. 4

Последним этапом исполнения нейронной сети (см. рис. 3) является формирование выходных сигналов управления по полученным значениям выходных сигналов от каждой нейронной сети состояний *s*1–*s*4. Способ заключается в том, что на вход нейронной сети состояний *s*1–*s*4 подается дополнительный вход соответствующего состояния. При нулевом значении данного входа (это означает, что в текущий момент времени исполняется другое состояние) сеть обучается выдавать нулевое управление. При значении единицы – сеть формирует выходные значения управления для соответствующего состояния. Далее результаты всех выходов нейронных сетей состояния объединяются по условию «или».

Реализация и обучение. Реализация нейронных сетей выполнялась в программном пакете tensorflow (v1.4.0). При формировании управляющих сигналов для состояния *s*1 используется многослойный персептрон с двумя скрытыми слоями в 100 и 30 нейронов. На вход сети подается двумерный массив координатной сети, на котором находятся две метки – положение робота-манипулятора и координаты, к которым он должен добраться. В качестве стоимостной функции достаточно использовать функцию перекрестной энтропии (cross entropy), а для обучения был выбран метод Adam (Adaptive Subgradient Methods for Online Learning and Stochastic Optimization) с параметром обучения 0,01. Обучение сети осуществлялось при помощи сгенерированных примеров, в которых управляющие сигналы определялись как направление движения по кратчайшему пути в графе от положения робота до искомой цели. Обучение выполнялось в течение 200 эпох (при большом количестве данных для обучения используется практика разбиения исходного набора данных на «пачки» меньшего размера с целью уменьшения памяти и увеличения скорости обучения. При этом нейронная сеть обучается на каждой «пачке» данных последовательно, что и называется эпохой обучения).

Для состояний *s*2–*s*4 структура сети и принцип генерации данных схожи, за исключением того, что в состоянии *s*3 (движение по периметру) изначально выполнялся переход к одной из «стенок» координатной сети, а затем движение по часовой стрелке с учетом препятствий (данные формировались также посредством наикратчайшего пути в графе по периметру).

Реализация перехода *t*1, который используется для определения появления координат цели, выполнена при помощи многослойного персептрона с одним скрытым слоем и обратной связью. Вход сети состоит из 226 элементов, из которых 225 – элементы для обработки входа координационной сети и 1 вход – для обработки обратной связи значения выхода нейронной сети. Данной структуры достаточно в силу того, что признаки на координационной сети сильно отличаются друг от друга (так как рассматриваются численные значения меток, а все остальные значения заполнены нулями). Однако при необходимости любая часть представленной нейронной системы может быть заменена на более сложную, в том числе и на сверточные нейронные сети для более точного распознавания или др. Реализация остальных переходов осуществляется аналогичным образом, и нейронные сети переходов *t*2–*t*4 имеют схожую структуру. Обучение данной сети выполнялось при помощи метода Adam на 1000 сгенерированных примерах с равномерным распределением наличия и отсутствия сигнала перехода в 10 000 эпох. Траектории перемещения робота-манипулятора при различных эпохах обучения приведены на рис. 5.



а *б*

Рис. 5

На рис. 5, *а* приведена визуализация траектории движения робота-манипулятора при различном количестве эпох обучения. После одной эпохи перемещение не совершается. Траектория перемещения после 10 эпох показана штриховой линией (цифра *1* на рис. 5, *а*); траектория перемещения после 50 эпох – пунктиром (цифра *2* на рис. 5, *а*); после 100 – штрихпунктиром (цифра *3* на рис. 5, *а*). Все вышеприведенные траектории зациклились и не завершились в заданных координатах. После 150 эпох обучения робот-манипулятор смог достичь заданных координат по траектории, изображенной сплошной линией (цифра *4* на рис. 5, *а*). На рис. 5, *б* изображены траектории передвижения робота-манипулятора при разном количестве эпох обучения.

В результате декомпозиции выявляются состояния, в которых может находиться робот-манипулятор, и признаки смены этих состояний, на основании которых можно представить исходную задачу в виде конечного автомата. Предлагается использовать полученную структуру конечного автомата для реализации нейронной сети. Реализация нейронной сети условно разбивается на реализацию нескольких нейронных подсетей: 1) для реализации смены состояний; 2) для определения признаков переходов между состояниями; 3) для формирования выходов нейронной сети для каждого состояния, а также объединения всех выходов нейронных подсетей каждого состояния и формирования выходов всей нейронной сети.

Предлагаемый способ, основанный на формировании структуры нейронной сети, позволяет упростить процесс реализации и обучения нейронной сети за счет разбиения изначальной задачи на более простые, реализация и обучение которых является известной задачей. Также данный подход позволяет получить частичную наблюдаемость нейронной сети, когда в каждый момент времени известно текущее состояние, значения признаков переходов и управляющие сигналы как всей нейронной сети, так и каждого из состояний.

СПИСОК ЛИТЕРАТУРЫ

1. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet Classification with Deep Convolutional Neural Networks // Proc. of Neural Information Processing Systems (NIPS). 2012. P. 1097–1105.

2. Karpathy A., Fei-Fei L. Deep Visual-Semantic Alignments for Generating Image Descriptions // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2016. Vol. 39. P. 664–676.

3. Playing Atari with Deep Reinforcement Learning / V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller // Proc. of Neural Information Processing Systems (NIPS) Workshop. 2013. P. 1160–117.

4. Learning from Demonstrations for Real World /Reinforcement Learning / T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. Agapiou, J. Z. Leibo, A. Gruslys // Proc. AAAI'16 Proc. of the Thirtieth AAAI Conf. on Artificial Intelligence. Phoenix. USA, 2017. P. 2094–2100.

5. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge: MIT Press, 2016. 800 p.

6. Prioritized Experience Replay / T. Schaul, J. Quan, I. Antonoglou, D. Silver // Proc. ICLR. New Orleans, 2016. P. 1260–1268.

7. Воевода А. А., Романников Д. О. Синтез нейронной сети для решения логико-арифметических задач // Тр. СПИИРАН. 2017. Вып. 54. С. 205–223.

A. A. Voevoda, D. O. Romannikov
Novosibirsk Technical University

FORMATION OF THE STRUCTURE OF A NEURON NETWORK THROUGH DECOMPOSITION OF THE INITIAL TASK ON A PARTICULAR EXAMPLE OF THE ROBOT-MANIPULATOR MANAGING PROBLEM

Currently, neural networks are widely used in the construction of many systems in which the development of an algorithm is a serious problem. However, neural networks are considered as a «black box», and its structure is in practice obtained by carrying out a lot of experiments and choosing the best option available. The article proposes a method of forming a neural network structure on the basis of the original problem decomposition, resulting in a set of states the system can be in, and the signs of the state changes. Next, it is proposed to construct a finite automaton whose structure is the main one for the realization of a neural network. Building of a neural network consists of three stages: 1) the implementation of the part for determining the signs of transitions between states; 2) the implementation of the part for determining the state of the system; 3) the implementation of a neural network for each state for generating the output signals and combining the output signals from different states. The resulting neural network allows for greater observability and the possibility of «debugging», as well as greatly simplifies the learning process by using simpler types of neural networks and problems being solved.

Neural networks, neural network structure, reinforcement training, automation, finite state machine, robot manipulator control